SIAM J. SCI. COMPUT.
Vol. 29, No. 1, pp. 1–21
© 2007 Society for Industrial and Applied Mathematics

A MULTIRESOLUTION APPROACH TO THE ELECTRIC FIELD INTEGRAL EQUATION IN ANTENNA PROBLEMS*

F. P. ANDRIULLI†, A. TABACCO†, AND G. VECCHI†

Abstract. This paper deals with a multiresolution approach to the finite-element solution of the electric field integral equation (EFIE) formulation of the boundary value problem for Maxwell equations. After defining a multiresolution set of discretized spaces, each of them is first separated into solenoidal and nonsolenoidal complementary spaces. The possibility of obtaining these two spaces with a scalar-to-vector space mapping is used to consider first two separate scalar wavelet decompositions, and then to transform them properly into the two desired vector finite-element sets, finally joined to obtain the complete multiresolution basis. Numerical results, obtained for real life antennas, are presented to verify the actual sparsity of the system matrix and to show the fast convergence behavior of iterative solvers.

Key words. multiresolution analysis, wavelets, integral equations, method of moments, triangular meshes, antennas

AMS subject classifications. 65N30, 35B25, 42C15

DOI. 10.1137/050634943

1. Introduction.

The problem of finding the electromagnetic field radiated by an antenna and its network parameters (impedance, scattering matrix) is of key importance in applied electromagnetics, and the related numerical simulation tools are part of all practical antenna design. The typical antenna structure is constituted of metal conductors (typically assumed ideal) and of dielectric bodies that are piecewise homogeneous; the related analysis thus amounts to the solution of Maxwell equations with boundary conditions to be enforced on the surface of material discontinuity. It is to be noted that a fully three-dimensional electromagnetic problem is intrinsically vector-valued and in general cannot be transformed in a straightforward manner to multiple scalar problems. Various numerical methods have been devised and applied to antenna analysis over the past decades, either based on the differential formulation of the boundary-value problem or on its transformation into an integral equation; for a review, one can, e.g., refer to [13].

In the following, we will explicitly consider antennas made of metal structures embedded in a homogeneous medium (air, vacuum, etc.) called "free space"; the metals will be considered perfect electric conductors (PECs). Although not directly addressed in the presented results, the method proposed here is directly applicable to the case of conductors embedded in a layered dielectric medium that can be considered infinite in two directions (as is commonly assumed in the analysis of "printed" antennas and circuits). We will consider the integral-equation formulation of the boundary-value problem for time-harmonic dependence, which is very well suited for this class of structures.

*Received by the editors July 1, 2005; accepted for publication (in revised form) May 31, 2006; published electronically January 8, 2007. This research was supported in part by the Italian funds MIUR Prin '05 "Analisi Armonica," MIUR Prin '03 "Innovative technologies for antennas for multimedia services," and FIRB "Primo."

http://www.siam.org/journals/sisc/29-1/63494.html

†Politecnico di Torino, Corso Duca degli Abruzzi 24, Torino, 10129, Italy (box.andriulli@tin.it, anita.tabacco@polito.it, giuseppe.vecchi@polito.it).

In our setting, we will consider antennas that can be modelled by a two-dimensional surface Γ that need not be close. The electric field integral equation (EFIE) for the boundary-value problem is obtained (see, e.g., [13]) by invoking the equivalence theorem ("Huygens' principle") to substitute the PEC surface with unknown equivalent surface currents $\boldsymbol{J}$ that radiate in free space; the total electric field $\boldsymbol{E}$ is then separated into a "primary" (or "incident") field $\boldsymbol{E}_{in}$ produced in free space by the independent sources, and a secondary (or "scattered") field $\boldsymbol{E}_{sc}$ produced by the equivalent currents, $\boldsymbol{E} = \boldsymbol{E}_{in} + \boldsymbol{E}_{sc}$. The boundary conditions on the conductor surface require that the total tangent field be zero, whose enforcement leads to the integral equation for the unknown current $\boldsymbol{J}$:

$$
\begin{aligned}
-i\omega\mu \iint_{\Gamma} \frac{e^{ik|\boldsymbol{r}-\boldsymbol{r}'|}}{4\pi\,|\boldsymbol{r}-\boldsymbol{r}'|}\ \boldsymbol{J}(\boldsymbol{r}')\ \mathrm{d}S(\boldsymbol{r}') + \\
+\frac{1}{\epsilon}\nabla_s \iint_{\Gamma} \frac{e^{ik|\boldsymbol{r}-\boldsymbol{r}'|}}{4\pi\,|\boldsymbol{r}-\boldsymbol{r}'|}\nabla_s\cdot\boldsymbol{J}(\boldsymbol{r}')\ \mathrm{d}S(\boldsymbol{r}') - \boldsymbol{E}_{in}^{tan}(\boldsymbol{r}) = 0,
\end{aligned}
\tag{1.1}
$$

where a time dependence $\exp(-i\omega t)$ has been assumed and suppressed, ϵ and μ denote the permeability and the permittivity of the free space, $k = \omega\sqrt{\epsilon\mu}$ is the wave-number, and ∇_s indicates the surface divergence. The operator setting of the EFIE (1.1) and the solution spaces are discussed in [8, 4]. The discussion of the above issues is, however, outside the scope of the present paper, as will be explained later.

The parameters of practical interest (radiated field, impedance, etc.) are obtained from operations on the surface current $\boldsymbol{J}$. In scattering problems, the source term $\boldsymbol{E}_{in}$ is typically an incident plane (or spherical) wave radiated by a distant source; the connection between the source term and the physical problem is more involved for antennas, where the feeding region is part of the problem. The typical choice is to relate the primary field $\boldsymbol{E}_{in}$ to the so-called voltage-gap between the antenna (circuit) terminals (e.g., [13]). A more satisfactory model involves magnetic-current sources [2, 5, 10], and it can be shown that in this case the impedance is a stationary (variational) functional of the current $\boldsymbol{J}$; however, implementation of this source modelling is clumsier than the widespread voltage-gap approximation.

The numerical solution of the problem is customarily obtained via finite-element approximation of the unknown $\boldsymbol{J}$ and weighted-residual (weak) enforcement of the EFIE (1.1). Summarizing the basic steps of the procedure, one begins with choosing an approximating space X of finite dimension d, spanned by a suitable basis x_i, $i = 1 \dots d$, yet to be discussed, and accordingly expressing the approximate sought-for solution via

$$
\boldsymbol{J} = \boldsymbol{J}_X + \boldsymbol{e}, \qquad \boldsymbol{J}_X = \sum_i I_i\, \boldsymbol{x}_i,
\tag{1.2}
$$

where $\boldsymbol{J}_X$ is the approximate solution, obtained by projection onto X, and $\boldsymbol{e}$ is the approximation error; when (1.2) is inserted into (1.1), the approximation error gives rise to a residual term. The equation (1.1) is then weakly enforced by requiring that the residual term be zero upon projection onto a specified finite-dimensional test space T; the most common choice is the Galerkin choice, in which $T = X$. This gives rise to the finite-dimensional linear system

$$
\left[Z^A + Z^{\phi}\right][I] = [V],
\tag{1.3}
$$

where

$$\left[Z^A\right]_{p,q} = i\omega\mu \int_\Gamma \boldsymbol{x}_p(\boldsymbol{r}) \cdot \int_\Gamma \frac{e^{ik|\ -\ '|}}{4\pi\,|\boldsymbol{r}-\boldsymbol{r'}|} \boldsymbol{x}_q(\boldsymbol{r'})\ \mathrm{d}S(\boldsymbol{r'})\,\mathrm{d}S(\boldsymbol{r}), \tag{1.4}$$

$$\left[Z^\phi\right]_{p,q} = -\frac{i}{\omega\epsilon} \int_\Gamma \nabla_s \cdot \boldsymbol{x}_p(\boldsymbol{r}) \cdot \int_\Gamma \frac{e^{ik|\ -\ '|}}{4\pi\,|\boldsymbol{r}-\boldsymbol{r'}|} \nabla_s \cdot \boldsymbol{x}_q(\boldsymbol{r'})\ \mathrm{d}S(\boldsymbol{r'})\,\mathrm{d}S(\boldsymbol{r}), \tag{1.5}$$

$$[V]_i = -\int_\Gamma \boldsymbol{x}_i(\boldsymbol{r}) \cdot E_i^{tan}(\boldsymbol{r})\ \mathrm{d}S(\boldsymbol{r}), \tag{1.6}$$

$$[I]_i = I_i. \tag{1.7}$$

The overall approach is a boundary-element method, but in computational electromagnetics the above approach is widely known with the name "method of moments" (MoM). For a general shape conductor, the construction of the approximation space X has two steps: (a) the construction of a mesh on the surface Γ, or surface discretization, and (b) the construction of an appropriate finite-element basis over the cells of the mesh. In the following, we will assume the most employed such basis, constructed upon a triangular-facet approximation of the actual geometry; the basis was first introduced in computational electromagnetics by Rao, Wilton, and Glisson in [15], and since then the basis functions have been called RWG; for a discussion on the history of this basis see [13]. The RWG functions are edge-defined (e.g., there is one RWG function per inner edge of the mesh) and are linear-tangential continuous-normal (LTCN) with respect to the defining edge; they are also div-conforming (refer to section 2 for the formal definition). The system matrix $[Z] = \left[Z^A\right] + \left[Z^\phi\right]$ is fully populated, with poor conditioning and poor convergence performance of iterative solvers for practical large problems. Furthermore, the condition number of the problem and the speed of convergence degrade with increasing density of the mesh (e.g., [23]).

Among the several attempts to overcome the difficulties of the standard approach, the properties of multiresolution representations have attracted researchers of computational electromagnetics. Chief among these is the property of having, under certain conditions, vanishing moments; this is usually considered the key to the sparsification of the MoM matrix, i.e., the clipping of entries below a given threshold. The earlier works (e.g., [18, 7]), and many of those that use more structured wavelets, are confined to the analysis of two-dimensional (i.e., scalar) problems, or to wire-type problems in which the current direction is one-dimensional. For the sake of brevity we will not attempt to list them here, since this work specifically concentrates on the issue of vector electromagnetic problems for surfaces. Due to the difficulties of extending wavelets to this latter class of problems, applications in this sense are more recent; in [1, 16, 9] the analysis of planar circuits on rectangular meshes is addressed, taking advantage of the separability of the current into two Cartesian components. A wavelet-related approach has been applied to scattering from plates in [19], again employing the separability of the problem, and to triangular grids in [20] separating the current directions into local Cartesian and diagonal components. In [12], Coifman intervallic wavelets are applied to scattering problems from closed smooth bodies, for which the surface current can be conveniently decomposed along two locally orthogonal directions (azimuth and elevation components on spheres and spheroids).

In our approach, we want to keep the geometric flexibility of the RWG approach, yet construct a multiresolution (MR) basis for X; our procedure will generate a hierarchical sequence of successively refined meshes over which the MR functions will be

defined, but in the end we will be able to express all of our MR basis as a linear combination of the RWG basis for the finest mesh. This has the important consequence that the present method allows the reuse of existing MoM codes for the RWG basis.

Another relevant feature of our approach is the use of the MR basis as a preconditioner; i.e., in addition to the more usual emphasis on the sparsifying potential of the MR bases, we will also focus on the effect of the MR transformation on the spectrum of the system matrix, as it results from the improved convergence properties of iterative solvers.

In our approach, we employ first a mapping between the approximate current $\boldsymbol{J}_X$ and two isotropically scalar quantities [22] that can be viewed as projection of the Helmholtz decomposition onto the RWG space X. This allows the use of scalar MR functions on triangular grids, and this issue has already been addressed in the literature [21]. The vector-to-scalar transformation will be done with a proper operator mapping, which ensures the preservation of the necessary wavelet properties. The employed mapping to scalars is based on the separation of the current $\boldsymbol{J}_X$ into a solenoidal component and a quasi-irrotational remainder. The side benefit of this decomposition is that it prevents the low-frequency problem of the EFIE, i.e., the instability that appears when the cell size is much smaller than the wavelength, as often happens when the size of the mesh cells is dictated by frequency-independent fine geometrical details and not by the operating wavelength.

The present approach follows the strategy outlined in [14] and applied there to planar and quasi-planar antennas and circuits on rectangular grids. Here, we extend that approach to a triangular grid; the present paper has to be considered a companion to [25]. The emphasis of the present paper is on the mathematical properties of the developed MR vector-valued basis; conversely, a detailed account of the algorithmic issues and their application is given in [25, 24].

The paper is organized as follows: In section 2 we describe the MR analysis setting. All the computation and justification are postponed to the appendices in order to avoid technicalities at a first reading. Finally, in section 3 we present our numerical results and we compare them with the previous approaches.

2. Multiresolution analysis. We start defining projection spaces X_j that will be used in the following. In doing this we need a triangular mesh on Γ. For sake of simplicity we will assume that Γ is a faceted structure; i.e., the triangularization can be obtained using planar triangles (examples can be found in Figures 3.1, 3.2, and 3.3). If this is not verified, a continuous bijection is needed in addition to the following construction [26]. In antenna problems, a widely used finite-element space is the one introduced in [11] and applied to computational electromagnetics in [15], spanned by the so called [illegible] (RWG) functions. These are defined on the inner edges of the mesh, and, referring to Figure 2.1, their expression in local coordinates is

$$\boldsymbol{f}_p = \begin{cases} \dfrac{l_n}{2A_p^+}\rho_p^+, & \boldsymbol{r} \in T_p^+, \\ \dfrac{l_n}{2A_p^-}\rho_p^-, & \boldsymbol{r} \in T_p^-, \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

Let us define

$$X_j^{\mathrm{RWG}} = \operatorname{span}\left\{\boldsymbol{f}_1, \ldots, \boldsymbol{f}_{n_{ed}^j}\right\},$$

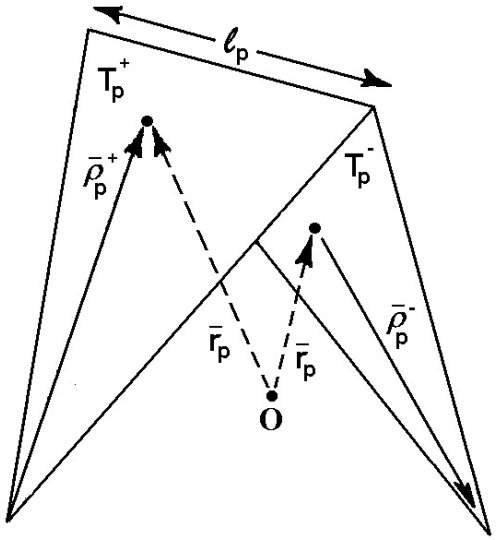


FIG. 2.1. *Rao–Wilton–Glisson function.*

where j labels a particular mesh and n_{ed}^j is the number of inner edges relative to the mesh j.

Considering the antenna surface Γ and a mesh at level j defined on it, we can obtain a finer mesh bisecting each edge (refer to Figure 2.2). The obtained mesh will be labelled $j+1$. Thus on each mesh one can define an RWG space, obtaining spaces X_j^{RWG}, X_{j+1}^{RWG}, $X_{j+2}^{\mathrm{RWG}}, \ldots$ which satisfy the inclusion relationship

$$X_j^{\mathrm{RWG}} \subset X_{j+1}^{\mathrm{RWG}} \tag{2.2}$$

(see Appendix B).

The relationship (2.2) makes spaces X_j^{RWG} eligible for a wavelet construction, according to the formal setting described in Appendix A.1. The coarsest mesh will be labelled $j = 0$. Referring to notation introduced in Appendix A.1, it is a natural choice to set $V_j = X_j^{\mathrm{RWG}}$. Now we look for spaces W_j^{RWG} that satisfy the decomposition relationship

$$X_{j+1}^{\mathrm{RWG}} = X_j^{\mathrm{RWG}} \oplus W_j^{\mathrm{RWG}}$$

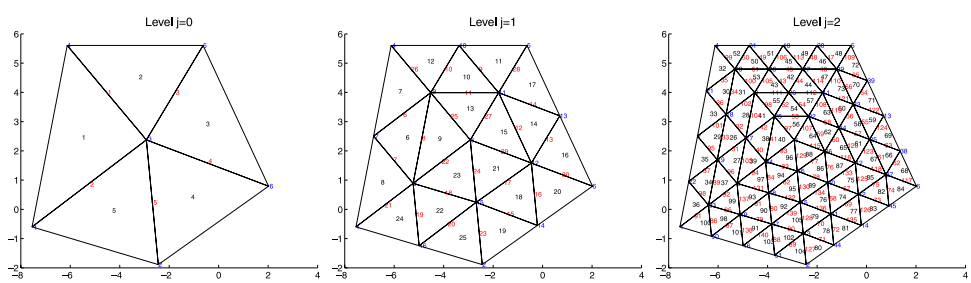


FIG. 2.2. *Different level meshes.*

or, equivalently,

$$X_l^{\mathrm{RWG}} = X_0^{\mathrm{RWG}} \oplus \bigoplus_{j=0}^{l} W_j^{\mathrm{RWG}}, \tag{2.3}$$

where X_0^{RWG} is the space spanned by the RWG functions defined on the coarsest level mesh.

The space W_j^{RWG} will be decomposed into two parts denoted W_j^T and W_j^B, respectively. The space W_j^T is obtained working separately on each cell. More precisely,

$$W_j^T = \bigoplus_{i=1}^{n_{cell}^0} W_{j,i}^T, \tag{2.4}$$

where the spaces $W_{j,i}^T$ are suitable detail spaces of functions with domains included in each of the single coarsest level cells i. The space W_j^B is instead a connection space in the sense that

$$W_j^{\mathrm{RWG}} = W_j^T \oplus W_j^B \tag{2.5}$$

and it takes into account functions with domain between two cells of the coarsest level.

In choosing the basis functions, it is convenient to separate the basis element belonging to the kernel of operator $\nabla_s \cdot$, the so-called [illegible] (see [22]). For this reason the spaces $W_{j,i}^T$ will be obtained by maintaining this separation; in other words,

$$W_{j,i}^T = W_{j,i}^{TE} \oplus W_{j,i}^{qTM} \tag{2.6}$$

with

$$(\nabla_s \cdot)\left(W_{j,i}^{TE}\right) = 0. \tag{2.7}$$

2.1. The solenoidal spaces $\boldsymbol{W}^{\boldsymbol{E}}$. We recall that when i is fixed, we are looking for a decomposition starting from a single triangular cell. The space spanned by RWG functions defined on the mesh obtained by bisecting j times the triangular cell i will be called $T_{j,i}^{\mathrm{RWG}}$, and clearly $T_{j,i}^{\mathrm{RWG}} \subset X_j^{\mathrm{RWG}}$. A basis for the kernel of $\nabla_s \cdot$ acting on $T_{j,i}^{\mathrm{RWG}}$ (with fixed j) can be found by considering the loop functions (as introduced in [22]):

$$\boldsymbol{l}_n^j(\boldsymbol{r}) = \nabla_s \times \hat{n}\Lambda_n^j(\boldsymbol{r}), \quad n = 1, \ldots, n_{nodes}, \tag{2.8}$$

where $\hat{n}$ is the direction normal to the antenna surface, Λ_n^j is the scalar linear Lagrange nodal interpolating function defined on inner node n (equal to one at the node and linearly going to zero on all neighboring nodes), and n_{nodes}^j is the number of inner nodes of the mesh j (refer to Figure 2.3).

Observe that the elements of the space $\nabla_s \cdot (T_{j,i}^{\mathrm{RWG}})$ are the functions piecewise constant on each cell at level j with zero mean value. Indeed, referring to notation used in Figure 2.1,

$$\nabla_s \cdot \boldsymbol{f}_p = \begin{cases} l/A_p^+ & \text{when } \boldsymbol{r} \in T_p^+, \\ -l/A_p^+ & \text{when } \boldsymbol{r} \in T_p^-, \\ 0 & \text{elsewhere.} \end{cases} \tag{2.9}$$

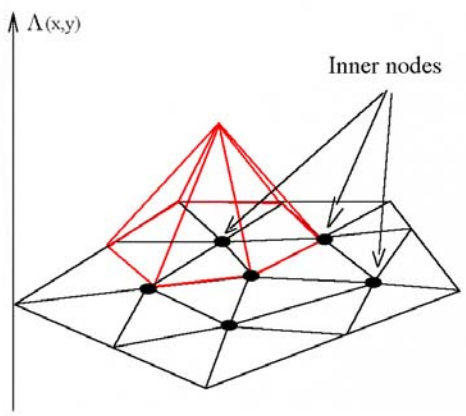


FIG. 2.3. Λ *function defined on an inner node.*

Moreover, the dimension of $\nabla_s \cdot (T_{j,i}^{\mathrm{RWG}})$ is equal to $n_{cell}^j - 1$, where n_{cell}^j is the number of cells at level j. In fact, from the Euler theorem

$$n_{cell}^j + n_{nodes}^j = n_{ed}^j + 1;$$

thus

$$\dim \ker(\nabla_s \cdot|_{T_{j,i}^{\mathrm{RWG}}}) = \dim T_j^{\mathrm{RWG}} - \dim \nabla_s \cdot \left(T_j^{\mathrm{RWG}}\right) = n_{ed}^j - n_{cell}^j + 1 = n_{nodes}^j.$$

Therefore, the functions $\boldsymbol{l}_n^j(\boldsymbol{r})$ belong to the kernel and are linearly independent.

Calling $V_{j,i}^{\Lambda}$ the space obtained by the span of Lagrange functions Λ_n^j on inner nodes

$$\ker(\nabla_s \cdot)|_{T_{j,i}^{\mathrm{RWG}}} = \nabla_s \times \hat{n}\left(V_{j,i}^{\Lambda}\right),$$

it is possible to obtain a wavelet construction for the spaces $V_{j,i}^{\Lambda}$ (refer, for example, to [26]). A basis of $V_{j+1,i}^{\Lambda}$ is obtained by adding to a basis of $V_{j,i}^{\Lambda}$ the functions Λ_n^{j+1}, where $n \in N_{j+1,i}^{new}$, the set of nodes of level $j+1$. In other words, we define

$$W_{j,i}^{\Lambda} = \mathrm{span}\left\{\Lambda_n^{j+1},\, n \in N_{j+1,i}^{new}\right\}$$

so that

$$V_{j+1,i}^{\Lambda} = V_{j,i}^{\Lambda} \oplus W_{j,i}^{\Lambda}.$$

According to the construction presented in section A.2 relative to the operator $\mathcal{A} = \nabla_s \times$, we obtain

$$W_{j,i}^{TE} = \mathrm{span}\left\{\boldsymbol{l}_n^{j+1},\, n \in N_{j+1,i}^{new}\right\} \tag{2.10}$$

and

$$\ker(\nabla_s \cdot|_{T_{j,i}^{\mathrm{RWG}}}) = \bigoplus_{y=0}^{j} W_{y,i}^{TE}. \tag{2.11}$$

2.2. The complementary spaces $\boldsymbol{W}_{\ }^{\ \boldsymbol{M}}$. We recall that $W_{j,i}^{qTM}$ is a space so that

$$T_{j+1,i}^{\mathrm{RWG}} = T_{j,i}^{\mathrm{RWG}} \oplus W_{j,i}^{TE} \oplus W_{j,i}^{qTM}. \tag{2.12}$$

Since we are working with finite dimension spaces, it is ensured that the decomposition

$$T_{j,i}^{\mathrm{RWG}} = \ker(\nabla_s \cdot|_{T_{j,i}^{\mathrm{RWG}}}) \oplus \ker(\nabla_s \cdot|_{T_{j,i}^{\mathrm{RWG}}})^c \tag{2.13}$$

holds. Taking into account the relationship (2.11), the equality (2.12) would be satisfied if

$$\ker(\nabla_s \cdot|_{T_{j,i}^{\mathrm{RWG}}})^c = \bigoplus_{y=0}^{j} W_{y,i}^{qTM};$$

in other words, it is sufficient to find a decomposition for $\ker(\nabla_s \cdot|_{T_{j,i}^{\mathrm{RWG}}})^c$ to satisfy (2.12). We will use again the approach explained in Appendix A.2 starting first from a scalar construction. The space $\nabla_s \cdot (T_{j,i}^{\mathrm{RWG}})$ considered before can be decomposed using a generalized Haar basis [26]. This is obtained, referring to Figure 2.4, by considering in each cell of level j the three functions

$$\psi_{1,k}^{j+1,i}(\boldsymbol{r}) = \begin{cases} 1, & \boldsymbol{r} \in T_4, \\ -1, & \boldsymbol{r} \in T_1, \\ 0 & \text{otherwise}, \end{cases} \tag{2.14}$$

$$\psi_{2,k}^{j+1,i}(\boldsymbol{r}) = \begin{cases} 1, & \boldsymbol{r} \in T_4, \\ -1, & \boldsymbol{r} \in T_2, \\ 0 & \text{otherwise}, \end{cases} \tag{2.15}$$

$$\psi_{3,k}^{j+1,i}(\boldsymbol{r}) = \begin{cases} 1, & \boldsymbol{r} \in T_4, \\ -1, & \boldsymbol{r} \in T_3, \\ 0 & \text{otherwise}. \end{cases} \tag{2.16}$$

In this way, letting $n_{cell}^{j,i}$ be the number of cells at level j obtained by bisecting j times edges of coarsest level cell i and defining the space

$$W_{j,i}^{H} = \operatorname{span}\left\{ \psi_{q,k}^{j+1,i},\ k = 1 \ldots, n_{cell}^{j,i},\ q = 1,2,3 \right\},$$

we have

$$\nabla_s \cdot \left(T_{j,i}^{\mathrm{RWG}}\right) = \bigoplus_{y=0}^{j} W_{y,i}^{H}. \tag{2.17}$$

We observe that the divergence operator can be inverted if restricted to the space $\ker(\nabla_s \cdot|_{T^{\mathrm{RWG}}})^c$ that excludes the div-free functions. Therefore, we consider the operator

$$\mathcal{A} = \left(\nabla_s \cdot|_{\ker(\nabla_s \cdot|_{T_{j,i}^{\mathrm{RWG}}})^c} \right)^{-1} \tag{2.18}$$

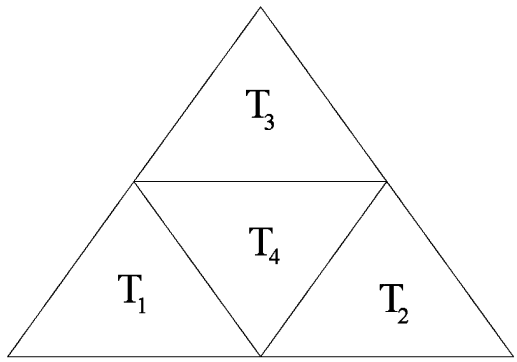


FIG. 2.4. *Cell conventions.*

and observe that

$$\mathcal{A}:\ \nabla_s\cdot\left(T_{j,i}^{\mathrm{RWG}}\right)\quad\mapsto\quad\ker(\nabla_s\cdot|_{T_{j,i}^{\mathrm{RWG}}})^c.$$

Using relationships (A.11), (2.17), and (2.18), we obtain

$$W_{j,i}^{qTM}=\mathrm{span}\left\{\mathcal{A}\left(\psi_{q,k}^{j+1,i}\right),\ k=1\ldots,n_{cell}^{j},\ q=1,2,3\right\}.$$

The interested reader can find a general approach to discretize the operator $\mathcal{A}$ in [22] (by setting to zero the solenoidal component of equation (22)); this allows one to find the RWG coefficients of image of any charge distribution, i.e., for any divergence of RWG functions. However, a generic implementation of $\mathcal{A}(\psi_{q,k}^{j+1,i})$ may result in spreading the support of the charge functions $\psi_{q,k}^{j+1,i}$ to significant portions of the structure, Γ. In this case, due to the simplicity of a single triangle domain (which is without loops), this is avoided by restricting the divergence operator to the complementary kernel with the mapping that leaves unchanged the function domains. In other words, the restriction is such that $\mathcal{A}\left(\psi_{q}^{k,j+1}\right)$ is simply the RWG with the same domain of the starting function $\psi_{q}^{k,j+1}$. A further modification of the mapping is necessary in order to obtain better moment properties for the basis: on each cell k of level j, we will consider the three functions

$$\begin{pmatrix}\boldsymbol{g}_{1,k}^{j+1,i}(\boldsymbol{r})\\ \boldsymbol{g}_{2,k}^{j+1,i}(\boldsymbol{r})\\ \boldsymbol{g}_{3,k}^{j+1,i}(\boldsymbol{r})\end{pmatrix}=\begin{pmatrix}-1&-1&1\\1&-1&-1\\1&1&1\end{pmatrix}\begin{pmatrix}\mathcal{A}\left(\psi_{1}^{k,j+1}\right)\\ \mathcal{A}\left(\psi_{2}^{k,j+1}\right)\\ \mathcal{A}\left(\psi_{3}^{k,j+1}\right)\end{pmatrix}$$

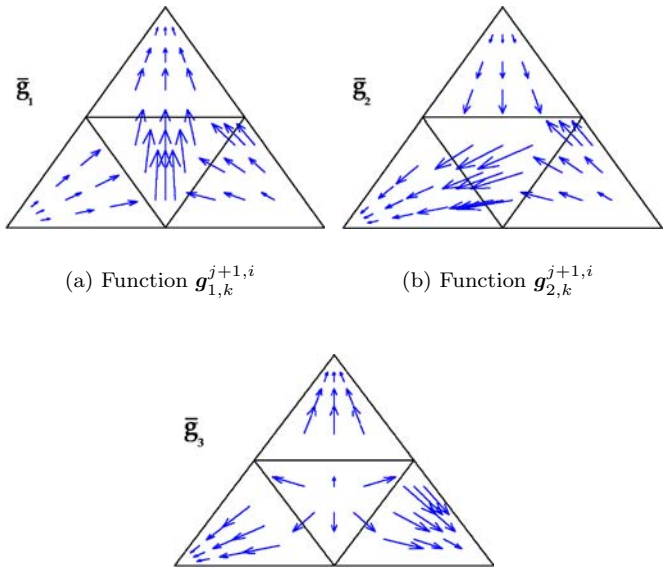
that are represented in Figures 2.5(a), 2.5(b), and 2.5(c).

Finally,

$$W_{j,i}^{qTM}=\mathrm{span}\left\{\boldsymbol{g}_{q}^{k,j+1},\ k=1\ldots,n_{cell}^{j},\ q=1,2,3\right\}\tag{2.19}$$

and

$$\ker(\nabla_s\cdot|_{T_{j,i}^{\mathrm{RWG}}})^c=\bigoplus_{y=0}^{j}W_{y,i}^{qTM}.\tag{2.20}$$



(a) Function $\boldsymbol{g}_{1,k}^{j+1,i}$ (b) Function $\boldsymbol{g}_{2,k}^{j+1,i}$

(c) Function $\boldsymbol{g}_{3,k}^{j+1,i}$

FIG. 2.5. *Basis functions for the* $W_{j,i}^{qTM}$ *space.*

2.3. The connection spaces W^B. Since spaces $T_{j,i}^{\mathrm{RWG}}$ are spanned by the RWG of level j completely included in the coarsest level cell i, it is clear that the spaces

$$T_j^{\mathrm{RWG}} = \bigoplus_{i=1}^{n_{cell}^0} T_{j,i}^{\mathrm{RWG}}$$

are spanned by RWG functions of level j completely included in one of the coarsest level cells. Moreover, from relationships (2.20), (2.11), (2.13), (2.6), (2.4), (2.5), and (2.3), we obtain

$$X_j^{\mathrm{RWG}} = T_j^{\mathrm{RWG}} \oplus X_0^{\mathrm{RWG}} \oplus \bigoplus_{y=0}^{j} W_y^B;$$

in other words, $X_0^{\mathrm{RWG}} \oplus \bigoplus_{y=0}^{j} W_y^B$ is the space of RWG of level j defined between two cells of coarsest level. Referring to the notation introduced in Appendix A.2, starting from X_0^{RWG} each RWG between two cells of a level j have to be substituted, at level $j+1$, by the two corresponding RWGs labelled I and II (refer to Figure B.1). For the relationship shown between the RWG function and the eight sub-RWGs in its domain, instead of taking the two $j+1$ level RWGs it is possible to maintain the RWG of level j and add a linear combination of the two $j+1$ level functions I and II. The most symmetric choice is to take the loop spanned by functions I, III, IV, II, VII, and VI. The linear independence of the chosen element is ensured by the fact that the two functions I and II have to be taken in the same direction to span the

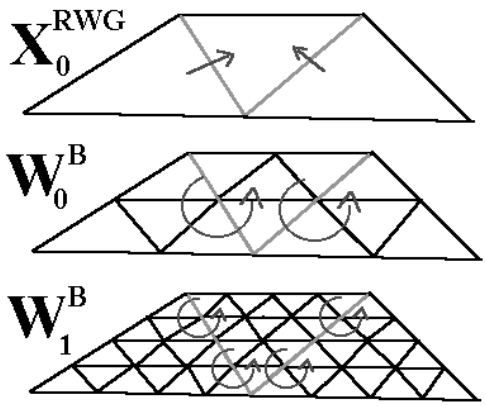


FIG. 2.6. *Spaces* W^B.

whole RWG and in opposite directions to span the loop. From these considerations it follows that

$$W_j^B = \text{span}\left\{\boldsymbol{l}_n^{j+1},\, n \in N_{j+1,edg}^{new}\right\}, \tag{2.21}$$

where $N_{j+1,edg}^{new}$ is the set of new nodes of level $j+1$. This is exemplified in Figure 2.6. With this last space, the decomposition of X_j^{RWG} is complete. We recall here that the completeness property of the decomposition is ensured, according to (2.3), (2.5), and (2.6), since we have proved the relationships (2.11), (2.20), and (2.21). Summarizing, the combination of the basis sets (2.10), (2.19) for each cell i and (2.21) represent an alternative basis for the space X_j^{RWG} that now will be used instead of RWG functions in (1.4), (1.5), and (1.6).

3. Numerical results. The surfaces Γ that will be considered represent some antenna structures that are actually usable in applications. The first structure is a wide band dipole known as a bowtie antenna. In Figure 3.1 the structure is presented with a mesh that is relative to the coarsest level $j = 0$. Two levels ($j = 0, 1, 2$) of the wavelets presented in this work have been used for this structure, so each edge of the coarsest level presented in Figure 3.1 has been bisected twice. The second structure is a sub-array of four printed patches (Figure 3.2, coarse level), a frequent configuration in the microstrip technology. Also for this structure the figure represents the coarsest level mesh, and two level of wavelets have been used. The last structure is a so-called log-periodic antenna (Figure 3.3, coarse level), which is a well-known wide band antenna; for this structure one level of wavelets ($j = 0, 1$) has been used. In the following, we will compare the performances of the MR basis with that of the conventional RWG basis. We remark here that the final spaces spanned by the RWG and by the wavelet basis are the same; in other words, the comparison is made by keeping constant the error on the discretization. The number of MR levels depends on the (average) cell dimension of the coarsest ($j = 0$) and finest meshes; the size of the coarsest mesh is essentially dictated by geometry, and the number of levels is then related to the desired level of RWG mesh fineness. It can be observed that the larger the number of levels, the better, but this obviously entails a larger number of unknowns. The numbers of levels adopted in the examples derive from a preliminary convergence study (with respect to finest mesh size) that is essentially the same as one

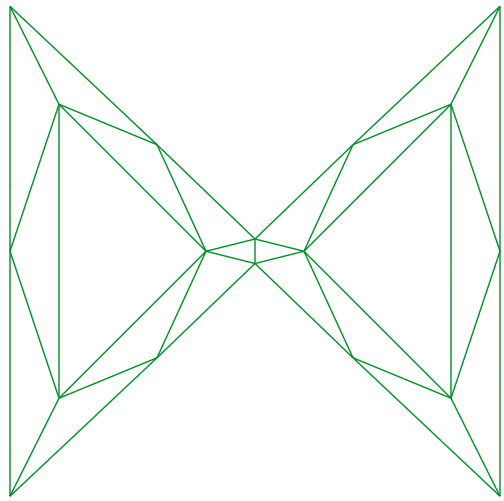


FIG. 3.1. Γ: *Bowtie.*

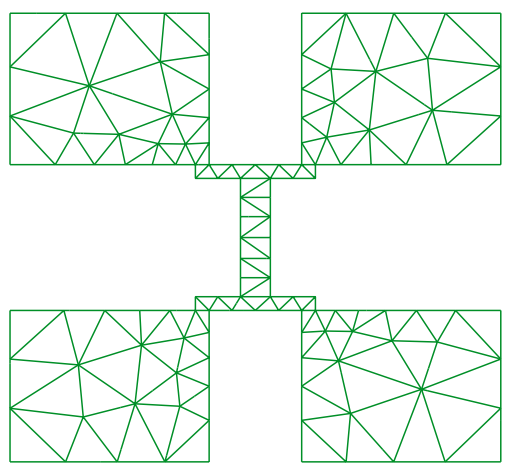


FIG. 3.2. Γ: *Patch array.*

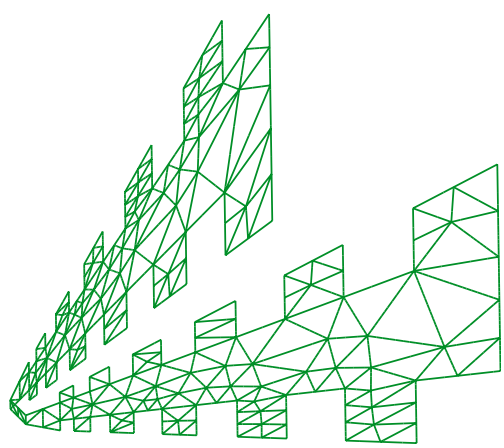


FIG. 3.3. Γ: *Log-periodic.*

would need to assess the smallest problem size versus accuracy using the usual RWG basis. On the other hand, the adopted numbers of levels appear sufficient to achieve the benefits of the multiscale nature of the basis, as will emerge from the results.

Prior to the solution, a [illegible] preconditioning is always effected, which is an integral constituent of the method (i.e., the basis allows a diagonal preconditioner to actually condition the system).

To solve the linear system (1.3) obtained by the numerical method, the modified Laczos conjugate gradient presented in [17] has been used. Before solving the system, the abovementioned diagonal preconditioning is used, and the functions of level $j = 0$ are made orthogonal using the singular value decomposition [25]. The performances of the new basis will be tested considering the number of iterations needed by the conjugate gradient (CG) to obtain a fixed relative residual on the solution; we have considered a rather restrictive threshold, 10^{-6}, to make sure that no convergence stagnation could be hidden by termination. The number of iterations of the modified conjugate gradient needed to solve the system obtained with the usual RWG basis is compared with the number of iterations needed with the basis presented in this work. This comparison is made for several frequencies $f = \frac{\omega}{2\pi}$, where ω is the constant which appears in (1.1). It is well known that the actual CG solution error is related to the residual and the matrix condition number. The results below, and those more extensively reported in [25], show that the condition number of the matrix obtained with MR is significantly lower than with RWG; therefore, comparing iterations counts for a fixed residual is indeed a conservative estimate of the MR performance with respect to RWG (the RWG solution error will be larger than with MR).

In Figure 3.4, we present the results for the bowtie antenna; the same analysis is performed for the other two structures, and the results are presented in Figures 3.5 and 3.6.

In order to put into better perspective the preconditioning performances of the proposed MR method, we have compared them with those achievable with the well-known incomplete LU (ILU) preconditioner. Other preconditioners have been developed for specific problems, including similar endeavors (e.g., [27]), but comparison with ILU can be considered a benchmarking. The test has been performed on the bowtie structure (Figure 3.7), and the parameter of the preconditioner at each frequency is set so that the preconditioner sparsity is always equal to $\frac{1}{3}$. This makes the

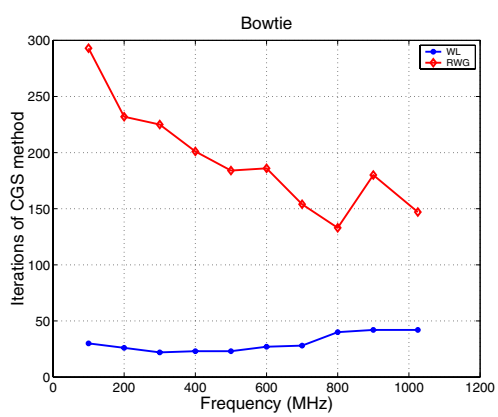


FIG. 3.4. *Bowtie: number of CG iterations of* RWG *basis versus wavelet basis.*

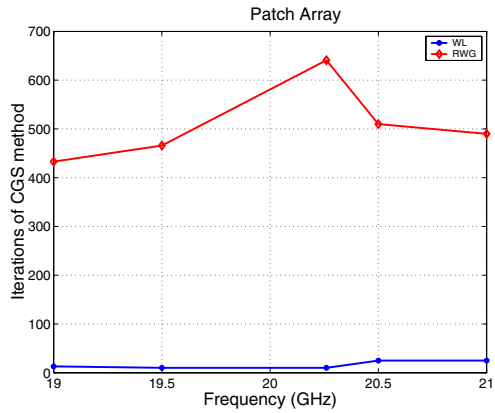


FIG. 3.5. *Patch array: number of CG iterations of* RWG *basis versus wavelet basis.*

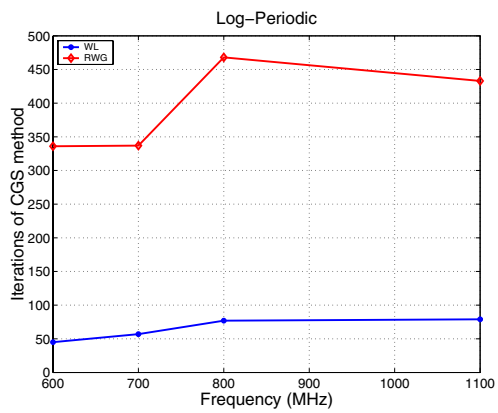


FIG. 3.6. *Log-periodic: number of CG iterations of* RWG *basis versus wavelet basis.*

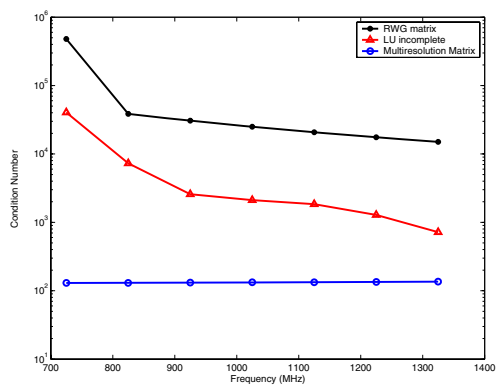


FIG. 3.7. *Condition number comparison for the* RWG *basis,* RWG *preconditioned with incomplete LU basis, and wavelet basis.*

cost of the LU preconditioning operation scale as $O(\frac{N}{3}^3)$, a choice that we consider impractical and clearly conservative. It can be observed that the MR always outper-

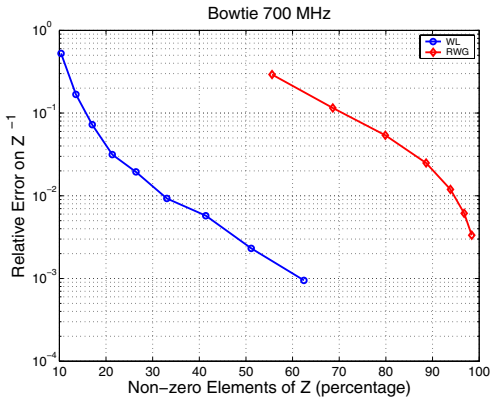


FIG. 3.8. *Bowtie:* RWG *basis versus wavelet basis sparsification properties.*

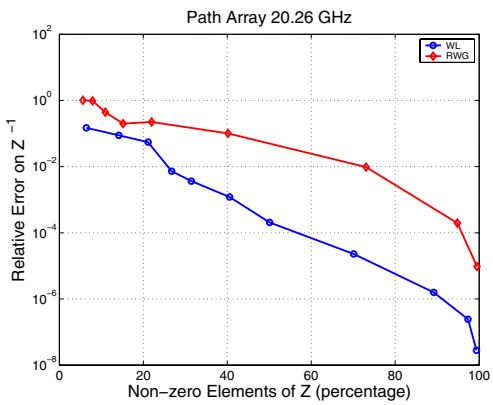


FIG. 3.9. *Patch array:* RWG *basis versus wavelet basis sparsification properties.*

forms the ILU in terms of condition number, even with the present parameter setting that makes the ILU preconditioning cost not comparable to that of MR.

As already stated in the introduction, a good condition number is not the only benefit that can be obtained using an MR basis; in fact, attention to conditioning properties is rather uncommon in wavelet applications in this field. The more usual achievement of a wavelet basis is that the matrix of the system can be sparsified with a low error on the solution by a thresholding operation, i.e., the setting to zero of all matrix entries below (in absolute value) a properly chosen threshold. We remark that the results discussed above refer to the convergence history of the matrix without any sparsification. Turning to sparsification, for the bowtie antenna this property is tested in Figure 3.8, where increasing sparsification percentages (number of zero elements over the total number of elements), due to the increasing threshold levels, are plotted versus the obtained relative error on the system solution. The test is performed with $f = 800$ MHz since this is a resonance frequency for the structure considered. With the same criterion, sparsification performances are tested for the other two structures in Figures 3.9 and 3.10, respectively. We finally observe that the condition number and iteration count do not change when the MR matrix is sparsified (not shown).

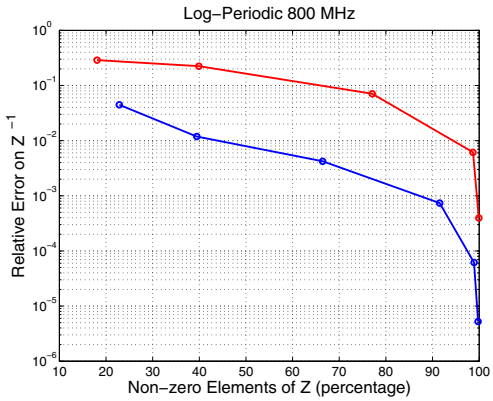


FIG. 3.10. *Log-periodic:* RWG *basis versus wavelet basis sparsification properties.*

Appendix A. Multilevel decompositions. We will recall how to define a multilevel decomposition of a separable Banach space V, with norm denoted by $\|\cdot\|$. For more details and proofs we refer to, among others, [3, 6].

A.1. Abstract setting. Let $\{V_j\}_{j\in\mathbb{N}}$ be a family of closed subspaces of V such that $V_j \subset V_{j+1}$ for all $j \in \mathbb{N}$. For all $j \in \mathbb{N}$, let $P_j : V \to V_j$ be a continuous linear operator satisfying the following properties:

$$\|P_j\|_{\mathcal{L}(V,V_j)} \le C \qquad (\text{independent of } \cdot), \tag{A.1}$$

$$P_j v = v \qquad \forall v \in V_j, \tag{A.2}$$

$$P_j \circ P_{j+1} = P_j. \tag{A.3}$$

Observe that (A.1) and (A.2) imply

$$\|v - P_j v\| \le C \inf_{u\in V_j} \|v - u\| \qquad \forall v \in V,$$

where C is a constant independent of j. Through P_j, we define another set of operators $Q_j : V \to V_{j+1}$ by

$$Q_j v = P_{j+1} v - P_j v \qquad \forall v \in V,\ \forall j \in \mathbb{N},$$

and the detail spaces

$$W_j := \operatorname{Im} Q_j \qquad \forall j \in \mathbb{N},$$

where $\operatorname{Im} Q_j$ is the range of Q_j. Thanks to the assumptions (A.1)–(A.3), every Q_j is a continuous linear projection on W_j, the sequence $\{Q_j\}_{j\in\mathbb{N}}$ is uniformly bounded in $\mathcal{L}(V, V_{j+1})$, and each space W_j is a complement space of V_j in V_{j+1}, i.e.,

$$V_{j+1} = V_j \oplus W_j. \tag{A.4}$$

By iterating the decomposition (A.4), we get for any integer j_n,

$$V_{j_n} = V_0 \oplus \bigoplus_{j=0}^{j_n-1} W_j, \tag{A.5}$$

so that every element in V_{j_n} can be viewed as a rough approximation of itself on a coarse level plus a sum of refinement details. For each $j \in \mathbb{N}$, let us fix a basis for the subspaces V_j,

$$\Phi_j = \{\varphi_{jk} \,:\, k \in \breve{\mathcal{K}}_j\}, \tag{A.6}$$

and for the subspaces W_j,

$$\Psi_j = \{\psi_{jk} \,:\, k \in \hat{\mathcal{K}}_j\}, \tag{A.7}$$

with $\breve{\mathcal{K}}_j$ and $\hat{\mathcal{K}}_j$ suitable sets of indices.

We can represent the operators P_j and Q_j in the form

$$P_j v = \sum_{k \in \breve{\mathcal{K}}_j} \breve{v}_{jk} \varphi_{jk}, \qquad Q_j v = \sum_{k \in \hat{\mathcal{K}}_j} \hat{v}_{jk} \psi_{jk} \quad \forall v \in V. \tag{A.8}$$

Functions φ_{jk} are often referred to as [illegible] while ψ_{jk} are referred to as [illegible].

A.2. Decomposition with a linear operator. We start with a wavelet decomposition of a space V as described earlier. We consider a linear, bounded, surjective operator

$$A : V \longmapsto X$$

mapping V onto a linear space X. Starting from a decomposition in V we will find, through A, a decomposition in X. Assume first that A is invertible. Defining the subspaces $X_j = A\,(V_j)$, we immediately obtain the inclusion relationship

$$X_{j+1} \subset X_j. \tag{A.9}$$

Moreover, defining

$$P_j^X = A P_j A^{-1}, \tag{A.10}$$

it is not difficult to show that P_j^X satisfy conditions (A.1)–(A.3).

Similarly, the operators

$$Q_j^X = A Q_j A^{-1} = P_{j+1}^X - P_{j+1}^X$$

allow us to define the subspaces W_j^X as

$$W_j^X = \operatorname{Im} Q_j.$$

Since A is invertible, it maps linearly independent vectors into linearly independent ones; in other words, starting from bases of V_j and W_j, we obtain bases of V_j^X and W_j^X through the relations

$$\varphi_{jk}^X = A\varphi_{jk} \quad \text{and} \qquad \psi_{jk}^X = A\psi_{jk}. \tag{A.11}$$

We can now extend the construction to a noninvertible operator A. We may assume a decomposition of V of the form

$$V = \ker A \oplus (\ker A)^c\,,$$

where $(\ker A)^c$ is a complementary space to the kernel. The existence of at least one of these spaces $(\ker A)^c$ is ensured when V is a Hilbert space; in fact, since A is bounded, the decomposition

$$V = \ker A \oplus (\ker A)^{\perp}$$

exists. Thus

$$\dim (\ker A)^c = \dim (\ker A)^{\perp} = \dim (\operatorname{Im} A)\,;$$

it follows that A restricted to $(\ker A)^c$ is invertible. In other words, we can apply the previous construction to the operator

$$A' = A|_{(\ker A)^c}.$$

Appendix B. Inclusion relationship between spaces $\boldsymbol{X^{\mathrm{RWG}}}$. We will make explicit the coefficients that allow us to express an RWG function of a given level j as a linear combination of RWG functions of level $j+1$. Referring to Figure B.1, the following conventions are used: RWG functions are identified by the edge on which they are defined, and the positive sign on a cell indicates that it is a T^+ cell (according to conventions used in Figure B.1) for each of the RWG functions on it and, similarly, the minus sign on a cell indicates that it is of the kind T^-. We shall use the following conventions. The Greek letter α will indicate coefficients of linear combinations, and their index will indicate a particular edge; similarly, the letter l corresponds to the length of an edge, $\mathcal{A}$ to the area of a cell. Vertices are labelled with capital letters and are thought of as vectors; an independent vector variable is denoted by $\boldsymbol{x}$. We will proceed considering a single cell at a time. We start from cell number 1. On this cell we have to enforce, according to (2.1),

$$\alpha_{\mathrm{I}}\, l_{\mathrm{I}}\, \frac{\boldsymbol{x}-D}{2\mathcal{A}_1} + \alpha_{\mathrm{III}} l_{\mathrm{III}} \frac{\boldsymbol{x}-A}{2\mathcal{A}_1} = l_{Tot} \frac{\boldsymbol{x}-F}{2\mathcal{A}^+_{Tot}}, \qquad \boldsymbol{x} \in 1,$$

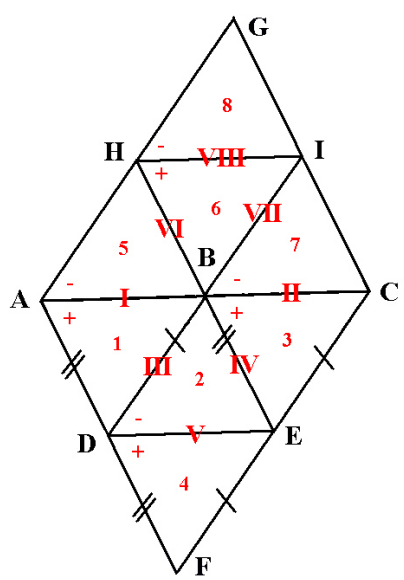


FIG. B.1. *Conventions used for functions.*

where $\mathcal{A}^+_{Tot} = \mathcal{A}_1 + \mathcal{A}_2 + \mathcal{A}_3 + \mathcal{A}_4$ and $l_{Tot} = l_{\rm I} + l_{\rm II}$. This is equivalent to the two conditions

$$\begin{cases} \alpha_{\rm I}\dfrac{l_{\rm I}}{\mathcal{A}_1} + \alpha_{\rm III}\dfrac{l_{\rm III}}{\mathcal{A}_1} &= \dfrac{l_{Tot}}{\mathcal{A}^+_{Tot}}, \\ \alpha_{\rm I}\dfrac{l_{\rm I}}{\mathcal{A}_1} D + \alpha_{\rm III}\dfrac{l_{\rm III}}{\mathcal{A}_1} A &= \dfrac{l_{Tot}}{\mathcal{A}^+_{Tot}} F; \end{cases} \tag{B.1}$$

thus

$$\alpha_{\rm I}\frac{l_{\rm I}}{\mathcal{A}_1}(D-A) = \frac{l_{Tot}}{\mathcal{A}^+_{Tot}}(F-A).$$

Since the two vectors $(A-D)$ and $(F-D)$ have the same direction (refer to Figure B.1), this is actually a scalar condition, obtained taking the vectors' magnitude

$$\alpha_{\rm I}\frac{l_{\rm I}}{\mathcal{A}_1}\,|A-D| = \frac{l_{Tot}}{\mathcal{A}^+_{Tot}}\,|A-F|\,. \tag{B.2}$$

Combining with B.1, we get

$$\alpha_{\rm I} = \frac{l_{Tot}}{l_{\rm I}}\frac{\mathcal{A}_1}{\mathcal{A}^+_{Tot}}\frac{|A-F|}{|A-D|}, \tag{B.3}$$

$$\alpha_{\rm III} = \frac{l_{Tot}}{l_{\rm III}}\frac{\mathcal{A}_1}{\mathcal{A}^+_{Tot}}\left(1-\frac{|A-F|}{|A-D|}\right). \tag{B.4}$$

For symmetry reasons,

$$\alpha_{\rm II} = \frac{l_{Tot}}{l_{\rm II}}\frac{\mathcal{A}_3}{\mathcal{A}^+_{Tot}}\frac{|C-F|}{|C-E|}, \tag{B.5}$$

$$\alpha_{\rm IV} = \frac{l_{Tot}}{l_{\rm IV}}\frac{\mathcal{A}_3}{\mathcal{A}^+_{Tot}}\left(1-\frac{|C-F|}{|C-E|}\right). \tag{B.6}$$

Consider now cell number 4; we have

$$\alpha_{\rm V}(\boldsymbol{x}-F)\frac{l_{\rm V}}{2\mathcal{A}_4} = (\boldsymbol{x}-F)\frac{l_{Tot}}{2\mathcal{A}_{Tot}}, \qquad \boldsymbol{x}\in 4,$$

so that

$$\alpha_{\rm V} = \frac{\mathcal{A}_4}{\mathcal{A}^+_{Tot}}\frac{l_{Tot}}{l_V}. \tag{B.7}$$

Now we have to verify that everything is coherent also on cell number 2. We get

$$\alpha_{\rm III}\frac{l_{\rm III}}{\mathcal{A}_2}(E-\boldsymbol{x}) + \alpha_{\rm IV}\frac{l_{\rm IV}}{\mathcal{A}_2}(D-\boldsymbol{x}) + \alpha_{\rm V}\frac{l_{\rm V}}{\mathcal{A}_2}(B-\boldsymbol{x}) = \frac{l_{Tot}}{\mathcal{A}_{Tot}}(\boldsymbol{x}-F),$$

which is equivalent to

$$\begin{cases} \alpha_{\rm III}\dfrac{l_{\rm III}}{\mathcal{A}_2} + \alpha_{\rm IV}\dfrac{l_{\rm IV}}{\mathcal{A}_2} + \alpha_{\rm V}\dfrac{l_{\rm V}}{\mathcal{A}_2} &= -\dfrac{l_{Tot}}{\mathcal{A}_{Tot}}, \\ \alpha_{\rm III}\dfrac{l_{\rm III}}{\mathcal{A}_2} E + \alpha_{\rm IV}\dfrac{l_{\rm IV}}{\mathcal{A}_2} D + \alpha_{\rm V}\dfrac{l_{\rm V}}{\mathcal{A}_2} B &= -\dfrac{l_{Tot}}{\mathcal{A}^+_{Tot}} F. \end{cases} \tag{B.8}$$

Substituting (B.4), (B.6), and (B.7) in (B.8), we get from the first equation the constraint

$$\mathcal{A}_{Tot} = \mathcal{A}_1 \frac{|A-F|}{|A-D|} + \mathcal{A}_3 \frac{|C-F|}{|C-E|}. \tag{B.9}$$

This relation is verified when the triangle edges are bisected, as in our case. In fact, $\frac{|A-F|}{|A-D|} = 2$ and $\mathcal{A}_{Tot} = 2\mathcal{A}_1 + 2\mathcal{A}_3$ holds since $\mathcal{A}_1$ and $\mathcal{A}_3$ are both equal to $\mathcal{A}_{Tot}/4$. From the second equation we get

$$\mathcal{A}_1 \left(1 - \frac{|A-F|}{|A-D|}\right) E + \mathcal{A}_3 \left(1 - \frac{|C-F|}{|C-E|}\right) D + \mathcal{A}_4 B = -F \mathcal{A}_2. \tag{B.10}$$

Again this holds in our case since the $\mathcal{A}_i$ are all equal and $\frac{|A-F|}{|A-D|} = \frac{|C-F|}{|C-E|} = 2$, so that

$$B - D = E - F.$$

In regard to the remaining coefficients, they are easily obtained by symmetry considerations in the following way:

$$\alpha_{\mathrm{I}}^{-} = \frac{l_{Tot}}{l_{\mathrm{I}}} \frac{\mathcal{A}_5}{\mathcal{A}_{Tot}^{-}} \frac{|A-G|}{|A-H|}, \tag{B.11}$$

$$\alpha_{\mathrm{II}}^{-} = \frac{l_{Tot}}{l_{\mathrm{II}}} \frac{\mathcal{A}_7}{\mathcal{A}_{Tot}^{-}} \frac{|C-G|}{|C-I|}, \tag{B.12}$$

$$\alpha_{\mathrm{VI}} = \frac{l_{Tot}}{l_{\mathrm{VI}}} \frac{\mathcal{A}_5}{\mathcal{A}_{Tot}^{-}} \left(1 - \frac{|A-G|}{|A-H|}\right), \tag{B.13}$$

$$\alpha_{\mathrm{VII}} = \frac{l_{Tot}}{l_{\mathrm{VII}}} \frac{\mathcal{A}_7}{\mathcal{A}_{Tot}^{-}} \left(1 - \frac{|C-G|}{|C-I|}\right), \tag{B.14}$$

$$\alpha_{\mathrm{VIII}} = \frac{\mathcal{A}_8}{\mathcal{A}_{Tot}^{-}} \frac{l_{Tot}}{l_{\mathrm{VIII}}}. \tag{B.15}$$

Moreover, $\alpha_{\mathrm{I}}^{-} = \alpha_{\mathrm{I}}$ and $\alpha_{\mathrm{II}}^{-} = \alpha_{\mathrm{II}}$ since the sides are bisected.

Acknowledgments. The authors would like to thank C. Canuto for useful comments at different stages of the work, and F. Vipiana and P. Pirinoli, who contributed with the background finite-element code and the first MR codes.

REFERENCES

[1] K. Bubke, G. Oberschmidt, and A. Jacob, *Investigation of different non-uniform grids in wavelet based planar circuit analysis*, in Proceedings of the Third International Kharkov Symposium on Physics and Engineering of Millimeter and Submillimeter Waves—MSMW, Vol. 1, 1998, pp. 329–331.

[2] W. C. Chew, Z. Nie, Q. Liu, and Y. Lo, *Analysis of a probe-fed microstrip disk antenna*, IEE Proceedings-H, 138 (1991), pp. 185–191.

[3] A. Cohen, *Numerical Analysis of Wavelet Methods*, Stud. Math. Appl. 32, North–Holland, Amsterdam, 2003.

[4] D. Colton and R. Kress, *Integral Equation Methods in Scattering Theory*, Wiley, New York, 1983.

[5] T. J. Cui and W. C. Chew, *Accurate analysis of wire structures from very-low frequency to microwave frequency*, IEEE Trans. Antennas and Propagation, 50 (2002), pp. 301–307.

[6] I. Daubechies, *Ten Lectures on Wavelets*, CBMS-NSF Regional Conf. Ser. in Appl. Math. 61, SIAM, Philadelphia, 1992.
[7] J. Goswami, A. Chan, and C. Chui, *On solving first-kind integral equations using wavelets on a bounded interval*, IEEE Trans. Antennas and Propagation, 43 (1995), pp. 614–622.
[8] G. C. Hsiao and R. E. Kleinman, *Mathematical foundation for error estimation in numerical solution of integral equations in electromagnetics*, IEEE Trans. Antennas and Propagation, 45 (1997), pp. 316–328.
[9] R. Loison, R. Gillard, J. Citerne, G. Piton, and H. Legay, *Multiresolution MoM analysis of multiport structures using matched terminations*, IEEE Trans. Microwave Theory Tech., 49 (2001), pp. 119–127.
[10] R. Maggiora, G. Vecchi, V. Lancellotti, and V. Kyrytsya, *Efficient $3D/1D$ self-consistent integral-equation analysis of ICRH antennae*, Nucl. Fusion, 44 (2004), pp. 846–868.
[11] J. C. Nedelec, *A new family of mixed finite element*, Numer. Math., 50 (1986), pp. 57–81.
[12] G. Pan, M. Toupikov, and B. Gilbert, *On the use of Coifman intervallic wavelets in the method of moments for fast construction of wavelet sparsified matrices*, IEEE Trans. Antennas and Propagation, 47 (1999), pp. 1189–1200.
[13] A. F. Peterson, S. L. Ray, and R. Mittra, *Computational Methods for Electromagnetics*, IEEE Press, New York, Oxford University Press, Oxford, UK, 1998.
[14] P. Pirinoli, G. Vecchi, and L. Matekovits, *Multiresolution analysis of printed antennas and circuits: A dual-scalar approach*, IEEE Trans. Antennas and Propagation, 49 (2001), pp. 858–874.
[15] S. M. Rao, D. R. Wilton, and A. W. Glisson, *Electromagnetic scattering by surfaces of arbitrary shape*, IEEE Trans. Antennas and Propagation, AP-30 (1982), pp. 409–418.
[16] G. Schneider and A. Jacob, *Adapted grids for the wavelet-based simulation of complex planar circuits*, in International Conference on Mathematical Methods in Electromagnetic Theory—MMET, Vol. 1, 2002, pp. 163–168.
[17] P. Sonneveld, *CGS: A fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 36–52.
[18] B. Z. Steinberg and Y. Leviatan, *On the use of wavelet expansions in the method of moments*, IEEE Trans. Antennas and Propagation, 41 (1993), pp. 610–619.
[19] C. Su and T. Sarkar, *Adaptive multiscale moment method (AMMM) for analysis of scattering from perfectly conducting plates*, IEEE Trans. Antennas and Propagation, 48 (2000), pp. 932–939.
[20] C. Su and T. K. Sarkar, *Adaptive multiscale moment method (AMMM) for analysis of scattering from three-dimensional perfectly conducting structures*, IEEE Trans. Antennas and Propagation, 50 (2002), pp. 444–450.
[21] W. Sweldens, *The lifting scheme: A construction of second generation wavelets*, SIAM J. Math. Anal., 29 (1998), pp. 511–546.
[22] G. Vecchi, *Loop star decomposition of basis functions in the discretization of the EFIE*, IEEE Trans. Antennas and Propagation, 47 (1999), pp. 339–346.
[23] G. Vecchi, L. Matekovits, P. Pirinoli, and M. Orefice, *A numerical regularization of the EFIE for three-dimensional planar structures in layered media*, Int. J. Microwave Millimeter-Wave CAE, 7 (1997), pp. 410–431.
[24] F. Vipiana, *A Multi-Resolution Approach to the Analysis and Synthesis of Antennas*, Ph.D. dissertation, Politecnico di Torino, Torino, Italy, 2004.
[25] F. Vipiana, P.Pirinoli, and G.Vecchi, *A multi-resolution method of moments for triangular meshes*, IEEE Trans. Antennas and Propagation, 53 (2005), pp. 2247–2258.
[26] T. von Petersdorff and C. Schwab, *Fully discrete multiscale Galerkin BEM*, in Multiscale Wavelet Methods for Partial Differential Equations, Academic Press, San Diego, 1997.
[27] Y. Xie, J. He, A. Sullivan, and L. Carin, *A simple preconditioner for electric-field integral equations*, Microwave and Opt. Tech. Letts., 30 (2001), pp. 51–54.

SIAM J. SCI. COMPUT.
Vol. 29, No. 1, pp. 22–48
© 2007 Society for Industrial and Applied Mathematics

NUMERICAL SENSITIVITY ANALYSIS FOR THE QUANTITY OF INTEREST IN PDE-CONSTRAINED OPTIMIZATION*

ROLAND GRIESSE† AND BORIS VEXLER†

Abstract. In this paper, we consider the efficient computation of derivatives of a functional (the quantity of interest) which depends on the solution of a PDE-constrained optimization problem with inequality constraints and which may be different from the cost functional. The optimization problem is subject to perturbations in the data. We derive conditions under with the quantity of interest possesses first and second order derivatives with respect to the perturbation parameters. An algorithm for the efficient evaluation of these derivatives is developed, with considerable savings over a direct approach, especially in the case of high-dimensional parameter spaces. The computational cost is shown to be small compared to that of the overall optimization algorithm. Numerical experiments involving a parameter identification problem for Navier–Stokes flow and an optimal control problem for a reaction-diffusion system are presented which demonstrate the efficiency of the method.

Key words. sensitivity analysis, PDE-constrained optimization, quantity of interest

AMS subject classifications. 49J20, 49K40, 90C31

DOI. 10.1137/050637273

1. Introduction.

In this paper we consider PDE-constrained optimization problems with inequality constraints. The optimization problems are formulated in a general setting including optimal control as well as parameter identification problems. The problems are subject to perturbation in the data. We suppose we are given a quantity of interest (output functional), which depends on both the state and the control variables and which may be different from the cost functional used during the optimization.

The quantity of interest is shown to possess first and, under tighter assumptions, second order derivatives with respect to the perturbation parameters. In the presence of control constraints, strict complementarity and compactness of certain derivatives of the state equation are assumed; for second order derivatives, stability of the active set is required in addition. The precise conditions are given in section 3. The main contribution of this paper is to devise an efficient algorithm to evaluate these sensitivity derivatives which offers considerable savings over a direct approach, especially in the case of high-dimensional parameter spaces. We show that the derivatives of the quantity of interest can be computed with little additional numerical effort in comparison to the corresponding derivatives of the cost functional. Moreover, the computational cost for the evaluation of the gradient of the quantity of interest is independent of the dimension of the parameter space and low compared to that of the overall optimization algorithm. The cost to evaluate the Hessian grows linearly with the dimension of the parameter space. We refer to Table 3.1 for details.

The parametric derivatives of the quantity of interest offer a significant amount of additional information on top of an optimal solution. The derivative information can be used to assess the stability of an optimal solution, or to compute a Taylor

*Received by the editors August 1, 2005; accepted for publication (in revised form) June 27, 2006; published electronically January 8, 2007.
http://www.siam.org/journals/sisc/29-1/63727.html
†Johann Radon Institute for Computational and Applied Mathematics (RICAM), Austrian Academy of Sciences, Linz, Austria (roland.griesse@oeaw.ac.at, boris.vexler@oeaw.ac.at).

expansion which allows the fast prediction of the perturbed value of the quantity of interest in a neighborhood of a reference parameter.

We note that a quantity of interest different from the cost functional is often natural. For instance, an optimization problem in fluid flow may aim at minimizing the drag of a given body, e.g., by adjusting the boundary conditions. The quantity of interest, however, may be the lift coefficient of the optimal configuration. We also mention the applicability of our results to bilevel optimization problems where the outer variable is the "perturbation" parameter and the outer objective is the output functional, whose derivatives are needed to employ efficient optimization algorithms.

The necessity to compute higher order derivatives may impose possible limitations to the applicability of the methods presented in this paper. Second order derivatives of the cost functional and the PDE constraint are required to evaluate the gradient of the quantity of interest, and third order derivatives are required to evaluate the Hessian.

Let us put our work into perspective. The existence of first and second order sensitivity derivatives of the [illegible] (cost functional) in optimal control of PDEs with control constraints has been proved in [7, 18]. Moreover, [8] addresses the numerical computation of these derivatives. Recently, the computation of the gradient of the quantity of interest in the [illegible] has been discussed in [3].

Problem setting. We consider the PDE-constrained optimization problem in the following abstract form: The state variable u in an appropriate Hilbert space $\mathcal{V}$ with scalar product $(\cdot,\cdot)_{\mathcal{V}}$ is determined by a PDE ([illegible]) in weak form:

$$a(u,q,p)(\phi) = f(\phi) \quad \forall \phi \in \mathcal{V}, \tag{1.1}$$

where q denotes the control, or, more generally, design variable, in the Hilbert space $\mathcal{Q} = L^2(\omega)$ with the standard scalar product $(\cdot,\cdot)$. Typically, ω is a subset of the computational domain Ω or a subset of its boundary $\partial\Omega$. In case of finite-dimensional controls we set $\mathcal{Q} = \mathbb{R}^n$ and identify this space with $L^2(\omega)$, where $\omega = \{1, 2, \dots, n\}$ to keep the notation consistent. The parameter p from a normed linear space $\mathcal{P}$ describes the perturbations of the data.

For fixed $p \in \mathcal{P}$, the semilinear form $a(\cdot,\cdot,p)(\cdot)$ is defined on the Hilbert space $\mathcal{V} \times \mathcal{Q} \times \mathcal{V}$. Semilinear forms are written with two parentheses; the first one refers to the nonlinear arguments, whereas the second one embraces all linear arguments. The partial derivatives of the semilinear form $a(\cdot,\cdot,p)(\cdot)$ are denoted by $a'_u(\cdot,\cdot,p)(\cdot,\cdot)$, $a'_q(\cdot,\cdot,p)(\cdot,\cdot)$, etc. The linear functional $f \in \mathcal{V}'$ represents the right-hand side of the state equation, where $\mathcal{V}'$ denotes the dual space of $\mathcal{V}$. For the cost functional (objective functional) we assume the form

$$J(u,p) + \frac{\alpha}{2}\|q - \overline{q}\|^2_{\mathcal{Q}}, \tag{1.2}$$

which is typical in PDE-constrained optimization problems. Here, $\alpha > 0$ is a regularization parameter and $\overline{q} \in \mathcal{Q}$ is a reference control. The functional $J : \mathcal{V} \times \mathcal{P} \to \mathbb{R}$ is also subject to perturbation. It is possible to extend our analysis to more general cost functionals than (1.2). In particular, only notational changes are necessary if J contains linear terms in q, and if α and $\overline{q}$ also depend on the perturbation parameter. However, full generality of the cost functional comes at the expense of additional assumptions which would unnecessarily complicate the discussion.

In order to cover additional control constraints we introduce a nonempty closed convex subset $\mathcal{Q}_{ad} \subset \mathcal{Q}$ by

$$\mathcal{Q}_{ad} = \{q \in \mathcal{Q} \mid b_-(x) \le q(x) \le b_+(x) \text{ a.e. on } \omega\},$$

with bounds $b_- \le b_+ \in \mathcal{Q}$. In the case of finite-dimensional controls the inequality $b_- \le q \le b_+$ is meant to hold componentwise.

The problem under consideration is to

$$(\mathbf{OP}(p)) \qquad \begin{aligned} &\text{minimize (1.2) over } \mathcal{Q}_{ad} \times \mathcal{V} \\ &\text{subject to the state equation (1.1)} \end{aligned}$$

for fixed $p \in \mathcal{P}$. We assume that in a neighborhood of a reference parameter p_0, there exist functions $u = U(p)$ and $q = Q(p)$, which map the perturbation parameter p to a local solution (u, q) of the problem $(\mathbf{OP}(p))$. In section 3, we give sufficient conditions ensuring the existence and differentiability of these functions. Our results complement previous findings in [7, 10, 18].

The quantity of interest is denoted by a functional

$$I : \mathcal{V} \times \mathcal{Q} \times \mathcal{P} \to \mathbb{R}. \tag{1.3}$$

This gives rise to the definition of the reduced quantity of interest $i : \mathcal{P} \to \mathbb{R}$,

$$i(p) = I(U(p), Q(p), p). \tag{1.4}$$

Likewise, we denote by $j : \mathcal{P} \to \mathbb{R}$ the reduced cost functional

$$j(p) = J(U(p), p) + \frac{\alpha}{2}\|Q(p) - \overline{q}\|^2_{\mathcal{Q}}. \tag{1.5}$$

As stated above, the main contribution of this paper is to devise an efficient algorithm to evaluate the first and second derivatives of the reduced quantity of interest $i(p)$.

The outline of the paper is as follows: In the next section we specify the first order necessary optimality conditions for the problem under consideration. We recall a primal-dual active set method for its solution. The core step of this method is described in some detail since it is also used for the problems arising during the sensitivity computation. In section 3 we use duality arguments for the efficient evaluation of the first and second order sensitivities of the quantity of interest with respect to perturbation parameters. Throughout, we compare the standard sensitivity analysis for the reduced cost functional $j(p)$ with our analysis for the reduced quantity of interest $i(p)$. In the last section we discuss two numerical examples illustrating our approach. The first example deals with a parameter identification problem for a channel flow described by the incompressible Navier–Stokes equations. In the second example we consider the optimal control of time-dependent three-species reaction-diffusion equations under control constraints.

2. Optimization algorithm. In this section we recall the first order necessary conditions for the problem $(\mathbf{OP}(p))$ and describe the optimization algorithm with active set strategy which we use in our numerical examples. In particular, we specify the Newton step taking into account the active sets since the sensitivity problems arising in section 3 are solved by the same technique.

Throughout the paper we make the following assumption.

ASSUMPTION 2.1.

1. $a(\cdot,\cdot,\cdot)(\cdot)$ [illegible] (u,q,p)
2. $J(\cdot,\cdot)$ [illegible] (u,p)
3. $I(\cdot,\cdot,\cdot)$ [illegible] (u,q,p)

In order to establish the optimality system, we introduce the Lagrangian $\mathcal{L} : \mathcal{V} \times \mathcal{Q} \times \mathcal{V} \times \mathcal{P} \to \mathbb{R}$ as follows:

$$\mathcal{L}(u,q,z,p) = J(u,p) + \frac{\alpha}{2}\|q-\overline{q}\|^2_{\mathcal{Q}} + f(z) - a(u,q,p)(z), \tag{2.1}$$

where $z \in \mathcal{V}$ denotes the adjoint state. The first order necessary conditions for the problem $(\mathbf{OP}(p))$ read

$$\mathcal{L}'_u(u,q,z,p)(\delta u) = 0 \quad \forall \delta u \in \mathcal{V}, \tag{2.2}$$

$$\mathcal{L}'_q(u,q,z,p)(\delta q - q) \geq 0 \quad \forall \delta q \in \mathcal{Q}_{ad}, \tag{2.3}$$

$$\mathcal{L}'_z(u,q,z,p)(\delta z) = 0 \quad \forall \delta z \in \mathcal{V}. \tag{2.4}$$

They can be explicitly rewritten as follows:

$$J'_u(u,p)(\delta u) - a'_u(u,q,p)(\delta u, z) = 0 \quad \forall \delta u \in \mathcal{V}, \tag{2.5}$$

$$\alpha(q-\overline{q}, \delta q - q) - a'_q(u,q,p)(\delta q - q, z) \geq 0 \quad \forall \delta q \in \mathcal{Q}_{ad}, \tag{2.6}$$

$$f(\delta z) - a(u,q,p)(\delta z) = 0 \quad \forall \delta z \in \mathcal{V}. \tag{2.7}$$

For given u, q, z, p, we introduce an additional Lagrange multiplier $\mu \in L^2(\omega)$ by the following identification:

$$\begin{aligned}(\mu, \delta q) &:= -\mathcal{L}'_q(u,q,z,p)(\delta q) \\ &= -\alpha(q-\overline{q}, \delta q) + a'_q(u,q,p)(\delta q, z) \quad \forall \delta q \in L^2(\omega).\end{aligned}$$

The variational inequality (2.6) is known to be equivalent to the following pointwise conditions a.e. on ω:

$$q(x) = b_-(x) \Rightarrow \mu \leq 0, \tag{2.8}$$

$$q(x) = b_+(x) \Rightarrow \mu \geq 0, \tag{2.9}$$

$$b_-(x) < q(x) < b_+(x) \Rightarrow \mu = 0. \tag{2.10}$$

In addition to the necessary conditions above, in the following lemma we recall second order sufficient optimality conditions.

LEMMA 2.2 (sufficient optimality conditions). [illegible] $x = (u,q,z)$ [illegible] (2.2)–(2.4) [illegible] $(\mathbf{OP}(p))$ [illegible] $a'_u(u,q,p) \colon \mathcal{V} \to \mathcal{V}'$ [illegible] $\rho > 0$ [illegible]

$$\begin{pmatrix}\delta u & \delta q\end{pmatrix} \begin{bmatrix}\mathcal{L}''_{uu}(x,p) & \mathcal{L}''_{uq}(x,p) \\ \mathcal{L}''_{qu}(x,p) & \mathcal{L}''_{qq}(x,p)\end{bmatrix} \begin{pmatrix}\delta u \\ \delta q\end{pmatrix} \geq \rho \left(\|\delta u\|^2_{\mathcal{V}} + \|\delta q\|^2_{\mathcal{Q}}\right)$$

[illegible] $(\delta u, \delta q)$ [illegible]

$$a'_u(u,q,p)(\delta u, \varphi) + a'_q(u,q,p)(\delta q, \varphi) = 0 \quad \forall \varphi \in \mathcal{V},$$

[illegible] (u,q) [illegible] $(\mathbf{OP}(p))$

For the proof we refer the reader to [19].

For the solution of the first order necessary conditions (2.5)–(2.7) for fixed $p \in \mathcal{P}$, we employ a nonlinear primal-dual active set strategy; see [4, 12, 15, 16]. In the following we sketch the corresponding algorithm on the continuous level:

Nonlinear primal-dual active set strategy

1. Choose initial guess u^0, q^0, z^0, μ^0 and $c > 0$ and set $n = 1$
2. While not converged
3. Determine the active sets $\mathcal{A}_+^n$ and $\mathcal{A}_-^n$

$$\mathcal{A}_-^n = \{x \in \omega \mid q^{n-1} + \mu^{n-1}/c - b_- \leq 0\},$$
$$\mathcal{A}_+^n = \{x \in \omega \mid q^{n-1} + \mu^{n-1}/c - b_+ \geq 0\}$$

4. Solve the equality-constrained optimization problem

$$\text{Minimize} \quad J(u^n, p) + \frac{\alpha}{2}\|q^n - \overline{q}\|_{\mathcal{Q}}^2 \text{ over } \mathcal{V} \times \mathcal{Q}$$

subject to (1.1) and to

$$q^n(x) = b_-(x) \text{ on } \mathcal{A}_-^n, \qquad q^n(x) = b_+(x) \text{ on } \mathcal{A}_+^n$$

with adjoint variable z^n

5. Set $\mu^n = -\alpha(q^n - \overline{q}) + a_q'(u^n, q^n, p)(\cdot, z^n)$
6. Set $n = n + 1$ and go to 2.

. . .. 2.3.

1. The initial guess for the Lagrange multiplier μ^0 can be taken according to step 5. Another possibility is choosing $\mu^0 = 0$ and $q^0 \in \mathcal{Q}_{ad}$, which leads to solving the optimization problem (step 4) without control constraints in the first iteration.
2. The convergence in step 2 can be determined conveniently from agreement of the active sets in two consecutive iterations. The problem is then solved only on the inactive set $\mathcal{I}^n = \omega \setminus (A_-^n \cup A_+^n)$, e.g., by applying Newton's method to the corresponding first order necessary conditions.

Later on, the above algorithm is applied on the discrete level. The concrete discretization schemes are described in section 4 for each individual example.

Clearly, the main step in the primal-dual algorithm is the solution of the equality-constrained nonlinear optimization problem in step 4. We shall describe the Lagrange Newton SQP method for its solution in some detail since exactly the same procedure may be used to solve the sensitivity problems in section 3, which are the main focus of our paper.

For given active and inactive sets $\mathcal{A} = \mathcal{A}_+ \cup \mathcal{A}_-$ and $\mathcal{I} = \omega \setminus \mathcal{A}$, let us define the

"restriction" operator $R_{\mathcal{I}} : L^2(\omega) \to L^2(\omega)$ by

$$R_{\mathcal{I}}(q) = q \cdot \chi_{\mathcal{I}},$$

where $\chi_{\mathcal{I}}$ is a characteristic function of the set $\mathcal{I}$. Similarly, the operators $R_{\mathcal{A}}$, $R_{\mathcal{A}_+}$, and $R_{\mathcal{A}_-}$ are defined. Note that $R_{\mathcal{I}}$, etc., are obviously self-adjoint.

The first order necessary conditions for the purely equality-constrained problem in step 4 are (compare (2.2)–(2.4), respectively, (2.5)–(2.7)):

$$\mathcal{L}'_u(u,q,z,p)(\delta u) = 0 \quad \forall \delta u \in \mathcal{V}, \tag{2.11}$$

$$\mathcal{L}'_q(u,q,z,p)(\delta q) = 0 \quad \forall \delta q \in L^2(\mathcal{I}^n), \tag{2.12}$$

$$q - b_- = 0 \quad \text{on } A^n_-, \tag{2.13}$$

$$q - b_+ = 0 \quad \text{on } A^n_+, \tag{2.14}$$

$$\mathcal{L}'_z(u,q,z,p)(\delta z) = 0 \quad \forall \delta z \in \mathcal{V}, \tag{2.15}$$

with the inactive set $\mathcal{I}^n = \omega \backslash (\mathcal{A}^n_- \cup \mathcal{A}^n_+)$. Using the restriction operators, (2.12)–(2.14) can be reformulated as

$$\mathcal{L}'_q(u,q,z,p)(R_{\mathcal{I}^n}\delta q) + (q - b_-, R_{\mathcal{A}^n_-}\delta q) + (q - b_+, R_{\mathcal{A}^n_+}\delta q) = 0 \quad \forall \delta q \in \mathcal{Q}.$$

The Lagrange Newton SQP method is defined as Newton's method, applied to (2.11)–(2.15). To this end, we define B as the Hessian operator of the Lagrangian $\mathcal{L}$, i.e.,

$$B(x,p) = \begin{bmatrix} \mathcal{L}''_{uu}(x,p)(\cdot,\cdot) & \mathcal{L}''_{uq}(x,p)(\cdot,\cdot) & \mathcal{L}''_{uz}(x,p)(\cdot,\cdot) \\ \mathcal{L}''_{qu}(x,p)(\cdot,\cdot) & \mathcal{L}''_{qq}(x,p)(\cdot,\cdot) & \mathcal{L}''_{qz}(x,p)(\cdot,\cdot) \\ \mathcal{L}''_{zu}(x,p)(\cdot,\cdot) & \mathcal{L}''_{zq}(x,p)(\cdot,\cdot) & 0 \end{bmatrix}. \tag{2.16}$$

To shorten the notation, we abbreviate $x = (u,q,z)$ and $\mathcal{X} = \mathcal{V} \times \mathcal{Q} \times \mathcal{V}$. Note that $B(x,p)$ is a bilinear operator on the space $\mathcal{X}$. By "multiplication" of B with an element $\delta x \in \mathcal{X}$ from the left, we mean the insertion of the components of δx into the first argument. Similarly we define the "multiplication" of B with an element $\delta x \in \mathcal{X}$ from the right as insertion of the components of δx into the second argument. When only one element is inserted, B is interpreted as a linear operator $B : \mathcal{X} \to \mathcal{X}'$. In what follows, we shall omit the $(\cdot,\cdot)$ notation if no ambiguity arises.

In the absence of control constraints, the Newton update $(\Delta u, \Delta q, \Delta z)$ for (2.11)–(2.15) at the current iterate (u_k, q_k, z_k) is given by the solution of

$$B(x_k,p)\begin{pmatrix} \Delta u \\ \Delta q \\ \Delta z \end{pmatrix} = -\begin{pmatrix} \mathcal{L}'_u(x_k,p) \\ \mathcal{L}'_q(x_k,p) \\ \mathcal{L}'_z(x_k,p) \end{pmatrix}. \tag{2.17}$$

With nonempty active sets $\mathcal{A}^n_-$ and $\mathcal{A}^n_+$, however, (2.17) is replaced by

$$\widetilde{B}(x_k,p)\begin{pmatrix} \Delta u \\ \Delta q \\ \Delta z \end{pmatrix} = -\begin{pmatrix} \mathcal{L}'_u(x_k,p) \\ R_{\mathcal{I}^n}\mathcal{L}'_q(x_k,p) + R_{\mathcal{A}^n_-}(q_k - b_-) + R_{\mathcal{A}^n_+}(q_k - b_+) \\ \mathcal{L}'_z(x_k,p) \end{pmatrix}, \tag{2.18}$$

where

$$\widetilde{B}(x_k,p) = \begin{pmatrix} id & & \\ & R_{\mathcal{I}^n} & \\ & & id \end{pmatrix} B(x_k,p) \begin{pmatrix} id & & \\ & R_{\mathcal{I}^n} & \\ & & id \end{pmatrix} + \begin{pmatrix} 0 & & \\ & R_{\mathcal{A}^n} & \\ & & 0 \end{pmatrix}. \tag{2.19}$$

In other words, $\widetilde{B}$ is obtained from B by replacing those components in the derivatives with respect to the control q by the identity which belongs to the active set. In our practical realization, we reduce the system (2.18) to the control space $L^2(\omega)$ using Schur complement techniques; see, e.g., [17]. The reduced system is solved iteratively using the conjugate gradient method, where each step requires the evaluation of a matrix-vector product for the reduced Hessian, which in turn requires the solution of one tangent and one dual problem; see, e.g., [13] or [2] for a detailed description of this procedure in the context of space-time finite element discretization of the problem. In fact, the reduced system needs to be solved only on the currently inactive part $L^2(\mathcal{I}^n)$ of the control space since on the active sets, the update Δq satisfies the trivial relation $R_{\mathcal{A}_\pm^n}(\Delta q) = R_{\mathcal{A}_\pm^n}(b_\pm - q_{k-1})$.

The Newton step is completed by applying the update

$$(u_{k+1}, q_{k+1}, z_{k+1}) = (u_k, q_k, z_k) + (\Delta u, \Delta q, \Delta z).$$

3. Sensitivity analysis. In this section we analyze the behavior of local optimal solutions for (**OP**(p)) under perturbations of the parameter p. We derive formulas for the first and second order derivatives of the reduced quantity of interest and develop an efficient method for their evaluation.

To set the stage, we outline the main ideas in section 3.1 by means of a finite-dimensional optimization problem, without partitioning the optimization variables into states and controls, and in the absence of control constraints. To facilitate the discussion of the infinite-dimensional case, we treat the case of no control constraints in section 3.2 and turn to problems with these constraints in section 3.3. Throughout, we compare the standard sensitivity analysis for the reduced cost functional $j(p)$ (1.5) with our analysis for the reduced quantity of interest $i(p)$ (1.4). The main results can be found in Theorem 3.6 for the unconstrained case and in Theorems 3.18 and 3.21 for the case with control constraints. An algorithm at the end of section 3 summarizes the necessary steps to evaluate the various sensitivity quantities.

3.1. Outline of ideas. Let us consider the nonlinear finite-dimensional equality-constrained optimization problem

$$\text{Minimize } J(x,p) \text{ s.t. } g(x,p) = 0, \tag{3.1}$$

where $x \in \mathbb{R}^n$ denotes the optimization variable, $p \in \mathbb{R}^d$ is the perturbation parameter, and $g : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}^m$ collects a number of equality constraints. The Lagrangian of (3.1) is $\mathcal{L}(x,p) = J(x,p) - z^\top g(x,p)$, and under standard constraint qualifications, a local minimizer x_0 of (3.1) at the reference parameter p_0 has an associated Lagrange multiplier $z_0 \in \mathbb{R}^m$ such that

$$\begin{aligned} &\mathcal{L}'_x(x_0, z_0, p_0) = J'_x(x_0, p_0) - z_0^\top g'_x(x_0, p_0) = 0, \\ &\mathcal{L}'_z(x_0, z_0, p_0) = g(x_0, p_0) = 0 \end{aligned} \tag{3.2}$$

hold. If we assume second order sufficient conditions to hold in addition, then the implicit function theorem yields the local existence of functions $X(p)$ and $Z(p)$ which satisfy (3.2) with p instead of p_0, and $X(p_0) = x_0$ and $Z(p_0) = z_0$ hold. Moreover, (3.2) can be differentiated totally with respect to the parameter and we obtain

$$\begin{pmatrix} \mathcal{L}''_{xx}(x_0, z_0, p_0) & g'_x(x_0, p_0)^\top \\ g'_x(x_0, p_0) & 0 \end{pmatrix} \begin{pmatrix} X'(p_0)\,\delta p \\ Z'(p_0)\,\delta p \end{pmatrix} = - \begin{pmatrix} \mathcal{L}''_{xp}(x_0, z_0, p_0)\,\delta p \\ g'_p(x_0, p_0)\,\delta p \end{pmatrix}. \tag{3.3}$$

The solution of (3.3) is a directional derivative of $X(p)$ (and $Z(p)$) at $p = p_0$, and we note that it is equivalent to the solution of a linear-quadratic optimization problem. Hence the evaluation of the full Jacobian $X'(p_0)$ requires $d = \dim \mathcal{P}$ solves of (3.3) with different δp. In our context of large-scale problems, iterative solvers need to be used and the numerical effort to evaluate the full Jacobian scales linearly with the number of right-hand sides, i.e., with the dimension of the parameter space $d = \dim \mathcal{P}$.

We adapt the definition of the reduced cost functional and the reduced quantity of interest to our current setting, $j(p) = J(X(p), p)$ and $i(p) = I(X(p), p)$. Since we wish to compare the effort to compute the first and second order derivatives of both, we begin by recalling the following result.

LEMMA 3.1.

$$
\begin{aligned}
j'(p_0)\,\delta p &= \mathcal{L}'_p(x_0, z_0, p_0)\,\delta p,\\
\delta p^\top j''(p_0)\,\widehat{\delta p} &= \delta p^\top \big[\mathcal{L}''_{px}(x_0, z_0, p_0)X'(p_0)\,\widehat{\delta p} + \mathcal{L}''_{pz}(x_0, z_0, p_0)Z'(p_0)\,\widehat{\delta p}\\
&\qquad + \mathcal{L}''_{pp}(x_0, z_0, p_0)\,\widehat{\delta p}\big].
\end{aligned}
$$

. We have $j(p) = \mathcal{L}(X(p), Z(p), p)$ and hence by the chain rule $j'(p_0) = \mathcal{L}'_x(x_0, z_0, p_0)X'(p_0) + \mathcal{L}'_z(x_0, z_0, p_0)Z'(p_0) + \mathcal{L}'_p(x_0, z_0, p_0)$, where the first two terms vanish in view of (3.2). Differentiating again totally with respect to p yields the expression for the second derivative. ∎

Lemma 3.1 shows that the evaluation of the gradient of $j(\cdot)$ does not require any linear solves of the sensitivity system (3.3), while the evaluation of the Hessian requires $d = \dim \mathcal{P}$ such solves. The corresponding results for the infinite-dimensional case can be found below in Propositions 3.5 and 3.16 for the unconstrained and control constrained cases.

We will show now that the derivatives of the reduced quantity of interest $i(\cdot)$ can be evaluated efficiently, requiring just one additional system solve. This is a significant improvement over a direct approach; see Table 3.1.

From a first look at

$$
i'(p_0)\,\delta p = I'_x(x_0, p_0)X'(p_0)\,\delta p + I'_p(x_0, p_0)\,\delta p
$$

it seems that the evaluation of the gradient $i'(p_0)$ requires $d = \dim \mathcal{P}$ solves of the system (3.3). This is referred to as the direct approach in Table 3.1. However, using (3.3), we may rewrite this as

$$
i'(p_0)\,\delta p = \big[-\big(I'_x(x_0, p_0), 0\big)\,B_0^{-1}\big] \begin{pmatrix} \mathcal{L}''_{xp}(x_0, z_0, p_0)\,\delta p \\ g'_p(x_0, p_0)\,\delta p \end{pmatrix} + I'_p(x_0, p_0)\,\delta p,
$$

where B_0 is the matrix on the left-hand side of (3.3). Realizing that $I'_x(x_0, p_0)$ has just one row, evaluating the term in square brackets amounts to only one linear system solve. We define the dual quantities (v, y) by

$$
B_0^\top \begin{pmatrix} v \\ y \end{pmatrix} = - \begin{pmatrix} I'_x(x_0, p_0) \\ 0 \end{pmatrix}
$$

and finally obtain

$$
i'(p_0)\,\delta p = v^\top \mathcal{L}''_{xp}(x_0, z_0, p_0)\,\delta p + y^\top \mathcal{L}''_{zp}(x_0, z_0, p_0)\,\delta p + I'_p(x_0, p_0)\,\delta p. \tag{3.4}
$$

We refer to this as a dual approach. In our context, B_0 is symmetric and hence the computation of the dual quantities requires just one solve of (3.3) with a modified right-hand side; see again Table 3.1.

For the second derivative, we differentiate (3.4) totally with respect to p. From the chain rule we infer that the sensitivities $X'(p_0)$ and $Z'(p_0)$ now come into play. In addition, v and y need to be differentiated with respect to p, but again a duality technique can be used in order to avoid computing these extra terms. Hence the extra computational cost to evaluate the Hessian of $i(\cdot)$ amounts to $d = \dim \mathcal{P}$ solves for the evaluation of the sensitivity matrices $X'(p_0)$ and $Z'(p_0)$; see Table 3.1. Details can be found in the proofs of Theorem 3.6 for the unconstrained case and Theorems 3.18 and 3.21 for the case with control constraints.

3.2. The case of no control constraints. Throughout this and the following section, we denote by $p_0 \in \mathcal{P}$ a given reference parameter and by $x_0 = (u_0, q_0, z_0)$ a solution to the corresponding first order optimality system (2.11)–(2.15). Moreover, we make the following regularity assumption which we require throughout.

ASSUMPTION 3.2. $a'_u(u_0, q_0, p_0) : \mathcal{V} \to \mathcal{V}'$

In the case of no control constraints, i.e., $\mathcal{Q}_{ad} = \mathcal{Q}$, the first order necessary conditions (2.11)–(2.15) simplify to

$$\mathcal{L}'_u(u, q, z, p)(\delta u) = 0 \quad \forall \delta u \in \mathcal{V}, \tag{3.5}$$

$$\mathcal{L}'_q(u, q, z, p)(\delta q) = 0 \quad \forall \delta q \in \mathcal{Q}, \tag{3.6}$$

$$\mathcal{L}'_z(u, q, z, p)(\delta z) = 0 \quad \forall \delta z \in \mathcal{V}. \tag{3.7}$$

The analysis in this subsection is based on the classical implicit function theorem. We denote by $B_0 = B(x_0, p_0)$ the previously defined Hessian operator at the given reference solution. For the results in this section we require that B_0 is boundedly invertible. This property follows from the second order sufficient conditions; see, for instance, [14].

LEMMA 3.3. 2.2 x_0 **OP**(p_0) B_0

The following lemma is a direct application of the implicit function theorem (see [5]) to the first order optimality system (3.5)–(3.7).

LEMMA 3.4. B_0 $\mathcal{N}(p_0) \subset \mathcal{P}$ p_0 $\mathcal{N}(x_0) \subset \mathcal{X}$ x_0 $(U, Q, Z) : \mathcal{N}(p_0) \to \mathcal{N}(x_0)$

(a) $p \in \mathcal{N}(p_0)$ $(U(p), Q(p), Z(p))$ (3.5)–(3.7) $\mathcal{N}(x_0)$

(b) $(U(p_0), Q(p_0), Z(p_0)) = (u_0, q_0, z_0)$

(c) (U, Q, Z) p_0 $\delta p \in \mathcal{P}$

$$B_0 \begin{pmatrix} U'(p_0)(\delta p) \\ Q'(p_0)(\delta p) \\ Z'(p_0)(\delta p) \end{pmatrix} = - \begin{pmatrix} \mathcal{L}''_{up}(x_0, p_0)(\cdot, \delta p) \\ \mathcal{L}''_{qp}(x_0, p_0)(\cdot, \delta p) \\ \mathcal{L}''_{zp}(x_0, p_0)(\cdot, \delta p) \end{pmatrix}. \tag{3.8}$$

In the following proposition we recall the first and second order sensitivity derivatives of the $j(p)$; compare this to [18].

PROPOSITION 3.5. B_0 $j(p) = J(U(p), p) + \frac{\alpha}{2}\|Q(p) - \overline{q}\|^2_{\mathcal{Q}}$ $\mathcal{N}(p_0)$

p_0 $\delta p \in \mathcal{P}$

$$j'(p_0)(\delta p) = \mathcal{L}'_p(x_0, p_0)(\delta p). \tag{3.9}$$

δp $\widehat{\delta p}$

$$\begin{aligned} j''(p_0)(\delta p, \widehat{\delta p}) &= \mathcal{L}''_{up}(x_0, p_0)(U'(p)(\delta p), \widehat{\delta p}) + \mathcal{L}''_{qp}(x_0, p_0)(Q'(p)(\delta p), \widehat{\delta p}) \\ &\quad + \mathcal{L}''_{zp}(x_0, p_0)(Z'(p)(\delta p), \widehat{\delta p}) + \mathcal{L}''_{pp}(x_0, p_0)(\delta p, \widehat{\delta p}). \end{aligned} \tag{3.10}$$

Since $(U(p), Q(p))$ satisfies the state equation, we have

$$j(p) = \mathcal{L}(U(p), Q(p), Z(p), p)$$

for all $p \in \mathcal{N}(p_0)$. By the chain rule, the derivative of $j(p)$ reads

$$\begin{aligned} j'(p_0)(\delta p) = \mathcal{L}'_u(x_0, p_0)(U'(p_0)(\delta p)) + \mathcal{L}'_q(x_0, p_0)(Q'(p_0)(\delta p)) + \mathcal{L}'_z(x_0, p_0)(Z'(p_0)(\delta p)) \\ + \mathcal{L}'_p(x_0, p_0)(\delta p). \end{aligned}$$

The three terms in the first line vanish in view of the optimality system (3.5)–(3.7). Differentiating (3.9) again totally with respect to p in the direction of $\widehat{\delta p}$ yields (3.10), which completes the proof. ∎

The previous proposition allows us to evaluate the first order derivative of the reduced cost functional computing the sensitivity derivatives of the state, control, and adjoint variables. That is, the effort to evaluate $j'(p_0)$ is negligible compared to the effort required to solve the optimization problem. In order to obtain second order derivative $j''(p_0)$, however, the sensitivity derivatives have to be computed according to formula (3.8). This corresponds to the solution of one additional linear-quadratic optimization problem per perturbation direction δp, whose optimality system is given by (3.8).

We now turn to our main result in the absence of control constraints. In the following theorem, we show that the first and second order derivatives of the can be evaluated at practically the same effort as those of the cost functional. To this end, we use a duality technique (see section 3.1) and formulate the following dual problem for the dual variables $v \in \mathcal{V}$, $r \in \mathcal{Q}$, and $y \in \mathcal{V}$:

$$B_0 \begin{pmatrix} v \\ r \\ y \end{pmatrix} = - \begin{pmatrix} I'_u(q_0, u_0, p_0) \\ I'_q(q_0, u_0, p_0) \\ 0 \end{pmatrix}. \tag{3.11}$$

We remark that this dual problem involves the same operator matrix B_0 as the sensitivity problem (3.8) since B_0 is self-adjoint.

THEOREM 3.6. B_0 $i(p)$ (1.4) $\mathcal{N}(p_0)$ p_0 $\delta p \in \mathcal{P}$

$$\begin{aligned} i'(p_0)(\delta p) = \mathcal{L}''_{up}(x_0, p_0)(v, \delta p) + \mathcal{L}''_{qp}(x_0, p_0)(r, \delta p) + \mathcal{L}''_{zp}(x_0, p_0)(y, \delta p) \\ + I'_p(u_0, q_0, p_0)(\delta p). \end{aligned} \tag{3.12}$$

δp $\widehat{\delta p}$

$$\begin{aligned} i''(p_0)(\delta p, \widehat{\delta p}) &= \langle v, \eta \rangle_{\mathcal{V} \times \mathcal{V}'} + \langle r, \kappa \rangle_{\mathcal{Q} \times \mathcal{Q}'} + \langle y, \sigma \rangle_{\mathcal{V} \times \mathcal{V}'} \\ &+ \begin{pmatrix} U'(p_0)(\delta p) \\ Q'(p_0)(\delta p) \\ \delta p \end{pmatrix}^\top \begin{pmatrix} I''_{uu}(q_0, u_0, p_0) & I''_{uq}(q_0, u_0, p_0) & I''_{up}(q_0, u_0, p_0) \\ I''_{qu}(q_0, u_0, p_0) & I''_{qq}(q_0, u_0, p_0) & I''_{qp}(q_0, u_0, p_0) \\ I''_{pu}(q_0, u_0, p_0) & I''_{pq}(q_0, u_0, p_0) & I''_{pp}(q_0, u_0, p_0) \end{pmatrix} \begin{pmatrix} U'(p_0)(\widehat{\delta p}) \\ Q'(p_0)(\widehat{\delta p}) \\ \widehat{\delta p} \end{pmatrix}. \end{aligned} \tag{3.13}$$

. $(\eta, \kappa, \sigma) \in \mathcal{V}' \times \mathcal{Q}' \times \mathcal{V}'$, , , .

$$
\begin{aligned}
\begin{pmatrix} \eta \\ \kappa \\ \sigma \end{pmatrix} &= \begin{pmatrix} \mathcal{L}'''_{upp}()(\cdot, \delta p, \widehat{\delta p}) \\ \mathcal{L}'''_{qpp}()(\cdot, \delta p, \widehat{\delta p}) \\ \mathcal{L}'''_{zpp}()(\cdot, \delta p, \widehat{\delta p}) \end{pmatrix} \\
&+ \begin{pmatrix} \mathcal{L}'''_{upu}()(\cdot, \delta p, U'(p_0)(\widehat{\delta p})) + \mathcal{L}'''_{upq}()(\cdot, \delta p, Q'(p_0)(\widehat{\delta p})) + \mathcal{L}'''_{upz}()(\cdot, \delta p, Z'(p_0)(\widehat{\delta p})) \\ \mathcal{L}'''_{qpu}()(\cdot, \delta p, U'(p_0)(\widehat{\delta p})) + \mathcal{L}'''_{qpq}()(\cdot, \delta p, Q'(p_0)(\widehat{\delta p})) + \mathcal{L}'''_{qpz}()(\cdot, \delta p, Z'(p_0)(\widehat{\delta p})) \\ \mathcal{L}'''_{zpu}()(\cdot, \delta p, U'(p_0)(\widehat{\delta p})) + \mathcal{L}'''_{zpq}()(\cdot, \delta p, Q'(p_0)(\widehat{\delta p})) \end{pmatrix} \\
&+ \Big(B'_u()(U'(p_0)(\widehat{\delta p})) + B'_q()(Q'(p_0)(\widehat{\delta p})) + B'_z()(Z'(p_0)(\widehat{\delta p})) + B'_p()(\widehat{\delta p}) \Big) \begin{pmatrix} U'(p_0)(\delta p) \\ Q'(p_0)(\delta p) \\ Z'(p_0)(\delta p) \end{pmatrix}.
\end{aligned}
$$

. . . . 3.7.

(a) In the definition of (η, κ, σ) we have abbreviated the evaluation at the point (x_0, p_0) by $()$.
(b) The bracket $\langle \cdot, \cdot \rangle_{\mathcal{V} \times \mathcal{V}'}$ in (3.13) denotes the duality pairing between $\mathcal{V}$ and its dual space $\mathcal{V}'$. For instance, the evaluation of $\langle v, \eta \rangle_{\mathcal{V} \times \mathcal{V}'}$ amounts to plugging in v instead of $\cdot$ in the definition of η. A similar notation is used for the control space $\mathcal{Q}$.
(c) It is tedious but straightforward to check that (3.13) coincides with (3.10) if the quantity of interest is chosen equal to the cost functional. In this case, it follows from (3.11) that the dual quantities v and r vanish and $y = z_0$ holds.

. . . . . 3.6. From the definition of the reduced quantity of interest (1.4), we infer that

$$
\begin{aligned}
i'(p_0)(\delta p) &= I'_u(u_0, q_0, p_0)(U'(p_0)(\delta p)) \\
&\quad + I'_q(u_0, q_0, p_0)(Q'(p_0)(\delta p)) + I'_p(u_0, q_0, p_0)(\delta p)
\end{aligned} \tag{3.14}
$$

holds. In virtue of (3.8) and (3.11), the sum of the first two terms equals

$$
- \begin{pmatrix} I'_u(u_0, q_0, p_0) \\ I'_q(u_0, q_0, p_0) \\ 0 \end{pmatrix}^\top B_0^{-1} \begin{pmatrix} \mathcal{L}''_{up}(x_0, p_0)(\cdot, \delta p) \\ \mathcal{L}''_{qp}(x_0, p_0)(\cdot, \delta p) \\ \mathcal{L}''_{zp}(x_0, p_0)(\cdot, \delta p) \end{pmatrix} = \begin{pmatrix} v \\ r \\ y \end{pmatrix}^\top \begin{pmatrix} \mathcal{L}''_{up}(x_0, p_0)(\cdot, \delta p) \\ \mathcal{L}''_{qp}(x_0, p_0)(\cdot, \delta p) \\ \mathcal{L}''_{zp}(x_0, p_0)(\cdot, \delta p) \end{pmatrix},
$$

which implies (3.12). In order to obtain the second derivative, we differentiate (3.14) totally with respect to p in the direction of $\widehat{\delta p}$. This yields

$$
\begin{aligned}
&i''(p_0)(\delta p, \widehat{\delta p}) \\
&= \begin{pmatrix} U'(p_0)(\delta p) \\ Q'(p_0)(\delta p) \\ \delta p \end{pmatrix}^\top \begin{pmatrix} I''_{uu}(q_0, u_0, p_0) & I''_{uq}(q_0, u_0, p_0) & I''_{up}(q_0, u_0, p_0) \\ I''_{qu}(q_0, u_0, p_0) & I''_{qq}(q_0, u_0, p_0) & I''_{qp}(q_0, u_0, p_0) \\ I''_{pu}(q_0, u_0, p_0) & I''_{pq}(q_0, u_0, p_0) & I''_{pp}(q_0, u_0, p_0) \end{pmatrix} \begin{pmatrix} U'(p_0)(\widehat{\delta p}) \\ Q'(p_0)(\widehat{\delta p}) \\ \widehat{\delta p} \end{pmatrix} \\
&\qquad + \begin{pmatrix} I'_u(u_0, q_0, p_0) \\ I'_q(u_0, q_0, p_0) \\ 0 \end{pmatrix}^\top \begin{pmatrix} U''(p_0)(\delta p, \widehat{\delta p}) \\ Q''(p_0)(\delta p, \widehat{\delta p}) \\ Z''(p_0)(\delta p, \widehat{\delta p}) \end{pmatrix}.
\end{aligned} \tag{3.15}
$$

From differentiating (3.8) totally with respect to p in the direction of $\widehat{\delta p}$, we obtain

$$B_0 \begin{pmatrix} U''(p_0)(\delta p, \widehat{\delta p}) \\ Q''(p_0)(\delta p, \widehat{\delta p}) \\ Z''(p_0)(\delta p, \widehat{\delta p}) \end{pmatrix} = - \begin{pmatrix} \eta \\ \kappa \\ \sigma \end{pmatrix}. \tag{3.16}$$

From here, (3.13) follows. □

The main statement of the previous theorem is that the first and second order derivatives of the reduced quantity of interest can be evaluated at the additional expense of just one dual problem (3.11), compared to the evaluation of the reduced cost functional's derivatives. More precisely, computing the gradient of $i(p)$ at p_0 requires only the solution of (3.11). In addition, in order to compute the Hessian of $i(p)$ at p_0, the sensitivity quantities $U'(p_0)$, $Q'(p_0)$, and $Z'(p_0)$ need to be evaluated in the directions of a collection of basis vectors of the parameter space $\mathcal{P}$. That is, $\dim \mathcal{P}$ sensitivity problems (3.8) need to be solved. These are exactly the same problems which have to be solved for the computation of the Hessian of the [illegible]; see Table 3.1. Note that in the combined effort $1 + \dim \mathcal{P}$, "1" refers to the same dual problem (3.11) that has already been solved during the computation of the gradient of $i(p)$. In case that the space $\mathcal{P}$ is infinite-dimensional, it needs to be discretized first. Finally, in order to evaluate the second order Taylor expansion for a given direction δp,

$$i(p_0 + \delta p) \approx i(p_0) + i'(p_0)(\delta p) + \frac{1}{2} i''(p_0)(\delta p, \delta p),$$

the same dual problem (3.11) and one sensitivity problem (3.8) in the direction of δp are needed; see Table 3.1.

TABLE 3.1
Number of linear-quadratic problems to be solved to evaluate the derivatives of $j(p)$ and $i(p)$.

	Reduced cost functional $j(p)$	Reduced quantity of interest $i(p)$	
		Dual approach	Direct approach
Gradient	0	1	$\dim \mathcal{P}$
Hessian	$\dim \mathcal{P}$	$1 + \dim \mathcal{P}$	$(\dim \mathcal{P})\ (\dim \mathcal{P} + 1)/2$

Note that the sensitivity and dual problems (3.8) and (3.11), respectively, are solved by the technique described in section 2. The solution of such problems amounts to the computation of one additional QP step (2.17), with a different right-hand side. Therefore, the numerical effort to compute, e.g., the second order Taylor expansion for a given direction is typically low compared to the solution of the nonlinear optimization problem $\mathbf{OP}(p_0)$.

3.3. The control-constrained case. The analysis is based on the notion of strong regularity for the problem $\mathbf{OP}(p)$. Strong regularity extends the previous assumption of bounded invertibility of B_0 used throughout section 3.2.

Below, we make use of $\mu_0 \in \mathcal{Q}$ given by the following identification:

$$(\mu_0, \delta q) = -\mathcal{L}_q'(x_0, p_0)(\delta q) \quad \forall \delta q \in \mathcal{Q}. \tag{3.17}$$

This quantity acts as a Lagrange multiplier for the control constraint $q \in \mathcal{Q}_{ad}$. For the definition of strong regularity we introduce the following linearized optimality system

which depends on $\varepsilon = (\varepsilon^u, \varepsilon^q, \varepsilon^z) \in \mathcal{V} \times \mathcal{Q} \times \mathcal{V}$:

(**LOS**(ε))

$$\begin{aligned} &\mathcal{L}''_{uu}(x_0,p_0)(\delta u, u-u_0) + \mathcal{L}''_{uq}(x_0,p_0)(\delta u, q-q_0) \\ &\quad + \mathcal{L}''_{uz}(x_0,p_0)(\delta u, z-z_0) + \mathcal{L}'_{u}(x_0,p_0)(\delta u) \\ &\quad + (\varepsilon^u, \delta u)_{\mathcal{V}} = 0 \quad \forall \delta u \in \mathcal{V}, \end{aligned} \tag{3.18}$$

$$\begin{aligned} &\mathcal{L}''_{uq}(x_0,p_0)(u-u_0, \delta q - q) + \mathcal{L}''_{qq}(x_0,p_0)(\delta q - q, q-q_0) \\ &\quad + \mathcal{L}''_{qz}(x_0,p_0)(\delta q - q, z-z_0) \\ &\quad + \mathcal{L}'_{q}(x_0,p_0)(\delta q - q) + (\varepsilon^q, \delta q - q) \geq 0 \quad \forall \delta q \in \mathcal{Q}_{ad}, \end{aligned} \tag{3.19}$$

$$\begin{aligned} &\mathcal{L}''_{zu}(x_0,p_0)(\delta z, u-u_0) + \mathcal{L}''_{zq}(x_0,p_0)(\delta z, q-q_0) \\ &\quad + \mathcal{L}'_{z}(x_0,p_0)(\delta z) + (\varepsilon^z, \delta z)_{\mathcal{V}} = 0 \quad \forall \delta z \in \mathcal{V}. \end{aligned} \tag{3.20}$$

In what follows, we refer to (3.18)–(3.20) as (**LOS**(ε)).

DEFINITION 3.8 (strong regularity). $p_0 \in \mathcal{P}$ $x_0 = (u_0, q_0, z_0)$ (2.5)–(2.7) $\mathcal{N}(0) \subset \mathcal{X} = \mathcal{V} \times \mathcal{Q} \times \mathcal{V}$ 0 $\mathcal{N}(x_0) \subset \mathcal{X}$ x_0

(a) $\varepsilon \in \mathcal{N}(0)$ $(u^\varepsilon, q^\varepsilon, z^\varepsilon)$ (3.18)–(3.20)

(b) $(u^\varepsilon, q^\varepsilon, z^\varepsilon)$ (3.18)–(3.20) $\mathcal{N}(x_0)$

(c) $(u^\varepsilon, q^\varepsilon, z^\varepsilon)$ ε $L > 0$

$$\|u^{\varepsilon_1} - u^{\varepsilon_2}\|_{\mathcal{V}} + \|q^{\varepsilon_1} - q^{\varepsilon_2}\|_{\mathcal{Q}} + \|z^{\varepsilon_1} - z^{\varepsilon_2}\|_{\mathcal{V}} \leq L \, \|\varepsilon_1 - \varepsilon_2\|_{\mathcal{X}} \tag{3.21}$$

$\varepsilon_1, \varepsilon_2 \in \mathcal{N}(0)$

(2.5)–(2.7) strongly regular x_0

Note that (u_0, q_0, z_0) solves (3.18)–(3.20) for $\varepsilon = 0$. It is not difficult to see that in the case of no control constraints, i.e., $\mathcal{Q} = \mathcal{Q}_{ad}$, strong regularity is nothing else than bounded invertibility of B_0 which we had to assume in section 3.2. In the following lemma we show that strong regularity holds under suitable second order sufficient optimality conditions, in analogy to Lemma 3.3. The proof can be carried out using the techniques presented in [21].

LEMMA 3.9. 2.2 x_0 **OP**(p_0) $\varepsilon \in \mathcal{X}$ (3.18)–(3.20) $(u^\varepsilon, q^\varepsilon, z^\varepsilon)$

$$\mathcal{X} \ni \varepsilon \mapsto (u^\varepsilon, q^\varepsilon, z^\varepsilon) \in \mathcal{X} \tag{3.22}$$

x_0

In the next step, we proceed to prove that the solution $(u^\varepsilon, q^\varepsilon, z^\varepsilon)$ of the linearized optimality system (3.18)–(3.20) is directionally differentiable with respect to the perturbation ε. To this end, we need the following assumption.

ASSUMPTION 3.10. (u_0, q_0, z_0)

1. $\mathcal{V} \ni u \mapsto a''_{qu}(u_0, q_0, p_0)(\cdot, u, z_0) \in \mathcal{Q}'$
2. $\mathcal{Q} \ni q \mapsto a''_{qq}(u_0, q_0, p_0)(\cdot, q, z_0) \in \mathcal{Q}'$
3. $\mathcal{V} \ni z \mapsto a'_{q}(u_0, q_0, p_0)(\cdot, z) \in \mathcal{Q}'$

3.11. The previous assumption is satisfied for the following important classes of PDE-constrained optimization problems on bounded domains $\Omega \subset \mathbb{R}^d$, $d \in \{1, 2, 3\}$:

(a) If (**OP**(p)) is a distributed optimal control problem for a semilinear elliptic PDE, e.g.,

$$-\Delta u = f(u) + q \text{ on } \Omega$$

with $\mathcal{V} = H_0^1(\Omega)$ and $\mathcal{Q} = L^2(\Omega)$, then $a''_{qu} = a''_{qq} = 0$ and a'_q is the compact injection of $\mathcal{V}$ into $\mathcal{Q}$.

(b) In the case of Neumann boundary control on $\partial\Omega$, e.g.,

$$-\Delta u = f(u) \text{ on } \Omega \quad \text{and} \quad \frac{\partial}{\partial n} u = q \text{ on } \partial\Omega,$$

we have $\mathcal{V} = H^1(\Omega)$ and $\mathcal{Q} = L^2(\partial\Omega)$. Again, $a''_{qu} = a''_{qq} = 0$ and a'_q is the compact Dirichlet trace operator from $\mathcal{V}$ to $\mathcal{Q}$.

(c) For bilinear control problems, e.g.,

$$-\Delta u = qu + f \text{ on } \Omega$$

with $\mathcal{V} = H_0^1(\Omega)$, $\mathcal{Q} = L^2(\Omega)$, and an appropriate admissible set $\mathcal{Q}_{ad}$, we have $a''_{qq} = 0$. Moreover, the operators $u \mapsto a''_{qu}(u_0, q_0, p_0)(\cdot, u, z_0) = (uz_0, \cdot)$ and $z \mapsto a'_q(u_0, q_0, z_0) = (u_0 z, \cdot)$ are compact from $\mathcal{V}$ to $\mathcal{Q}'$ since the pointwise product of two functions in $\mathcal{V}$ embeds compactly into $\mathcal{Q}$.

(d) For parabolic equations such as

$$u_t = \Delta u + f(u) + q$$

with solutions in $\mathcal{V} = \{u \in L^2(0, T; H_0^1(\Omega)) : u_t \in L^2(0, T; H^{-1}(\Omega))\}$ we have $a''_{qu} = a''_{qq} = 0$ and a'_q is the compact injection of $\mathcal{V}$ into $\mathcal{Q} = L^2(\Omega \times (0, T))$.

(e) Finally, Assumption 3.10 is always satisfied if the space $\mathcal{Q}$ is finite-dimensional. This includes all cases of parameter identification problems without any additional restrictions on the coupling between the parameters q and the state variable u. For instance, the Arrhenius law leads to reaction-diffusion equations of the form

$$-\Delta u = f(u) + e^{qu} \text{ on } \Omega$$

with unknown Arrhenius parameter $q \in \mathbb{R}$.

For the following theorem, we introduce the admissible set $\widehat{\mathcal{Q}}_{ad}$, defined as

$$\widehat{\mathcal{Q}}_{ad} = \{\hat{q} \in \mathcal{Q} : \widehat{b}_-(x) \le \hat{q}(x) \le \widehat{b}_+(x) \text{ a.e. on } \omega\}$$

with bounds

$$\widehat{b}_-(x) = \begin{cases} 0 & \text{if } \mu_0(x) \neq 0 \text{ or } q_0(x) = b_-(x), \\ -\infty & \text{else,} \end{cases}$$

$$\widehat{b}_+(x) = \begin{cases} 0 & \text{if } \mu_0(x) \neq 0 \text{ or } q_0(x) = b_+(x), \\ +\infty & \text{else.} \end{cases}$$

THEOREM 3.12. [illegible] 2.2 [illegible] x_0 [illegible] **OP**(p_0) [illegible] 3.10 [illegible]

(3.22) [illegible] $\varepsilon = 0$ [illegible] $\delta\varepsilon = (\delta\varepsilon^u, \delta\varepsilon^q, \delta\varepsilon^z) \in \mathcal{X}$ [illegible] $(\hat{u}, \hat{q})$ [illegible] $\hat{z}$ [illegible] $\mathbf{DQP}(\delta\varepsilon)$

($\mathbf{DQP}(\delta\varepsilon)$)

$$\text{[illegible]} \quad \frac{1}{2}\begin{pmatrix}\hat{u} & \hat{q}\end{pmatrix}\begin{pmatrix}\mathcal{L}''_{uu}(x_0,p_0) & \mathcal{L}''_{uq}(x_0,p_0)\\ \mathcal{L}''_{qu}(x_0,p_0) & \mathcal{L}''_{qq}(x_0,p_0)\end{pmatrix}\begin{pmatrix}\hat{u}\\ \hat{q}\end{pmatrix} + (\hat{u}, \delta\varepsilon^u)_{\mathcal{V}} + (\hat{q}, \delta\varepsilon^q)$$

[illegible] $\hat{q} \in \widehat{\mathcal{Q}}_{ad}$ [illegible]

$$a'_u(u_0,q_0,p_0)(\hat{u},\phi) + a'_q(u_0,q_0,p_0)(\hat{q},\phi) + (\delta\varepsilon^z,\phi) = 0 \quad \forall \phi \in \mathcal{V}.$$

[illegible]

$$\mathcal{L}''_{uu}(x_0,p_0)(\delta u,\hat{u}) + \mathcal{L}''_{uq}(x_0,p_0)(\delta u,\hat{q}) + \mathcal{L}''_{uz}(x_0,p_0)(\delta u,\hat{z}) + (\delta\varepsilon^u,\delta u) = 0 \quad \forall \delta u \in \mathcal{V}, \tag{3.23}$$

$$\mathcal{L}''_{uq}(x_0,p_0)(\hat{u},\delta q-\hat{q}) + \mathcal{L}''_{qq}(x_0,p_0)(\delta q-\hat{q},\hat{q}) + \mathcal{L}''_{qz}(x_0,p_0)(\delta q-\hat{q},\hat{z}) + (\delta\varepsilon^q,\delta q-\hat{q}) \geq 0 \quad \forall \delta q \in \hat{\mathcal{Q}}_{ad}, \tag{3.24}$$

$$\mathcal{L}''_{zu}(x_0,p_0)(\delta z,\hat{u}) + \mathcal{L}''_{zq}(x_0,p_0)(\delta z,\hat{q}) + (\delta\varepsilon^z,\delta z) = 0 \quad \forall \delta z \in \mathcal{V}. \tag{3.25}$$

[illegible]. Let $\delta\varepsilon = (\delta\varepsilon^u, \delta\varepsilon^q, \delta\varepsilon^z) \in \mathcal{X}$ be given and let $\{\tau_n\} \subset \mathbb{R}^+$ denote a sequence converging to zero. We denote by $(u_n, q_n, z_n) \in \mathcal{X}$ the unique solution of $\mathbf{LOS}(\varepsilon_n)$, where $\varepsilon_n = \tau_n \delta\varepsilon$. Note that (u_0, q_0, z_0) is the unique solution of $\mathbf{LOS}(0)$ and that $(u_n, q_n, z_n) \to (u_0, q_0, z_0)$ strongly in $\mathcal{X}$. From Lemma 3.9 we infer that

$$\left\|\frac{u_n-u_0}{\tau_n}\right\|_{\mathcal{V}} + \left\|\frac{q_n-q_0}{\tau_n}\right\|_{\mathcal{Q}} + \left\|\frac{z_n-z_0}{\tau_n}\right\|_{\mathcal{V}} \leq L\,\|\delta\varepsilon\|_{\mathcal{X}}.$$

This implies that a subsequence (still denoted by index n) of the difference quotients converges weakly to some limit element $(\hat{u}, \hat{q}, \hat{z}) \in \mathcal{X}$. The proof proceeds with the construction of the [illegible] limit $\widetilde{q}$ of $(q_n - q_0)/\tau_n$, which is later shown to coincide with $\hat{q}$. It is well known that the variational inequality (3.19) in $\mathbf{LOS}(\varepsilon_n)$ can be equivalently rewritten as

$$q_n(x) = \Pi_{[b_-(x),b_+(x)]}\big(d_n(x)\big) \text{ a.e. on } \omega, \tag{3.26}$$

where $\Pi_{[b_-(x),b_+(x)]}$ is the projection onto the interval $[b_-(x), b_+(x)]$ and

$$d_n = \bar{q} + \frac{1}{\alpha}\Big(a''_{qu}(u_0,q_0,p_0)(\cdot,u_n-u_0,z_0) + a''_{qq}(u_0,q_0,p_0)(\cdot,q_n-q_0,z_0) + a'_q(u_0,q_0,p_0)(\cdot,z_n) - \varepsilon^q_n\Big) \in \mathcal{Q}. \tag{3.27}$$

The linear operators in (3.27) are understood as their Riesz representations in $\mathcal{Q}$. Similarly, we have $q_0(x) = \Pi_{[b_-(x),b_+(x)]}\big(d_0(x)\big)$ a.e. on ω, where

$$d_0 = \bar{q} + \frac{1}{\alpha}a'_q(u_0,q_0,p_0)(\cdot,z_0) \in \mathcal{Q}. \tag{3.28}$$

Note that $d_n \to d_0$ strongly in $\mathcal{Q}$ since the Fréchet derivatives in (3.27) are bounded linear operators. From the compactness properties in Assumption 3.10 we infer that

$$\frac{d_n - d_0}{\tau_n} \to \hat{d} \text{ strongly in } \mathcal{Q}, \quad \text{where}$$

$$\hat{d} = \frac{1}{\alpha}\Big(a''_{qu}(u_0,q_0,p_0)(\cdot,\hat{u},z_0) + a''_{qq}(u_0,q_0,p_0)(\cdot,\hat{q},z_0) + a'_q(u_0,q_0,p_0)(\cdot,\hat{z}) - \delta\varepsilon^q\Big).$$

By taking another subsequence, we obtain that $d_n \to d_0$ and $(d_n - d_0)/\tau_n \to \hat{d}$ hold also pointwise a.e. on ω. The construction of the pointwise limit

$$\widetilde{q}(x) = \lim_{n\to\infty} \frac{q_n(x) - q_0(x)}{\tau_n}$$

uses the following partition of ω into five disjoint subsets:

$$\omega = \omega^I \cup \omega_0^+ \cup (\omega^+ \setminus \omega_0^+) \cup \omega_0^- \cup (\omega^- \setminus \omega_0^-), \tag{3.29}$$

where

$$\omega^I = \{x \in \omega : b_-(x) < q_0(x) < b_+(x)\} \quad \text{(inactive)}, \tag{3.30a}$$

$$\omega_0^+ = \{x \in \omega : \mu_0(x) > 0\} \quad \text{(upper strongly active)}, \tag{3.30b}$$

$$\omega^+ = \{x \in \omega : q_0(x) = b_+(x)\} \quad \text{(upper active)}, \tag{3.30c}$$

$$\omega_0^- = \{x \in \omega : \mu_0(x) < 0\} \quad \text{(lower strongly active)}, \tag{3.30d}$$

$$\omega^- = \{x \in \omega : q_0(x) = b_-(x)\} \quad \text{(lower active)}. \tag{3.30e}$$

The Lagrange multiplier μ_0 belonging to the constraint $q_0 \in \mathcal{Q}_{ad}$ defined in (3.17) allows the following representation:

$$\mu_0 = \alpha(d_0 - q_0). \tag{3.31}$$

Note that the five sets in (3.29) are guaranteed to be disjoint if $b_-(x) < b_+(x)$ holds a.e. on ω. However, one can easily check that $\widetilde{q}$ is well defined also in the case that the bounds coincide on all or part of ω. We now distinguish five cases according to the sets in (3.29).

1. For almost every x in the inactive subset ω^I, we have $q_0(x) = d_0(x)$ and $q_n(x) = d_n(x)$ for all sufficiently large n. Therefore,

$$\widetilde{q}(x) = \lim_{n\to\infty} \frac{q_n(x) - q_0(x)}{\tau_n} = \hat{d}(x).$$

2. For almost every $x \in \omega_0^+$, $\mu_0(x) > 0$ implies $d_0(x) > q_0(x)$ by (3.31). Therefore, $q_0(x) = b_+(x)$ and $d_n(x) > q_0(x)$ for sufficiently large n. Hence $q_n = b_+(x)$ for these n and

$$\widetilde{q}(x) = \lim_{n\to\infty} \frac{q_n(x) - q_0(x)}{\tau_n} = 0.$$

3. For almost every $x \in \omega^+ \setminus \omega_0^+$, we have $q_0(x) = b_+(x) = d_0(x)$.

(a) If $\hat{d}(x) > 0$, then $d_n(x) > b_+(x)$ for sufficiently large n. Therefore, $q_n(x) = b_+(x)$ for these n and hence $\widetilde{q}(x) = 0$.

(b) If $\hat{d}(x) = 0$, then $(q_n(x) - q_0(x))/\tau_n = \min\{0, d_n(x) - b_+(x)\}/\tau^n$ for sufficiently large n; hence $\widetilde{q}(x) = 0$.

(c) If $\hat{d}(x) < 0$, then $d_n(x) < b_+(x)$ and hence $q_n(x) = d_n(x)$ for sufficiently large n. Therefore, $\widetilde{q}(x) = \hat{d}(x)$ holds.

Case 3 can be summarized as

$$\widetilde{q}(x) = \lim_{n\to\infty} \frac{q_n(x) - q_0(x)}{\tau_n} = \min\{0, \hat{d}(x)\}.$$

4. For almost every $x \in \omega_0^-$, we obtain, similarly to Case 2,

$$\widetilde{q}(x) = \lim_{n\to\infty} \frac{q_n(x) - q_0(x)}{\tau_n} = 0.$$

5. For almost every $x \in \omega^- \setminus \omega_0^-$, we obtain, similarly to Case 3,

$$\widetilde{q}(x) = \lim_{n\to\infty} \frac{q_n(x) - q_0(x)}{\tau_n} = \max\{0, \hat{d}(x)\}.$$

Summarizing all previous cases, we have shown that

$$\widetilde{q}(x) = \Pi_{[\hat{b}_-(x), \hat{b}_+(x)]}(\hat{d}(x)). \tag{3.32}$$

We proceed by showing that

$$\frac{q_n - q_0}{\tau_n} \to \widetilde{q} \text{ strongly in } \mathcal{Q} = L^2(\omega). \tag{3.33}$$

From the Lipschitz continuity of the projection Π, it follows that

$$\begin{aligned}\left\| \frac{q_n - q_0}{\tau_n} - \widetilde{q} \right\|_{\mathcal{Q}} &= \left\| \frac{1}{\tau_n} \left(\Pi_{\mathcal{Q}_{ad}}(d_n) - \Pi_{\mathcal{Q}_{ad}}(d_0)\right) - \Pi_{\widehat{\mathcal{Q}}_{ad}}(\hat{d}) \right\|_{\mathcal{Q}} \\ &\le \left\| \frac{d_n - d_0}{\tau_n} \right\|_{\mathcal{Q}} + \|\hat{d}\|_{\mathcal{Q}} \to 2\|\hat{d}\|_{\mathcal{Q}}.\end{aligned}$$

From Lebesgue's dominated convergence theorem, (3.33) follows. Consequently, we have $\widetilde{q} = \hat{q}$. The projection formula (3.32) is equivalent to the variational inequality (3.24). Using (3.18) and (3.20) for (u_n, q_n, z_n) and for (u_0, q_0, z_0), we infer that the weak limit $(\hat{u}, \hat{q}, \hat{z})$ satisfies (3.23) and (3.25). It is readily checked that (3.23)–(3.25) are the first order necessary conditions for (**DQP**$(\delta\varepsilon)$). In view of the second order sufficient optimality conditions (Lemma 2.2), (**DQP**$(\delta\varepsilon)$) is strictly convex and thus it has a unique solution. In view of Assumption 3.2 and (3.25), we obtain

$$\left\| \frac{u_n - u_0}{\tau_n} - \hat{u} \right\|_{\mathcal{V}} \le C \left\| \frac{q_n - q_0}{\tau_n} - \hat{q} \right\|_{\mathcal{Q}},$$

where C is independent of n. Hence $\hat{u}$ is also the strong limit of the difference quotient in $\mathcal{V}$. The same argument holds for $\hat{z}$. Our whole argument remains valid if in the beginning, we start with an arbitrary subsequence of $\{\tau_n\}$. Since the limit $(\hat{u}, \hat{q}, \hat{z})$ is always the same, the convergence extends to the whole sequence. ∎

From the previous theorem we derive the following important corollary. The proof follows from a direct application of the implicit function theorem for generalized equations; see [6, Theorem 2.4].

COROLLARY 3.13.
$\mathcal{N}(p_0) \subset \mathcal{P}$ p_0 $\mathcal{N}(x_0) \subset \mathcal{X}$ x_0
$(U, Q, Z) : \mathcal{N}(p_0) \to \mathcal{N}(x_0)$
(a) $p \in \mathcal{N}(p_0)$ $(U(p), Q(p), Z(p))$
(2.5)–(2.7) $\mathcal{N}(x_0)$
(b) $(U(p_0), Q(p_0), Z(p_0)) = (u_0, q_0, z_0)$

(c) [illegible] (U, Q, Z) [illegible] p_0 [illegible] $\delta p \in \mathcal{P}$ [illegible] $\varepsilon \mapsto (u^\varepsilon, q^\varepsilon, z^\varepsilon)$ [illegible] $\varepsilon = 0$ [illegible]

$$\delta\varepsilon = \begin{pmatrix} \mathcal{L}''_{up}(x_0, p_0)(\cdot, \delta p) \\ \mathcal{L}''_{qp}(x_0, p_0)(\cdot, \delta p) \\ \mathcal{L}''_{zp}(x_0, p_0)(\cdot, \delta p) \end{pmatrix}, \tag{3.34}$$

[illegible] $(\hat{u}, \hat{q}, \hat{z})$ [illegible] **DQP**$(\delta\varepsilon)$

We remark that computing the sensitivity derivative of (U, Q, Z) for a given direction δp amounts to solving the linear-quadratic optimal control problem **DQP**$(\delta\varepsilon)$ for $\delta\varepsilon$ given by (3.34). Note that this problem, like the original one **OP**(p_0), is subject to pointwise inequality constraints for the control variable. Due to the structure of the admissible set $\widehat{\mathcal{Q}}_{ad}$, the directional derivative of (U, Q, Z) is in general not a linear function of the direction δp, but only positively homogeneous. Note, however, that if the admissible set $\widehat{\mathcal{Q}}_{ad}$ is a linear space (which follows from a condition known as strict complementarity; see below), then the directional derivative becomes a linear function of the direction (i.e., it is the Gateaux differential).

Definition 3.14 (strict complementarity). [illegible] (x_0, p_0)

$$\Big\{ x \in \omega \,:\, q_0(x) \in \{b_-(x), b_+(x)\} \quad \mu_0(x) = 0 \Big\}$$

[illegible]

A consequence of the strict complementarity condition is that the sensitivity derivatives are characterized by a linear system of equations set forth in the following lemma. We recall that $\widetilde{B}$ was defined in (2.19) and that R_I denotes the multiplication of a function in $L^2(\omega)$ with the characteristic function of the inactive set $\omega^I = \{x \in \omega : b_-(x) < q_0(x) < b_+(x)\}$; see section 2.

Lemma 3.15. [illegible] 3.12 [illegible] (x_0, p_0) [illegible] (U, Q, Z) [illegible] linear [illegible]

$$\widetilde{B}(x_0, p_0) \begin{pmatrix} U'(p_0)(\delta p) \\ Q'(p_0)(\delta p) \\ Z'(p_0)(\delta p) \end{pmatrix} = - \begin{pmatrix} \mathcal{L}''_{up}(x_0, p_0)(\cdot, \delta p) \\ R_I \mathcal{L}''_{qp}(x_0, p_0)(\cdot, \delta p) \\ \mathcal{L}''_{zp}(x_0, p_0)(\cdot, \delta p) \end{pmatrix}. \tag{3.35}$$

[illegible] $\widetilde{B}(x_0, p_0) : \mathcal{X} \to \mathcal{X}'$ [illegible]. By virtue of the strict complementarity property, the admissible set $\widehat{\mathcal{Q}}_{ad}$ defined in Theorem 3.12 becomes

$$\widehat{\mathcal{Q}}_{ad} = \Big\{ \hat{q} \in \mathcal{Q} \,:\, \hat{q}(x) = 0, \text{ where } q_0(x) \in \{b_-(x), b_+(x)\} \Big\}.$$

Consequently, the variational inequality (3.24) simplifies to the following equation for $Q'(p_0)(\delta p) \in \widehat{\mathcal{Q}}_{ad}$:

$$\begin{aligned} &\mathcal{L}''_{qu}(x_0, p_0)(\delta q, U'(p_0)(\delta p)) + \mathcal{L}''_{qq}(x_0, p_0)(\delta q, Q'(p_0)(\delta p)) \\ &\qquad + \mathcal{L}''_{qz}(x_0, p_0)(\delta q, Z'(p_0)(\delta p)) = -\mathcal{L}''_{qp}(x_0, p_0)(\delta q, \delta p) \quad \forall \delta q \in \widehat{\mathcal{Q}}_{ad}, \end{aligned}$$

which is equivalent to the middle equation in (3.35). The first and third equation in (3.35) coincide with (3.23) and (3.25), which proves the first claim. From Theorem 3.12 we conclude that $\widetilde{B}(x_0, p_0)$ is bijective. Since it a continuous linear operator from $\mathcal{X} \to \mathcal{X}'$, so is its inverse. ∎

We are now in the position to recall the first and second order sensitivity derivatives of the reduced cost functional $j(p)$; see again [18]. Note that we do not make use of strict complementarity in the following proposition.

PROPOSITION 3.16. [illegible] 3.12 [illegible]

$$j(p) = J(U(p), p) + \frac{\alpha}{2}\|Q(p) - \overline{q}\|_{\mathcal{Q}}^2$$

[illegible] $\mathcal{N}(p_0)$ [illegible] p_0 [illegible] $\delta p \in \mathcal{P}$ [illegible]

$$j'(p)(\delta p) = \mathcal{L}_p'(x_0, p_0)(\delta p). \tag{3.36}$$

[illegible] j [illegible]

$$\begin{aligned} j''(p_0)(\delta p, \widehat{\delta p}) = {} & \mathcal{L}_{up}''(x_0, p_0)(U'(p_0)(\delta p), \widehat{\delta p}) + \mathcal{L}_{qp}''(x_0, p_0)(Q'(p_0)(\delta p), \widehat{\delta p}) \\ & + \mathcal{L}_{zp}''(x_0, p_0)(Z'(p_0)(\delta p), \widehat{\delta p}) + \mathcal{L}_{pp}''(x_0, p_0)(\delta p, \widehat{\delta p}). \end{aligned} \tag{3.37}$$

[illegible]. As in the unconstrained case there holds

$$\begin{aligned} j'(p_0)(\delta p) = {} & \mathcal{L}_u'(x_0, p_0)(U'(p_0)(\delta p)) + \mathcal{L}_q'(x_0, p_0)(Q'(p_0)(\delta p)) \\ & + \mathcal{L}_z'(x_0, p_0)(Z'(p_0)(\delta p)) + \mathcal{L}_p'(x_0, p_0)(\delta p), \end{aligned}$$

and the terms $\mathcal{L}_u'$ and $\mathcal{L}_z'$ vanish. Moreover,

$$\mathcal{L}_q'(x_0, p_0)(Q'(p_0)(\delta p)) = -(\mu_0, Q'(p_0)(\delta p)) = 0$$

since $Q'(p_0)(\delta p)$ is zero on the strongly active set and μ_0 vanishes on its complement. The formula for the second order derivative follows as in Proposition 3.5 by total directional differentiation of the first order formula. ∎

[illegible] 3.17. We note that the expressions for the first and second order derivatives in Proposition 3.16 are the same as in the unconstrained case; see Proposition 3.5.

We now turn to our main result in the control-constrained case, concerning the differentiability and efficient evaluation of the sensitivity derivatives for the reduced quantity of interest (1.4). We recall that in the unconstrained case, we have made use of a duality argument for the efficient computation of the first and second order derivatives; see section 3.2. However, in the presence of control constraints, this technique seems to be applicable only in the case of strict complementarity since otherwise, the derivatives $(U'(p_0)(\delta p), \xi'(p_0)(\delta p), Z'(p_0)(\delta p))$ do not depend linearly on the direction δp. In analogy to (3.11) and (3.35), we define the dual quantities $(\widetilde{v}, \widetilde{r}, \widetilde{y}) \in \mathcal{X}$ by

$$\widetilde{B}(x_0, p_0) \begin{pmatrix} \widetilde{v} \\ \widetilde{r} \\ \widetilde{y} \end{pmatrix} = - \begin{pmatrix} I_u'(q_0, u_0, p_0) \\ R_I I_q'(q_0, u_0, p_0) \\ 0 \end{pmatrix}. \tag{3.38}$$

THEOREM 3.18. [illegible] 3.12 [illegible] $i(p)$ [illegible] p_0 [illegible] (x_0, p_0) [illegible]

p_0 $\delta p \in \mathcal{P}$

$$i'(p_0)(\delta p) = \mathcal{L}''_{up}(x_0,p_0)(\widetilde{v},\delta p) + \mathcal{L}''_{qp}(x_0,p_0)(R_I\widetilde{r},\delta p) + \mathcal{L}''_{zp}(x_0,p_0)(\widetilde{y},\delta p) + I'_p(u_0,q_0,p_0)(\delta p). \tag{3.39}$$

The proof is carried out similarly to the proof of Theorem 3.6 using Lemma 3.15. □

Our next goal is to consider second order derivatives of the reduced quantity of interest. In order to apply the approach used in the unconstrained case, we rely on the existence of second order directional derivatives of (U, Q, Z) at p_0. However, these second order derivatives do not exist without further assumptions, as seen from the following simple consideration: Suppose that near a given reference parameter $p_0 = 0$, the local optimal control is given by $Q(p)(x) = \max\{0, x + p\} \in L^2(\omega)$ for $x \in \omega = (-1, 1)$ and $p \in \mathbb{R}$. (An appropriate optimal control problem (**OP**(p)) can be easily constructed.) Then $Q'(p)(x) = H(x+p)$ (the Heaviside function), which is not directionally differentiable with respect to p and values in $L^2(\omega)$. Note that the point $x = -p$ of discontinuity marks the boundary between the active and inactive sets of (**OP**(p)). Hence we conclude that the reason for the nonexistence of the second order directional derivatives of Q lies in the change of the active set with p.

The preceding argument leads to the following assumption.

ASSUMPTION 3.19. $\mathcal{N}(p_0) \subset \mathcal{P}$ p_0 $p \in \mathcal{N}(p_0)$ $(U(p), Q(p), Z(p))$ (u_0, q_0, z_0)

3.20. The previous assumption seems difficult to satisfy in the general case. However, if the control variable is finite-dimensional and strict complementarity is assumed at the reference solution (u_0, q_0, z_0), then Assumption 3.19 is satisfied since the Lagrange multiplier $\mu(p) = -\mathcal{L}'_q(U(p), Q(p), Z(p), p)$ is continuous with respect to p and has values in $\mathbb{R}^n$.

We now proceed to our main result concerning second order derivatives of the reduced quantity of interest. In the theorem below, we use again () to denote evaluation at the point (x_0, p_0).

THEOREM 3.21. 3.12 3.19 $i(p)$ p_0 δp $\widehat{\delta p}$

$$i''(p_0)(\delta p, \widehat{\delta p}) = \langle \widetilde{v}, \eta\rangle_{\mathcal{V}\times\mathcal{V}'} + \langle \widetilde{r}, \kappa\rangle_{\mathcal{Q}\times\mathcal{Q}'} + \langle \widetilde{y}, \sigma\rangle_{\mathcal{V}\times\mathcal{V}'} + \begin{pmatrix} U'(p_0)(\delta p) \\ Q'(p_0)(\delta p) \\ \delta p \end{pmatrix}^\top \begin{pmatrix} I''_{uu}(q_0,u_0,p_0) & I''_{uq}(q_0,u_0,p_0) & I''_{up}(q_0,u_0,p_0) \\ I''_{qu}(q_0,u_0,p_0) & I''_{qq}(q_0,u_0,p_0) & I''_{qp}(q_0,u_0,p_0) \\ I''_{pu}(q_0,u_0,p_0) & I''_{pq}(q_0,u_0,p_0) & I''_{pp}(q_0,u_0,p_0) \end{pmatrix} \begin{pmatrix} U'(p_0)(\widehat{\delta p}) \\ Q'(p_0)(\widehat{\delta p}) \\ \widehat{\delta p} \end{pmatrix}. \tag{3.40}$$

$(\eta, \kappa, \sigma) \in \mathcal{V}' \times \mathcal{Q}' \times \mathcal{V}'$

$$\begin{pmatrix} \eta \\ \kappa \\ \sigma \end{pmatrix} = \begin{pmatrix} \mathcal{L}'''_{upp}()(\cdot, \delta p, \widehat{\delta p}) \\ \mathcal{L}'''_{qpp}()(\cdot, \delta p, \widehat{\delta p}) \\ \mathcal{L}'''_{zpp}()(\cdot, \delta p, \widehat{\delta p}) \end{pmatrix} \tag{3.41}$$

Evaluation of sensitivity derivatives

1. Evaluate $j'(p_0)\,\delta p$ according to (3.36)
2. Compute the sensitivities $U'(p_0)\,\delta p$, $Q'(p_0)\,\delta p$, and $Z'(p_0)\,\delta p$ from (3.35)
3. Evaluate $j''(p_0)(\delta p, \widehat{\delta p})$ according to (3.37)
4. Compute the dual quantities $(\widetilde{v}, \widetilde{r}, \widetilde{y})$ from (3.38)
5. Evaluate $i'(p_0)\,\delta p$ according to (3.39)
6. Compute the sensitivities $U'(p_0)\,\widehat{\delta p}$, $Q'(p_0)\,\widehat{\delta p}$, and $Z'(p_0)\,\widehat{\delta p}$ from (3.35)
7. Compute the auxiliary quantities (η, κ, σ) from (**??**)
8. Evaluate $i''(p_0)(\delta p, \widehat{\delta p})$ according to (3.40).

ALGORITHM 3.1. *Evaluating the first and second order derivatives of the reduced cost function j and the reduced quantity of interest i.*

$$+\begin{pmatrix} \mathcal{L}'''_{upu}()(\cdot,\delta p, U'(p_0)(\widehat{\delta p})) + \mathcal{L}'''_{upq}()(\cdot,\delta p, Q'(p_0)(\widehat{\delta p})) + \mathcal{L}'''_{upz}()(\cdot,\delta p, Z'(p_0)(\widehat{\delta p})) \\ \mathcal{L}'''_{qpu}()(\cdot,\delta p, U'(p_0)(\widehat{\delta p})) + \mathcal{L}'''_{qpq}()(\cdot,\delta p, Q'(p_0)(\widehat{\delta p})) + \mathcal{L}'''_{qpz}()(\cdot,\delta p, Z'(p_0)(\widehat{\delta p})) \\ \mathcal{L}'''_{zpu}()(\cdot,\delta p, U'(p_0)(\widehat{\delta p})) + \mathcal{L}'''_{zpq}()(\cdot,\delta p, Q'(p_0)(\widehat{\delta p})) \end{pmatrix}$$

$$+\left(\widetilde{B}'_u()(U'(p_0)(\widehat{\delta p})) + \widetilde{B}'_q()(Q'(p_0)(\widehat{\delta p})) + \widetilde{B}'_z()(Z'(p_0)(\widehat{\delta p})) + \widetilde{B}'_p()(\widehat{\delta p})\right)\begin{pmatrix} U'(p_0)(\delta p) \\ Q'(p_0)(\delta p) \\ Z'(p_0)(\delta p) \end{pmatrix}.$$

. The proof uses the same argument as the proof of Theorem 3.6. Note that in view of Assumption 3.19, $\widetilde{B}(U(p), Q(p), Z(p), p)$ is totally directionally differentiable with respect to p at p_0. In the direction $\widehat{\delta p}$, the derivative is

$$\widetilde{B}'_u()(U'(p_0)(\widehat{\delta p})) + \widetilde{B}'_q()(Q'(p_0)(\widehat{\delta p})) + \widetilde{B}'_z()(Z'(p_0)(\widehat{\delta p})) + \widetilde{B}'_p()(\widehat{\delta p}).$$

Due to the constant active sets, these partial derivatives have the following form:

$$\widetilde{B}'_u() = \begin{pmatrix} id & & \\ & R_I & \\ & & id \end{pmatrix} B'_u(x_0, p_0) \begin{pmatrix} id & & \\ & R_I & \\ & & id \end{pmatrix}.$$

In view of the bounded invertibility of $\widetilde{B}(x_0, p_0)$ (see Lemma 3.15), the second order partial derivatives of (U, Q, Z) at p_0 exist by the implicit function theorem. They satisfy the analogue of (3.16). ∎

We conclude this section by stating Algorithm 3.1 which collects the necessary steps to evaluate the first and second order sensitivity derivatives $j'(p_0)\,\delta p$ and $j''(p_0)(\delta p, \widehat{\delta p})$ as well as $i'(p_0)\,\delta p$ and $i''(p_0)(\delta p, \widehat{\delta p})$ for given $\delta p, \widehat{\delta p} \in \mathcal{P}$. We suppose that the original optimization problem (**OP**(p)) has been solved, e.g., by the primal-dual active set approach in section 2, for the nominal parameter p_0. We denote by

$\mathcal{A}_\pm$ and $\mathcal{I}$ the active and inactive sets belonging to the nominal solution (u_0, q_0) and adjoint state z_0. For the definition of $\widetilde{B}(x_0, p_0)$ appearing in (3.35) and (3.38), we refer the reader to (2.19).

4. Numerical examples. In this section we illustrate our approach using two examples from different areas. The first example is concerned with a parameter identification problem for the stationary Navier–Stokes system. No inequality constraints are present in this problem, and first and second order derivatives of the quantity of interest are obtained. In the second example, we consider a control-constrained optimal control problem for an instationary reaction-diffusion system subject to an infinite-dimensional parameter, which demonstrates the full potential of our approach.

4.1. Example 1. In this section we illustrate our approach using as an example a parameter identification flow problem without inequality constraints. We consider the configuration sketched in Figure 4.1.

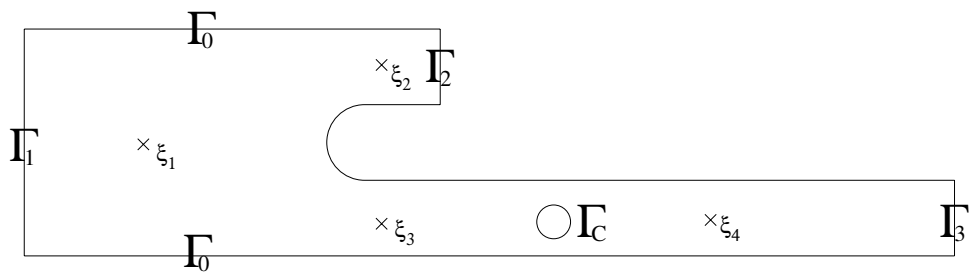


FIG. 4.1. *Configuration of the system of pipes with measurement points.*

The (stationary) flow in this system of pipes around the cylinder Γ_C is described by incompressible Navier–Stokes equations, with unknown viscosity q:

$$
\begin{aligned}
-q\Delta v + v\cdot\nabla v + \nabla p &= f && \text{in} && \Omega,\\
\nabla\cdot v &= 0 && \text{in} && \Omega,\\
v &= 0 && \text{on} && \Gamma_0\cup\Gamma_C,\\
v &= v_{in} && \text{on} && \Gamma_1,\\
q\tfrac{\partial v}{\partial n} - pn &= \pi n && \text{on} && \Gamma_2,\\
q\tfrac{\partial v}{\partial n} - pn &= 0 && \text{on} && \Gamma_3.
\end{aligned}
\tag{4.1}
$$

Here, the state variable $u = (v, p)$ consists of the velocity $v = (v^1, v^2) \in H^1(\Omega)^2$ and the pressure $p \in L^2(\Omega)$. The inflow Dirichlet boundary condition on Γ_1 is given by a parabolic inflow v_{in}. The outflow boundary conditions of the Neumann type are prescribed on Γ_2 and Γ_3 involving the perturbation parameter $\pi \in P = \mathbb{R}$. (Unlike in previous sections, we denote the perturbation parameter by π to avoid the confusion with the pressure p.) Physically, the perturbation parameter π describes the pressure difference between Γ_2 and Γ_3; see [11] for a detailed discussion of this type of outflow boundary condition. The reference parameter chosen is $\pi_0 = 0.029$.

The aim is to estimate the unknown viscosity $q \in Q = \mathbb{R}$ using the measurements of the velocity in four given points; see Figure 4.1. By the least squares approach, this results in the following parameter identification problem:

$$
\text{Minimize } \sum_{i=1}^{4}\sum_{j=1}^{2}(v^j(\xi_i) - \bar{v}_i^j)^2 + \alpha q^2, \quad \text{subject to (4.1).}
$$

Here, $\bar{v}_i^j$ are the measured values of the components of the velocity at the point ξ_i and α is a regularization parameter. For a priori error analysis for finite element discretization of parameter identification problems with pointwise measurements we refer the reader to [20].

The sensitivity analysis of previous sections allows us to study the dependence on the perturbation parameter π. To illustrate this, we define two functionals describing the possible quantities of interest:

$$I_1(u,q) = q, \qquad I_2(u,q) = c_d(u),$$

where $c_d(u)$ is the drag coefficient on the cylinder Γ_C defined as

$$c_d(u) = c_0 \int_{\Gamma_C} n \cdot \sigma \cdot d \, ds, \tag{4.2}$$

with a chosen direction $d = (1,0)$, a given constant c_0, and the stress tensor σ given by

$$\sigma = \frac{\nu}{2}(\nabla v + (\nabla v)^T) - pI.$$

For the discretization of the state equation we use conforming finite elements on a shape-regular quadrilateral mesh $\mathcal{T}_h$. The trial and test spaces consist of cellwise bilinear shape-functions for both pressure and velocities. We add further terms to the finite element formulation in order to obtain a stable formulation with respect to both the pressure-velocity coupling and convection dominated flow. This type of stabilization technique is based on local projections of the pressure first introduced in [1]. The resulting parameter identification problem is solved by Newton's method on the parameter space as described in [3] which is known to be mesh-independent. The nonlinear state equation is likewise solved by Newton's method, whereas the linear subproblems are computed using a standard multigrid algorithm. With these ingredients, the total numerical cost for the solution of this parameter identification problem on a given mesh behaves like $\mathcal{O}(N)$, where N is the number of degrees of freedom (dofs) for the state equation.

For the reduced quantities of interest $i_1(\pi)$ and $i_2(\pi)$ we compute the first and second derivatives using the representations from Theorem 3.6. In Tables 4.1 and 4.2 we collect the values of these derivatives for a sequence of uniformly refined meshes.

In order to verify the computed sensitivity derivatives, we make a comparison with the derivatives computed by the second order difference quotients. To this end we choose $\varepsilon = 10^{-4}$ and compute

$$di_l = \frac{i_l(\pi_0+\varepsilon) - i_l(\pi_0-\varepsilon)}{2\,\varepsilon}, \quad ddi_l = \frac{i_l(\pi_0+\varepsilon) - 2i_l(\pi_0) + i_l(\pi_0-\varepsilon)}{\varepsilon^2}$$

by solving the optimization problem additionally for $\pi = \pi_0 - \varepsilon$ and $\pi = \pi_0 + \varepsilon$. The results are shown in Table 4.3.

Example 4.1. The relative errors in Table 4.3 are of the order of the estimated finite difference truncation error. We therefore consider the correctness of our method to have been verified to within the accuracy of this test. The same holds for Example 2 and Table 4.4 below.

TABLE 4.1
The values of $i_1(\pi)$ and its derivatives on a sequence of uniformly refined meshes.

Cells	dofs	$i_1(\pi)$	$i_1'(\pi)$	$i_1''(\pi)$
60	270	1.0176e–2	–3.9712e–1	1.4065e–1
240	900	1.0086e–2	–3.9386e–1	–3.2022e–1
960	3240	1.0013e–2	–3.9613e–1	–8.5278e–1
3840	12240	1.0003e–2	–3.9940e–1	–1.0168e–0
15360	47520	1.0000e–2	–4.0030e–1	–1.0601e–0

TABLE 4.2
The values of $i_2(\pi)$ and its derivatives on a sequence of uniformly refined meshes.

Cells	dofs	$i_2(\pi)$	$i_2'(\pi)$	$i_2''(\pi)$
60	270	3.9511e–1	–13.4846	9.89988
240	900	3.9106e–1	–13.8759	–4.09824
960	3240	3.9293e–1	–13.8151	16.5239
3840	12240	3.9242e–1	–13.7357	19.3916
15360	47520	3.9235e–1	–13.7144	19.9385

TABLE 4.3
Comparison of the computed derivatives of i_l ($l = 1, 2$) with difference quotients, on the finest grid.

l	i_l'	di_l	$\frac{\mathbf{di_l - i_l'}}{\mathbf{i_l'}}$	i_l''	ddi_l	$\frac{\mathbf{ddi_l - i_l''}}{\mathbf{i_l''}}$
1	–0.399403	–0.399404	**2.5e–6**	–1.01676	–1.01678	**2.0e–5**
2	–13.73574	–13.73573	**–7.3e–7**	19.3916	19.3917	**5.2e–6**

4.2. Example 2. The second example concerns a control-constrained optimal control problem for an instationary reaction-diffusion model in three spatial dimensions. As the problem setup was described in detail in [9], we will be brief here. The reaction-diffusion state equation is given by

$$(c_1)_t = D_1 \Delta c_1 - k_1 c_1 c_2 \quad \text{in } \Omega \times (0, T), \tag{4.3a}$$

$$(c_2)_t = D_2 \Delta c_2 - k_2 c_1 c_2 \quad \text{in } \Omega \times (0, T), \tag{4.3b}$$

where c_i denotes the concentration of the ith substance, and hence $u = (c_1, c_2)$ is the state variable. Ω is a domain in $\mathbb{R}^3$, in this case an annular cylinder (Figure 4.2), and T is the given final time. The control q enters through the inhomogeneous boundary conditions

$$D_1 \frac{\partial c_1}{\partial n} = 0 \quad \text{in } \partial\Omega \times (0, T), \tag{4.4a}$$

$$D_2 \frac{\partial c_2}{\partial n} = q(t)\, \alpha(t, x) \quad \text{in } \partial\Omega_c \times (0, T), \tag{4.4b}$$

$$D_2 \frac{\partial c_2}{\partial n} = 0 \quad \text{in } (\partial\Omega \setminus \partial\Omega_c) \times (0, T), \tag{4.4c}$$

and α is a given shape function on the boundary, modeling a revolving nozzle on the control surface $\partial\Omega_c$, the upper annulus. Initial conditions

$$c_1(0, x) = c_{10}(x) \quad \text{in } \Omega, \tag{4.5a}$$

$$c_2(0, x) = c_{20}(x) \quad \text{in } \Omega \tag{4.5b}$$

are also given. The objective to be minimized is

$$J(c_1, c_2, q) = \frac{1}{2} \int_\Omega \alpha_1 \, |c_1(T, \cdot) - c_{1T}|^2 + \alpha_2 \, |c_2(T, \cdot) - c_{2T}|^2 \, \mathrm{d}x + \frac{\gamma}{2} \int_0^T |q - q_d|^2 \, \mathrm{d}t$$
$$+ \frac{1}{\varepsilon} \max \left\{ 0, \int_0^T q(t) \, \mathrm{d}t - q_c \right\}^3 ;$$

i.e., it contains contributions from deviation of the concentrations at the given terminal time T from the desired ones c_{iT}, plus control cost and a term stemming from a penalization of excessive total control action. We consider here the particular setup described in [9, Example 1], where substance c_1 is to be driven to zero at time T (i.e., we have $\alpha_1 = 1$ and $\alpha_2 = 0$) from given uniform initial state $c_{10} \equiv 1$. This problem features a number of parameters, and differentiability of optimal solutions with respect to these parameters was proved in [10]; hence, we may apply the results of section 3. The nominal as well as the sensitivity and dual problems were solved using a primal-dual active set strategy; see [9, 15]. The nominal control is depicted in Figure 4.2. One clearly sees that the upper and lower bounds with values 5 and 1, respectively, are active in the beginning and end of the time interval. All computations were carried out using piecewise linear finite elements on a tetrahedral grid with roughly 3300 vertices, 13200 tetrahedra, and 100 time steps.

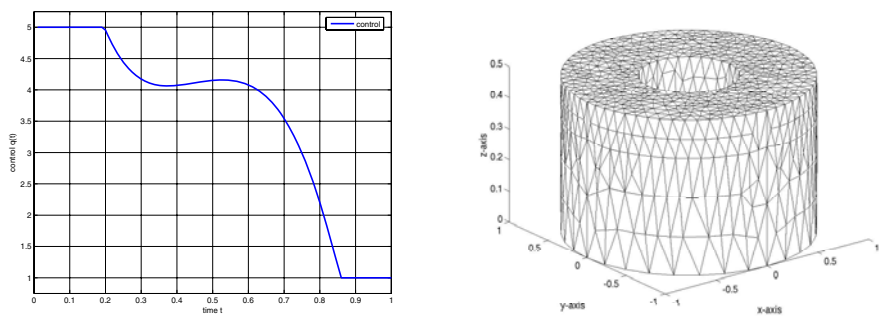


FIG. 4.2. *Optimal (unperturbed) control q (left) and computational domain (right).*

Since the control variable is infinite-dimensional and control constraints are active in the solution, the active sets will in general change even under arbitrarily small perturbations; hence second order derivatives of the reduced quantity of interest $i(p)$ may not exist (see the discussion before Asumption 3.19).

We choose as quantity of interest the total amount of control action

$$I(u, q) = \int_0^T q(t) \, \mathrm{d}t.$$

In contrast to the previous example, we consider now an infinite-dimensional parameter $p = c_{10}$, the initial value of the first substance. After discretization on the given spatial grid, the parameter space has a dimension $\dim P \approx 3300$. A look at Table 3.1 now reveals the potential of our method: The direct evaluation of the derivative $i'(p_0)$ would have required the solution of 3300 auxiliary linear-quadratic problems, an unbearable effort. By our dual approach, however, we need to solve only one additional such problem (3.38) for the dual quantities. The derivative $i'(p_0)$ is shown

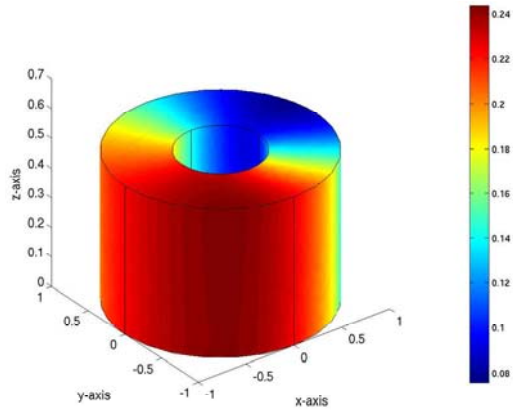


FIG. 4.3. *Gradient of the quantity of interest.*

in Figure 4.3 as a distributed function on Ω. In the unperturbed setup, the desired terminal state $c_1(T)$ is everywhere above the desired state $c_{1T} \equiv 0$. By increasing the value of the initial state c_{10}, the desired terminal state is even more difficult to reach, which leads to an increased control effort and thus an increased value of the quantity of interest. This is reflected by the sign of the function in Figure 4.3, which is everywhere positive. Moreover, one can identify the region of Ω where perturbations in the initial state have the greatest impact on the value of the quantity of interest.

In order to check the derivative, we use again a comparison with a difference quotient in the given direction of $\delta p \equiv 1$. Table 4.4 shows the analogue of Table 4.3 with $\varepsilon = 10^{-2}$ for this example.

TABLE 4.4
Comparison of the computed derivatives of i with difference quotients.

i'	di	$\frac{\mathbf{di}-\mathbf{i}'}{\mathbf{i}'}$
0.222770	0.222463	**−1.4e−3**

5. Conclusion. In this paper, we considered PDE-constrained optimization problems with inequality constraints, which depend on a perturbation parameter p. The differentiability of optimal solutions with respect to this parameter is shown in Theorem 3.12. This result complements previous findings in [7, 18] and makes precise the compactness assumptions needed for the proof.

We obtained sensitivity results for a quantity of interest which depends on the optimal solution and is different from the cost functional. The main contribution of this paper is to devise an efficient algorithm to evaluate these sensitivity derivatives. Using a duality technique, we showed that the numerical cost of evaluating the gradient or the Hessian of the quantity of interest is only marginally higher than the evaluation of the gradient or the Hessian of the cost functional. The small additional effort is spent for the solution of one additional linear-quadratic optimization problem for a suitable dual quantity. A comparison with a direct approach for the evaluation of the gradient and the Hessian revealed the tremendous savings of the dual approach especially in the case of a high-dimenensional parameter space. Two numerical examples confirmed the correctness of our derivative formulae and illustrated the applicability

of our results.

REFERENCES

[1] R. Becker and M. Braack, *A finite element pressure gradient stabilization for the Stokes equations based on local projections*, Calcolo, 38 (2001), pp. 173–199.

[2] R. Becker, D. Meidner, and B. Vexler, *Efficient numerical solution of parabolic optimization problems by finite element methods*, Optim. Methods Softw., in revision.

[3] R. Becker and B. Vexler, *Mesh refinement and numerical sensitivity analysis for parameter calibration of partial differential equations*, J. Comput. Phys., 206 (2005), pp. 95–110.

[4] M. Bergounioux, K. Ito, and K. Kunisch, *Primal-dual strategy for constrained optimal control problems*, SIAM J. Control Optim., 37 (1999), pp. 1176–1194.

[5] J. Dieudonné, *Foundations of Modern Analysis*, Academic Press, New York, 1969.

[6] A. Dontchev, *Implicit function theorems for generalized equations*, Math. Programming, 70 (1995), pp. 91–106.

[7] R. Griesse, *Parametric sensitivity analysis in optimal control of a reaction-diffusion system—part* I: *Solution differentiability*, Numer. Funct. Anal. Optim., 25 (2004), pp. 93–117.

[8] R. Griesse, *Parametric sensitivity analysis in optimal control of a reaction-diffusion system—part* II: *Practical methods and examples*, Optim. Methods Softw., 19 (2004), pp. 217–242.

[9] R. Griesse and S. Volkwein, *A primal-dual active set strategy for optimal boundary control of a nonlinear reaction-diffusion system*, SIAM J. Control Optim., 44 (2005), pp. 467–494.

[10] R. Griesse and S. Volkwein, *Parametric sensitivity analysis for optimal boundary control of a* 3*D reaction-diffusion system*, in Large-Scale Nonlinear Optimization, G. D. Pillo and M. Roma, eds., Nonconvex Optim. Appl. 83, Springer-Verlag, New York, 2006, pp. 127–149.

[11] J. Heywood, R. Rannacher, and S. Turek, *Artificial boundaries and flux and pressure conditions for the incompressible Navier–Stokes equations*, Internat. J. Numer. Methods Fluids, 22 (1996), pp. 325–352.

[12] M. Hintermüller, K. Ito, and K. Kunisch, *The primal-dual active set strategy as a semismooth Newton method*, SIAM J. Optim., 13 (2003), pp. 865–888.

[13] M. Hinze and K. Kunisch, *Second order methods for optimal control of time-dependent fluid flow*, SIAM J. Control Optim., 40 (2001), pp. 925–946.

[14] K. Ito and K. Kunisch, *Augmented Lagrangian-SQP-methods in Hilbert spaces and application to control in the coefficients problem*, SIAM J. Optim., 6 (1996), pp. 96–125.

[15] K. Ito and K. Kunisch, *The primal-dual active set method for nonlinear optimal control problems with bilateral constraints*, SIAM J. Control Optim., 43 (2004), pp. 357–376.

[16] K. Kunisch and A. Rösch, *Primal-dual active set strategy for a general class of constrained optimal control problems*, SIAM J. Optim., 13 (2002), pp. 321–334.

[17] F.-S. Kupfer, *An infinite-dimensional convergence theory for reduced SQP methods in Hilbert space*, SIAM J. Optim., 6 (1996), pp. 126–163.

[18] K. Malanowski, *Sensitivity analysis for parametric optimal control of semilinear parabolic equations*, J. Convex Anal., 9 (2002), pp. 543–561.

[19] H. Maurer and J. Zowe, *First and second order necessary and sufficient optimality conditions for infinite-dimensional programming problems*, Math. Programming, 16 (1979), pp. 98–110.

[20] R. Rannacher and B. Vexler, *A priori error estimates for the finite element discretization of elliptic parameter identification problems with pointwise measurements*, SIAM J. Control Optim., 44 (2005), pp. 1844–1863.

[21] F. Tröltzsch, *Lipschitz stability of solutions of linear-quadratic parabolic control problems with respect to perturbations*, Dyn. Contin. Discrete Impuls. Syst. Ser. A Math. Anal., 7 (2000), pp. 289–306.

SIAM J. SCI. COMPUT.
Vol. 29, No. 1, pp. 49–69
© 2007 Society for Industrial and Applied Mathematics

HIERARCHICAL NONLINEAR APPROXIMATION FOR EXPERIMENTAL DESIGN AND STATISTICAL DATA FITTING*

DANIEL BUSBY†, CHRIS L. FARMER‡, AND ARMIN ISKE§

Abstract. This paper proposes a hierarchical nonlinear approximation scheme for scalar-valued multivariate functions, where the main objective is to obtain an accurate approximation with using only very few function evaluations. To this end, our iterative method combines at any refinement step the selection of suitable evaluation points with kriging, a standard method for statistical data analysis. Particular improvements over previous nonhierarchical methods are mainly concerning the construction of new evaluation points at run time. In this construction process, referred to as experimental design, a flexible two-stage method is employed, where adaptive domain refinement is combined with sequential experimental design. The hierarchical method is applied to statistical data analysis, where the data is generated by a very complex and computationally expensive computer model, called a simulator. In this application, a fast and accurate statistical approximation, called an emulator, is required as a cheap surrogate of the expensive simulator. The construction of the emulator relies on computer experiments using a very small set of carefully selected input configurations for the simulator runs. The hierarchical method proposed in this paper is, for various analyzed models from reservoir forecasting, more efficient than existing standard methods. This is supported by numerical results, which show that our hierarchical method is, at comparable computational costs, up to ten times more accurate than traditional nonhierarchical methods, as utilized in commercial software relying on the response surface methodology (RSM).

Key words. nonlinear approximation, hierarchical approximation, experimental design, statistical data fitting, computer experiments, data-adaptive modeling, kriging, reservoir forecasting

AMS subject classifications. 62K20, 62P30, 65D15

DOI. 10.1137/050639983

1. Introduction.

Approximation of scalar-valued multivariate functions from scattered data is of primary importance in many applications from computational science and engineering. To describe the mathematical problem, let $f : \Omega \to \mathbb{R}$, for some domain $\Omega \subset \mathbb{R}^d$, denote an unknown multivariate function, whose scalar function values in $f\big|_X = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n))^T \in \mathbb{R}^n$, taken at a scattered set $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \Omega$ of pairwise distinct points, are given. Scattered data approximation requires the construction of a suitable function $s : \mathbb{R}^d \to \mathbb{R}$, which is [illegible] to f in a sense to be described. Usually, the quality of the approximation s is, for a fixed normed linear space $\mathcal{V}$ containing f and s, i.e., $f, s \in \mathcal{V}$, measured by the approximation error

$$\eta_{\mathcal{V}}(f, s) = \|f - s\|_{\mathcal{V}}.$$

*Received by the editors September 9, 2005; accepted for publication (in revised form) July 24, 2006; published electronically January 8, 2007.

http://www.siam.org/journals/sisc/29-1/63998.html

†IFP (Institut Français du Pétrole), Rueil-Malmaison, 92500, France (Daniel.BUSBY@ifp.fr). This author was supported by Schlumberger Abingdon Technology Center and by the European Union through the project FAMOUS (Fast Model Utilization and Screening), contract ENK6-CT-2002-50528.

‡Schlumberger Abingdon Technology Center, Abingdon OX14 1UJ, UK, and Oxford Centre for Industrial and Applied Mathematics, University of Oxford, Oxford OX1 3LB, UK (farmer5@slb.com). This author thanks the Royal Society for support through an Industry Fellowship at the University of Oxford.

§Department of Mathematics, University of Hamburg, Bundesstr. 55, D-20146 Hamburg, Germany (iske@math.uni-hamburg.de).

The construction of an approximation to f is usually accomplished by first specifying a suitable finite-dimensional approximation space $\mathcal{S} \subset \mathcal{V}$, before a $s^* \in \mathcal{S}$ to f is computed by minimizing the distance between f and $\mathcal{S}$, so that s^* satisfies

$$\min_{s \in \mathcal{S}} \|f - s\|_{\mathcal{V}} = \|f - s^*\|_{\mathcal{V}}.$$

In the above-mentioned approximation problem, the beforehand construction of a approximation space $\mathcal{S}$ usually depends on X, and so the resulting approximation scheme is . In such a setting, the sample points in X and the function values $f|_X$ are required to be .

We remark that there are only a few powerful approximation methods available for solving such linear scattered data approximation problems for multivariate data, where radial basis function methods are among the most popular tools. For a comprehensive treatment of scattered data approximation and radial basis function methods, we refer the reader to the textbooks [1, 10, 30], where in particular relevant aspects concerning numerical stability, approximation orders for suitable (native) functions spaces, $\mathcal{V}$, and the construction of best approximations are explained in great detail.

In this paper we wish to propose a approximation scheme for data fitting, where the nonlinearity is due to the variation of the sample points in X. The goal of our resulting computational method is to obtain a sufficiently accurate approximation to f by using as few as possible sample points in X; i.e., n is required to be very small. Moreover, we assume that the dimension of the input variables is very large, i.e., $d \gg 1$, and the input data are assumed to be . The latter essentially requires suitable methods for data fitting, i.e., statistical approximation schemes which are relying on suitable statistical models.

To this end, we prefer to work with , a well-established statistical method first introduced by Krige in the 1950's, then formalized as a fundamental geostatistical technique by Matheron [16] in 1963, and later utilized by Sacks, Welch, Mitchell, and Wynn [24] in computer experiments. We remark that kriging is under specific conditions equivalent to Bayesian methods (see [6] for details), and also to radial basis function interpolation (as recently shown in [8, 31]).

By combining kriging with adaptive domain decomposition, the proposed method of this paper results in a hierarchical nonlinear approximation scheme, where the nonlinearity is due to the adaptive selection of suitable sample points X at the algorithm's run time.

The motivation for this work is mainly given by specific applications from experimental design and statistical data fitting in computer experiments. In the general setting of this wide application field, many scientific phenomena are investigated by using complex computer models or codes. In oil reservoir forecasting, for instance, are used to model the evolution of time-dependent physical quantities in a hydrocarbon reservoir production scenario, where long time scales are common; see [11].

The simulation codes are often computationally expensive to run, which requires cheaper predictors, , as surrogates for the too expensive simulators. The construction of an emulator relies on a sequence of computer experiments, involving a number of costly simulator runs with various different input configurations, where the overall aim is to obtain a sufficiently accurate emulator at an as small as possible number of simulator runs. This requires a careful selection of input configurations for the simulator runs.

In such and similar application scenarios for computer experiments, the simulator output is [illegible]; i.e., rerunning the code with the same inputs gives identical observations, but major computational difficulties are often due to [illegible] and [illegible] input data. This requires customized methods for the construction of a sufficiently accurate emulator, in particular as experimenters are interested in [illegible] of the simulation output, [illegible] of the simulator by using external data from the physical system, such as in [illegible], and the optimization of adjustable input parameters.

In the mathematical formulation of this problem, the simulator is viewed as a multivariate target function $f : \mathbb{R}^d \to \mathbb{R}$, the emulator is an approximation $s : \mathbb{R}^d \to \mathbb{R}$ to f, and the selected input configurations, often referred to as [illegible], are the sample points in X.

In general, the construction of an efficient emulator involves the following steps.

1. [illegible]. The goal is to determine configurations of input parameters for running the simulator to build a sufficiently accurate emulator with an as small as possible number of simulator runs.
2. [illegible]. The aim is to reduce the dimensionality of the problem under investigation. This is usually done in a preprocessing step by performing a sensitivity analysis with the available data, in order to discard irrelevant parameters which were initially included in the model. Therefore, screening can also be viewed as an a priori model selection problem, where the [illegible] and [illegible] input parameters are determined.
3. [illegible]. Model parameters are tuned for the selection of an (optimal) emulator which is used to make accurate predictions on the response at untried inputs. This is usually done by optimizing a functional of the response. A common approach for analyzing statistical data is [illegible], as explained in section 3.

The proposed hierarchical approximation scheme aims at the efficient construction of an accurate emulator. This is accomplished by an iterative refinement strategy where the available data collected so far is used to apply at each iteration step an adaptive two-stage process for the experimental design, followed by a statistical data analysis to optimize the parameters of the emulator. Following standard concepts [18] concerning screening, we refrained from including screening in this hierarchical refinement process, although this may be an option in situations where the dimension d of the input data is extremely high, so that further eliminations of input variables are inevitably needed to reduce the required computational costs. A formal description of the proposed hierarchical approximation scheme is given by Algorithm 1.

Our nonlinear approximation scheme results in a multiresolution representation of the emulator, from coarse to fine. That is, with increasing accuracy at each iteration, to obtain a sufficiently accurate emulator at the finest level after merely a few iterations. In this way, the utilized refinement strategy allows control of the gradually increasing computational costs for building the hierarchical emulator sequence, and their increasing accuracy.

We remark that the proposed approach in this paper uses ideas from previous work [5, 6, 7, 13, 24, 26, 27, 29] concerning experimental (sequential) design, sensitivity analysis, and interpolation schemes in nonhierarchical methods.

Our approach aims at two important objectives in experimental design, that is, (1) reducing the uncertainty predicted by the model and (2) improving the model reliability. The first objective is addressed by entropy minimization, in combination

with a customized adaptive gridding algorithm. The second objective is achieved by adaptive cross validation, with acquiring further observations from input configurations where the model prediction is bad ( ). Although both kinds of adaptivity look similar at first sight, these two approaches are fundamentally different from a mathematical and conceptual point of view.

The outline of the paper is as follows. In section 2, the basic ideas of the hierarchical nonlinear approximation scheme are explained, before the kriging method is discussed in section 3. In section 4, the construction of an initial design is explained and some selected aspects concerning screening are briefly addressed. Sections 5 and 6 are devoted to details concerning adaptivity in experimental design, which works with a two-stage process. In section 5, we explain the global approach of adaptive gridding, before turning to local optimal design in section 6. A schematic description of the final algorithm is given in section 7. Finally, section 8 presents numerical results obtained by applying our method to a given function, being regarded as a simulator, and to two different reservoir forecasting problems. It is shown that our hierarchical scheme is for the analyzed reservoir models more efficient than existing methodologies. Indeed, our approximation method is, at comparable computational costs, up to ten times more accurate than the quadratic method in reservoir uncertainty analysis software, such as COUGAR [4], which relies on the response surface methodology (RSM).

2. Hierarchical nonlinear approximation. The proposed hierarchical approximation scheme constructs a sequence $s_1, s_2, \ldots, s_L$ of approximations to an unknown function $f : \mathbb{R}^d \to \mathbb{R}$ from samples of f taken at scattered locations $X = \{x_1, \ldots, x_m\}$. The construction of the approximation relies on a data hierarchy

$$X_1 \subset X_2 \subset \cdots \subset X_L \subset X \tag{2.1}$$

of nested subsets of X, where L denotes the number of levels. Then, the functions s_ℓ approximate f at the subsets X_ℓ of the level ℓ, $1 \le \ell \le L$, according to some specific approximation scheme. Examples for recent approximation schemes are discussed in [10]. Note that the sequence $s_1, s_2, \ldots, s_L$ of approximations to f is from coarse to fine. Indeed, the coarsest approximation s_1 is computed on the basis of the initial design X_1, which includes only very few data. In contrast, the approximations s_ℓ at finer levels ℓ contain gradually more information from their corresponding designs X_ℓ.

The approach taken in this paper is ; that is, the construction of the data hierarchy in (2.1) is done at run time in the approximation process, rather than in a preprocessing step. This adaptive approach, dating back to [12], allows control of both the cumulative computational cost and the accuracy of the current approximation s_ℓ at any level ℓ.

Recall that in the setting and language of this paper, the sets X_ℓ in (2.1) are constructed, one after the other, by an adaptive experimental design strategy. Moreover, any approximation s_ℓ is viewed as an emulator of the simulator f. Each emulator is computed by statistical data fitting at level ℓ, $1 \le \ell \le L$.

The following algorithm schema outlines our hierarchical approximation method, described in detail in sections 3–6.

ALGORITHM 1 (hierarchical nonlinear approximation).

INPUT: $f : \mathbb{R}^d \to \mathbb{R}$

(1) X_1 f X_1 $\{f(\mathbf{x}) : \mathbf{x} \in X_1\}$

(2)

(3) [illegible] s_1 [illegible]
(4) FOR $\ell = 2, 3, \ldots$ DO
(4a) [illegible] X_ℓ (2.1) [illegible]
(4b) [illegible] f [illegible] $X_\ell \setminus X_{\ell-1}$ [illegible] $\{f(\mathbf{x}) : \mathbf{x} \in X_\ell \setminus X_{\ell-1}\}$
(4c) [illegible] $\{f(\mathbf{x}) : \mathbf{x} \in X_\ell\}$ [illegible] s_ℓ [illegible] ℓ
OUTPUT: [illegible] $s_1, s_2, \ldots, s_L$ [illegible]

Note that the iterative approximation scheme performs a sequential refinement of the (emulator) model. In the following sections, we explain steps (1)–(3), (4a), and (4c) of Algorithm 1 in detail. The construction of the initial design X_1 in step (1) is done by using [illegible] (LHC), as discussed in section 4, where also screening, as required in step (2), is briefly addressed. The two-stage experimental design of step (4a) is explained in sections 5 and 6. The construction of the emulators s_ℓ in steps (3) and (4c) relies on kriging, a standard statistical approach, which for the reader's convenience is reviewed in the following section.

3. Kriging. The approximation process at each level of Algorithm 1 can be divided into two main steps: In the first step, a statistical model $S(\mathbf{x})$ for the given data (obtained from the simulator) is derived. In the second step, a deterministic predictor for the statistical model—the emulator—is computed. Therefore, the emulator can be viewed as a statistical approximation, and so the proposed method, Algorithm 1, provides a suitable tool for statistical data fitting.

The construction of the individual emulators s_ℓ, $\ell = 1, \ldots, L$, in Algorithm 1 uses [illegible]. Kriging, due to Matheron [16], is a statistical method for scattered data interpolation which can be used for prediction in computer experiments [24, 29]. To explain kriging, denote, at any fixed level $1 \le \ell \le L$, the emulator by $s \equiv s_\ell$ and let $X \equiv X_\ell$ be the corresponding design at level ℓ.

In the underlying statistical model of kriging, the deterministic response, say, $f(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}$, is considered as a realization of a random function. Therefore, the simulator f is treated as a realization of a stochastic process, whose form is assumed as

$$S(\mathbf{x}) = \sum_{j=1}^{k} \beta_j h_j(\mathbf{x}) + Z(\mathbf{x}). \tag{3.1}$$

The regression part in this stochastic model (3.1) is a linear combination of preselected real-valued functions $h_1, \ldots, h_k$, with coefficients $\beta = (\beta_1, \ldots, \beta_k)^T \in \mathbb{R}^k$, where h_1 is a constant. Moreover, Z in (3.1) is assumed to be a Gaussian random process with mean zero and covariance

$$\operatorname{cov}[\mathbf{x}, \mathbf{y}] = \mathrm{E}[Z(\mathbf{x})Z(\mathbf{y})] = \sigma^2 R(\mathbf{x}, \mathbf{y}) \tag{3.2}$$

between $Z(\mathbf{x})$ and $Z(\mathbf{y})$, where σ^2 denotes the process variance, $R(\mathbf{x}, \mathbf{y})$ is a specific correlation function, and the symbol E denotes the usual statistical expectation. The choice of σ and R is discussed later in this section.

In a Bayesian approach, the random process $S(\mathbf{x})$ is regarded as a [illegible] model and the response values $\{f(\mathbf{x}) : \mathbf{x} \in X\}$ are viewed as observations. It remains to determine a suitable [illegible] predictor—the emulator—from the given set of observations.

In kriging, for given design $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subset \mathbb{R}^d$ and for given responses $\mathbf{f}_X = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_m))^T \in \mathbb{R}^m$, a linear predictor

$$\hat{S}(\mathbf{x}) = \mathbf{a}^T(\mathbf{x})\mathbf{f}_X$$

of $f(\mathbf{x}) \in \mathbb{R}$ at an untried $\mathbf{x} \in \mathbb{R}^d$ is assumed (being the required statistical model for the emulator $s(\mathbf{x}) \equiv \hat{S}(\mathbf{x})$), where the entries in the coefficient vector $\mathbf{a}(\mathbf{x}) = (a_1(\mathbf{x}), \ldots, a_m(\mathbf{x}))^T \in \mathbb{R}^d$ are unknown. The [illegible] for the deterministic response $f(\mathbf{x})$ is then obtained by minimizing the [illegible]

$$\mathrm{MSE}[\hat{S}(\mathbf{x})] = \mathrm{E}[\mathbf{a}^T(\mathbf{x})\mathbf{f}_X - S(\mathbf{x})]^2 \tag{3.3}$$

under variation of $\mathbf{a}(\mathbf{x})$ and subject to the [illegible]

$$\mathrm{E}[\mathbf{a}^T(\mathbf{x})\mathbf{f}_X] = \mathrm{E}[S(\mathbf{x})]. \tag{3.4}$$

Recall that $\mathbf{f}_X$ is regarded as a realization of $S(\mathbf{x})$. Thus, using (3.4), the mean square error in (3.3) can be rewritten as

$$\sigma^2 \left[\mathbf{a}^T(\mathbf{x})\mathbf{R}\mathbf{a}(\mathbf{x}) - 2\mathbf{a}(\mathbf{x})^T\mathbf{r}(\mathbf{x}) + 1\right],$$

and the linear constraints in (3.4) can be rewritten as

$$\mathbf{H}^T\mathbf{a}(\mathbf{x}) = \mathbf{h}(\mathbf{x}),$$

where

$$\begin{aligned}
\mathbf{h}(\mathbf{x}) &= (h_1, h_2(\mathbf{x}), \ldots, h_k(\mathbf{x}))^T \in \mathbb{R}^k, \\
\mathbf{H} &= (h_j(\mathbf{x}_k))_{1\le k\le m; 1\le j\le k} \in \mathbb{R}^{m\times k}, \\
\mathbf{R} &= (R(\mathbf{x}_i, \mathbf{x}_j))_{1\le i,j\le m} \in \mathbb{R}^{m\times m}, \\
\mathbf{r}(\mathbf{x}) &= (R(\mathbf{x}_1, \mathbf{x}), \ldots, R(\mathbf{x}_m, \mathbf{x}))^T \in \mathbb{R}^m.
\end{aligned}$$

Here, $\mathbf{H}$ is the [illegible], $\mathbf{R}$ is the [illegible] between the values of Z at the design sites X, and $\mathbf{r}(\mathbf{x})$ is the correlation vector between Z at the sites X and an untried input $\mathbf{x}$.

Therefore, minimizing (3.3) subject to (3.4) amounts to solving the linear system

$$\begin{bmatrix} \mathbf{R} & \mathbf{H} \\ \mathbf{H}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{a}(\mathbf{x}) \\ \mathbf{b}(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \mathbf{r}(\mathbf{x}) \\ \mathbf{h}(\mathbf{x}) \end{bmatrix}, \tag{3.5}$$

where $\mathbf{b}(\mathbf{x}) = (b_1(\mathbf{x}), \ldots, b_k(\mathbf{x}))^T \in \mathbb{R}^k$ are the Lagrange multipliers for the linear constraints of the quadratic optimization problem.

By solving the linear system (3.5), the best linear unbiased prediction $\hat{S}(\mathbf{x})$ can be written as

$$\hat{S}(\mathbf{x}) = \mathbf{h}^T(\mathbf{x})\hat{\beta} + \mathbf{r}^T(\mathbf{x})\mathbf{R}^{-1}(\mathbf{f}_X - \mathbf{H}\hat{\beta}),$$

where

$$\hat{\beta} = (\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{R}^{-1}\mathbf{f}_X \tag{3.6}$$

is the usual generalized least squares estimate of coefficients $\beta \in \mathbb{R}^k$ in (3.1).

Note that the kriging method not only gives an estimate of the posterior mean $\hat{S}(\mathbf{x})$ of the stochastic process $S(\mathbf{x})$, but it also provides by

$$\mathrm{MSE}[S(\mathbf{x})] = \sigma^2 \left[1 - [\mathbf{r}^T(\mathbf{x}), \mathbf{h}^T(\mathbf{x})] \begin{bmatrix} \mathbf{R} & \mathbf{H} \\ \mathbf{H}^T & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{r}(\mathbf{x}) \\ \mathbf{h}(\mathbf{x}) \end{bmatrix} \right]$$

an estimate of the posterior variance of the random process [24]. For further details, see Matheron [16] or the more recent papers [24, 31]. A different formulation based on a maximum probability interpolant can be found in [8].

Now let us finally turn to the model selection problem, where we need to determine the unknown functions and parameters in (3.1). The selection of the functions $h_2, \ldots, h_k$ of the regression part in the stochastic model (3.1) relies on prior knowledge concerning the model response. For instance, linear or quadratic terms are preferred in situations where there is strong evidence to suggest weakly nonlinear trends in the response, or whenever the number of affordable simulations is particularly small.

When using kriging, in combination with hierarchical adaptive design, it is often sufficient to merely include the constant term h_1 in the polynomial regression (3.1) in order to obtain emulators s_ℓ, each of whose performance (in terms of their prediction quality) is at least as good or even superior to that of comparable one-level emulators. This is supported by our numerical results in section 8.

The selection of the correlation function, R, also requires prior knowledge concerning the stochastic process. To this end, we follow the common approach in [5, 6, 13, 24, 29], where it is suggested to use a correlation function of the form

$$R(\mathbf{x}, \mathbf{y}) = r(\mathbf{x} - \mathbf{y}) = \exp\left(- \sum_{j=1}^{d} \theta_j |x_j - y_j|^p \right) \tag{3.7}$$

with $\theta_j > 0$, $1 \le j \le d$, and $0 < p \le 2$.

Moreover, by following the recommendations in [29], the unknown coefficients $\theta = (\theta_1, \ldots, \theta_d) \in \mathbb{R}^d$ and p in (3.7) are determined by [illegible] (MLE). Likewise, the parameter σ for the process variance in (3.2) and $\beta \in \mathbb{R}^k$ are determined by MLE; see [29] for details. In this case, the log-likelihood for the Gaussian process $S(\mathbf{x})$ is given by

$$\ell(\beta, \sigma, \theta, p) \sim -1/2 \left(m \ln \sigma^2 + \ln \det(\mathbf{R}) + (\mathbf{f}_X - \mathbf{H}\beta)^T \mathbf{R}^{-1} (\mathbf{f}_X - \mathbf{H}\beta)/\sigma^2 \right). \tag{3.8}$$

In order to reduce the required computational complexity, it is reasonable to work with the following simplification. Note that with assuming the parameters θ_j, $1 \le j \le d$, and p as fixed, the MLE for $\beta \in \mathbb{R}^k$ is given by the generalized least squares estimate $\hat{\beta} \in \mathbb{R}^k$ in (3.6). Moreover, the MLE estimate for the process variance σ^2 in (3.2) is given by

$$\hat{\sigma}^2 = \frac{1}{m} (\mathbf{f}_X - \mathbf{H}\hat{\beta})^T \mathbf{R}^{-1} (\mathbf{f}_X - \mathbf{H}\hat{\beta}).$$

Using these estimates $\hat{\beta}$ and $\hat{\sigma}^2$ in the log-likelihood (3.8), the coefficients θ_j, $1 \le j \le d$, and p are then determined by maximizing the simpler expression

$$\ell(\theta, p) \sim -1/2 \left(m \ln \sigma^2 + \ln \det(\mathbf{R}) \right). \tag{3.9}$$

Maximizing the expression (3.9) requires global optimization, which can be computationally too expensive. However, as pointed out in [25], MLE is in practice regarded as sufficiently reliable. But in situations where several regression terms $h_1, \ldots, h_n$ are included in the model (3.1), a [illegible] (REML) is recommended [25]. In this case, parameter estimation by REML is accomplished by maximizing the likelihood for transformed data

$$W = \mathbf{C}\mathbf{f}_X \sim \mathrm{N}[\mathbf{C}\mathbf{H}\beta = 0, \sigma^2 \mathbf{C}\mathbf{R}(\theta)\mathbf{C}^T],$$

with a transformation matrix $\mathbf{C} \in \mathbb{R}^{(m-k)\times m}$ satisfying $\mathbf{C}\mathbf{H} = 0$, and where $\mathrm{N}(\mu, \sigma^2\mathbf{R})$ denotes the usual statistical normal distribution with mean μ and covariance $\sigma^2\mathbf{R}$. Further details on MLE and REML are immaterial for the purposes of this paper, and so we prefer to refer the interested reader to the textbook [25]. We finally remark that within our hierarchical approximation, Algorithm 1, the computational costs required for the performance of MLE and REML are negligible.

4. Initial design and screening. This section first explains in subsection 4.1 the construction of the utilized initial design, before some relevant aspects concerning screening are briefly addressed in subsection 4.2.

4.1. Initial design. To determine the initial design X_1 in Algorithm 1, we prefer to work with LHCs [17]. This standard technique is easy to use and has mainly two advantages. First, with using LHC most regions of the input domain are well covered by a well-balanced distribution of the design sites. Second, if some of the input variables are found inactive, then the initial design can [illegible] be projected onto the restricted space of the remaining active variables. Commonly used alternative designs, such as fractional factorial designs (FFDs) [7], as they are used in RSM [7, 18], usually lack these two important properties, even though they are usually better suited for sensitivity analysis.

LHCs were first introduced by McKay, Beckman, and Conover [17] in computer experiments. Later on, their utility in numerical integration of real-valued functions of many variables were discussed in several papers by Owen [19]. As shown in [19], LHC is closely related to quasi–Monte Carlo methods [3].

For the numerical experiments of section 8, a simplified version of LHC is utilized. To explain this simplified LHC, assume that the range of any input variable is the interval $I = [-1, 1]$. On a preselected number m of input configurations, each of the m input sites in the initial design $X_1 = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ is assumed to have coordinates from

$$I_m = \{-1 + 2i/(m-1) : 0 \le i \le m-1\},$$

where for every input parameter x_i, $1 \le i \le d$, the m corresponding components $\mathbf{x}_1^{(i)}, \ldots, \mathbf{x}_m^{(i)}$ of the sites in X_1 are required to be pairwise distinct. To this end, according to the construction of LHC, the m components $\mathbf{x}_1^{(i)}, \ldots, \mathbf{x}_m^{(i)}$ are chosen from I_m in random order; i.e., for each $1 \le i \le d$, the components $\mathbf{x}_1^{(i)}, \ldots, \mathbf{x}_m^{(i)}$ are a permutation of the m numbers in I_m.

Note that in our formulation of LHC the input coordinates are randomly selected from a [illegible] set, I_m, rather than from a [illegible] parameter interval. The reason for doing so is mainly for the sake of stability. To be more precise, we are aiming at the construction of well-separated design sites. This serves to stabilize the required maximization of matrix determinants by MLE in the subsequent local optimal design. Details on this are explained in section 6.

Moreover, note that our construction guarantees that the data points in the design X_1 are uniquely projectable to low-dimensional point sets; see the second remark at the outset of this section. This is particularly useful for dimension reduction, especially in situations where we expect only a small set of variables from the tried responses. In this case, we want to discard the remaining variables. Using LHC, this can, unlike in FFD, be done unambiguously and without any modeling conflicts by just removing the components of the inactive variables from the design X_1.

As suggested in [29], LHC does also support efficient estimation of the correlation lengths $\lambda_j = 1/\theta_j > 0$ in (3.7). This will be very important in the subsequent steps of our method. As further explained in [29], the correlation lengths λ_j should be used in the screening process. In fact, a very large correlation length λ_j indicates that the response output is not very sensitive to the variation of the input parameter x_j, $1 \le j \le d$.

4.2. Screening. Screening usually works with a linear polynomial model, where a specific statistical test, called the , is applied. The Student test relies on least squares estimates for the coefficients in the linear model, in order to evaluate their contribution to the response variability. These estimates are used in order to recursively eliminate inactive terms from the model. More details on this standard statistical technique can be found in [18] or in the COUGAR [4] user manual.

As supported by our numerical experiments, a screening criterion should be preferred to avoid premature discarding of input variables, as they may become significant at a later stage of the hierarchical approximation algorithm. In our implementation, this is accomplished by combining several independent screening criteria, in which case an input variable will only be discarded if it satisfies screening criteria involved, rather than just . For a more detailed discussion on the related sensitivity analysis from a Bayesian perspective, we refer the reader to the recent work of Oakley and O'Hagan [14].

5. Adaptive gridding. In comparison to the rather easy task of initial design, as described in subsection 4.1, is a much more challenging task, requiring rather sophisticated optimization techniques. Global sequential optimal designs, as suggested in previous papers [5, 6, 24], are commonly used iterative methods for optimal design in computer experiments. In these methods, a sequence of design sites is constructed for the computational domain Ω, which usually involves very expensive computations to determine an design point in each step of the iteration. Due to their large computational costs, the applicability of such methods is rather limited. Moreover, in these global strategies for optimal design, design points often cluster in some region of Ω, whereas they are sparse in other regions of Ω [24]. In fact, sequential optimal designs are often not too efficient, due to a too heterogeneous distribution of the design sites. This section proposes a more efficient approach for hierarchical optimal design, called , as utilized in the design step (4a) of Algorithm 1.

Adaptive gridding is a domain decomposition strategy which splits the computational domain Ω into smaller subdomains, , according to a suitable splitting criterion [12]. In our method, Ω is adaptively partitioned into anisotropic cells of equal size, where the splitting criterion is according to the correlation of the response $f(\mathbf{x})$. By using (3.7), this correlation is chosen as a function $R(\mathbf{x}, \mathbf{y})$ which is rapidly decaying at growing distance between the points $\mathbf{x}$ and $\mathbf{y}$.

Our approach combines global adaptive gridding with local optimal design. To this end, the construction of the design sites is done locally, by applying a local sequential optimal design to the (highly correlated) cells which are output by global adaptive gridding. The correlation is measured by the correlation lengths $\lambda_i = 1/\theta_i$ of the corresponding input parameters x_i, $1 \le i \le d$.

We shall be more precise on this two-stage procedure in this and in the following section. But let us first make a few more general remarks concerning computational aspects. Note that adaptive gridding, in combination with local optimal design, allows us to distribute the required computations to different processors. Therefore, at any design level ℓ, several computations for new design sites and corresponding simulations may be run, one for each cell, at the same time. In relevant applications, such as in oil reservoir forecasting or climate modeling, where each simulation run may take several hours or even several days, parallelization is in fact essential in order to reduce the required computational time.

Now the basic idea of adaptive gridding is to construct a partitioning of Ω into cells of equal size, where the cell edges have different lengths depending on their spatial direction. Moreover, each cell edge is aligned with one coordinate axis x_i, corresponding to one specific input parameter. The aim of adaptive gridding is to perform the splitting, such that the edge lengths are proportional to the [illegible] λ_i of the corresponding input parameter x_i, $1 \le i \le d$.

We further explain adaptive gridding in subsection 5.1, before we turn to the construction of a customized error indicator in subsection 5.2, which allows us to evaluate the prediction quality at each of the individual cells. The latter will establish an important link between global adaptive gridding and local optimal design.

5.1. Adaptive gridding algorithm. To define adaptive gridding, suppose that the domain Ω is a hypercuboid in $\mathbb{R}^d$,

$$\Omega = [x_\ell^{(1)}, x_r^{(1)}] \times \cdots \times [x_\ell^{(d)}, x_r^{(d)}] \subset \mathbb{R}^d,$$

and so Ω is defined by its lower left corner point $\mathbf{x}_\ell = (x_\ell^{(1)}, \ldots, x_\ell^{(d)}) \in \mathbb{R}^d$ and its upper right corner point $\mathbf{x}_r = (x_r^{(1)}, \ldots, x_r^{(d)}) \in \mathbb{R}^d$.

We intend to decompose Ω into a collection $\mathcal{L} = \{\omega\}_{\omega \in \mathcal{L}}$ of smaller hypercuboidal subdomains, [illegible], of equal sizes and with pairwise disjoint interior, so that

$$\Omega = \bigcup_{\omega \in \mathcal{L}} \omega$$

yields a partitioning of Ω.

The decomposition of Ω is computed by splitting the $d \times 2^{d-1}$ edges of Ω. Note that each edge e is aligned with one coordinate axis x_i, $1 \le i \le d$. That is, any edge e in Ω corresponds to one variable x_i, $1 \le i \le d$, and therefore e may be viewed as an interval I_i on coordinate axis x_i. In adaptive gridding, each interval I_i, corresponding to one edge e (and thus to one variable x_i), is uniformly split into n_i pairwise disjoint intervals $I_i^{(1)}, \ldots, I_i^{(n_i)}$, $1 \le i \le d$, of equal length, so that

$$I_i = \bigcup_{j=1}^{n_i} I_i^{(j)}, \qquad \text{with } |I_i^{(j)}| \equiv |I_i|/n_i \quad \text{for } 1 \le j \le n_i,\ 1 \le i \le d.$$

Now the basic idea of adaptive gridding is to perform the splitting of each interval I_i, such that the resulting constant length of the n_i subintervals $|I_i^{(j)}| \equiv |I_i|/n_i$,

$1 \le j \le n_i$, is of the order of the correlation length λ_i of variable x_i; i.e., $\lambda_i \sim |I_i|/n_i$, or $n_i \sim |I_i|/\lambda_i$ for $1 \le i \le d$, so that $\prod_{i=1}^d n_i \sim \prod_{i=1}^d |I_i|/\lambda_i$ is the number of cells.

In this way, each of the resulting cells $\omega \in \mathcal{L}$ has long edges along coordinate directions where the variation of the response f is small, whereas short edges of ω correspond to variables with stronger fluctuation of the response. Therefore, adaptive gridding captures preferred directions of the response f quite effectively. This in turn leads to an adaptive distribution of the design points, according to the global behavior of f.

The adaptive gridding algorithm outputs a uniform partitioning $\mathcal{L} = \{\omega\}_{\omega \in \mathcal{L}}$ of Ω into $\prod_{i=1}^d n_i$ smaller cells of equal size, where each cell $\omega \in \Omega$ is a hypercuboid in $\mathbb{R}^d$. For the illustration, we refer the reader to Figure 8.2 of subsection 8.1, where examples for partitionings of a planar domain $\Omega \subset \mathbb{R}^2$ are shown.

Recall that the adaptive gridding algorithm is performed at each level $\ell \ge 2$ in step (4a) of Algorithm 1. In particular, the correlation lengths $\lambda_i \equiv \lambda_i^{(\ell)}$ are reevaluated at each level $\ell \ge 2$, and so the partitioning $\mathcal{L} = \{\omega\}_{\omega \in \mathcal{L}}$ is updated accordingly.

5.2. Cell prediction errors. Having applied adaptive gridding to obtain a partitioning $\mathcal{L} = \{\omega\}_{\omega \in \mathcal{L}}$ of the domain Ω into smaller cells ω, further design points are to be added to the cells $\omega \in \mathcal{L}$. This is done by following ideas from our previous work [9], where again adaptivity plays an important role.

To this end, we first assign for the current design X_ℓ a prediction error η_ω to each cell ω, by cross validation. More precisely, let $X_\omega = X_\ell \cap \omega$ denote the set of current design sites lying in cell ω, so that by

$$X_\ell = \bigcup_{\omega \in \mathcal{L}} X_\omega$$

this yields a partitioning of X_ℓ.

Note that the set X_ω of design sites contained in a cell ω may be empty, in which case we let $\eta_\omega = \eta_{\max}$ for the prediction error of cell ω, where $\eta_{\max}$ is a (very high) preselected value. Otherwise, if ω is not empty, we work with a cell prediction error of the form

$$\eta_\omega = \max_{\mathbf{x}_\omega \in X_\omega} |f(\mathbf{x}_\omega) - s_{X_\ell \setminus \mathbf{x}_\omega}(\mathbf{x}_\omega)|, \tag{5.1}$$

where $s_{X_\ell \setminus \mathbf{x}_\omega}$ is the kriging emulator (see section 3), whose construction uses [illegible] design sites from X_ℓ except $\mathbf{x}_\omega$. This way, the cell prediction error (5.1) reflects the [illegible] approximation behavior, i.e., the quality of the emulator's local prediction at the cell ω.

Indeed, if η_ω is small, then the response f is usually smooth on ω and the emulator makes good predictions, whereas for large η_ω the response is usually subject to strong variation in ω (e.g., for nonlinear behavior of f), in which case further design points should be added to ω in order to improve the prediction of the emulator.

The cell prediction error is used in combination with a preselected [illegible], $\eta^* \le \eta_{\max}$, where further points are to be added to every cell ω satisfying $\eta_\omega \ge \eta^*$. We call any cell ω, where (by $\eta_\omega \ge \eta^*$) the emulator's prediction is rather poor, a [illegible] cell; otherwise (if $\eta_\omega < \eta^*$) the cell ω is called a [illegible] cell. Moreover, an empty cell is said to be a bad cell, unless the prediction error η_ω of [illegible] nonempty cell ω lies below the global threshold η^*. In this case, there are no bad cells in the partitioning $\mathcal{L} = \{\omega\}_{\omega \in \mathcal{L}}$.

Now the insertion of new design points is done by applying a local optimal design (as explained in section 6) to each bad cell ω, where we add—primarily for the sake of computational efficiency—only [illegible] point per (bad) cell ω. Note that the chosen refinement strategy allows one to combine the construction of new design sites with parallel computing, and so to further reduce the computational time required for the experimental design.

The utilized refinement strategy makes up an important link between global adaptive splitting and local optimal design through this two-stage construction of new design sites. Indeed, our construction reduces the likelihood that some unexplored areas of the computational domain Ω are classified as [illegible] at a too early refinement level by a too optimistic prediction indicator.

Finally let us remark that the target η^* should ideally reflect the desired accuracy threshold of the computer experiment. However, one should note that η^* can only be a rough estimate of the emulator's global prediction error. Nevertheless, as supported by our numerical experiments, the accuracy η^* can in fact be regarded as a suitable estimate for the emulator accuracy. Moreover, the estimate turns out to be useful for subsequent optimization and postprocessing calibration.

In practice, however, the stopping criterion is usually dominated by the maximum number of affordable simulations rather than by the desired problem accuracy. Therefore, any beforehand selection of the target accuracy η^* should not only depend on the required problem accuracy, but also on the available computational power and hardware capacity. Note that by the iterative construction in our hierarchical approximation scheme, the pay-off in terms of number of simulations versus accuracy can be monitored quite effectively during the performance of Algorithm 1.

6. Local optimal design. This section explains how the local optimal design is performed. Recall that this step concerns construction of new design sites, to be added to the current set $X \equiv X_\ell$ in step (4a) of Algorithm 1.

The local design is performed on a cell ω, which is constructed by adaptive gridding, as explained in section 5. The local optimal design is, for any cell ω, done by using an [illegible]. In this approach, dating back to Lindley [15], a design is determined by minimizing the posterior Shannon's entropy [22].

As later shown by Shewry and Wynn [23], this approach is equivalent to maximizing the prior entropy. For the special case of a Gaussian prior, this in turn is equivalent to maximizing the determinant of the covariance matrix $\sigma^2\mathbf{R}$. For a more detailed discussion on this, we refer to [6, 23]. As a result, regarding β as fixed and considering the variance σ^2 as stationary, this amounts to maximizing the determinant $\det(\mathbf{R}_X)$ of the correlation matrix

$$\mathbf{R}_X = (R(\mathbf{x}_i, \mathbf{x}_j))_{1\le i,j\le m} \in \mathbb{R}^{m\times m}$$

under variation of design sites $X = \{\mathbf{x}_1, \dots, \mathbf{x}_m\}$; see [6].

Therefore, we wish to insert one point $\mathbf{x} = \mathbf{x}^*$ into ω, which maximizes the determinant $\det(\mathbf{R}_{X\cup\mathbf{x}})$ of the resulting correlation matrix $\mathbf{R}_{X\cup\mathbf{x}}$ among all points in ω. To determine $\mathbf{x}^* \in \omega$, the search for a maximum of $\det(\mathbf{R}_{X\cup\mathbf{x}})$ on ω is done by varying the position of the point $\mathbf{x} = \mathbf{x}_\omega$ in the cell ω, while keeping the design sites in X fixed. The required numerical optimization relies on the algorithm [illegible], a constrained version of the quasi-Newton method, due to Byrd et al. [2]. In this local optimization algorithm, we apply the L-BFGS-B algorithm with different choices for the initial point $\mathbf{x}_0 \in \omega$, namely, with the center of ω and with a small number $n \ll 2^d$ of randomly chosen points from the 2^d corners of ω. We then take a point

$\mathbf{x}^* \in \omega$, where the maximum of the determinant $\det(\mathbf{R}_{X\cup\mathbf{x}})$ among these $n+1$ points is attained.

In our local optimal design, the points in X are, for each cell $\omega \in \mathcal{L}$, given by the union of the points in X_ω and a [illegible] set of neighbors around ω. Therefore, the size of the matrix $\mathbf{R}_X$ is [illegible]. In our numerical experiments, we considered using a number of n_ω [illegible] neighbors of the center $\mathbf{c}_\omega$, where n_ω is between 15 and 20. In order to determine these nearest neighbors for $\mathbf{c}_\omega \in \omega$, we considered working with the [illegible]

$$\mathbf{d}(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{j=1}^{n}\left(\frac{x_j - y_j}{\lambda_j}\right)^2} \qquad \text{for } \mathbf{x} = (x_1,\dots,x_d),\ \mathbf{y} = (y_1,\dots,y_d),$$

for measuring the distance between two points. This is in order to take different statistical variations for the different input parameters better into account.

Note that this procedure can recursively be applied to any cell ω, just in case one wants to add more than one new design point into ω. This then complies with the ideas of (local) sequential optimal design.

7. Formulation of the hierarchical approximation algorithm. Now that each step of Algorithm 1 has been explained in detail we provide a more precise schematic description in the following algorithm.

ALGORITHM 2 (hierarchical nonlinear approximation).

INPUT: [illegible] $f : \mathbb{R}^d \to \mathbb{R}$

(1) [illegible] X_1

(2) [illegible] θ_j, p [illegible] (3.7) [illegible]

(3) [illegible] f [illegible] X_1 [illegible]
$\mathbf{f}_{X_1} = \{f(\mathbf{x}) : \mathbf{x} \in X_1\}$

(4) [illegible] $\theta_j \sim 0$ [illegible] $1 \le j \le d$ [illegible]

(5) [illegible] s_1 [illegible] X_1 [illegible] $\mathbf{f}_{X_1}$ [illegible]

(6) FOR $\ell = 2, 3, \dots$ DO

(6a) [illegible] θ_j, p [illegible] (3.7) [illegible]

(6b) [illegible] ω [illegible]

(6c) [illegible] η_ω [illegible] ω [illegible]
[illegible] $\eta_\omega \ge \eta^*$

(6d) [illegible] ω [illegible] ω [illegible]
[illegible] ω [illegible] $D_\ell = X_\ell \setminus X_{\ell-1}$

(6e) [illegible] f [illegible] D_ℓ [illegible]
$\mathbf{f}_{D_\ell} = \{f(\mathbf{x}) : \mathbf{x} \in D_\ell\}$

(6f) [illegible] s_ℓ [illegible] X_ℓ [illegible] $\mathbf{f}_{X_\ell}$ [illegible]

OUTPUT: [illegible] $s_1, s_2, \dots, s_L$ [illegible]

8. Numerical results. We have implemented the proposed hierarchical approximation scheme, Algorithm 2, for arbitrary space dimension d by using the programming language R [20]. This section investigates the performance of our algorithm. To this end, we work with three different model problems, one involving an analytical function (subsection 8.1), and two model problems from reservoir forecasting (subsection 8.2), called [illegible] and the [illegible] model. In the two examples concerning reservoir forecasting, the method is compared with the standard RSM (see [7, 18]) which is utilized in reservoir uncertainty analysis software, such as COUGAR [4].

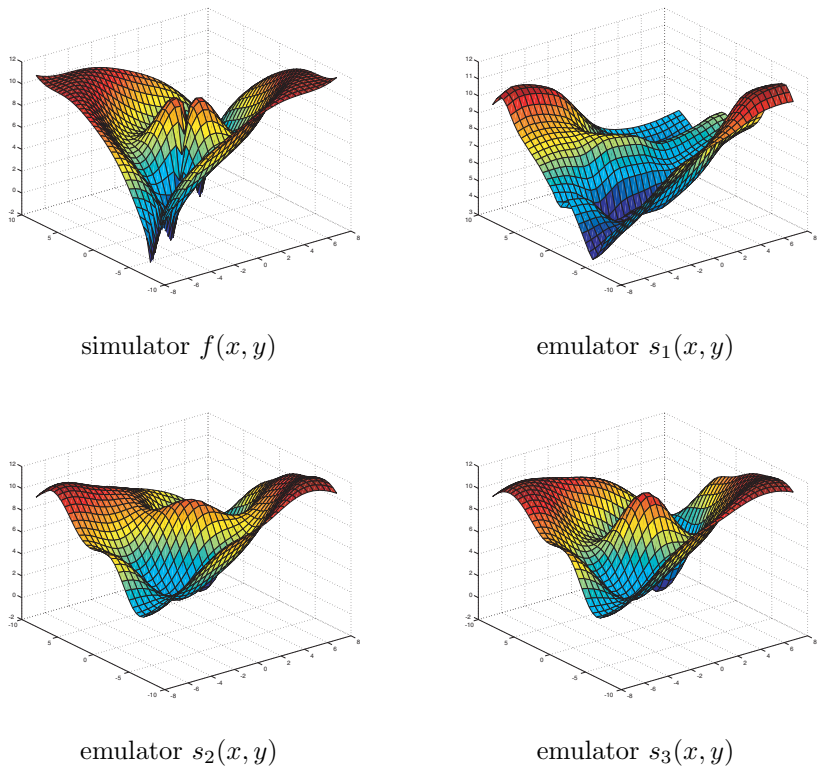


FIG. 8.1. *Hierarchical approximation. Graphs of the simulator f and emulators s_1, s_2, s_3.*

In each numerical comparison, the required number of simulations is fixed, and the quality of the resulting emulators is compared by their prediction accuracy. Note that the required number of simulations reflects the required computational costs, since each simulator run takes much more time than the computational time required for building the emulator, no matter which method is employed.

8.1. Analytical test example. In this first test case, consider the trivariate function

$$f(x, y, z) = 7\,\frac{\sin(\sqrt{x^2 + y^2}) + \epsilon}{\sqrt{x^2 + y^2}} + 3|x - y|^{1/2} + 0.01z,$$

where $\epsilon = 10^{-7}$, which is regarded as the "simulator." Note that f is linear in variable z, but nonlinear in x and y. In fact, the two variables x and y are identified as [illegible] variables, whereas z is classified as [illegible]. This conclusion was reached in the preprocessing screening, where it was found that $\lambda_3 \gg \max(\lambda_1, \lambda_2)$. This allows us to view $f \equiv f_z(x, y)$ for fixed z as a bivariate surface, where $z \equiv 0$ after screening. The graph of the "true response surface" f is shown in Figure 8.1 (top left).

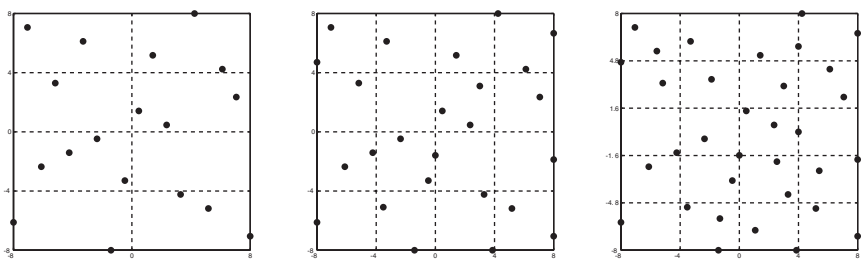
The correlation lengths λ_1 and λ_2 belonging to the remaining two active variables, x and y, are shown, along with the parameter p in (3.7), in Table 8.1 for three levels $\ell = 1, 2, 3$. For the construction of the initial design X_1, we apply an LHC to generate 18 points, and so the size of the initial point set X_1 is $|X_1| = 18$.

Then, the surface $f : \Omega \to \mathbb{R}$ is approximated on the computational domain

TABLE 8.1

Analytic test example. Correlation parameters at three levels, $\ell = 1, 2, 3$, and prediction accuracy of emulator sequence s_1, s_2, s_3. Note the definitions of η_1, η_2, and η_∞ in (8.1), (8.2), and (8.3); n_1 and n_2 are the grid dimensions.

ℓ	$\lvert X_\ell \rvert$	λ_1	λ_2	p	n_1	n_2	η_1	η_2	η_∞
1	18	13.5	4.9	1.4	2	4	1.3	1.8	7.6
2	25	11.4	10.3	2.0	4	4	0.8	1.1	5.0
3	33	14.4	11.0	2.0	4	5	0.5	1.1	4.8



X_1 X_2 X_3

FIG. 8.2. *Hierarchical approximation. Adaptive grids and designs X_1, X_2, X_3.*

$\Omega = [-8, 8] \times [-8, 8]$ by an emulator, whose construction is done by using our hierarchical approximation scheme. In this method, we apply kriging without including a regression part in the stochastic model (3.1).

Figure 8.1 shows the graphs of the three resulting emulators, s_1, s_2, s_3, from coarse to fine, and Figure 8.2 shows the corresponding sequence X_1, X_2, X_3 of three design sets.

Note from the visual appearance of the surface graphs in Figure 8.1 that the prediction quality of emulators s_1, s_2, s_3 is gradually improved. This is confirmed by our numerical results, which are shown in Table 8.1. Table 8.1 displays for each level $\ell = 1, 2, 3$ the number of points $|X_\ell|$ in the design set X_ℓ, the resulting

(8.1) $$\eta_1 = \|f - s_\ell\|_1/|G| = \frac{1}{|G|}\sum_{\mathbf{x}\in G} |f(\mathbf{x}) - s_\ell(\mathbf{x})|,$$

(8.2) $$\eta_2 = \left(\|f - s_\ell\|_2^2/|G|\right)^{1/2} = \sqrt{\frac{1}{|G|}\sum_{\mathbf{x}\in G} |f(\mathbf{x}) - s_\ell(\mathbf{x})|^2},$$

and

(8.3) $$\eta_\infty = \|f - s_\ell\|_\infty = \max_{\mathbf{x}\in G} |f(\mathbf{x}) - s_\ell(\mathbf{x})|,$$

where G denotes a fine uniform grid contained in Ω.

Note that by our adaptive design, that is, by the efficient distribution of the sites in X_1, X_2, X_3, the features of f are captured remarkably well, especially in regions where f has large derivatives.

TABLE 8.2
PUNQS model. Correlation parameters at different levels.

ℓ	$\lvert X_\ell \rvert$	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7	p
1	50	3.6	5.2	3.5	-	2.0	7.3	3.9	1.6
2	93	4.8	3.2	0.3	-	2.0	7.3	3.9	1.6
3	134	182	0.7	1.3	-	1000	2.3	0.2	1.3
4	200	147	0.8	1.0	-	1000	2.0	0.5	1.3
5	258	141	0.8	0.8	-	1000	2.1	0.4	1.2

8.2. Test cases from reservoir forecasting.

8.2.1. The PUNQS reservoir test case. In this second example, we analyze the case of a reservoir simulator containing seven uncertain scalar input variables and one scalar output. The example is generated by using the PUNQS [7] reservoir model in combination with Schlumberger's multiphase flow simulator ECLIPSE [21]. The input variables and their range of variation were suggested by reservoir engineers. In this model problem, the utilized input variables are corresponding to the aquifer strength, the residual gas oil saturation, the residual water oil saturation, and the vertical and horizontal permeability multipliers of the low and of the high quality sands. We have analyzed several outputs of the ECLIPSE simulator. The numerical results, presented in this subsection, were obtained by regarding the seven input variables, and the output oil production rate at 13 years of a given production well (called PRO-15 in the model [7]).

An LHC is used to generate the initial design X_1 comprising 50 different configurations, i.e., $|X_1| = 50$. In the screening preprocess, the input parameter x_4 is classified as inactive. In fact, the corresponding correlation length $\lambda_4 \gg 1000$, as predicted by MLE, is much larger than the initial correlation lengths of the other six input variables. Therefore, x_4 is discarded, the correlation lengths of the other six variables are for each level $\ell = 1, \ldots, 5$ shown in Table 8.2.

Recall that by the LHC, the projection of the initial seven-dimensional design to the active six-dimensional design is injective, and so it does not produce coincident data points.

The construction of the emulators s_ℓ is done by using our hierarchical approximation scheme, Algorithm 2. In this method, we apply kriging with using a quadratic regression part in the stochastic model (3.1) at each level ℓ, where the regression terms $x_1, x_2, x_3, x_5, x_6, x_1x_5, x_1^2$ were found to be dominant by the utilized statistical data analysis.

In order to analyze the accuracy of the emulators s_ℓ, we compare the emulators' function values $s_\ell(\mathbf{x})$ with the response output $f(\mathbf{x})$ of the simulator at 200 randomly selected configurations of input parameters $\mathbf{x}$, which are not contained in the design sites X_ℓ. We let $\eta^* = 3$ for the target accuracy. Our numerical results are shown in Table 8.3.

The numerical results in Table 8.3 document significant gain in performance between consecutive levels, in terms of the emulators' accuracy; see the different errors in Table 8.3. At final level $\ell = 5$, the target accuracy, $\eta^* = 3$, is approximated sufficiently well, as the mean absolute error drops from $\eta_1 = 4.62$ (at initial level $\ell = 1$) to $\eta_1 = 2.88$ (at $\ell = 5$), and so we obtain $\eta_1 \leq \eta^*$ at final level $\ell = 5$.

In order to further evaluate the performance of our hierarchical design, we have performed another test run based on a LHC with 258 points, matching

TABLE 8.3

*PUNQS model. Results obtained from our hierarchical approximation and comparison with kriging using LHC (***LHC***) and with RSM (***RSM***). Note the definitions of* η_1, η_2, *and* η_∞ *in* (8.1), (8.2), *and* (8.3).

ℓ	$\lvert X_\ell \rvert$	η_1	η_2	η_∞
1	50	4.62	5.60	14.80
2	93	3.86	4.90	15.70
3	134	3.58	4.70	15.10
4	200	3.18	4.30	13.30
5	258	2.88	4.20	13.80
LHC	258	2.90	4.70	16.30
RSM	79	3.93	4.84	13.10
RSM	258	3.93	4.80	12.90

the number of points $|X_5| = 258$ in the hierarchical design at the final level $\ell = 5$. The numerical results are shown in the sixth row (**LHC**) of Table 8.3. Note that our hierarchical approximation method is superior to this plain LHC in terms of the emulator accuracy. Indeed, while the mean absolute error η_1 of the emulator s_5 and that of the one-level LHC emulator, say, s_{LHC}, are about equal, the emulator s_5 reduces the root mean square error η_2 of s_{LHC} by about 10%, and the maximum error η_∞ of s_{LHC} is reduced by even about 20%.

We can explain the significant gain concerning the reduction of the maximum error η_∞ as follows. Due to the hierarchical design, the design sites are adaptively inserted, such that at each refinement step new points are inserted in regions (by local optimal design) where the current maximum error is likely to be highest. This interactive prediction cannot be achieved by a plain one-level LHC, where ,, design sites are selected beforehand in a preprocessing step.

We have also compared our hierarchical approximation method with a polynomial regression model, as usually used in the standard RSM [7, 18]. Two experiments were made; the first uses 79 simulations selected from a central composite face centered design, and the second uses 258 design sites generated from a one-level LHC.

In the first experiment the regression terms were selected by using the step by step fit used in COUGAR [4] based on sensitivity analysis arguments similar to the ones described in section 4 for the screening process. The results, displayed in the penultimate row of Table 8.3, are about comparable to the ones obtained by our emulator s_2, at a slightly smaller maximum error though. The difference in maximum error is mainly due to small overshoots of the kriging emulator in local areas of the domain, where the simulator f is subject to strong variation. The emulator s_{RSM} manages to somewhat reduce these undesired local overshoots.

However, in the second experiment, displayed in the last row (**RSM**) of Table 8.3, our emulator s_5 is clearly superior to the comparable one-level method RSM. Indeed, the emulator s_5 of our hierarchical method reduces the mean absolute error η_1 obtained by RSM (i.e., by emulator s_{RSM}), by more than 25%, and the root mean square error η_2 of s_{RSM} is reduced by about 15%. The resulting maximum error η_∞ of s_5, however, is again slightly larger than that of the emulator s_{RSM}.

Our numerical results also show that the emulator s_{RSM} does not significantly improve its accuracy at the insertion of more design points; see the last two rows in Table 8.3. In contrast, our hierarchical emulator gradually reduces the prediction errors η_1 and η_2 at increasing refinement level.

TABLE 8.4
*IC Fault model. Results obtained from our hierarchical approximation method; comparison with RSM using FFD (***RSM***), and kriging using LHC (***KRG***). Note the definitions of η_1, η_2, and η_∞ in* (8.1), (8.2), *and* (8.3).

ℓ	$\lvert X_\ell \rvert$	η_1	η_2	η_∞
1	22	3.6	5.4	28.7
2	49	1.4	3.2	25.2
3	72	1.1	1.8	12.5
4	102	0.8	1.5	11.3
5	124	0.7	1.3	8.6
6	146	0.6	1.0	5.4
RSM	22	8.7	10.0	25.8
RSM	146	7.2	8.4	22.9
KRG	146	1.0	1.5	11.7

We finally remark that the variation of the response surface of the PUNQS model is rather small. More complicated test case scenarios from relevant applications are dealing with discontinuous response surfaces of strong variation. One example is the challenging IC Fault model treated in the following subsection.

8.2.2. The IC Fault model. In this final example, we apply our hierarchical approximation method to a more difficult test case from Imperial College, called the [illegible] model, which has previously been used for parameter identification by history matching. The IC Fault model incorporates a rather complicated discontinuous response surface. Therefore, the IC Fault model is regarded as a very challenging test example, where standard methods for history matching often fail [28]. For a detailed description on the IC Fault model, see [28].

Let us first discuss only some of the relevant features of the IC Fault model. The simplified reservoir of the IC Fault model has three uncertain input parameters, corresponding to the fault throw and the good and the poor sand permeability multipliers. The analyzed output from the response surface is in this test case the oil production rate at 10 years. The three input parameters are referred to as x_1, x_2, and x_3. According to [28], the input configurations are uniformly distributed in the computational domain $\Omega = [0, 50] \times [100, 200] \times [0, 50]$.

Having rescaled each of the variables' range to the interval $[-1, 1]$, the reservoir simulator ECLIPSE is run on an LHC consisting of $|X_1| = 22$ different configurations for the initial design X_1. We then apply our hierarchical approximation method to the IC Fault model, using Algorithm 2, where we let $\eta^* = 0.5$ for the target accuracy. This results in a hierarchical approximation with six design levels, that is, $L = 6$.

The construction of the emulators s_ℓ is done by using our hierarchical approximation scheme, Algorithm 2. In this method, we apply kriging with using a quadratic regression part in the stochastic model (3.1) at each level ℓ, where the regression terms x_2, x_3, x_3^2 were found to be dominant by the utilized statistical data analysis.

In order to evaluate the prediction accuracy of emulator s_ℓ at each design level ℓ, $1 \le \ell \le 6$, the simulator was run at an additional set X comprising 200 points, where the points in X are generated by another LHC, being different from the one used to generate the initial design X_1 in our hierarchical approximation method.

We have recorded the mean absolute error η_1, the root mean square error η_2, as well as the maximum error η_∞ for our emulator s_ℓ at each level. The numerical results are shown in Table 8.4. Note that the mean absolute error η_1 is at the final level $\ell = 6$ very close to the target accuracy $\eta^* = 0.5$, after only 146 simulations.

TABLE 8.5
IC Fault model. Correlation parameters at different levels.

ℓ	$\lvert X_\ell \rvert$	λ_1	λ_2	λ_3	p
1	22	16.40	1.70	0.40	2.00
2	49	24.00	42.20	0.13	1.80
3	72	22.90	89.90	0.40	1.20
4	102	34.00	98.80	0.12	1.40
5	124	16.10	37.40	0.10	1.50
6	146	10.40	37.70	0.30	1.00

For the purpose of comparison, we consider two alternative (one-level) methods to compare their numerical results with those of our emulator. The first emulator, s_{RSM}, is obtained by RSM in combination with an FFD. For the emulator s_{RSM} a quadratic regression polynomial was used. The second emulator, s_{KRG}, is obtained by using kriging in combination with an LHC. Table 8.4 shows the numerical results obtained by using these two emulators, s_{RSM} and s_{KRG}.

Note that the final emulator s_6 of our hierarchical approximation method (see Table 8.4) is superior to both the kriging emulator s_{KRG} (see Table 8.4, last row) and the emulator s_{RSM} (see Table 8.4, second last row). Indeed, the mean absolute error η_1 and the maximum error η_∞ obtained by s_6 is about half of the corresponding error obtained by s_{KRG}, whereas the root mean square error η_2 of s_{KRG} is reduced by a third. Note that a comparable prediction accuracy of s_{KRG} is obtained by s_3 after only three steps of our hierarchical approximation method, with using only $|X_3| = 72$ simulator runs rather than $|X_6| = 146$ runs.

As for the comparison between our emulator and that of RSM, s_{RSM}, the performance of our hierarchical emulator is much better than that of s_{RSM}. Indeed, note that the accuracy of our final emulator s_6 is for the mean absolute error η_1 more than ten times smaller than that of the emulator s_{RSM}, when using $|X_6| = 146$ design points, whereas its root mean square error η_2 is about eight times smaller than that of s_{RSM}, and the maximum error η_∞ of s_6 is only about one quarter of the maximum error of s_{RSM}.

Note that the accuracy of our initial emulator, s_1, is superior to the accuracy of the emulator s_{RSM}, when using only $|X_1| = 22$ design points. Indeed, while the maximum error η_∞ of s_1 is comparable to that of s_{RSM}, the mean absolute error η_1 of s_1 is less than half of the corresponding mean absolute error of s_{RSM}. Moreover, the root mean square error η_2 of s_1 is about half the corresponding mean square error of s_{RSM}; see Table 8.4, first row, and Table 8.4, third to last row. That s_1 outperforms s_{RSM} is not too surprising, since s_1 is the best linear unbiased predictor for the deterministic response $\mathbf{f}_{X_1}$; see the discussion in section 3.

Finally, Table 8.5 shows the correlation lengths λ_i, $1 \le i \le 3$, of the correlation parameters for the three different input parameters, at the six different design levels. Note that input parameter x_3 has, in comparison with x_1 and x_2, a rather short correlation length λ_3. This behavior is well treated by the adaption strategy (i.e., global adaptive gridding in combination with local optimal design) of our hierarchical approximation method, which prefers to insert further design points (at run time of Algorithm 2) in the direction of coordinate axis x_3. This enhanced flexibility gives our hierarchical approximation method another advantage over commonly used nonhierarchical methods which are not designed to determine potentially dominant input parameters.

9. Conclusion and future work. We have proposed a hierarchical nonlinear approximation scheme for scalar-valued multivariate functions. The main objective of the utilized iterative refinement is to obtain a sufficiently accurate approximation at very few function evaluations. The hierarchical approximation method has been applied to relevant applications from experimental design and statistical data analysis in computer experiments. Our hierarchical method combines different important requirements for efficient designs in computer experiments, such as screening, entropy minimization, statistical data fitting, and parallelization. This results in a flexible and efficient iterative method, whose main components are adaptive gridding, local optimal design, and kriging, in combination with MLE. As supported by numerical examples, including two real-world model problems from reservoir forecasting, the performance of our hierarchical approximation method is in terms of the resulting emulators' prediction accuracy superior to that of traditional nonhierarchical methods, such as the widely used RSM. Further important aspects of this research are uncertainty reduction and calibration, which we will treat in a forthcoming paper.

Acknowledgments. We wish to thank Jonathan Rougier for many useful suggestions and fruitful discussions. Moreover, we received much technical help and advice from our colleagues at Schlumberger. Last but not least, we thank the anonymous referees for their useful comments and constructive suggestions, which helped improve a previous version of the paper.

REFERENCES

[1] M. D. Buhmann, *Radial Basis Functions*, Cambridge University Press, Cambridge, UK, 2003.
[2] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, *A limited memory algorithm for bound constrained optimization*, SIAM J. Sci. Comput., 16 (1995), pp. 1190–1208.
[3] R. E. Caflisch, *Monte Carlo and quasi-Monte Carlo methods*, Acta Numer., 1998, pp. 1–49.
[4] The COUGAR Project, http://consortium.ifp.fr/cougar/.
[5] P. S. Craig, M. Goldstein, J. C. Rougier, and A. H. Seheult, *Bayesian forecasting for complex systems using computer simulators*, J. Amer. Statist. Assoc., 96 (2001), pp. 717–729.
[6] C. Currin, T. Mitchell, M. Morris, and D. Ylvisaker, *A Bayesian Approach to the Design and Analysis of Computer Experiments*, ORNL technical report 6498, National Technical Information Service, Springfield, VA, 1988.
[7] J. P. Dejean and G. Blanc, *Managing uncertainties on production predictions using integrated statistical methods*, SPE J., SPE 56696, 1999.
[8] C. L. Farmer, *Geological modelling and reservoir simulation*, in Mathematical Methods and Modelling in Hydrocarbon Exploration and Production, A. Iske and T. Randen, eds., Springer-Verlag, Berlin, 2005, pp. 119–212.
[9] T. Gutzmer and A. Iske, *Detection of discontinuities in scattered data approximation*, Numer. Algorithms, 16 (1997), pp. 155–170.
[10] A. Iske, *Multiresolution Methods in Scattered Data Modelling*, Springer-Verlag, Berlin, 2004.
[11] A. Iske and T. Randen, eds., *Mathematical Methods and Modelling in Hydrocarbon Exploration and Production*, Springer-Verlag, Berlin, 2005.
[12] A. Iske and J. Levesley *Multilevel scattered data approximation by adaptive domain decomposition*, Numer. Algorithms, 39 (2005), pp. 187–198.
[13] M. C. Kennedy and A. O'Hagan, *Bayesian calibration of computer models*, J. Roy. Statist. Soc. Ser. B, 63 (2000), pp. 425–464.
[14] J. Oakley and A. O'Hagan, *Probabilistic sensitivity analysis of complex models: A Bayesian approach*, J. Roy. Statist. Soc. Ser. B, 16 (2004), pp. 751–769.
[15] D. V. Lindley, *On a measure of the information provided by an experiment*, Ann. Math. Statist., 27 (1956), pp. 986–1005.
[16] G. Matheron, *Principles of geostatistics*, Economic Geology, 58 (1963), pp. 1246–1266.
[17] M. D. McKay, R. J. Beckman, and W. J. Conover, *A comparison of three methods for selecting values of input variables in the analysis of output from a computer code*, Technometrics, 21 (1979), pp. 239–245.

[18] R. H. Myers and D. C. Montgomery, *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*, 2nd ed., Wiley-Interscience, New York, 2002.
[19] A. B. Owen, *Monte Carlo variance of scrambled net quadrature*, SIAM J. Numer. Anal., 34 (1997), pp. 1884–1910.
[20] R Development Core Team, *R: A Language and Environment for Statistical Computing*, http://www.R-project.org, 2004.
[21] Schlumberger GeoQuest, *ECLIPSE Technical Description Manual*, Schlumberger GeoQuest, Houston, TX, 2001.
[22] C. E. Shannon, *A mathematical theory of communication*, Bell System Tech. J., 27 (1948), pp. 379–423, 623–656.
[23] M. C. Shewry and H. P. Wynn, *Maximum entropy sampling*, J. Appl. Statist., 14 (1987), pp. 165–170.
[24] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, *Design and analysis of computer experiments*, Statist. Sci., 4 (1989), pp. 409–435.
[25] T. J. Santner, B. J. Williams, and W. I. Notz, *The Design and Analysis of Computer Experiments*, Springer-Verlag, New York, 2003.
[26] M. Sambridge, *Geophysical inversion with a neighbourhood algorithm* I: *Searching a parameter space*, Geophys. J. Int., 138 (1999), pp. 479–494.
[27] M. Sambridge, *Geophysical inversion with a neighbourhood algorithm* II: *Appraising the ensemble*, Geophys. J. Int., 138 (1999), pp. 727–746.
[28] Z. Tavassoli, J. N. Carter, and P. R. King, *Errors in history matching*, SPE J., SPE 86883, 2004.
[29] W. J. Welch, R. J. Buck, J. Sacks, H. P. Wynn, T. J. Mitchell, and M. D. Morris, *Screening, predicting, and computer experiments*, Technometrics, 34 (1992), pp. 15–25.
[30] H. Wendland, *Scattered Data Approximation*. Cambridge University Press, Cambridge, UK, 2005.
[31] T. Werther, *Optimal multivariate interpolation*, in Mathematical Methods and Modelling in Hydrocarbon Exploration and Production, A. Iske and T. Randen, eds., Springer-Verlag, Berlin, 2005, pp. 389–407.

SIAM J. SCI. COMPUT.
Vol. 29, No. 1, pp. 70–92
© 2007 Society for Industrial and Applied Mathematics

EFFICIENT ALGORITHMS FOR PROBLEMS WITH FRICTION*

S. HÜEBER†, A. MATEI‡, AND B. I. WOHLMUTH†

Abstract. In this paper, a nonconforming discretization method for the frictional contact between two bodies subjected to antiplane shear deformation is considered. The method is based on a mixed variational formulation where dual basis functions are used for the discretization of the Lagrange multiplier. Under some regularity assumptions on the solution, an optimal a priori error estimate is obtained. To solve the discrete nonlinear problem an inexact primal-dual active set strategy is introduced. Finally, numerical examples confirming the theoretical result for the a priori error estimate and illustrating the performance of the algorithm are presented. The results can easily be generalized to the case of Coulomb friction. For this case a numerical example is given.

Key words. antiplane shear deformation, frictional contact, mixed formulation, dual Lagrange multipliers, nonconforming meshes, optimal a priori error estimates, inexact primal-dual active set strategy

AMS subject classifications. 65K10, 65N15, 65N30, 65N55, 74B10, 74M10, 74M15, 74S05

DOI. 10.1137/050634141

1. Introduction.

In this paper, we study the antiplane shear deformation of two elastic bodies in frictional contact on their common boundary. For a cylindrical body subject to antiplane shear, the displacement is parallel to the generators of the cylinder and is independent of the axial coordinate. The displacement field is described by a scalar function, and therefore the governing equations and boundary conditions are quite simple. The antiplane shear (or longitudinal shear, generalized shear) may be viewed as complementary to [illegible] and represents the mode III fracture mode for crack problems. This model was considered by many authors; see, for instance, [BHP02, HMM02, Hor95, HS01, HS04, Kno76, Kno77, MMS01]. An excellent reference concerning antiplane shear deformation in solid mechanics is the review article [Hor95], where modern developments for the antiplane shear model and its applications are described for both linear and nonlinear solid mechanics. Note that in [HMM02, MMS01] quasi-static antiplane contact problems were considered, and the study was made within the framework of variational inequalities. For the basics in the theory of variational inequalities, we refer to [GLT81, Glo84].

To model the friction, we use Tresca's law; a description of this law can be found in [DL76, Pan85] and more recently in [HS02, SST04]. Our study is based on a mixed variational formulation with dual Lagrange multipliers, the well-posedness of this weak formulation being guaranteed by arguments in the saddle point theory; details on the saddle point theory and its applications can be found, e.g., in

*Received by the editors June 21, 2005; accepted for publication (in revised form) April 25, 2006; published electronically January 12, 2007. This work was supported in part by the Deutsche Forschungsgemeinschaft, SFB 404, B8, and by the Research Project *Nonlinear approximation and adaptivity: breaking complexity in numerical modeling and data representation* carried out in the framework of the European Community Program *Improving the Human Potential Network* (HPRN-CT-2002-00286).

http://www.siam.org/journals/sisc/29-1/63414.html

†Institut für Angewandte Analysis und Numerische Simulation (IANS), Universität Stuttgart, Pfaffenwaldring 57, D-70569 Stuttgart, Germany (hueeber@ians.uni-stuttgart.de, wohlmuth@ians.uni-stuttgart.de).

‡Department of Mathematics, University of Craiova, A.I. Cuza street 13, 200585, Craiova, Romania (AndaluziaMatei@k.ro).

[Bra97, BF91, ET76, Hcc96]. To discretize our frictional problem, we use a mortar technique on nonconforming meshes. For mortar techniques with standard Lagrange multipliers, we refer to [BR03, BHL99, CHLS01, Hil00, HL02]. Mortar techniques with dual Lagrange multipliers were recently used in [HW05b, HW05a] within the framework of frictionless unilateral multibody contact problems. Due to the use of dual basis functions for the Lagrange multiplier, we can apply a primal-dual active set strategy based on [Sta04] to solve the resulting nonlinear algebraic form of the discrete problem. Furthermore, the dual Lagrange multiplier allows us in the multibody case to easily apply a basis transformation of the finite element basis such that the problem in the new basis for the multibody case has exactly the same structure as for the one body case. In contrast, to standard Lagrange multiplier basis functions, this transformation is obtained by a local scaling of the coupling matrix. As a consequence, the constraints can be verified pointwise for each node on the slave side. This is not the case for the standard Lagrange multiplier. The considered primal-dual active set strategy can be interpreted as a semi-smooth quasi-Newton method. A further advantage of the approach is that the nonlinear friction conditions are applied as Dirichlet or Neumann boundary conditions on the interface. This makes the implementation extremly simple, and an existing finite element code can be easily adapted to this solution strategy. Since there is no need for a penalty parameter in the primal-dual active set strategy, the condition number of the stiffness matrix is the same as in a linear setting. Therefore, we combine the considered method with an optimal multigrid solver (see [Woh05]), yielding an inexact method. This approach results in an efficient nonlinear multigrid method for the contact problem with a negligible additional effort compared to solving a linear problem.

The structure of the rest of the paper is as follows. In section 2, we describe the frictional antiplane contact problem involving two elastic bodies, and we provide a weak formulation. In section 3, we propose a nonconforming discretization method based on dual Lagrange multipliers; see [Woh00]. We obtain an optimal a priori error estimate of order $h^{1/2}+\nu$, $0<\nu\leq\frac{1}{2}$, for the solution and for the Lagrange multiplier if the solution is regular enough. In section 4, the inexact primal-dual active set algorithm is presented. Finally, numerical examples confirming our theoretical results and illustrating the performance of our algorithm are given in section 5. Section 6 describes the generalization of our algorithm to the case of Coulomb friction.

We mention that the idea of a primal-dual active set strategy for frictional problems can also be generalized to the three-dimensional case. Here the use of a more general nonlinear complementarity function involving the direction is necessary (see also [HSW06]).

2. An antiplane frictional contact problem. Let us consider two deformable solids, $\mathcal{B}^m$, $\mathcal{B}^s\subset\mathbb{R}^3$, that are cylinders, with the generators parallel to the x_3-axis of a rectangular Cartesian coordinate system $Ox_1x_2x_3$. The subscripts m and s are motivated by the presence of the mortar setting in the numerical analysis of our model. Everywhere below, we will use a superscript k to indicate that a quantity is related to the cylinder $\mathcal{B}^k$, $k=m,s$. We assume that the bodies are homogeneous, isotropic and elastic media; more precisely, we will use the constitutive law:

$$\boldsymbol{\sigma}^k=\lambda^k\mathrm{tr}(\boldsymbol{\varepsilon}(\boldsymbol{u}^k))\,\mathrm{Id}+2\mu^k\boldsymbol{\varepsilon}(\boldsymbol{u}^k)\quad\text{in}\quad\mathcal{B}^k, \tag{2.1}$$

where $\boldsymbol{\sigma}^k=(\boldsymbol{\sigma}^k_{ij})$ denotes the stress field, $\boldsymbol{\varepsilon}(\boldsymbol{u}^k)=(\varepsilon_{ij}(\boldsymbol{u}^k))$ the linearized strain tensor, $\lambda^k>0$ and $\mu^k>0$ are the Lamé coefficients, $\mathrm{tr}(\boldsymbol{\varepsilon}(\boldsymbol{u}^k))=\sum_{i=1}^3\varepsilon_{ii}(\boldsymbol{u}^k)$, and Id is the unit tensor in $\mathbb{R}^3$. Moreover, we assume that the generators are sufficiently

long so that end effects in the axial direction are negligible. Let us denote by Ω^k a cross-section, which is a domain in $\mathbb{R}^2$. Thus, $\mathcal{B}^k = \Omega^k \times (-\infty, +\infty)$. For each domain Ω^k, we assume that its boundary Γ^k is Lipschitz continuous and is divided into three disjoint measurable parts Γ_1^k, Γ_2^k and Γ_3^k, with meas $(\Gamma_1^k) > 0$. We assume that the bodies are clamped on Γ_1^k, body forces of density $\boldsymbol{f}_0^k$ act on Ω^k, and surface traction of density $\boldsymbol{f}_2^k$ act on Γ_2^k. Moreover, we assume that the bodies in the initial configuration are in contact on their common boundary part $\Gamma_3 = \Gamma_3^1 = \Gamma_3^2$ and that $\overline{\Gamma}_3$ is a compact subset of $\partial\Omega^k \backslash \overline{\Gamma}_1^k$, $k = m$, s. We load the solid in a special way as follows:

$$\boldsymbol{f}_0^k = (0, 0, f_0^k), \quad f_0^k = f_0^k(x_1, x_2) : \Omega^k \to \mathbb{R}, \tag{2.2}$$

$$\boldsymbol{f}_2^k = (0, 0, f_2^k), \quad f_2^k = f_2^k(x_1, x_2) : \Gamma_2^k \to \mathbb{R}. \tag{2.3}$$

The unit outward normal on $\Gamma^k \times (-\infty, +\infty)$ is denoted by $\boldsymbol{n}^k$, $\boldsymbol{n}^k = (n_1^k, n_2^k, 0)$ and is defined a.e.. We note that on Γ_3,

$$\boldsymbol{n} := \boldsymbol{n}^s = -\boldsymbol{n}^m, \quad \boldsymbol{\sigma}^s\, \boldsymbol{n}^s = -\boldsymbol{\sigma}^m\, \boldsymbol{n}^m, \quad \boldsymbol{\sigma}_\tau := \boldsymbol{\sigma}_\tau^s = -\boldsymbol{\sigma}_\tau^m.$$

For a vector $\boldsymbol{v}^k$, we denote by v_n^k and $\boldsymbol{v}_\tau^k$ its [illegible] and [illegible] parts on the boundary, given by $v_n^k := \boldsymbol{v}^k \cdot \boldsymbol{n}^k$ and $\boldsymbol{v}_\tau^k := \boldsymbol{v}^k - v_n^k \boldsymbol{n}^k$, respectively. Furthermore, for a given stress field $\boldsymbol{\sigma}^k$, we denote by σ_n^k and $\boldsymbol{\sigma}_\tau^k$ the [illegible] and the [illegible] parts on the boundary, that is $\sigma_n^k := (\boldsymbol{\sigma}^k \boldsymbol{n}^k) \cdot \boldsymbol{n}^k$ and $\boldsymbol{\sigma}_\tau^k := \boldsymbol{\sigma}^k \boldsymbol{n}^k - \sigma_n^k \boldsymbol{n}^k$, respectively.

The body forces (2.2) and the surface traction (2.3) would be expected to give rise to a deformation of the elastic cylinder $\mathcal{B}^k$, such that the displacement $\boldsymbol{u}^k$ is of the form

$$\boldsymbol{u}^k = (0, 0, u^k), \quad u^k = u^k(x_1, x_2) : \Omega^k \to \mathbb{R}. \tag{2.4}$$

Consequently, keeping in mind (2.1), the traction vector on $\Gamma^k \times (-\infty, +\infty)$ is given by

$$\boldsymbol{\sigma}^k \boldsymbol{n}^k = (0, 0, \mu^k \partial_n u^k),$$

where, as usual, $\partial_n u^k := \nabla \boldsymbol{u}^k \cdot \boldsymbol{n}^k$. Moreover, we note that

$$\text{Div}\, \boldsymbol{\sigma}^k = (0, 0, \mu^k\, \Delta u^k). \tag{2.5}$$

According to the physical setting, we have

$$\text{Div}\, \boldsymbol{\sigma}^k + \boldsymbol{f}_0^k = \boldsymbol{0} \qquad \text{in} \qquad \mathcal{B}^k, \tag{2.6}$$

$$\boldsymbol{u}^k = \boldsymbol{0} \qquad \text{on} \qquad \Gamma_1^k \times (-\infty, +\infty), \tag{2.7}$$

$$\boldsymbol{\sigma}^k \boldsymbol{n}^k = \boldsymbol{f}_2^k \qquad \text{on} \qquad \Gamma_2^k \times (-\infty, +\infty). \tag{2.8}$$

Finally, we have to describe the frictional contact condition on $\Gamma_3 \times (-\infty, +\infty)$. Everywhere in what follows, we will use $|\cdot|$ to denote the Euclidean norm on $\mathbb{R}^d$, $d = 1$ or 3. Since $u_n^s = u_n^m = 0$, on $\Gamma_3 \times (-\infty, +\infty)$ the contact is bilateral. Let us model the friction using Tresca's law,

$$\left\{ \begin{array}{l} |\boldsymbol{\sigma}_\tau| \le g, \\ |\boldsymbol{\sigma}_\tau| < g \;\Rightarrow\; \boldsymbol{u}_\tau^s - \boldsymbol{u}_\tau^m = 0, \\ |\boldsymbol{\sigma}_\tau| = g \;\Rightarrow\; \boldsymbol{\sigma}_\tau = -\beta(\boldsymbol{u}_\tau^s - \boldsymbol{u}_\tau^m), \quad \beta > 0, \end{array} \right. \qquad \text{on} \quad \Gamma_3 \times (-\infty, +\infty), \tag{2.9}$$

where g is a given function, called the [illegible]. We note that $\boldsymbol{u}_\tau^k = (0, 0, u^k)$ and $\boldsymbol{\sigma}_\tau^k = (0, 0, \sigma_\tau^k)$ with $\sigma_\tau^k = \mu^k \, \partial_n u^k$. In addition, we have

$$\partial_n u := \mu^s \partial_n u^s = -\mu^m \partial_n u^m \quad \text{on} \quad \Gamma_3. \tag{2.10}$$

Using (2.6)–(2.9) and keeping in mind (2.4)–(2.5), the study of our three-dimensional mechanical model reduces to the study of the following mathematical problem.

PROBLEM 2.1. [illegible] $u^k : \Omega \to \mathbb{R}$, $k = m, s$, [illegible]

$$\mu^k \Delta u^k + f_0^k = 0 \qquad \text{[illegible]} \qquad \Omega^k, \tag{2.11}$$

$$u^k = 0 \qquad \text{[illegible]} \qquad \Gamma_1^k, \tag{2.12}$$

$$\mu^k \partial_n u^k = f_2^k \qquad \text{[illegible]} \qquad \Gamma_2^k, \tag{2.13}$$

[illegible]

$$\left\{ \begin{array}{l} |\partial_n u| \le g, \\ |\partial_n u| < g \Rightarrow u^s - u^m = 0, \\ |\partial_n u| = g \Rightarrow \partial_n u = -\beta(u^s - u^m), \quad \beta > 0, \end{array} \right. \qquad \text{[illegible]} \quad \Gamma_3 \tag{2.14}$$

[illegible]

In the study of Problem 2.1, we assume that

$$f_0^k \in L^2(\Omega^k), \quad f_2^k \in L^2(\Gamma_2^k), \tag{2.15}$$

$$g \in L^2(\Gamma_3), \qquad g \ge 0 \ \text{ a.e. on } \Gamma_3. \tag{2.16}$$

In order to obtain a weak formulation for Problem 2.1, we introduce the Hilbert space

$$V^k := \left\{ v^k \in H^1(\Omega^k) \,|\, v^k = 0 \quad \text{on} \quad \Gamma_1^k \right\}$$

endowed with the inner product given by

$$(u^k, v^k)_{V^k} := \int_{\Omega^k} \nabla u^k \cdot \nabla v^k \, dx, \qquad u^k, v^k \in V^k,$$

and the associated norm,

$$\|v^k\|_{V^k} := \|\nabla v^k\|_{L^2(\Omega^k)^2}, \qquad v^k \in V^k,$$

which is equivalent to $\|\cdot\|_{1,\Omega^k}$. Let us consider the product space $V := V^m \times V^s$ and let $a : V \times V \to \mathbb{R}$ be the bilinear form

$$a(u, v) := \sum_{k=1}^{2} \mu^k (u^k, v^k)_{V^k}.$$

Clearly, this form is continuous with the continuity constant $M_a = \mu^m + \mu^s$, and V-elliptic with the V-ellipticity constant $m_a = \min\{\mu^m, \mu^s\}$. Defining $f \in V$ such that

$$(f, v)_V = \sum_{k=m,s} \left(\int_{\Omega^k} f_0^k \, v^k \, dx + \int_{\Gamma_2^k} f_2^k \, v^k \, ds \right),$$

due to Green's formula, for regular functions u^m and u^s satisfying (2.11)–(2.13), we get

$$a(u,\, v) = (f, v)_V + \int_{\Gamma_3} (\mu^m \partial_n u^m\, v^m + \mu^s \partial_n u^s\, v^s)\, ds, \qquad v = (v^m, v^s) \in V. \tag{2.17}$$

Keeping in mind (2.10), we define a Lagrange multiplier λ in $M := (H^{1/2}(\Gamma_3))'$ by

$$\langle \lambda, w\rangle_{\Gamma_3} := -\int_{\Gamma_3} \partial_n u\, w\, ds, \qquad w \in H^{1/2}(\Gamma_3), \tag{2.18}$$

where $\langle\cdot,\cdot\rangle_{\Gamma_3}$ denotes the duality pairing. Furthermore, we introduce a nonempty closed convex set,

$$\Lambda := \left\{ \zeta \in M \,:\, \langle \zeta,\, w\rangle_{\Gamma_3} \le \int_{\Gamma_3} g\,|w|\, ds, \quad w \in H^{1/2}(\Gamma_3) \right\}. \tag{2.19}$$

Clearly, $\lambda \in \Lambda$ and we can rewrite (2.17) as

$$a(u,\, v) = (f, v)_V \,-\, \langle \lambda, [v]\rangle_{\Gamma_3}, \qquad v \in V, \tag{2.20}$$

where $[v]$ denotes the [illegible] defined on Γ_3 by

$$[v] := v^s - v^m, \qquad v = (v^m, v^s) \in V.$$

Let us denote by $b : V \times M \to \mathbb{R}$ the bilinear and continuous form

$$b(v,\, \zeta) := \langle \zeta, [v]\rangle_{\Gamma_3}. \tag{2.21}$$

Keeping in mind the properties of the trace operator, it can be shown that the form $b(\cdot,\cdot)$ satisfies the [illegible]: there exists a constant $\alpha > 0$ such that

$$\inf_{0\neq\mu\in M} \sup_{0\neq v\in V} \frac{b(v,\zeta)}{\|v\|_V \|\zeta\|_M} \ge \alpha,$$

where $\|\cdot\|_M$ denotes the natural norm of $(H^{1/2}(\Gamma_3))'$. The friction law (2.14) leads to the identity

$$\int_{\Gamma_3} \partial_n u\, [u]\, ds = -\int_{\Gamma_3} g|[u]|\, ds.$$

Taking into account the definition of λ, (2.18), the definition of $b(\cdot,\cdot)$, (2.21), and the definition of Λ, (2.19), we can write

$$b(u,\lambda) = \int_{\Gamma_3} g|[u]|\, ds \quad \text{and} \quad b(u,\zeta) \le \int_{\Gamma_3} g|[u]|\, ds, \qquad \zeta \in \Lambda.$$

Consequently, we get the inequality

$$b(u, \zeta - \lambda) \le 0, \qquad \zeta \in \Lambda.$$

Combining this, (2.20), and (2.21), we obtain the following weak formulation of Problem 2.1.

PROBLEM 2.2. [illegible] $u \in V$ [illegible] $\lambda \in \Lambda$ [illegible]

$$\begin{aligned} a(u, v) + b(v, \lambda) &= (f, v)_V, \quad v \in V, \\ b(u, \zeta - \lambda) &\leq 0, \qquad \zeta \in \Lambda. \end{aligned}$$

The well-posedness of Problem 2.2 is given by the following theorem.

THEOREM 2.1. [illegible] (2.15) [illegible] (2.16) [illegible] $(u, \lambda) \in V \times \Lambda$ [illegible] 2.2 [illegible] f_1, f_2 [illegible] V [illegible] $(f_0, f_2)_1, (f_0, f_2)_2$, [illegible]

$$\|u_1 - u_2\|_V + \|\lambda_1 - \lambda_2\|_M \leq \frac{\alpha + m_a + M_a}{\alpha\, m_a} \|f_1 - f_2\|_V, \tag{2.22}$$

[illegible] (u_1, λ_1), (u_2, λ_2) [illegible] 2.2 [illegible] $f_1, f_2 \in V$, [illegible]

In order to prove the existence and the uniqueness of the solution of Problem 2.2, we can use Theorem 3.11 in [Hcc96]. The stability result can be obtained by standard techniques and by using the inf-sup property of the bilinear form $b(\cdot, \cdot)$.

3. Nonconforming discretization and optimal a priori error estimates. In this section we give the discretization of Problem 2.2, and we derive an optimal a priori error estimate for the discretization error. To do so, we introduce a suitable basis transformation of our nodal finite element basis. Let us assume that the bodies Ω^k, $k = m, s$, are polygonal domains. To approximate V, we use linear finite elements on simplicial triangulations and bilinear finite elements on quadrilateral triangulations. The finite element space associated with the shape regular triangulation $\mathcal{T}_{h,\Omega^k}$ is denoted by $S_1(\Omega^k, \mathcal{T}_{h,\Omega^k})$. The mesh size h is defined by the maximal diameter of the elements in $\mathcal{T}_{h,\Omega^m}$ and $\mathcal{T}_{h,\Omega^s}$. For simplicity we assume that Γ_1^k, $k = m, s$, and that Γ_3 can be written as the union of edges, Γ_3 both from the triangulation $\mathcal{T}_{h,\Omega^s}$ of the slave side and from the triangulation $\mathcal{T}_{h,\Omega^m}$ of the master side. Before introducing the discrete spaces, we decompose the set of all vertices into three disjoint sets $\mathcal{S}$, $\mathcal{M}$, and $\mathcal{N}$. By the first subset $\mathcal{S}$ we denote all vertices on $\overline{\Gamma}_3$ of the triangulation $\mathcal{T}_{h,\Omega^s}$ on the slave side, and by $\mathcal{M}$ we denote all vertices on $\overline{\Gamma}_3$ of the triangulation $\mathcal{T}_{h,\Omega^m}$ on the master side. The set $\mathcal{N}$ contains all remaining ones. Then we have for the discrete spaces V_h^k

$$V_h^k := \left\{ v_h^k \in S_1\left(\Omega^k,\, \mathcal{T}_{h,\Omega^k}\right) \,:\, v_h^k = 0 \quad \text{on} \quad \Gamma_1^k \right\} \subset V^k,$$

and we define $V_h := V_h^m \times V_h^s$. For the discretization of the Lagrange multiplier space we use dual shape functions, introduced in [Woh00]. In the case of linear or bilinear finite elements in two dimensions, the dual basis functions are associated with the vertices. We use discontinuous piecewise linear functions having value two at the associated vertex and value minus one at the two neighboring vertices as basis functions. We denote this discrete Lagrange multiplier space by $M_h = \operatorname{span}\{\psi_p,\ p \in \mathcal{S}\}$, where ψ_p denotes the basis function associated with the vertex p. Then the biorthogonality of the basis functions yields

$$\langle \psi_p, \phi_q \rangle_{\Gamma_3} \,=\, \delta_{pq} \int_{\Gamma_3} \phi_q \, ds, \qquad p, q \in \mathcal{S}, \tag{3.1}$$

where ϕ_q are the standard nodal basis functions of $S_1(\Omega^s, \mathcal{T}_{h,\Omega^s})$ associated with the vertex q. The finite element space V_h can be written in terms of the standard finite element basis $\boldsymbol{\phi}$ as $V_h = \operatorname{span}\{\phi_p,\ p \in \mathcal{S} \cup \mathcal{M} \cup \mathcal{N}\}$. In addition to the basis $\boldsymbol{\phi}$, we introduce the constrained finite element basis $\hat{\boldsymbol{\phi}}$ (see [WK03]). To introduce these basis functions, we define the entries of the coupling matrices D and M between the finite element basis functions ϕ_p and the basis functions for the Lagrange multiplier space ψ_p by

$$\begin{aligned} D[p,q] &:= \langle \psi_p, \phi_q \rangle_{\Gamma_3}, \quad p,\, q \in \mathcal{S}, \\ M[p,q] &:= \langle \psi_p, \phi_q \rangle_{\Gamma_3}, \quad p \in \mathcal{S},\, q \in \mathcal{M}. \end{aligned}$$

Due to the biorthogonality (3.1), the matrix D is diagonal. Defining $\hat{M} := D^{-1}M$, we obtain the constrained basis $\hat{\boldsymbol{\phi}}$ of V_h from the nodal basis $\boldsymbol{\phi}$ of V_h by the transformation

$$\hat{\boldsymbol{\phi}} := \begin{pmatrix} \hat{\phi}_{\mathcal{N}} \\ \hat{\phi}_{\mathcal{M}} \\ \hat{\phi}_{\mathcal{S}} \end{pmatrix} := \begin{pmatrix} \mathrm{Id} & 0 & 0 \\ 0 & \mathrm{Id} & \hat{M}^\top \\ 0 & 0 & \mathrm{Id} \end{pmatrix} \begin{pmatrix} \phi_{\mathcal{N}} \\ \phi_{\mathcal{M}} \\ \phi_{\mathcal{S}} \end{pmatrix} =: Q\boldsymbol{\phi}. \tag{3.2}$$

We note that only basis functions associated with a node $p \in \mathcal{M}$ are changed and that by definition

$$b(\hat{\phi}_q, \psi_p) = 0, \quad p \in \mathcal{S},\, q \in \mathcal{M}.$$

For simplicity of notation, we use the same symbol for a function in V_h and M_h as for its algebraic representation with respect to the nodal basis. Let v_h be the algebraic representation of an element $v_h \in V_h$ with respect to the basis $\boldsymbol{\phi}$ and let $\hat{v}_h$ be the corresponding algebraic representation with respect to the constrained basis $\hat{\boldsymbol{\phi}}$. Then we have the relation $v_h = Q^\top \hat{v}_h$. Now, after an easy computation and taking into account the biorthogonality (3.1), we get

$$b(v_h, \mu_h) = \langle \mu_h, \sum_{p\in\mathcal{S}} v_p\phi_p - \sum_{q\in\mathcal{M}} v_q\phi_q \rangle_{\Gamma_3} = \langle \mu_h, \sum_{p\in\mathcal{S}} \hat{v}_p\hat{\phi}_p \rangle_{\Gamma_3} = \langle \mu_h, \sum_{p\in\mathcal{S}} \hat{v}_p\phi_p \rangle_{\Gamma_3}.$$

Before introducing the discrete set Λ_h for the admissible Lagrange multipliers, we define for $v_h \in V_h$ the restriction to the slave side of the interface Γ_3 by

$$v_{h,\mathcal{S}} := \sum_{p\in\mathcal{S}} v_p\phi_p.$$

Now we define the discrete convex set Λ_h by

$$\Lambda_h := \Big\{ \mu_h \in M_h \,:\, \langle \mu_h, v_{h,\mathcal{S}} \rangle_{\Gamma_3} \le \int_{\Gamma_3} g|v_{h,\mathcal{S}}|_h \, ds, \quad v_h \in V_h \Big\}, \tag{3.3}$$

where the mesh dependent absolute value $|v_{h,\mathcal{S}}|_h$ of the function $v_{h,\mathcal{S}}$ is given by

$$|v_{h,\mathcal{S}}|_h := \sum_{p\in\mathcal{S}} |v_p|\phi_p.$$

We remark that in general $|v_{h,\mathcal{S}}|_h \neq |v_{h,\mathcal{S}}|$. Henceforth in this section, we assume that g is a strictly positive constant. In this case, the convex set Λ_h can be written equivalently as

$$\Lambda_h := \Big\{ \mu_h \in M_h \,:\, \mu_h = \sum_{p\in\mathcal{S}} \gamma_p\psi_p, \quad |\gamma_p| \le g, \quad p \in \mathcal{S} \Big\}. \tag{3.4}$$

The discrete formulation of Problem 2.2 is as follows.

PROBLEM 3.1. [illegible] $u_h \in V_h$ [illegible] $\lambda_h \in \Lambda_h$ [illegible]

$$\begin{aligned} a(u_h, v_h) + b(v_h, \lambda_h) &= (f, v_h)_V, \qquad && v_h \in V_h, \\ b(u_h, \zeta_h - \lambda_h) &\leq 0, && \zeta_h \in \Lambda_h. \end{aligned}$$

Using the discrete inf-sup property for the spaces M_h and V_h (see, e.g., [Woh00]), we get the existence and the uniqueness of the solution. In order to obtain an optimal a priori error estimate, several lemmas will be proved. We note that $\Lambda_h \not\subset \Lambda$.

Before presenting the first lemma, we have to consider for a function $v_h \in V_h$ the discrete jump $\hat{v}_{h,\mathcal{S}}$ on the interface Γ_3 in the constrained basis and its mesh dependent absolute value by

$$\hat{v}_{h,\mathcal{S}} = \sum_{p \in \mathcal{S}} \hat{v}_p \phi_p, \quad |\hat{v}_{h,\mathcal{S}}|_h = \sum_{p \in \mathcal{S}} |\hat{v}_p| \phi_p,$$

respectively. Now we can prove the following lemma.

LEMMA 3.1. [illegible] (u, λ) [illegible] 2.2 [illegible] (u_h, λ_h) [illegible] 3.1 [illegible]

$$b(u, \lambda) = \int_{\Gamma_3} g|[u]|\, ds, \qquad b(u_h, \lambda_h) = \int_{\Gamma_3} g|\hat{u}_{h,\mathcal{S}}|_h\, ds.$$

[illegible]. In order to prove the first equality, we define $\mu \in M$ by

$$\langle \mu, v \rangle_{\Gamma_3} := \int_{\Gamma_3} g\ \mathrm{sgn}([u])\, v\, ds, \quad v \in H^{1/2}(\Gamma_3).$$

Thus, we have $\mu \in \Lambda$. Using the definition of Λ and taking into account that

$$b(u, \mu - \lambda) \leq 0, \qquad \mu \in \Lambda,$$

we can write

$$\int_{\Gamma_3} g\,|[u]|\, ds \geq \langle \lambda, [u] \rangle_{\Gamma_3} \geq \langle \mu, [u] \rangle_{\Gamma_3} = \int_{\Gamma_3} g\ \mathrm{sgn}([u])\,[u]\, ds = \int_{\Gamma_3} g\,|[u]|\, ds.$$

Keeping in mind (2.21), we have the first relation.

In order to prove the second equality, we consider $\zeta_h = \sum_{p \in \mathcal{S}} g\ \mathrm{sgn}(\hat{u}_p)\, \psi_p$. Obviously, we have $\zeta_h \in \Lambda_h$. Due to the relation

$$b(u_h, \zeta_h - \lambda_h) \leq 0, \quad \zeta_h \in \Lambda_h,$$

we deduce that

$$b(u_h, \lambda_h) \geq \langle \sum_{p \in \mathcal{S}} g\ \mathrm{sgn}(\hat{u}_p) \psi_p, \hat{u}_{h,\mathcal{S}} \rangle_{\Gamma_3}.$$

Using the biorthogonality relation of the basis function (3.1), we get

$$\begin{aligned} \left\langle \sum_{p \in \mathcal{S}} g\ \mathrm{sgn}(\hat{u}_p)\, \psi_p, \hat{u}_{h,\mathcal{S}} \right\rangle_{\Gamma_3} &= \sum_{p \in \mathcal{S}} g\ \mathrm{sgn}(\hat{u}_p)\, \hat{u}_p \int_{\Gamma_3} \phi_p\, ds = \sum_{p \in \mathcal{S}} g\,|\hat{u}_p| \int_{\Gamma_3} \phi_p\, ds \\ &= \int_{\Gamma_3} g\,|\hat{u}_{h,\mathcal{S}}|_h\, ds. \end{aligned}$$

From this, we obtain

$$b(u_h, \lambda_h) = \langle \lambda_h, \hat{u}_{h,\mathcal{S}} \rangle_{\Gamma_3} \geq \int_{\Gamma_3} g\, |\hat{u}_{h,\mathcal{S}}|_h \, ds.$$

Taking into account (2.21) and the definition of Λ_h, we have

$$b(u_h, \lambda_h) \leq \int_{\Gamma_3} g |\hat{u}_{h,\mathcal{S}}|_h \, ds.$$

Combining the last inequalities, we deduce the second relation. □

Furthermore, the following result holds.

LEMMA 3.2. [illegible] $(u, \lambda) \in V \times \Lambda$ [illegible] 2.2 [illegible] $(u_h, \lambda_h) \in V_h \times \Lambda_h$ [illegible] 3.1 [illegible] C [illegible] h, [illegible] $v_h \in V_h$, $\mu_h \in M_h$,

$$\|u - u_h\|_V^2 + \|\lambda - \lambda_h\|_M^2 \leq C \left\{ \|u - v_h\|_V^2 + \|\lambda - \mu_h\|_M^2 \right\} + b(u, \lambda_h - \lambda).$$

[illegible]. The proof follows directly from the arguments given in [HL02]. For the convenience of the reader, we adapt the proof to our situation and take into account the different definitions of Λ and Λ_h. Evaluating $a(u - u_h, u - u_h)$, we find for $v_h \in V_h$ the relation

$$\begin{aligned} a(u - u_h, u - u_h) &= a(u - u_h, u - v_h) + a(u - u_h, v_h - u_h) \\ &= a(u - u_h, u - v_h) - b(v_h - u_h, \lambda) + b(v_h - u_h, \lambda_h) \\ &= a(u - u_h, u - v_h) - b(v_h - u, \lambda - \lambda_h) - b(u - u_h, \lambda - \lambda_h). \end{aligned}$$

Due to the V-ellipticity of $a(\cdot, \cdot)$ and continuity of $a(\cdot, \cdot)$ and $b(\cdot, \cdot)$, we deduce from this expression, using a trace theorem, the inequality

$$\|u - u_h\|_V^2 \leq C \left(\|u - u_h\|_V + \|\lambda - \lambda_h\|_M \right) \|u - v_h\|_V - b(u - u_h, \lambda - \lambda_h). \tag{3.5}$$

Here and below, we denote by C a positive constant independent of the mesh size, whose value may change from place to place. Using Lemma 3.1, we can write for the last term on the previous estimate

$$-b(u - u_h, \lambda - \lambda_h) = b(u, \lambda_h - \lambda) + b(u_h, \lambda) - \int_{\Gamma_3} g |\hat{u}_{h,\mathcal{S}}|_h \, ds,$$

and from this, since $b(u_h, \lambda) \leq \int_{\Gamma_3} g |\hat{u}_{h,\mathcal{S}}| \, ds$ and $|\hat{u}_{h,\mathcal{S}}| \leq |\hat{u}_{h,\mathcal{S}}|_h$, we deduce that

$$-b(u - u_h, \lambda - \lambda_h) \leq b(u, \lambda_h - \lambda). \tag{3.6}$$

On the other hand, taking into account the inf-sup property of the form $b(\cdot, \cdot)$, we find that

$$\begin{aligned} \|\mu_h - \lambda_h\|_M &\leq \alpha \sup_{0 \neq w_h \in V_h} \frac{b(w_h, \mu_h - \lambda_h)}{\|w_h\|_V} \\ &= \alpha \sup_{0 \neq w_h \in V_h} \frac{b(w_h, \mu_h - \lambda) + a(u_h - u, w_h)}{\|w_h\|_V} \\ &\leq C(\|\mu_h - \lambda\|_M + \|u_h - u\|_V), \end{aligned}$$

and, using the triangle inequality, we can write

$$\|\lambda - \lambda_h\|_M \le C(\|\mu_h - \lambda\|_M + \|u_h - u\|_V). \tag{3.7}$$

We conclude our proof using (3.5), (3.6), and (3.7). □

To estimate the first two terms on the right side of the inequality of Lemma 3.2, we use the approximation properties of the spaces V_h and M_h. In addition, we have to estimate the residual term $b(u, \lambda_h - \lambda)$. To this end, we define $\overline{\gamma_{sl}} := \text{supp}\,[u] \subset \Gamma_3$, $\gamma_{st} := \Gamma_3 \backslash \gamma_{sl}$, and we introduce the sets

$$\mathcal{W}^* := \overline{\mathring{\gamma}_{st}} \cap \overline{\gamma_{sl}},$$
$$\mathcal{W}^0 := \Big\{ w \in \Gamma_3 \,:\, [u](w) = 0 \text{ and } \exists\, r > 0 \,:\, [u](w-\varepsilon)\,[u](w+\varepsilon) < 0, \quad 0 < \epsilon < r \Big\},$$
$$\mathcal{W} := \mathcal{W}^* \cup \mathcal{W}^0.$$

Everywhere below we will use the following assumption.

3.1. The number of points in $\mathcal{W}$ is finite.

The minimum distance between elements in $\mathcal{W}$ is denoted by a, i.e.,

$$a := \inf\{|w_j - w_k| \,:\, 1 \le j \ne k \le N_w\},$$

where N_w denotes the number of points in $\mathcal{W}$. By Assumption 3.1, $N_w < \infty$ and thus $a > 0$. For $h < \frac{a}{2} =: h_0$, we find between two neighboring points in $\mathcal{W}$ at [illegible] in $\mathcal{S}$. Then the following lemma holds.

LEMMA 3.3. [illegible] $(u, \lambda) \in V \times \Lambda$ [illegible] 2.2 [illegible] $(u_h, \lambda_h) \in V_h \times \Lambda_h$ [illegible] 3.1 [illegible] 3.1 [illegible] $u^k \in H^{\frac{3}{2}+\nu}(\Omega^k)$ $0 < \nu \le \frac{1}{2}$ $k = m, s,$ [illegible]

$$b(u, \lambda_h - \lambda) \le C h^{\frac{1}{2}+\nu} \|\lambda - \lambda_h\|_M \sum_{k=m,s} |u^k|_{\frac{3}{2}+\nu,\Omega^k}$$

[illegible] C [illegible] $h < h_0$

[illegible]. For each element $w \in \mathcal{W}$, we can choose an element in $\mathcal{S}$ denoted by $p_{j(w)}$, such that

$$|p_{j(w)} - w| = \min_{p \in \mathcal{S},\, [u](p) \ne 0} |w - p| \quad \text{if} \quad w \in \mathcal{W} \backslash \mathcal{S},$$
$$p_{j(w)} = w \qquad\qquad\qquad\qquad \text{if} \quad w \in \mathcal{S}.$$

Let us consider the set $\tilde{\mathcal{S}}$, defined by

$$\tilde{\mathcal{S}} := \Big\{ p \in \mathcal{S} \,:\, \exists\, w \in \mathcal{W} \text{ s.t. } p = p_{j(w)} \Big\}.$$

For this set, we consider the disjoint decomposition

$$\tilde{\mathcal{S}} = \mathcal{S}^* \cup \mathcal{S}^l \cup \mathcal{S}^r,$$

where

$$\mathcal{S}^* := \Big\{ p \in \mathcal{S} \,:\, \exists\, w \in \mathcal{W}^* \text{ s.t. } p = p_{j(w)} \Big\},$$
$$\mathcal{S}^l := \Big\{ p \in \mathcal{S} \,:\, \exists\, w \in \mathcal{W}^0 \text{ s.t. } p = p_{j(w)} \text{ and } p_{j(w)} \text{ is on the left side of } w \Big\},$$
$$\mathcal{S}^r := \Big\{ p \in \mathcal{S} \,:\, \exists\, w \in \mathcal{W}^0 \text{ s.t. } p = p_{j(w)} \text{ and } p_{j(w)} \text{ is on the right side of } w \Big\};$$

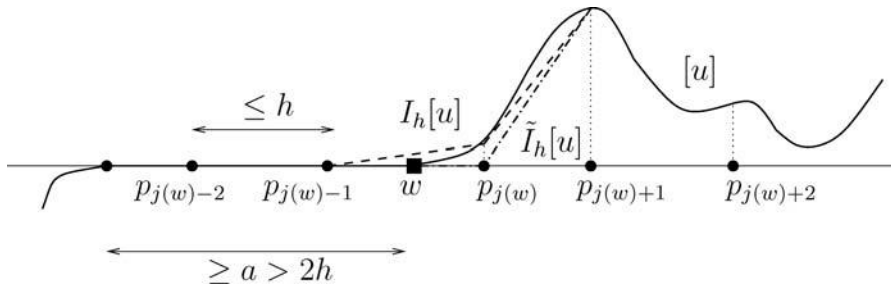


FIG. 3.1. $w \in \mathcal{W}^*$, $p_{j(w)} \in \mathcal{S}^*$, $[u]$, $I_h[u]$, $\tilde{I}_h[u]$.

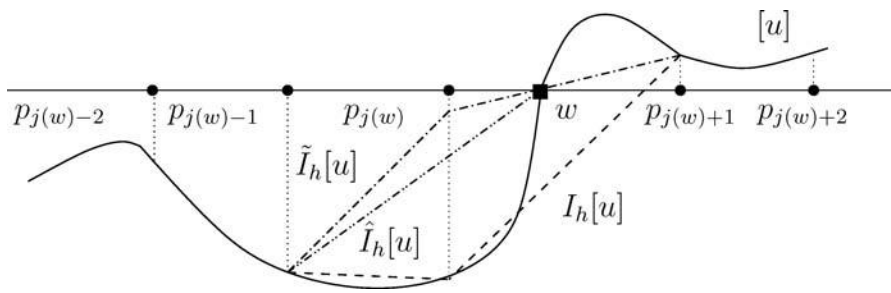


FIG. 3.2. $w \in \mathcal{W}^0$, $p_{j(w)} \in \mathcal{S}^l$, $[u]$, $I_h[u]$, $\tilde{I}_h[u]$, $\hat{I}_h[u]$.

see Figure 3.1 for an example of an element in $\mathcal{S}^*$ and Figure 3.2 for an example of an element in $\mathcal{S}^l$. Let I_h be the standard interpolation operator; that is, on Γ_3

$$I_h[u] = \sum_{p \in \mathcal{S}} [u](p)\phi_p.$$

Let $\tilde{I}_h$ be a modified interpolation operator such that

$$(\tilde{I}_h[u])(p) = \begin{cases} [u](p) & \text{if} \quad p \in \mathcal{S} \backslash \tilde{\mathcal{S}}, \\ [u](p_{j(w)+1})\frac{s^l}{s^l-1} & \text{if} \quad p = p_{j(w)} \in \mathcal{S}^l, \ s^l := \frac{|w-p_{j(w)}|}{|p_{j(w)+1}-p_{j(w)}|}, \\ [u](p_{j(w)-1})\frac{s^r}{s^r-1} & \text{if} \quad p = p_{j(w)} \in \mathcal{S}^r, \ s^r := \frac{|p_{j(w)}-w|}{|p_{j(w)}-p_{j(w)-1}|}, \\ 0 & \text{if} \quad p = p_{j(w)} \in \mathcal{S}^*; \end{cases}$$

see, e.g., Figures 3.1 and 3.2. In the construction of $\tilde{I}_h$ the value of I_h at the point $p_{j(w)}$ is modified such that $(\tilde{I}_h[u])(w) = 0$ holds. In order to simplify the writing, everywhere below we will use the notation $p_{j\pm l}$ instead of $p_{j(w)\pm l}$ for $l \in \{0, 1, 2\}$. We note that $0 < s^l, s^r \leq \frac{1}{2}$. Moreover, the identity

$$[u]\ \tilde{I}_h[u] \geq 0 \tag{3.8}$$

holds on Γ_3. Finally, we introduce the Lagrange interpolation operator $\hat{I}_h$ with respect to the nodes $(\mathcal{S} \backslash (\mathcal{S}^l \cup \mathcal{S}^r)) \cup \mathcal{W}^0$. Due to the shape regularity of the grid given by the nodes in $\mathcal{S}$, and because of $0 < s^l, s^r \leq \frac{1}{2}$, the new grid is also shape regular.

Let us evaluate $b(u, \lambda_h - \lambda)$ using the interpolation operators introduced above. First, using I_h and $\tilde{I}_h$, we can write

$$b(u, \lambda_h - \lambda) \leq \langle \lambda_h - \lambda, [u] - I_h[u] \rangle_{\Gamma_3} + \langle \lambda_h - \lambda, I_h[u] - \tilde{I}_h[u] \rangle_{\Gamma_3} + \langle \lambda_h - \lambda, \tilde{I}_h[u] \rangle_{\Gamma_3}. \tag{3.9}$$

For the first term in the right side of the previous inequality, we get

$$\langle \lambda_h - \lambda, [u] - I_h[u] \rangle_{\Gamma_3} \leq \|\lambda_h - \lambda\|_M \|[u] - I_h[u]\|_{\frac{1}{2},\Gamma_3},$$

and from this, we obtain

$$\langle \lambda_h - \lambda, [u] - I_h[u] \rangle_{\Gamma_3} \leq C\|\lambda_h - \lambda\|_M h^{1/2+\nu} \sum_{k=m,s} |u^k|_{\frac{3}{2}+\nu,\Omega^k}. \tag{3.10}$$

In order to evaluate the second term, we use the inverse inequality to write

$$\|I_h[u] - \tilde{I}_h[u]\|^2_{\frac{1}{2},\Gamma_3} \leq \frac{C}{h}\|I_h[u] - \tilde{I}_h[u]\|^2_{0,\Gamma_3},$$

and from this, we find that

$$\|I_h[u] - \tilde{I}_h[u]\|^2_{\frac{1}{2},\Gamma_3} \leq \frac{C}{h} \sum_{p_j \in \mathcal{S}^l \cup \mathcal{S}^r} \|I_h[u] - \tilde{I}_h[u]\|^2_{0,[p_{j-1},p_{j+1}]} + C \sum_{p_j \in \mathcal{S}^*} \Big([u](p_j)\Big)^2. \tag{3.11}$$

We note that for each $p_j \in \mathcal{S}^*$ we have

$$0 = (\tilde{I}_h[u])(p_j) \neq (I_h[u])(p_j) = [u](p_j).$$

Let p_j be an element in $\mathcal{S}^l$ and let us evaluate $\|I_h[u] - \tilde{I}_h[u]\|^2_{0,[p_{j-1},p_{j+1}]}$. First, we note that

$$\|I_h[u] - \tilde{I}_h[u]\|^2_{0,[p_{j-1},p_{j+1}]} \leq Ch\Big((I_h[u] - \tilde{I}_h[u])(p_j)\Big)^2.$$

Furthermore we have

$$(I_h[u] - \tilde{I}_h[u])(p_j) = \frac{1}{1-s^l}\Big((1-s^l)[u](p_j) + s^l[u](p_{j+1})\Big),$$

and from this, we obtain the identity

$$(I_h[u] - \tilde{I}_h[u])(p_j) = \frac{1}{1-s^l}(I_h[u])(w). \tag{3.12}$$

Since $0 < s^l \leq \frac{1}{2}$, we get, using also (3.12),

$$\|I_h[u] - \tilde{I}_h[u]\|^2_{0,[p_{j-1},p_{j+1}]} \leq Ch\Big((I_h[u] - \hat{I}_h[u])(w)\Big)^2.$$

Now we deduce that

$$\|I_h[u] - \tilde{I}_h[u]\|^2_{0,[p_{j-1},p_{j+1}]} \leq C\|I_h[u] - \hat{I}_h[u]\|^2_{0,[p_{j-1},p_{j+1}]}.$$

Using the triangle inequality and the interpolation error for Lagrange interpolation operators, we find that

$$\|I_h[u] - \tilde{I}_h[u]\|^2_{0,[p_{j-1},p_{j+1}]} \leq Ch\Big(h^{1/2+\nu}|[u]|_{1+\nu,[p_{j-1},p_{j+1}]}\Big)^2. \tag{3.13}$$

In a similar way, for each p_j in $\mathcal{S}^r$, we can also find the estimate (3.13). In order to estimate the last term in (3.11), let us introduce the notation

$$\mathcal{S}^{*l} := \big\{p_j \in \mathcal{S}^* \,:\, [u](p_{j+1}) = 0\big\}, \qquad \mathcal{S}^{*r} := \big\{p_j \in \mathcal{S}^* \,:\, [u](p_{j-1}) = 0\big\}. \tag{3.14}$$

We note that for each $p_j \in \mathcal{S}^{*l}$, $[u] \neq 0$ on $[p_j, w]$ and $[u] = 0$ on $(w, p_{j+2}]$. In addition, for each $p_j \in \mathcal{S}^{*r}$, $[u] \neq 0$ on $[w, p_j]$ and $[u] = 0$ on $[p_{j-2}, w)$. The regularity assumption on u yields $[u] \in H^{1+\nu}(\Gamma_3)$, and following [HW05a], for each point $p_j \in \mathcal{S}^{*l}$,

$$\Big([u](p_j)\Big)^2 \leq Ch^{1+2\nu}|[u]|^2_{1+\nu,[p_j,\,p_{j+2}]},$$

and for each point $p_j \in \mathcal{S}^{*r}$,

$$\Big([u](p_j)\Big)^2 \leq Ch^{1+2\nu}\,|[u]|^2_{1+\nu,[p_{j-2},\,p_j]}.$$

Using these last estimates for each $p_j \in \mathcal{S}^*$ and (3.13) for each $p_j \in \mathcal{S}^r \cup \mathcal{S}^l$, by summing and using a trace theorem we get

$$\|I_h[u] - \tilde{I}_h[u]\|_{\frac{1}{2},\Gamma_3} \leq C\,h^{\frac{1}{2}+\nu} \sum_{k=m,s} |u^k|_{\frac{3}{2}+\nu,\Omega^k}. \tag{3.15}$$

In the final step, we show that

$$\langle \lambda_h - \lambda, \tilde{I}_h[u]\rangle_{\Gamma_3} \leq 0. \tag{3.16}$$

First, we can find a constant $\delta > 0$ such that

$$\operatorname{sgn}[u] = \operatorname{sgn}\Big([u] - \delta(\tilde{I}_h[u])\Big) \quad \text{on} \quad \Gamma_3.$$

Using (3.8) we can write, on Γ_3,

$$|[u]| = |\delta(\tilde{I}_h[u])| + |[u] - \delta(\tilde{I}_h[u])|.$$

Taking into account Lemma 3.1, we get

$$\langle \lambda, [u]\rangle_{\Gamma_3} = \int_{\Gamma_3} g\,|\delta(\tilde{I}_h[u])|\,ds + \int_{\Gamma_3} g\,|[u] - \delta(\tilde{I}_h[u])|\,ds. \tag{3.17}$$

On the other hand, we have

$$\langle \lambda, [u]\rangle_{\Gamma_3} = \langle \lambda, \delta(\tilde{I}_h[u])\rangle_{\Gamma_3} + \langle \lambda, [u] - \delta(\tilde{I}_h[u])\rangle_{\Gamma_3}, \tag{3.18}$$

and due to the definition of Λ, we obtain

$$\langle \lambda, \delta(\tilde{I}_h[u])\rangle_{\Gamma_3} \leq \int_{\Gamma_3} g\,|\delta(\tilde{I}_h[u])|\,ds, \tag{3.19}$$

$$\langle \lambda, [u] - \delta(\tilde{I}_h[u])\rangle_{\Gamma_3} \leq \int_{\Gamma_3} g\,|[u] - \delta(\tilde{I}_h[u])|\,ds. \tag{3.20}$$

Using (3.17)–(3.20), we deduce that in (3.19) and (3.20) we can write identities. Since $\delta > 0$ we get

$$\langle \lambda, \tilde{I}_h[u]\rangle_{\Gamma_3} = \int_{\Gamma_3} g\,|\tilde{I}_h[u]|\,ds. \tag{3.21}$$

Now, let us write λ_h as a linear combination of the dual basis functions ψ_p and $\tilde{I}_h[u]$ as a linear combination of the standard basis functions ϕ_p; that is,

$$\lambda_h = \sum_{p\in\mathcal{S}} \gamma_p \psi_p, \quad \tilde{I}_h[u] = \sum_{p\in\mathcal{S}} \alpha_p \phi_p, \text{ where } \alpha_p = \tilde{I}_h[u](p).$$

Keeping in mind (3.1) and (3.21), we obtain

$$\langle \lambda_h - \lambda, \tilde{I}_h[u]\rangle_{\Gamma_3} = \int_{\Gamma_3} \left(\sum_{p\in(\mathcal{S}\setminus\mathcal{S}^*)} \gamma_p\alpha_p\phi_p - g\Big| \sum_{p\in(\mathcal{S}\setminus\mathcal{S}^*)} \alpha_p\phi_p\Big| \right) ds. \tag{3.22}$$

If we consider the subsets of Γ_3 defined by

$$\Gamma_3^+ := \Big\{ \boldsymbol{x} \in \Gamma_3 \,:\, [u](\boldsymbol{x}) > 0 \Big\} \quad \text{and} \quad \Gamma_3^- := \Big\{ \boldsymbol{x} \in \Gamma_3 \,:\, [u](\boldsymbol{x}) < 0 \Big\},$$

then, taking into account (3.8), we deduce that

$$\begin{aligned} &\tilde{I}_h[u](p) > 0, \qquad p \in \Gamma_3^+ \cap (\mathcal{S}\setminus\mathcal{S}^*), \\ &\tilde{I}_h[u](p) < 0, \qquad p \in \Gamma_3^- \cap (\mathcal{S}\setminus\mathcal{S}^*), \\ &\operatorname{supp} \tilde{I}_h[u] \subset \operatorname{supp}[u]. \end{aligned} \tag{3.23}$$

From this, using (3.22), we can write

$$\langle \lambda_h - \lambda, \tilde{I}_h[u]\rangle_{\Gamma_3} = \int_{\Gamma_3^+} \sum_{p\in(\mathcal{S}\setminus\mathcal{S}^*)} (\gamma_p - g)\alpha_p\phi_p \, ds + \int_{\Gamma_3^-} \sum_{p\in(\mathcal{S}\setminus\mathcal{S}^*)} (\gamma_p + g)\alpha_p\phi_p \, ds.$$

Keeping in mind (3.4) and (3.23), we get (3.16); i.e., the last term in the right side of the inequality (3.9) is negative. Taking into account (3.9)–(3.16) we conclude the proof of Lemma 3.3. □

Based on the results obtained in Lemmas 3.2 and 3.3 and using the well-known approximation properties for the spaces V_h and M_h, by applying Young's inequality, the following theorem holds.

THEOREM 3.4. [illegible] $(u,\lambda) \in V\times\Lambda$ [illegible] 2.2 [illegible] $(u_h,\lambda_h) \in V_h \times \Lambda_h$ [illegible] 3.1 [illegible] 3.1 [illegible] $u^k \in H^{\frac{3}{2}+\nu}(\Omega^k)$ $0 < \nu \le \frac{1}{2}$ $k = m, s,$ [illegible]

$$\|u - u_h\|_V + \|\lambda - \lambda_h\|_M \le C h^{\frac{1}{2}+\nu} \sum_{k=m,s} |u^k|_{\frac{3}{2}+\nu,\Omega^k}$$

[illegible] C [illegible] $h < h_0$.

4. Inexact primal-dual active set strategy. In this section, we present our algorithm to solve the discrete nonlinear Problem 3.1. We use an inexact primal-dual active set strategy. Primal-dual active set strategies for problems with friction were also considered in [Sta04]. In contrast to [Sta04], we do not use a penalty parameter and use a different nonlinear complementarity function. In [Sta04] the method can be interpreted as a semismooth Newton method. The nonlinear complementarity function used in the present approach can only be interpreted as a semismooth quasi-Newton method. Let $u_h \in V_h$ and $\lambda_h \in \Lambda_h$ be the solution of the discrete Problem

3.1. For the discrete solution (u_h, λ_h) of Problem 3.1, we get due to $\lambda_h \in \Lambda_h$ and from the definition (3.3) of the space Λ_h the condition

$$|\lambda_{p,s}| \leq g_p := \int_{\Gamma_3} g\phi_p \, ds, \qquad p \in \mathcal{S}, \tag{4.1}$$

where the scaled value $\lambda_{p,s}$ of λ_p is given by $\lambda_{p,s} := \lambda_p D[p,p]$. In terms of Lemma 3.1, the inequality condition of Problem 3.1 results for $g > 0$ in the algebraic system

$$\lambda_{p,s}\hat{u}_p = g_p|\hat{u}_p|, \qquad p \in \mathcal{S}. \tag{4.2}$$

We decompose $\mathcal{S}$ into two disjoint sets $\mathcal{A}$ and $\mathcal{I}$ such that

$$|\lambda_{p,s}| = g_p \ \text{ for } \ p \in \mathcal{A}, \qquad \hat{u}_p = 0 \ \text{ for } \ p \in \mathcal{I}, \tag{4.3}$$

and we refer to $\mathcal{A}$ and $\mathcal{I}$ as active and inactive sets, respectively. We observe that the nonlinear constraints (4.1) and (4.2) are the discrete constraints of (2.14). The constraints (4.1) and (4.2) can at each point with $\lambda_p \neq 0$ equivalently be written as

$$|\lambda_{p,s}| \leq g_p, \qquad \big(\lambda_p\hat{u}_p\big)\big(|\lambda_{p,s}| - g_p\big) = 0, \qquad \lambda_p\hat{u}_p \geq 0, \qquad p \in \mathcal{S}. \tag{4.4}$$

We introduce a nonlinear complementarity function $C(\hat{u}_p,\, \lambda_p)$ defined by

$$C(\hat{u}_p,\, \lambda_p) := \hat{u}_p\lambda_p - \max\big\{0,\, \hat{u}_p\lambda_p + c\,(|\lambda_{p,s}| - g_p)\big\}, \qquad c > 0. \tag{4.5}$$

The equivalence between the constraints (4.4) to the condition $C(\hat{u}_p,\, \lambda_p) = 0$ can be easily shown by a straightforward computation. We remark that in the nonlinear complementarity function we use the nodal value λ_p and the scaled nodal value $\lambda_{p,s}$. To guarantee equivalence between $C(\hat{u}_p,\, \lambda_p) = 0$, $p \in \mathcal{S}$, to (4.1) and (4.2), our algorithm has to make sure that if $C(\hat{u}_p,\, \lambda_p) = 0$ and $\lambda_p = 0$ we also have $\hat{u}_p = 0$. This point will be discussed after introducing the algorithm. To determine the sets $\mathcal{A}$ and $\mathcal{I}$ we use an iterative scheme based on the nonlinear complementarity function (4.5). Let k be the corresponding iteration index. The sign of $\lambda^k_{p,s}$ for $p \in \mathcal{A}_k$ is defined by the previous step. With the decomposition of the vertices introduced in section 3, we set the solution vector $u_h = (u_\mathcal{N},\, u_\mathcal{M},\, u_\mathcal{S})^\top$ and for the Lagrange multiplier we set $\lambda_h = \lambda_\mathcal{S}$. For the first line of Problem 3.1, the algebraic system in the standard basis $\boldsymbol{\phi}$ reads $A_h u_h + B_h\lambda_h = f_h$ with $B = (0,\, -M^\top,\, D)^\top$. The transformation of this algebraic system to the constrained basis $\hat{\boldsymbol{\phi}}$, introduced in (3.2), leads to the system $\hat{A}_h\hat{u}_h + \hat{B}_h\lambda_h = \hat{f}_h$, where we have the relations $\hat{A}_h = QA_hQ^\top$ and $\hat{f}_h = Qf_h$. The coupling matrix $\hat{B}_h$ in the constrained basis $\hat{\boldsymbol{\phi}}$ has the form $\hat{B}_h = (0,\, 0,\, D)^\top$. Combining this system with (4.3) and eliminating the Lagrange multiplier leads to

$$\begin{pmatrix} \hat{A}_{\mathcal{N}\mathcal{N}} & \hat{A}_{\mathcal{N}\mathcal{M}} & \hat{A}_{\mathcal{N}\mathcal{A}} & \hat{A}_{\mathcal{N}\mathcal{I}} \\ \hat{A}_{\mathcal{M}\mathcal{N}} & \hat{A}_{\mathcal{M}\mathcal{M}} & \hat{A}_{\mathcal{M}\mathcal{A}} & \hat{A}_{\mathcal{M}\mathcal{I}} \\ \hat{A}_{\mathcal{A}\mathcal{N}} & \hat{A}_{\mathcal{A}\mathcal{M}} & \hat{A}_{\mathcal{A}\mathcal{A}} & \hat{A}_{\mathcal{A}\mathcal{I}} \\ 0 & 0 & 0 & \mathrm{Id}_{\mathcal{I}\mathcal{I}} \end{pmatrix} \begin{pmatrix} \hat{u}_\mathcal{N} \\ \hat{u}_\mathcal{M} \\ \hat{u}_\mathcal{A} \\ \hat{u}_\mathcal{I} \end{pmatrix} = \begin{pmatrix} \hat{f}_\mathcal{N} \\ \hat{f}_\mathcal{M} \\ \hat{f}_\mathcal{A} + g_\mathcal{A} \\ 0 \end{pmatrix}, \tag{4.6}$$

where $\hat{A}_{k,l}$, $k,l \in \{\mathcal{N}, \mathcal{M}, \mathcal{A}, \mathcal{I}\}$, are the block stiffness matrices of $\hat{A}_h$ associated with the corresponding basis functions. The entries of the vectors $g_\mathcal{A}$ are the values $\pm g_p$ with $p \in \mathcal{A}$. We remark that the sign of the entry g_p of $g_\mathcal{A}$ at the point $p \in \mathcal{A}$ will be specified by the sign of λ_p in the previous iteration step. The Lagrange multiplier can be computed by the residual, i.e.,

$$\lambda_h = D^{-1}\big(\hat{f}_\mathcal{S} - \hat{A}_{\mathcal{S}\mathcal{N}}\hat{u}_\mathcal{N} - \hat{A}_{\mathcal{S}\mathcal{M}}\hat{u}_\mathcal{M} - \hat{A}_{\mathcal{S}\mathcal{S}}\hat{u}_\mathcal{S}\big). \tag{4.7}$$

To solve this linear problem, we use an optimal multigrid method. We use a $\mathcal{W}$-cycle with symmetric Gauß–Seidel iteration as smoother. We apply three pre-smoothing and three post-smoothing steps. Due to the use of a multigrid method as an iterative solver, we can formulate the inexact primal-dual active set algorithm. Based on the condition $C(\hat{u}_p, \lambda_p) = 0$, we introduce the primal-dual active set strategy as an iterative scheme. We note that primal-dual active set methods in combination with exact solvers can be analyzed as a Newton method; see, e.g., [HIK03, Sta04]. In combination with a multigrid method it can be interpreted as a nonlinear multigrid method.

ALGORITHM 1: Inexact primal-dual active set strategy

(0) Set $k = 1$, choose $c > 0$ and $m \in \mathbb{N}$.
Set $\mathcal{A}_1 = \emptyset$ and $\mathcal{I}_1 = \mathcal{S}$.
Initialize $\hat{u}_h^{1,0}$ as an initial solution for the multigrid method.

(1) For $i = 1, \ldots, m$, compute

$$\hat{u}_h^{k,i} = MG(\hat{u}_h^{k,i-1}, \mathcal{A}_k, \mathcal{I}_k).$$

(2) If $\|\hat{u}_h^{k,0} - \hat{u}_h^{k,m}\| / \|\hat{u}_h^{k,m}\|$ small enough, then stop.

(3) Set $\mathcal{A}_{k+1}$ and $\mathcal{I}_{k+1}$ to

$$\begin{aligned} \mathcal{A}_{k+1} &:= \left\{ p \in \mathcal{S} \,:\, \hat{u}_p^{k,m}\lambda_p^{k,m} + c(|\lambda_{p,s}^{k,m}| - g_p) > 0 \right\}, \\ \mathcal{I}_{k+1} &:= \left\{ p \in \mathcal{S} \,:\, \hat{u}_p^{k,m}\lambda_p^{k,m} + c(|\lambda_{p,s}^{k,m}| - g_p) \leq 0 \right\}. \end{aligned} \tag{4.8}$$

(4) Set $\hat{u}_h^{k+1,0} = \hat{u}_h^{k,m}$, set

$$(g_{\mathcal{A}})_p = \operatorname{sign}(\lambda_p^{k,m}) g_p \quad \text{for} \quad p \in \mathcal{A}_{k+1},$$

set $k = k + 1$ and go to step (1).

In this algorithm we denote by $\hat{u}_h^{k,i} = MG(\hat{u}_h^{k,i-1}, \mathcal{A}_k, \mathcal{I}_k)$ the iterate after one multigrid step for the linear system (4.6) with $\mathcal{A} = \mathcal{A}_k$ and $\mathcal{I} = \mathcal{I}_k$. The values $\lambda_p^{k,m}$ are computed by (4.7). We remark that for $m = \infty$, we solve the linear system (4.6) exactly, and we have the case of the exact primal-dual active set strategy. To motivate the choice of the active and inactive sets as presented in (4.8), we refer to Table 4.1. This table contains for any primal-dual pair $(\hat{u}_h^k, \lambda_h^k)$ satisfying $\hat{u}_p^k = 0$ for $p \in \mathcal{I}_k$ and $|\lambda_{p,s}^k| = g_p$ for $p \in \mathcal{A}_k$ the choice of the active and inactive sets $\mathcal{A}_{k+1}$ and $\mathcal{I}_{k+1}$. We point out that for each node $p \in \mathcal{A}_{k+1}$ we have $\lambda_p^{k,m} \neq 0$. Thus the vector $g_{\mathcal{A}}$ in (4.6) is always well defined. The choice of $\mathcal{I}_{k+1}$ guarantees that from $\lambda_p^{k,m} = 0$, it follows that $\hat{u}_p^{k+1,m} = 0$.

TABLE 4.1
Choice for the active set $\mathcal{A}_{k+1}$ and the inactive set $\mathcal{I}_{k+1}$ for a primal-dual pair $(\hat{u}_h^k, \lambda_h^k)$ satisfying $\hat{u}_p^k = 0$ for $p \in \mathcal{I}_k$ and $|\lambda_{p,s}^k| = g_p$ for $p \in \mathcal{A}_k$.

		$p \in$	$\hat{u}_p^k\lambda_p^k + c(\lvert\lambda_{p,s}^k\rvert - g_p)$	choice
$\hat{u}_p^k = 0$	$\lvert\lambda_{p,s}^k\rvert \leq g_p$	$\mathcal{I}_k$	≤ 0	$\mathcal{I}_{k+1}$
	$\lvert\lambda_{p,s}^k\rvert > g_p$	$\mathcal{I}_k$	> 0	$\mathcal{A}_{k+1}$
$\lvert\lambda_{p,s}^k\rvert = g_p$	$\hat{u}_p^k\lambda_p^k > 0$	$\mathcal{A}_k$	> 0	$\mathcal{A}_{k+1}$
	$\hat{u}_p^k\lambda_p^k \leq 0$	$\mathcal{A}_k$	$\leqq 0$	$\mathcal{I}_{k+1}$

We mention that the value of the positive constant c has no influence on the choice of the active and inactive sets $\mathcal{A}$ and $\mathcal{I}$ when we solve the linear system exactly. It only influences the choice in the inexact approach. Our algorithm guarantees that for an inadmissible Lagrange multiplier $\lambda_p^{k,m}$ the sign is preserved in the next step, where the node p is active.

5. Numerical examples. In this last section, we confirm our theoretical results for the discretization error and show the performance of our inexact primal-dual active set Algorithm 1. The implementation of this algorithm is based on the finite element toolbox UG; see [BBJ+97]. We use the material constants for granite, so that Young's modulus $E = 106$ and Poisson's ratio $\nu = 0.25$. These are related to the Lamé's parameters λ and μ by $\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}$ and $\mu = \frac{E}{2(1+\nu)}$. In all examples we assume conditions of plane strain. We focus on the discretization errors of the solution u in the L^2- and the H^1-norm. The error of the Lagrange multiplier is considered in a weighted L^2-norm $\|\cdot\|_{-\frac{1}{2},h,\Gamma_3}$ given by

$$\|\lambda\|^2_{-\frac{1}{2},h,\Gamma_3} := \sum_{e\in\Gamma_3^s} h_e \|\lambda\|^2_{0,e},$$

where we denote by e the edges of the discretization of Γ_3^s and by h_e the length of the edge e. We remark that using an inverse estimate it is also possible to establish a priori error estimates for the Lagrange multiplier in the weighted L^2-norm. Since there is in general no analytical solution available, we use for computing the discretization errors a reference solution u_{ref} and λ_{ref} on a fine mesh satisfying $h_{\text{ref}} \le 1/4\, h$. The use of $h_{\text{ref}} = 1/2\, h$ as reference mesh size does not yield reliable values for the discretization errors.

5.1. First example. In this section, we consider the example shown in Figure 5.1(a). We set the lower rectangle $\Omega^m = [-1, 1] \times [0, 1]$ and the upper as $\Omega^s = [-1, 1] \times [1, 2]$. On Γ_1^m, being the bottom of the lower rectangle Ω^m, we set the Dirichlet values equal to $u^m = 0$. We assign at the top boundary of the upper body Ω^s given Dirichlet values $u^s = 0.22\exp(-60x_1^2)$. On all the remaining boundary segments, we set homogeneous Neumann boundary conditions. For the friction bound we use $g = 0.6$. The contour lines of the numerical solution u_h are presented in Figure 5.1(b), while Figure 5.1(c) shows the Lagrange multiplier λ_h and the jump u_h amplified to the factor 150 at the interface Γ_3. The discretization errors and the convergence rates are presented in Table 5.1. Here we used the finite element solution on level 8 as reference solution u_{ref}. The grid on level 0 is shown in Figure 5.1(a). On

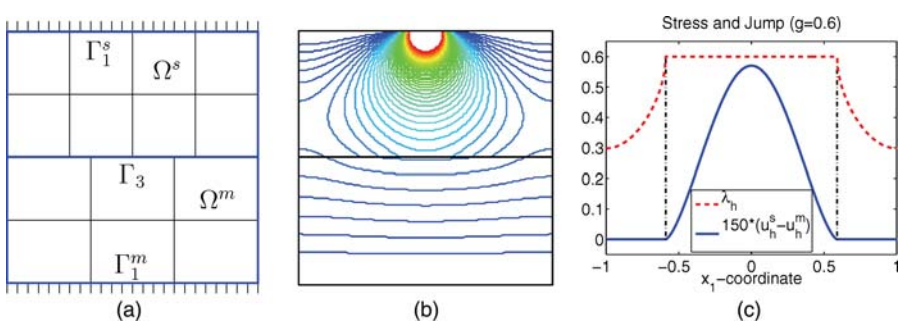


FIG. 5.1. (a) *Problem definition and grid on level* 0*;* (b) *contour lines of the solution;* (c) *Lagrange multiplier* λ_h *and amplified jump* $[u_h]$ *on* Γ_3.

TABLE 5.1
First example: Relative $L^2(\Omega)$-error and relative $H^1(\Omega)$-error of u_h with respect to u_{ref}, weighted $L^2(\Gamma_3)$-error of λ_h with respect to λ_{ref} and the numerical convergence rates.

level	$\|u_h - u_{\text{ref}}\|_{0,\Omega}/\|u_{\text{ref}}\|_{0,\Omega}$		$\|u_h - u_{\text{ref}}\|_{1,\Omega}/\|u_{\text{ref}}\|_{1,\Omega}$		$\|\lambda_h - \lambda_{\text{ref}}\|_{-\frac{1}{2},h,\Gamma_3}$	
0	$1.210198e+00$	–	$8.679373e-01$	–	$1.525284e-01$	–
1	$1.762545e-01$	2.78	$5.503170e-01$	0.66	$6.264323e-02$	1.28
2	$4.855714e-02$	1.86	$3.744576e-01$	0.56	$7.003110e-03$	3.16
3	$1.342324e-02$	1.85	$1.969682e-01$	0.93	$1.907398e-03$	1.88
4	$3.395710e-03$	1.98	$9.866146e-02$	1.00	$7.817159e-04$	1.29
5	$8.468612e-04$	2.00	$4.907663e-02$	1.01	$2.646597e-04$	1.56
6	$2.062731e-04$	2.04	$2.395136e-02$	1.03	$8.864475e-05$	1.58

level 8, we have 525 825 nodes on Ω^s and 394 497 on Ω^m. The convergence rates in the L^2-norm are asymptotically 2.0 and, in the H^1-norm, we get 1.0. As also observed in the linear case, we get better convergence rates for the Lagrange multiplier. The best approximation error in the weighted $L^2(\Gamma_3)$-norm is of order $h^{1.5}$.

5.2. Second example. Now we consider the problem depicted in Figure 5.2(a). Here the interface is a curved one. By means of this example, we illustrate the performance and flexibility of our algorithm. To fix the geometry, we set the three points P_1, P_2 and P_3 equal to $P_1 = (0, 1.5)$, $P_2 = (-1, 0)$ and $P_3 = (1, 0)$. The radius r of the upper half disc is set to be $r = 1$, and for the angle ϕ we choose $\phi = \pi/4$. The lower body plays the role of the slave side Ω_s, and the upper one the role of the master side Ω^m. We fix the domain at the bottom of the master side Ω^m. At the top boundary of the upper body Ω^s, we set the Dirichlet value to $u^s = 0.055x_1$. As done in the first example, we set the friction bound equal to $g = 0.6$.

Figure 5.2(c) shows the contour lines of the solution. Figure 5.3 shows the Lagrange multiplier λ_h, the jump $[u_h]$ amplified to the factor 50, and the solution of the master and slave side u_h^m, u_h^s on the interface Γ_3. The discretization errors and the corresponding convergence rates are listed in Table 5.2. Again we use as reference solution the solution on level 8. The grid on level 1 is presented in Figure 5.2(b).

To illustrate the performance of the inexact primal-dual active set Algorithm 1, we use as multigrid solver for the linear problems a $\mathcal{W}$-cycle with 3 pre-smoothing and post-smoothing steps of the symmetric Gauß–Seidel iteration. For the constant c in Algorithm 1, we use $c = 0.01$. We observe that the value of c has to be small enough to get convergence of the active sets. This value depends on the relation between the stresses λ_p and the displacements $\hat{u}_p$ and therefore on the material data. On each level, we start the multigrid iteration with $\hat{u}_h^{1,0} = 0$. We compare the inexact approach ($m = 1$), where we update the active set $\mathcal{A}$ after each multigrid step, with the exact approach ($m = \infty$), where we solve for each choice of the set $\mathcal{A}$ the resulting

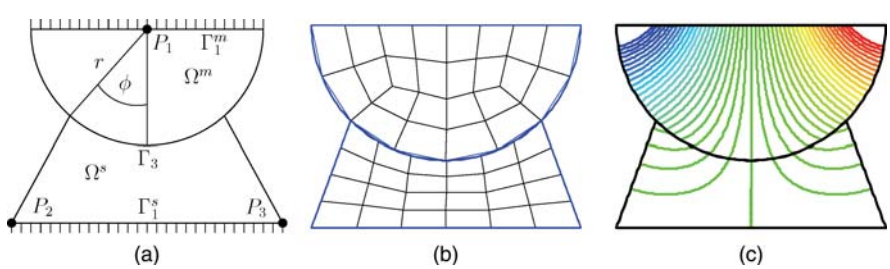


FIG. 5.2. (a) *Problem definition;* (b) *grid on level* 1*;* (c) *contour lines of the solution.*

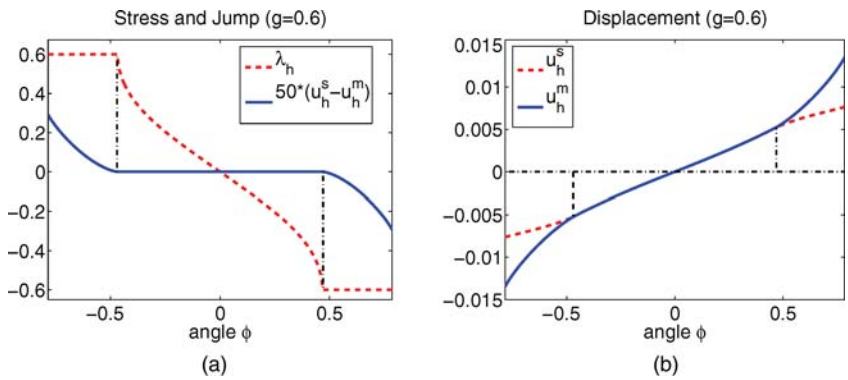


FIG. 5.3. (a) *Lagrange multiplier* λ_h *and amplified jump* $[u_h]$ *for* $g = 0.6$*;* (b) *displacement* u_h^s *on slave side and* u_h^m *on master side for* $g = 0.6$.

TABLE 5.2

Second example: Relative $L^2(\Omega)$*-error and relative* $H^1(\Omega)$*-error of* u_h *with respect to* u_{ref}*, weighted* $L^2(\Gamma_3)$*-error of* λ_h *with respect to* λ_{ref} *and the numerical convergence orders.*

level	$\|u_h - u_{\text{ref}}\|_{0,\Omega}/\|u_{\text{ref}}\|_{0,\Omega}$		$\|u_h - u_{\text{ref}}\|_{1,\Omega}/\|u_{\text{ref}}\|_{1,\Omega}$		$\|\lambda_h - \lambda_{\text{ref}}\|_{-\frac{1}{2},h,\Gamma_3}$	
0	$8.934322e-02$	–	$3.231679e-01$	–	$7.718450e-02$	–
1	$3.096022e-02$	1.53	$1.893279e-01$	0.77	$2.216194e-02$	1.80
2	$8.651862e-03$	1.84	$1.034272e-01$	0.87	$8.133636e-03$	1.45
3	$2.427230e-03$	1.83	$5.643122e-02$	0.87	$3.781621e-03$	1.10
4	$6.665729e-04$	1.86	$3.041368e-02$	0.89	$1.303146e-03$	1.54
5	$1.791767e-04$	1.90	$1.613158e-02$	0.91	$3.521447e-04$	1.89
6	$4.652240e-05$	1.95	$8.311285e-03$	0.96	$1.671239e-04$	1.08

linear system. For each of these two approaches, we compare the case where we start on each level from $\mathcal{A}_1 = \emptyset$ to the nested approach. Here we inherit the active set $\mathcal{A}$ from level $l-1$ to l and use the results as $\mathcal{A}_1$.

For the exact strategy, we denote by K_l the step of the active set iteration on level l in which the correct active sets are found. For the inexact approach, M_l denotes the number of the multigrid iterations after which the correct active sets are found and kept. Table 5.3 shows K_l and M_l for the two choices of $\mathcal{A}_1$. Here we denote by $|\mathcal{A}_k|$ the number of active nodes in the k-th. step. For the case $\mathcal{A}_1 = \emptyset$, we observe that almost on each level $K_l = M_l$. As a result, there is no need to solve the resulting linear system exactly. Furthermore we remark that the required number of steps depends linearly on the level. For the nested approach, we observe $K_l \leq M_l$ and the number of necessary steps is bounded independently of the refinement level. For the exact strategy, we get $K_l \leq 2$ and for the inexact strategy $M_l \leq 4$. Although $M_l \geq K_l$, the inexact strategy is much more efficient because each step requires only one multigrid iteration.

To compare the computational cost to solve the nonlinear problem for the four cases we consider the necessary numbers of multigrid iterations. These numbers are shown in Table 5.4. The last column contains the necessary number of multigrid iterations to solve the nonlinear problem with the exact active and inactive sets $\mathcal{A}$ and $\mathcal{I}$ given. Then the nonlinear problem reduces to a linear one. It turns out that the number of necessary multigrid iterations for the inexact approach with the nested strategy is almost the same as that for the linear problem. So, the additional effort solving the nonlinear problem entailed in is negligible. For the exact approach starting

TABLE 5.3

Comparison between exact and inexact active set strategy for the second example given in Figure 5.2 for the two approaches $\mathcal{A}_1 = \emptyset$ and the nested approach.

level	$\mathcal{A}_1 = \emptyset$		nested approach							
	exact	inexact	exact			inexact				
	K_l	M_l	K_l	$\lvert\mathcal{A}_k\rvert$		M_l	$\lvert\mathcal{A}_k\rvert$			
0	2	2	2	0	2	2	0	2		
1	3	3	2	2	4	2	2	4		
2	3	3	1	6		1	6			
3	4	5	1	10		3	10	12	10	
4	5	5	2	18	20	2	18	20		
5	6	6	2	38	40	2	38	40		
6	7	7	1	78		3	78	80	78	
7	8	8	2	154	156	3	154	158	156	
8	9	9	2	310	312	4	310	314	310	312

TABLE 5.4

Necessary number of multigrid iterations for the four considered cases for the second example given in Figure 5.2 for the two approaches.

level	$\mathcal{A}_1 = \emptyset$		nested approach		lin. problem
	exact	inexact	exact	inexact	
1	13	7	10	6	6
2	16	7	6	6	6
3	26	11	8	8	8
4	33	11	14	8	9
5	36	10	12	7	7
6	42	11	6	6	6
7	49	12	12	7	7
8	54	12	12	7	7

with $\mathcal{A}_1 = \emptyset$ on each level, the number depends linearly on the level. We find that the inexact strategy is cheaper compared to the exact one. In combination with the nested approach, the inexact strategy yields a very efficient iterative solver for our nonlinear problem.

Finally, we mention that our inexact primal-dual active set Algorithm 1 can also handle nonconstant friction bounds g. If we use $g = (x_1^2 - 1) + 1.2$ instead of $g = 0.6$ in the second example, we find the results shown in Figure 5.4.

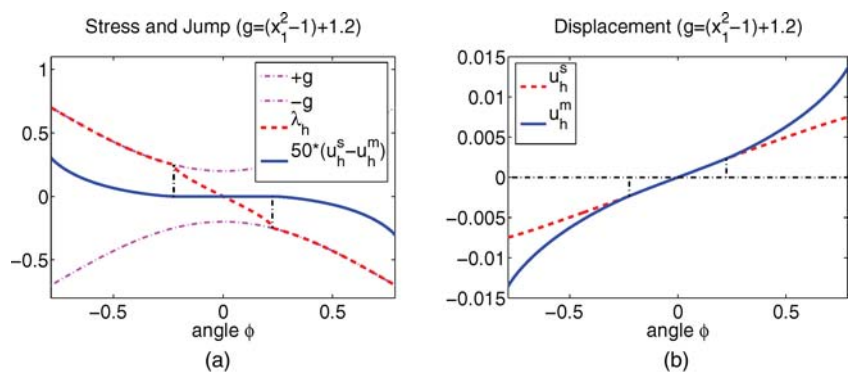


FIG. 5.4. (a) *Lagrange multiplier λ_h and amplified jump $u_h^s - u_h^m$ for $g = (x_1^2 - 1) + 1.2$;* (b) *displacement u_h^s on slave side and u_h^m on master side for $g = (x_1^2 - 1) + 1.2$.*

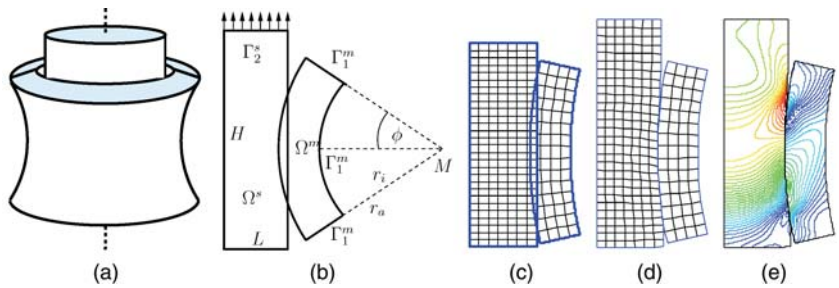


FIG. 6.1. (a) *Problem definition in three dimensions;* (b) *problem definition in two dimensions;* (c) *initial grid on level* 0*;* (d) *distorted grid on level* 0*;* (e) *effective von Mises stress on level* 2.

6. Generalization of Algorithm 1 to Coulomb friction. In this last section, we generalize our algorithm to a multibody contact problem with Coulomb friction. In this case, we replace the constant friction bound g in (2.9) by $\mathfrak{F}|\boldsymbol{\sigma}_n|$, where $\boldsymbol{\sigma}_n := (\boldsymbol{\sigma}\boldsymbol{n}) \cdot \boldsymbol{n}$ and $\mathfrak{F}$ denotes the friction coefficient. The contact condition in the normal direction is that of nonpenetration condition of the two bodies. More details on nonlinear contact conditions can be found, e.g., in [KO88].

Now we have to find two active sets. The active set of nodes on the contact interface are in contact in the normal direction with the second body and the active set for the Lagrange multiplier in the tangential direction. For the first one we use the algorithm given in [HW05b]. For the second one we use Algorithm 1. We remark that here we have to update the values for the friction bound g_p after each step. Combining the inexact versions of these two algorithms leads to an efficient iterative solver for the complex nonlinear problem of Coulomb friction. We mention that after each multigrid step we update both of the two active sets and the solution dependent friction bound.

We consider an example in the three-dimensional case. The geometry is shown in Figure 6.1(a). Due to axisymmetry the problem can be reduced to a two-dimensional setting. The resulting geometry can be seen in Figure 6.1(a). For the geometry we choose $L = 5$, $H = 15$, $r_i = 27$, $r_a = 30$, $M = (34.5, 7.0)^\top$ and $\phi = 13$. For the softer inner body Ω^s, playing the role of the slave side, we use for the material data $\lambda^s = 57.6923$ and $\mu^s = 38.4615$. For the outer body Ω^m, playing the role of the master side, we set $\lambda^m = 576.923$ and $\mu^m = 384.615$. At the top of the inner cylinder, we apply Neumann data $f_2^s = 17.0$. Due to the axisymmetry, we have on the left side of Ω_s, where $r = 0$ holds, homogeneous Dirichlet data in the radial direction. We fix Ω^m at the three segments Γ_1^m, as shown in Figure 6.1(b). For the friction coefficient $\mathfrak{F}$ we choose $\mathfrak{F} = 0.7$. We remark that the two bodies penetrate in the reference configuration.

For the Lamé operator in cylindrical coordinates, we refer to [JL01]. We denote by u_r the displacement in the radial direction, by u_z the displacement in the axial direction and by u_θ the displacement in circumferential direction. Since axisymmetric conditions we have $u_\theta = 0$ and therefore we get the relations $\boldsymbol{\varepsilon}_{r\theta} = \boldsymbol{\varepsilon}_{\theta r} = \boldsymbol{\varepsilon}_{z\theta} = \boldsymbol{\varepsilon}_{\theta z} = 0$. For the other entries of the strain tensor $\boldsymbol{\varepsilon}$ we get

$$\boldsymbol{\varepsilon}_{rr} = \frac{\partial u_r}{\partial r}, \quad \boldsymbol{\varepsilon}_{zz} = \frac{\partial u_z}{\partial z}, \quad \boldsymbol{\varepsilon}_{\theta\theta} = \frac{1}{r}\left(\frac{\partial u_\theta}{\partial \theta} + u_r\right), \quad \boldsymbol{\varepsilon}_{rz} = \boldsymbol{\varepsilon}_{zr} = \frac{1}{2}\left(\frac{\partial u_r}{\partial z} + \frac{\partial u_z}{\partial r}\right).$$

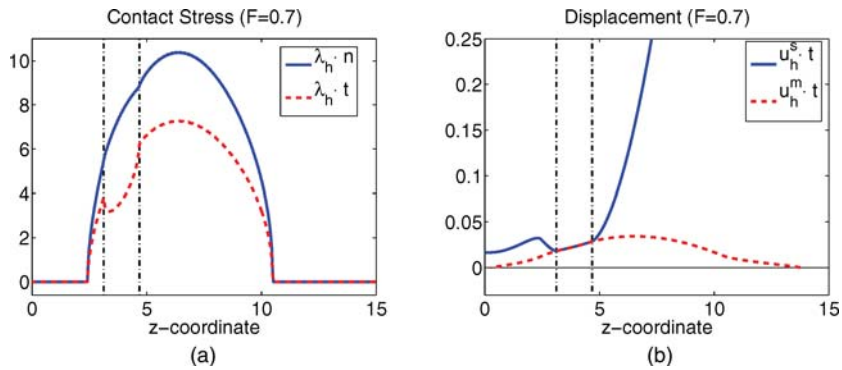


FIG. 6.2. (a) *Normal part* $\lambda_h \cdot n$ *and tangential part* $\lambda_h \cdot t$ *of the Lagrange multiplier* λ_h*;* (b) *tangential part* $u_h^s \cdot t$ *of the displacement on the slave side and tangential part* $u_h^m \cdot t$ *of the displacement on the master side.*

The stress strain relation is now

$$\begin{pmatrix} \boldsymbol{\sigma}_{rr} \\ \boldsymbol{\sigma}_{zz} \\ \boldsymbol{\sigma}_{\theta\theta} \\ \boldsymbol{\sigma}_{rz} \end{pmatrix} = \begin{pmatrix} \lambda + 2\mu & \lambda & \lambda & 0 \\ \lambda & \lambda + 2\mu & \lambda & 0 \\ \lambda & \lambda & \lambda + 2\mu & 0 \\ 0 & 0 & 0 & \mu \end{pmatrix} \begin{pmatrix} \boldsymbol{\varepsilon}_{rr} \\ \boldsymbol{\varepsilon}_{zz} \\ \boldsymbol{\varepsilon}_{\theta\theta} \\ 2\boldsymbol{\varepsilon}_{rz} \end{pmatrix}.$$

Figure 6.2 shows the normal and the tangential parts of the discrete Lagrange multiplier on the contact interface. We remark that the two bodies are in contact where we have $\lambda_h \cdot n \neq 0$. In the tangential direction, we get two active zones where we have $|\lambda_h \cdot t| = \mathfrak{F}|\lambda_h \cdot n|$. Between these two sets the two bodies are glued together in the tangential direction, i.e., $(u_h^s - u_h^m) \cdot t = 0$. The tangential part of the displacements on the slave and the master side is shown in Figure 6.2(b).

REFERENCES

[BBJ+97] P. BASTIAN, K. BIRKEN, K. JOHANNSEN, S. LANG, N. NEUSS, H. RENTZ–REICHERT, AND C. WIENERS, *UG—a flexible software toolbox for solving partial differential equations*, Comput. Vis. Sci., 1 (1997), pp. 27–40.

[BF91] F. BREZZI AND M. FORTIN, *Mixed and Hybrid Finite Element Methods*, Springer-Verlag, New York, 1991.

[BHL99] F. BEN BELGACEM, P. HILD, AND P. LABORDE, *Extension of the mortar finite element method to a variational inequality modeling unilateral contact*, Math. Models Methods Appl. Sci., 9 (1999), pp. 287–303.

[BHP02] A. BORRELLI, C. HORGAN, AND C. PATRIA, *Saint-Venant's principle for antiplane shear deformations of linear piezoelectric materials*, SIAM J. Appl. Math., 62 (2002), pp. 2027–2044.

[BR03] F. BEN BELGACEM AND Y. RENARD, *Hybrid finite element methods for the Signorini problem*, Math. Comp., 72 (2003), pp. 1117–1145.

[Bra97] D. BRAESS, *Finite Elements*, Cambridge University Press, Cambridge, UK, 1997.

[CHLS01] P. COOREVITS, P. HILD, K. LHALOUANI, AND T. SASSI, *Mixed finite element methods for unilateral problems: convergence analysis and numerical studies*, Math. Comp., 71 (2001), pp. 1–25.

[DL76] G. DUVOUT AND J.-L. LIONS, *Inequalities in Mechanics and Physics*, Springer, Berlin, 1976.

[ET76] I. EKELAND AND R. TEMAM, *Convex Analysis and Variational Problems*, Stud. Math. Appl. 1, North-Holland, Amsterdam, 1976.

[Glo84] R. GLOWINSKI, *Numerical Methods for Nonlinear Variational Problems*, Springer, New York, 1984.

[GLT81] R. Glowinski, J.-L. Lions, and R. Tremolières, *Numerical Analysis of Variational Problems*, North-Holland, Amsterdam, 1981.

[Hcc96] J. Haslinger, I. Hlaváček, and J. Nečas, *Numerical methods for unilateral problems in solid mechanics*, in Handbook of Numerical Analysis, Vol. IV, J.-L. Lions and P. Ciarlet, eds., North-Holland, Amsterdam, 1996, pp. 313–485.

[HIK03] M. Hintermüller, K. Ito, and K. Kunisch, *The primal-dual active set strategy as a semi–smooth Newton method*, SIAM J. Optim., 13 (2003), pp. 865–888.

[Hil00] P. Hild, *Numerical implementation of two nonconforming finite element methods for unilateral contact*, Comput. Methods Appl. Mech. Engrg., 184 (2000), pp. 99–123.

[HL02] P. Hild and P. Laborde, *Quadratic finite element methods for unilateral contact problems*, Appl. Numer. Math., 41 (2002), pp. 410–421.

[HMM02] T.-V. Hoarau-Mantel and A. Matei, *Analysis of a viscoelastic antiplane contact problem with slip dependent friction*, Internat. J. Appl. Math. Comput. Sci., 12 (2002), pp. 101–108.

[Hor95] C. Horgan, *Anti-plane shear deformations in linear and nonlinear solid mechanics*, SIAM Rev., 37 (1995), pp. 53–81.

[HS01] C. Horgan and G. Saccomandi, *Antiplane shear deformations for non-gaussian isotropic, incompressible hyperelastic materials*, in Proceedings of Royal Society of London, Ser. A, 2001, pp. 1999–2017.

[HS02] W. Han and M. Sofonea, *Quasistatic Contact Problems in Viscoelasticity and Viscoplasticity*, AMS/IP Stud. Adv. Math., AMS, Providence, RI, 2002.

[HS04] C. Horgan and G. Saccomandi, *Superposition of generalized plane strain on anti-plane shear deformations in isotropic incompressible hyperelastic materials*, J. Elasticity, 73 (2004), pp. 221–235.

[HSW06] S. Hüeber, G. Stadler, and B. Wohlmuth, *A Primal-Dual Active Set Algorithm for Three-Dimensional Contact Problems with Coulomb Friction*, Tech. report 06-16, Pré-publicações do Departamento de Matemática da Universidade de Coimbra, Coimbra, Portugal, 2006.

[HW05a] S. Hüeber and B. Wohlmuth, *An optimal a priori error estimate for nonlinear multibody contact problems*, SIAM J. Numer. Anal., 43 (2005), pp. 157–173.

[HW05b] S. Hüeber and B. Wohlmuth, *A primal-dual active set strategy for non–linear multibody contact problems*, Comput. Methods Appl. Mech. Engrg., 194 (2005), pp. 3147–3166.

[JL01] M. Jung and U. Langer, *Methode der finiten Elemente für Ingenieure*, Teubner, Leipzig, 2001.

[Kno76] J. Knowles, *On finite anti-plane shear for incompressible elastic materials*, J. Aust. Math. Soc. Ser. B, 419 (1976), pp. 400–415.

[Kno77] J. Knowles, *The finite anti-plane shear field near the tip of a crack for a class of incompressible elastic solids*, Internat. J. Fracture, 13 (1977), pp. 611–639.

[KO88] N. Kikuchi and J. Oden, *Contact Problems in Elasticity: A Study of Variational Inequalities and Finite Element Methods*, SIAM Stud. Appl. Math. 8, SIAM, Philadelphia, 1988.

[MMS01] A. Matei, V. Motreanu, and M. Sofonea, *A quasistatic antiplane contact problem with slip dependent friction*, Adv. Nonlinear Var. Inequal., 4 (2001), pp. 1–21.

[Pan85] P. Panagiotopoulos, *Inequality Problems in Mechanics and Applications*, Birkhäuser, Boston, 1985.

[SST04] M. Shillor, M. Sofonea, and J. Telega, *Models and Analysis of Quasistatic Contact: Variational Methods*, Lecture Notes in Phys. 655, Springer, Berlin, Heidelberg, 2005.

[Sta04] G. Stadler, *Semismooth Newton and augmented Lagrangian methods for a simplified friction problem*, SIAM J. Optim., 15 (2004), pp. 39–62.

[WK03] B. Wohlmuth and R. Krause, *Monotone methods on non-matching grids for nonlinear contact problems*, SIAM J. Sci. Comput., 25 (2003), pp. 324–347.

[Woh00] B. Wohlmuth, *A mortar finite element method using dual spaces for the Lagrange multiplier*, SIAM J. Numer. Anal., 38 (2000), 989–1012.

[Woh05] B. I. Wohlmuth, *A V-cycle multigrid approach for mortar finite elements*, SIAM J. Numer. Anal., 42 (2005), pp. 2476–2495.

SIAM J. SCI. COMPUT.
Vol. 29, No. 1, pp. 93–117
© 2007 Society for Industrial and Applied Mathematics

ACCURATE EVALUATION OF EUROPEAN AND AMERICAN OPTIONS UNDER THE CGMY PROCESS*

ARIEL ALMENDRAL† AND CORNELIS W. OOSTERLEE†

Abstract. A finite-difference method for integro-differential equations arising from Lévy driven asset processes in finance is discussed. The equations are discretized in space by the collocation method and in time by an explicit backward differentiation formula. The discretization is shown to be second-order accurate for a relevant parameter range determining the degree of the singularity in the Lévy measure. The singularity is dealt with by means of an integration by parts technique. An application of the fast Fourier transform gives the overall amount of work $O(N_t N \log N)$, rendering the method fast.

Key words. partial integro-differential equations, collocation method, option pricing

AMS subject classifications. 45K05, 45D05, 65R20, 91B28

DOI. 10.1137/050637613

1. Introduction.

In a seminal paper from 1973, Black and Scholes [8] derived a PDE for option prices, when asset prices behave according to the geometric Brownian motion. The pricing formulas obtained in that paper represented a major breakthrough in understanding financial derivatives, to such an extent that financial institutions and traders immediately adopted the new methodology. Later empirical studies revealed that the normality of the log-returns, as assumed by Black and Scholes, could not capture features like heavy tails and asymmetries observed in market-data log-return densities [12]. The Black–Scholes model assumes in addition constant problem parameters, interest rate, and volatility for the time until the option's expiration (also called maturity). This contradicts the existence of the so-called volatility smile: A numerical inversion of the Black–Scholes formula based on data from different strike prices and fixed maturity resembles a skew or a smile, implying nonconstant volatility. This inconsistency is said to be one of the causes for famous market crashes.

To explain these empirical observations, a number of alternate models have appeared in the financial literature: stochastic volatility [25, 28], deterministic local volatility [17, 20], jump-diffusion [29, 33], and infinite activity Lévy models [7, 15, 21]. Each of these models has its advantages and disadvantages. Jump-diffusion and infinite activity Lévy models are attractive since they can capture the jump patterns exhibited by some stocks and they are more realistic when pricing options close to maturity [18]. Processes with infinite activity, [illegible], represent a family that describes the high activity of the prices, while at the same time they reflect the empirical features desirable in a good model. The pricing equations are, however, numerically more challenging, and the market turns out to be incomplete in the sense that a strategy leading to an instantaneous risk-free portfolio does not exist in general [16, 34].

*Received by the editors August 5, 2005; accepted for publication (in revised form) July 6, 2006; published electronically January 12, 2007. This research was supported by the Dutch government through the national program BSIK: Knowledge and Research Capacity, in the ICT project BRICKS (http://www.bsik-bricks.nl), theme MSV1.

http://www.siam.org/journals/sisc/29-1/63761.html

†Delft University of Technology, Faculty of Electrical Engineering, Mathematics and Computer Science, Mekelweg 4, 2628 CD Delft, The Netherlands (Ariel.Almendral.Vazquez@nr.no, c.w.oosterlee@math.tudelft. nl).

Due to the close link between the stochastic martingale approach and the PDE approach [24, 30] the field of computational finance has gained a tremendous impulse from well-established numerical techniques for PDEs. For an overview we refer the reader to the introductory books [26, 37]. The development of analogous reliable techniques under Lévy markets is a subject of present research [5, 6, 19] for jump-diffusions and for the infinite activity case [3, 4, 16, 27, 32]. In these papers the numerical solution of a partial integro-differential equation (PIDE) has been addressed as a tool to calculate option prices.

When valuing and risk managing exotic derivatives, practitioners demand fast and accurate prices and sensitivities. Aside from nonstandard exotic derivatives, plain vanilla options in many stock markets are actually of the American (i.e., early exercise) type, for which no analytic solutions exist. As any pricing and risk management system has to be able to calibrate to these plain vanilla options; it is of the utmost importance to be able to value these American options quickly and accurately. This paper deals with a fast, accurate evaluation of options when the underlying process is a Carr–Geman–Madan–Yor (CGMY) Lévy process of infinite activity and finite variation, considered in finance in [13]. In previous work [3, 4], we computed the integral term in the PIDE by the trapezoidal rule, a choice that produced a first-order accurate overall convergence. Our contribution here is to show that second-order convergence may be obtained for a large range of the problem parameters ($0 < \alpha < 1$ in (2) below).

The approach in this paper differs from existing methods in the literature. We use an integration by parts technique to rewrite the integro-differential operator in terms of Volterra operators with a weakly singular kernel. The vast classical numerical literature on this type of operator [10, 11, 31] proves very useful to set up a high-order discretization method. Here we focus on one particular method that has been thoroughly tested in [11], namely, the collocation method for Volterra equations. The integro-differential operator in this new setting is not in the standard form in the literature, so the techniques need to be adapted to this particular example. The computation of integrals is expensive, since the number of operations involved is in general of order $O(N^2)$. With the help of the fast Fourier transform (FFT) algorithm it is possible to speed up computations to gain an almost linear complexity ($O(N \log N)$), provided the grid employed is uniform. The method proposed allows the application of the FFT algorithm. In the context of weakly singular Volterra (integro-differential) equations, this has been implemented in [23].

The paper is organized as follows. Section 2 offers a brief introduction to financial derivatives under exponential Lévy models together with the integro-differential equations of interest. Section 3 contains the core of the paper: the transformation of the equation by integration by parts, the collocation method on the resulting weakly singular Volterra equation, and a time integration method. A series of numerical experiments is presented in section 4, confirming the second-order accuracy of the discretization. One of the advantages of the collocation method is that one can deal with complicated payoffs, e.g., for butterfly options, straddles, etc., without affecting the accuracy. The numerical experiments in section 4 will confirm this statement. Technical details regarding the discretization are placed in appendices. A stability analysis for a model problem is also given in an appendix.

2. The exponential Lévy model in finance. Let us briefly introduce the exponential Lévy model. By a Lévy process $\{L_t\}_{t\geq 0}$ we mean any process starting at the origin, with stationary, independent increments. We restrict ourselves to those

Lévy processes that can be expressed as

$$L_t = (r - q + \zeta)t + \sigma W_t + Z_t, \tag{1}$$

where $r \geq 0$ and $q \geq 0$ denote the risk-free interest rate and the continuous dividend paid by some asset S_t, respectively. The parameter σ denotes the volatility. This process has a drift term controlled by ζ, a Brownian component $\{W_t\}_{t\geq 0}$, and a pure-jump component $\{Z_t\}_{t\geq 0}$. We focus on the case where the Lévy measure associated to Z_t (see [16]) can be written as $d\nu(x) = k(x)dx$, where the weight $k(x)$ is defined as

$$k(x) = \begin{cases} C\dfrac{\exp(-G\,|x|)}{|x|^{1+\alpha}} & \text{if} \quad x < 0, \\ \\ C\dfrac{\exp(-M\,|x|)}{|x|^{1+\alpha}} & \text{if} \quad x > 0 \end{cases} \tag{2}$$

for constants $C > 0$, $G \geq 0$, $M \geq 0$, and $\alpha < 2$. The process $\{Z_t\}_{t\geq 0}$ is known in the literature as the CGMY process [13]; it generalizes a jump-diffusion model by Kou [29] ($\alpha = -1$) and the Variance Gamma (VG) process [15] ($\alpha = 0$). The CGMY process is in turn a particular case of the Kobol process studied in [9], where the constant C is allowed to take on different values on the positive and negative semiaxes. For convenience of notation, we have chosen parameter α instead of the Y in [13].

2.1. Option pricing under the CGMY process. Let a market consist of one risky asset $\{S_t\}_{t\geq 0}$ and a bank account $\{B_t\}_{t\geq 0}$. Assume that the asset process $\{S_t\}_{t\geq 0}$ evolves according to the geometric law

$$S_t = S_0 \exp(L_t), \tag{3}$$

where $\{L_t\}_{t\geq 0}$ is the Lévy process defined in (1), and the bank account follows the law $B_t = \exp(rt)$.

The standard tool to assign prices to options under Lévy processes consists in changing to a convenient probability measure and taking the expectation of the discounted prices. The new probability measure Q is known in the financial literature as the equivalent martingale measure (EMM). It has the same null sets as the market probability, and the discounted process $\{e^{-(r-q)t}S_t\}_{t\geq 0}$ becomes a martingale. The so-called EMM condition $E_Q[S_t] = S_0 e^{t(r-q)}$ together with the formula for the characteristic exponent of L_t [16],

$$E_Q[\exp(L_t)] = \exp\left\{t\left[(r - q + \zeta) + \sigma^2/2 + \int_{\mathbb{R}} (e^x - 1)k(x)dx)\right]\right\}, \tag{4}$$

implies the following "risk-neutral" form for ζ (denoted by ϖ):

$$\varpi + \sigma^2/2 + \int_{\mathbb{R}} (e^x - 1)k(x)dx = 0. \tag{5}$$

Techniques used in Appendix A produce the following expression for ϖ:

$$\varpi = -\frac{\sigma^2}{2} - C\Gamma(-\alpha)\left\{(M-1)^\alpha - M^\alpha + (G+1)^\alpha - G^\alpha\right\} \tag{6}$$

with $\Gamma(\cdot)$ the gamma function. Note that ζ controls the drift of the market process, whereas ϖ is a new parameter controlling the drift in the artificial world described by the new probability measure Q, under which options can be priced straightforwardly. The same notation is kept for the risk-neutral parameters G and M. The other parameters σ, C, and α remain unchanged when passing to the Q measure; see, e.g., [16]. M must be larger than one for ϖ to be well defined.

European vanilla options. Consider a European option on the asset $\{S_t\}_{t\geq 0}$, with maturity time T, and strike price K. Let $\hat{u}(s,t)$ denote its price, with t denoting the time to expiration. It is well known that $\hat{u}$ satisfies a PIDE; see, for example, [16]. In order to write a convenient expression for the PIDE, let $s = \exp(x)$ and consider the log-prices $u(x,t)$. Then in general u solves the Cauchy problem

$$\begin{cases} u_t - \mathcal{L}u = 0, \quad t \in (0,T], \quad x \in \mathbb{R}, \\ \\ u(x,0) = \psi(x), \quad x \in \mathbb{R}, \end{cases} \tag{7}$$

where $\mathcal{L}$ is an integro-differential operator of the form

$$\begin{aligned} \mathcal{L}\varphi := \frac{\sigma^2}{2}\varphi_{xx} + \left(r - q - \frac{\sigma^2}{2}\right)\varphi_x - r\varphi \\ + \int_{\mathbb{R}} \left[\varphi(x+y,t) - \varphi(x,t) - (e^y - 1)\varphi_x(x,t)\right] k(y)dy, \end{aligned} \tag{8}$$

and the so-called payoff function ψ defines the nature of the contract. For example, for a put option $\psi(x) = (K - e^x)^+$, meaning that the holder of the option is entitled to receive the payment $K - S_T$ if the asset value falls below the threshold K; otherwise the option expires worthless.

For the particular case under consideration, namely, $\alpha < 1$, the third term in the integral operator may be computed explicitly (cf. (5)), resulting in the expression

$$\mathcal{L}\varphi := \frac{\sigma^2}{2}\varphi_{xx} + (r - q + \varpi)\varphi_x - r\varphi + \int_{\mathbb{R}} \left[\varphi(x+y,t) - \varphi(x,t)\right] k(y)dy. \tag{9}$$

This operator reduces to the Black–Scholes operator if no jumps are present, i.e., $k \equiv 0$.

American vanilla options. Consider as a matter of example an American put option written on the underlying asset $\{S_t\}_{t\geq 0}$. It is also well known that its price may be found by solving the free-boundary problem [16]

$$\begin{cases} u_t - \mathcal{L}u = 0, \quad t > 0, \quad x > x_f(t), \\ \\ u(x,t) = K - e^x, \quad t > 0, \quad x \leq x_f(t), \\ \\ u(x,t) \geq (K - e^x)^+, \quad t > 0, \quad x \in \mathbb{R}, \\ \\ u_t - \mathcal{L}u \geq 0, \quad t > 0, \quad x \in \mathbb{R}, \\ \\ u(0,x) = (K - e^x)^+, \quad x \in \mathbb{R}, \end{cases} \tag{10}$$

where the operator $\mathcal{L}$ is defined in (8) and the free-boundary is given by

$$x_f(t) = \inf\left\{x \in \mathbb{R} \mid u(x,t) > (K - e^x)^+\right\}, \quad t \in (0,T]. \tag{11}$$

The set $\{x \in \mathbb{R} \mid x \leq x_f(t)\}$ represents the exercise region for the logarithmic prices. Hence, for asset prices $s \leq s_f(t) := \exp(x_f(t))$, the American put should be exercised.

For numerical purposes, a useful reformulation of (10) is the following linear complementarity problem (LCP):

$$\begin{cases} u_t - \mathcal{L}u \geq 0 & \text{in } [0,T] \times \mathbb{R}, \\ u \geq \psi & \text{in } [0,T] \times \mathbb{R}, \\ (u_t - \mathcal{L}u)\,(u - \psi) = 0 & \text{in } [0,T] \times \mathbb{R}, \\ u(x,0) = \psi(x). \end{cases} \tag{12}$$

This formulation does not make use of the free-boundary explicitly, and besides, it does not rely on the form of the continuation region. The free-boundary can be obtained in a postprocessing step by using the relation (11).

3. Discretization of the integral term.

3.1. Integral equation over the positive semiaxis. The core of the numerical method that we propose here lies on a proper discretization of integral expressions of the form

$$f(x) = \int_0^\infty (u(x+y) - u(x)) \frac{e^{-My}}{y^{1+\alpha}} dy \tag{13}$$

that appear in the definition of $\mathcal{L}$ in (9). For ease of notation we have omitted the time variable. We suppose that u is known and we would like to approximate $f(x)$. The reason for not solving with respect to u in the first place is that we will use an explicit time-stepping scheme for solving (7). In the applications, the function $u(x)$ can be continuous but need not be differentiable at certain points.

Integration by parts of (13) yields

$$f(x) = -[u(x+y) - u(x)]\hat{k}(y) \Big|_0^\infty + \int_0^\infty u_y(x+y)\hat{k}(y)dy, \tag{14}$$

where we have introduced the function

$$\hat{k}(w) := \int_w^\infty \frac{e^{-M\zeta}}{\zeta^{1+\alpha}} d\zeta, \qquad w > 0. \tag{15}$$

This new kernel may be written in terms of the upper incomplete gamma function $\Gamma(\cdot,\cdot)$; see (50) in Appendix A for a definition and also [1] for some properties. The following expression holds:

$$\hat{k}(w) = M^\alpha \Gamma(-\alpha, Mw). \tag{16}$$

Under rather general conditions on the function $u(x)$, the first term in (14) vanishes. A possible set of sufficient conditions is

- The function u satisfies the Lipschitz condition; i.e., there is a constant C_0 such that

$$|u(z_1) - u(z_2)| \leq C_0 \,|z_1 - z_2| \qquad \forall z_1, z_2 \in \mathbb{R}. \tag{17}$$

- There exist constants $C_1 \geq 0$ and $0 \leq C_2 < M$ such that

$$|u(z)| \leq C_1 e^{C_2 z}, \qquad z \to +\infty. \tag{18}$$

If these two conditions hold, it is not difficult to show that

$$\lim_{y\to 0}[u(x+y)-u(x)]\hat{k}(y) = \lim_{y\to\infty}[u(x+y)-u(x)]\hat{k}(y) = 0. \tag{19}$$

For commonly used functions in option pricing like $u(x) = (K - e^x)^+$ and $u(x) = (e^x - K)^+$, both conditions above hold.

Changing variable $z = x+y$, integral equation (14) reduces to a Volterra equation of the first kind:

$$f(x) = \int_x^\infty u_z(z)\hat{k}(z-x)dz. \tag{20}$$

This kind of equation has been intensively studied in the last century, both theoretically and numerically; see [10, 11, 31] and the references therein. In this paper we follow very closely the notation in [11].

3.1.1. Collocation method on positive semiaxis. The first observation concerns the order of the singularity of the kernel $\hat{k}$. It is not difficult to realize that $\hat{k}$ behaves like $y^{-\alpha}$, as $y \to 0$. Hence, the integral equation (20) classifies as a weakly singular Volterra equation.

Motivated by the above observations, let us introduce a notation for a new kernel $\widetilde{k}$ and the derivative v of u, i.e.,

$$\widetilde{k}(w) := w^\alpha \hat{k}(w), \qquad v := u_z. \tag{21}$$

From now on we focus on the discretization of the integral equation

$$f(x) = \int_x^\infty (z-x)^{-\alpha} v(z)\widetilde{k}(z-x)dz, \tag{22}$$

following [11] closely. In this form it resembles the well-studied Abel equation [31]. In the examples considered in this paper, the functions u have a rapidly decaying derivative at infinity. Thus, one may truncate this integral to a finite interval $x \in [a, b]$ while having control of the error. An analysis of the size of the truncation domain is given in section 3.1.3 and a numerical test is presented in section 3.1.4.

Let us consider the following (not necessarily uniform) spatial mesh

$$x_0 = a; \qquad x_{n+1} = x_n + h_n, \qquad n = 0, \ldots, N-1; \qquad x_N = b. \tag{23}$$

Equation (22) is discretized by the collocation method. A discretization with two collocation points for each mesh point is considered here. For a more general treatment we refer the reader to [11].

The first step is to choose two numbers c_1 and c_2 such that

$$0 \leq c_1 < c_2 < 1. \tag{24}$$

With these constants the collocation mesh is defined as follows (see Figure 1):

$$x_{n,j} = x_n + c_j h_n, \qquad n = 0, \ldots, N-1; \qquad j = 1, 2. \tag{25}$$

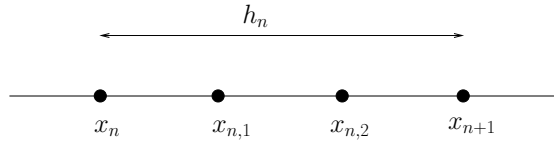


FIG. 1. *Collocation mesh and spatial mesh.*

The polynomial spline collocation method now consists, roughly speaking, in finding a polynomial spline $V(z)$, replacing the function $v(z)$ such that (22) holds exactly at each of the collocation points $x_{n,j}$. The technical details of the treatment of the Volterra equation when the variable of integration runs through the positive semiaxis are presented in Appendix B. For the negative semiaxis the discretization is very similar and is explained in Appendix C. Considering the collocation parameters $0 < c_1 < c_2 \leq 1$, the collocation mesh for the negative semiaxis is defined as in (23); however, these two meshes are typically different. This time $c_1 \neq 0$. In section 3.2 we couple both integrals with the use of a common spatial mesh.

REMARK 3.1.

[22] [10, Chapter 6]

(22)

x ∞

c_1

c_2

3.1.2. Smoothing the kernel. A simple numerical experiment with (22) reveals that the order of the discretization is only $O(h)$ (cf. Table 1). At first glance it seems that the problem is caused by the nonsmooth payoff. However, the cause of this loss of accuracy is the lack of smoothness of the kernel $\widetilde{k}$. This can be easily remedied by a convenient rewriting of the integral equation. One possible remedy is based on an expansion of the kernel such that the nonsmooth component can be identified. We will add a subscript "+" to the kernel to indicate that we work with the positive of the Lévy measure. A first integration by parts yields

$$\hat{k}(x) = \int_x^\infty \frac{e^{-My}}{y^{1+\alpha}} dy = \frac{e^{-Mx}x^{-\alpha}}{\alpha} - \frac{M}{\alpha}\int_x^\infty e^{-My}y^{-\alpha}dy.$$

A second integration by parts for the integral on the right-hand side gives rise to the identity

$$\int_x^\infty \frac{e^{-My}}{y^{1+\alpha}} dy = \frac{e^{-Mx}x^{-\alpha}}{\alpha} + \frac{Me^{-Mx}x^{1-\alpha}}{\alpha(1-\alpha)} - \frac{M^2}{\alpha(1-\alpha)}\int_x^\infty e^{-My}y^{1-\alpha}dy.$$

So, if we define the constant

$$\theta_+ := \frac{M^2}{\alpha(1-\alpha)}\int_0^\infty e^{-My}y^{1-\alpha}dy,$$

the new kernel

$$k_+(x) := \widetilde{k}(x) + \theta_+ x^\alpha = \frac{e^{-Mx}}{\alpha} + \frac{Me^{-Mx}x}{\alpha(1-\alpha)} - \frac{M^2x^\alpha}{\alpha(1-\alpha)}\int_0^x e^{-My}y^{1-\alpha}dy$$

becomes a differentiable function, in contrast to the original kernel $\widetilde{k}(x)$ in (21) which is only continuous. Moreover, due to the second integration by parts, k_+ is even twice continuously differentiable. Indeed, differentiating twice the relevant term on the right-hand side and neglecting the smooth terms give

$$x^{\alpha-2}\int_0^x e^{-My}y^{1-\alpha}dy,$$

which is continuous, but not differentiable. A final observation is that θ_+ may be written in terms of the gamma function $\Gamma(\cdot)$; i.e.,

$$\theta_+ = \frac{M^\alpha\Gamma(2-\alpha)}{\alpha(1-\alpha)}. \tag{26}$$

The observations lead to the integral equation

$$f(x) = \int_x^\infty (z-x)^{-\alpha}v(z)k_+(z-x)dz - \theta_+\int_x^\infty v(z)dz. \tag{27}$$

Notice that (27) is merely a rewriting of (22), but the kernels present in (27) are smoother. One could continue integrating by parts and extract the smooth components of $\widetilde{k}$. For numerical efficiency this is however not advisable as each additional term means a new integral to be computed numerically, increasing therefore the complexity. In order to avoid truncation errors we treat both terms on the right-hand side in (27) numerically.

3.1.3. Analysis of the truncation error. We sketch an argument that provides an estimate for the truncation error. The details have been omitted for clarity.

It is known that the solution u of (7) admits the stochastic interpretation

$$u(x,t) = e^{-rt}E_Q[\psi(x+L_t)]. \tag{28}$$

Consider the case of a put option, i.e., $\psi(x) = (K-e^x)^+$. Denoting by $\hat{q}(y)$ the probability density function of the variable L_t, we may write

$$u(x,t) = e^{-rt}\int_{-\infty}^{\log K-x}(K-e^{x+y})\hat{q}(y)dy.$$

Differentiating this expression with respect to x yields

$$u_x(x,t) = -e^{-rt+x}\int_{-\infty}^{\log K-x}e^y\hat{q}(y)dy.$$

Since $e^y \le Ke^{-x}$ for $y \le \log(K)-x$, the following estimate holds:

$$|u_x(x,t)| \le Ke^{-rt}. \tag{29}$$

This is a global estimate. A more detailed analysis shows that the derivative of a put decays exponentially, giving a sharper estimate. For our purposes this bound is sufficient. An estimate for the truncation error may be obtained as follows:

$$\begin{aligned}\left|\int_{-\infty}^a u_z(z,t)\hat{k}_-(x-z)dz + \int_b^{+\infty}u_z(z,t)\hat{k}_+(z-x)dz\right| \\ \le Ke^{-rt}\left[\int_{-\infty}^a \hat{k}_-(x-z)dz + \int_b^{+\infty}\hat{k}_+(z-x)dz\right].\end{aligned} \tag{30}$$

Both integrals on the right-hand side have exponential decay as $a \to -\infty$ and $b \to +\infty$. Namely, from (15), a rough estimate is the following:

$$\hat{k}_+(w) \le Ce^{-Mw}. \tag{31}$$

Therefore, the second integral on the right-hand side may be bounded by

$$\frac{C^+}{M} e^{-M(b-x)}.$$

An analogous analysis shows that the first integral is controlled by

$$\frac{C^-}{G} e^{-G(x-a)}.$$

These estimates show that, away from the end points a and b, the error decreases exponentially. Close to the end points the exponential decay of the derivative u_x (for a put option) will play a role in reducing the error by truncation.

3.1.4. First numerical experiments. In this section we first show that the collocation method applied to (27) gives at least a quadratic discretization error. We choose the payoff function for a put option $u(x) = (1 - e^x)^+$ and compare it with the analytic formula for $f(x)$ derived in Appendix A (52). Moreover, we check that indeed the same method applied to (22) gives only a linear order, so that the smoothing of the integrand as explained in section 3.1.2 is necessary for second-order accuracy. The results are displayed in Table 1. The parameters used are $\alpha = 0.8$ and $M = 5$ and the mesh is uniform. The truncation of the domain is in accordance with the analysis in section 3.1.3. In particular, the computational domain in this example is the interval $[-5, 5]$. The truncation does not influence the accuracy negatively.

The same convergence results hold on a nonuniform grid, but we are then not able to apply the FFT algorithm (discussed in section 3.4 and in detail in [3]) to make computations fast. On the other hand, the discretization order depends on the choice of the collocation parameters c_1 and c_2, as Table 1 shows. We actually observe superconvergence at the collocation points with the Radau parameters $c_1 = 0$ and $c_2 = 2/3$. This is in accordance with the results from [11].

TABLE 1
Discretization errors with the collocation method.

	Eq. (27)				Eq. (22)	
	$c_1 = 0, c_2 = 0.5$		$c_1 = 0, c_2 = 2/3$		$c_1 = 0, c_2 = 0.5$	
N	ℓ_∞-error	rate	ℓ_∞-error	rate	ℓ_∞-error	rate
20	0.0839		0.02314		0.248	
40	0.0240	3.4	0.00366	6.3	0.148	1.6
80	0.0065	3.6	0.00057	6.4	0.079	1.8
160	0.0017	3.8	0.00009	6.3	0.041	1.9

In order to obtain superconvergence on the negative semiaxis, the Radau points must be chosen as $c_1 = 1/3$ and $c_2 = 1$.

3.2. Assembling the PIDE. Recall that our main goal is to solve a PIDE of the form (7) with the operator $\mathcal{L}$ given by (9). The interesting case is when $\sigma = 0$, i.e., no smoothing takes place caused by a diffusion component. In case we deal with $\sigma > 0$, the ideas discussed in this paper still apply, but then special care of the diffusion needs

to be taken by means of an implicit-explicit scheme as in [3, 4, 16], where the diffusion part is taken implicitly and early exercise is dealt with by projection methods.

The considerations from previous sections lead us to consider the following rewriting of (7):

$$u_t + ru = C\mathcal{I}_+(u_z) + C\mathcal{I}_-(u_z) + \mu u_x, \tag{32}$$

with the constant (cf. (6))

$$\mu := r - q + \varpi, \tag{33}$$

and the operators

$$\mathcal{I}_+(u_z) := \int_x^{+\infty} (x-z)^{-\alpha} u_z(z) k_+(z-x)dz - \theta_+ \int_x^{+\infty} u_z(z)dz, \tag{34}$$

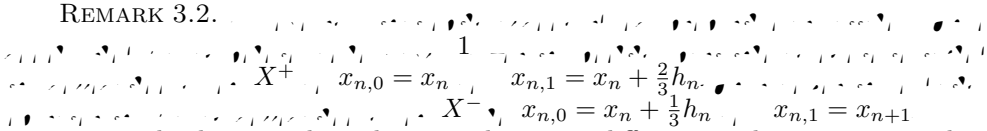
$$\mathcal{I}_-(u_z) := -\int_{-\infty}^{x} (x-z)^{-\alpha} u_z(z) k_-(z-x)dz + \theta_- \int_{-\infty}^{x} u_z(z)dz. \tag{35}$$

Note that these operators involve both the function u and its derivative $v = u_z$. To our knowledge, PIDEs like (32)–(35) have not been considered in the existing literature on Volterra equations.

If one applies a time integration procedure, one faces the problem of finding the numerical derivative of the unknown for a next time step, discussed in section 3.2.1. This is related to the treatment of the term μu_x in a consistent way.

In (32) we make a "superposition" of two operators. The positive operator behaves essentially as a "convection term" in that the solution is "transported" in the negative direction. The negative operator behaves exactly opposite, "transporting" the solution in the positive direction.

REMARK 3.2.



To couple the partial results over these two different meshes we propose here a straightforward interpolation: the evaluation of the integral on the X^+ and X^- meshes is interpolated to the "base mesh" $X^+ \cup X^-$. We use a simple linear interpolation, though a higher-order interpolation is in principle also applicable.

3.2.1. Local differentiation formulas due to the collocation method. There are several choices to discretize the numerical derivative in (34)–(35). The stable one is to apply once more the collocation method to the equation

$$u(x) = -\int_x^b v(s)ds + u(b) \tag{36}$$

for the positive integral. For the negative integral one should instead use the formula

$$u(x) = \int_a^x v(s)ds - u(a). \tag{37}$$

The formulas for the negative operator were derived in [11]. We have derived them for the positive operator. The formulas are summarized below for the sake of completeness. The formulas associated to (36) are called "forward formulas," and the ones

associated to (37) "backward formulas." As remarked in [11], to be able to obtain stable local differentiation formulas, one of the collocation points must coincide with one extreme of the interval, namely, $0 = c_1 < c_2 < 1$. For the backward formulas one must have $0 < c_1 < c_2 = 1$. One picks two arbitrary numbers $\nu_1, \nu_2 \in [0, 1]$, that need not be the same for the forward and backward formulas.

Forward formulas. [illegible] $x_{n,1}$

$$V_{n,1} = \frac{1}{h_n c_2(1-c_2)}[c_2(2\nu_1 - c_2)u(x_n + h_n) + (1-2\nu_1)u(x_n + c_2 h_n) + ((c_2)^2 + 2(1-c_2)\nu_1 - 1)u(x_n)] \tag{38}$$

[illegible] $x_{n,2}$

$$V_{n,2} = \frac{1}{h_n c_2(1-c_2)}[c_2(2\nu_2 - c_2)u(x_n + h_n) + (1-2\nu_2)u(x_n + c_2 h_n) + ((c_2)^2 + 2(1-c_2)\nu_2 - 1)u(x_n)] \tag{39}$$

Backward formulas [11].

$$V_{n,1} = \frac{1}{h_n c_1(1-c_1)}[c_1(2\nu_1 - c_1)u(x_n + h_n) + (1-2\nu_1)u(x_n + c_1 h_n) + ((c_1)^2 + 2(1-c_1)\nu_1 - 1)u(x_n)] \tag{40}$$

$$V_{n,2} = \frac{1}{h_n c_1(1-c_1)}[c_1(2\nu_2 - c_1)u(x_n + h_n) + (1-2\nu_2)u(x_n + c_1 h_n) + ((c_1)^2 + 2(1-c_1)\nu_2 - 1)u(x_n)]. \tag{41}$$

We now explain the treatment of the term μu_x. The consistent way of discretizing this term is by taking into account the transport of information produced by each of the integral operators. We may summarize the criterion as follows.

REMARK 3.3. [illegible] μu_x [illegible] μ [illegible] (38) (39) [illegible] $\mu > 0$ [illegible] (40) (41) [illegible] $\mu < 0$

3.3. Time integration. Let k be the time step size, $t_m = mk$, for $m = 0, \ldots, N_t$. For a European option, the explicit backward differentiation formula (BDF) method reads

$$\frac{3}{2}u^m - 2u^{m-1} + \frac{1}{2}u^{m-2} + kru^m = k[I^+(\bar{v}^m) + I^-(\bar{v}^m) + \mu\bar{v}^m], \tag{42}$$

where $\bar{v}^m = 2v^{m-1} - v^{m-2}$, and I^+, I^- denote the discretizations by collocation of the operators I_+ and I_-, respectively. The addition on the right-hand side needs to be properly defined, as we add quantities that are not defined over the same mesh. To give a meaning to this sum, consider the base mesh $X^+ \cup X^-$, i.e., the mesh where these additions make sense. Let us consider the case of Radau points. Let P_+ denote the prolongation operator from X^+ to $X^+ \cup X^-$. An example of such a prolongation is the following:

$$P_+(w)_{3k} = w_{k,1}, \tag{43}$$

$$P_+(w)_{3k+1} = (w_{k,1} + w_{k,2})/2, \tag{44}$$

$$P_+(w)_{3k+2} = w_{k,2} \tag{45}$$

for $k = 0, \ldots, N-1$. A negative prolongation P_- may be defined likewise. Then, the sum of integrals in (42) is in the sense $P_+(I^+(\bar{v}^m)) + P_-(I^-(\bar{v}^m))$. If the collocation parameters are chosen to be $c_1 = 0, c_2 = 0.5$ for the positive situation and $c_1 = 0.5, c_2 = 1$ for the negative situation, one does not need to interpolate as the meshes coincide, except for boundary points. However, this choice is less accurate as pointed out in section 3.1.4.

Each derivative v^m present in the integrands is evaluated by the local differentiation formulas of section 3.2.1. An iteration of the form

$$u^m = J(v^{m-1}, v^{m-2}) \tag{46}$$

is then found after collecting the prolongation terms. The first input to this iteration is the collocation derivative of the payoff v^0, and the second, v^1, is the result of differentiating one explicit Euler iteration.

For American options the iteration is similar:

$$\begin{cases} u^m \geq J(v^{m-1}, v^{m-2}), \\ u^m \geq \psi, \\ (u^m - J(v^{m-1}, v^{m-2}), u^m - \psi) = 0, \\ u^0 = \psi. \end{cases} \tag{47}$$

Here $(\cdot, \cdot)$ is the standard inner product in $\mathbb{R}^{3N}$, because these iterations are carried out in a mesh of size $3N$ due to the collocation points and the size of the base mesh. The solution of this discrete LCP is simply $u^m = \max(J(v^{m-1}, v^{m-2}), \psi)$.

3.4. The algorithm. We summarize the techniques discussed in an algorithm. The notations v_f and v_b stand for the forward and backward collocation derivative, respectively, as in section 3.2.1. The operator $P_+(u)$ means the prolongation operator by interpolation from collocation mesh X^+ to the base mesh $X^+ \cup X^-$. The operator $P_-(u)$ defines likewise the prolongation from X^-. Finally, the term μu_x is prolongated similarly and added to one of the integrals according to Remark 3.3.

Algorithm

(1) Initialize u^0, and define the starting derivatives v_f^0, v_b^0 according to (38)-(41).
(2) Compute the integral $I_+^0 := I_+(v_f^0)$ from (66) and $I_-^0 := I_-(v_b^0)$ from (79). Make prolongations $P_+(I_+^0)$, $P_-(I_+^0)$ using (44)–(45) and build the right-hand side vector $J(v^0)$ (similar to (46)).
(3) Compute one step Euler u^1; thereafter compute v_f^1 and v_b^1.
(4) Compute $I_+^1 := I_+(v_f^1)$ and $I_-^1 := I_-(v_b^1)$. Make prolongations $P_+(I_+^1)$, $P_-(I_-^1)$ and build the right-hand side vector $J(v^0, v^1)$ as in (46).
(5) Compute a BDF2 step u^2 (cf. (42)), to obtain v_f^2 and v_b^2.
(6) Update $u^1 \to u^0$, $u^2 \to u^1$, $v^1 \to v^0$, $v^2 \to v^1$. Let $m \to m+1$ and repeat steps (4)–(6) until $m = N_t$.

Whenever we compute the integrals I_+ and I_- we apply the FFT algorithm to accelerate the computation of the convolution. This is standard practice so the details are left out. In previous work [3, 4] we made use of this technique to reduce computational time. We only point out that formulas (64) and (82) take on a special form

when the mesh is uniform, representing the multiplication of a Toeplitz matrix by a vector. These matrix-vector multiplications can be efficiently implemented following the explanation in [36].

4. Numerical experiments. The experiments in this section are performed on an AMD Athlon(TM) XP 2700. The first experiment consists in computing European put options with different values of α in order to verify that the order of convergence is quadratic and independent of α; see Table 2. Here, "rate" is defined as the ratio of errors. So, a factor 4 corresponds to second-order accuracy. The numerical values are compared with the Carr–Madan formula [14]. We note that with increasing α, the number of time steps should be increased to obtain a stable solution. The numerical tests for the option pricing problems confirm that in order for the solution to exhibit a stable behavior, a stability condition $k \le C(\alpha)h^{\alpha}$ should be fulfilled. The experiments show also that the stability constant $C(\alpha)$ decreases when α approaches 1, which is in accordance with the stability analysis in Appendix D.

Since the grid convergence rate is also deteriorating with increasing α (Table 2), one does not expect the superconvergence for α close to one. However, $O(h^2)$ accuracy is clearly observed.

A second experiment consists in computing an American put option (Table 3). We again observe at least second-order accuracy of the scheme. An accuracy higher than two seems awkward at first sight with a second-order time integration scheme. Starting with $N = N_t = 20$, however, and doubling both N and N_t, one observes (not shown here) in fact that the explicit BDF2 collocation method is second-order in time. The higher rate observed in Table 3 is basically the effect of having a higher order in space than in time, combined with a special choice of h and k.

The CPU times behave as predicted by the application of the FFT for each time step, with $O(N_t N \log N)$ complexity. Notice that the computations will be significantly more rapid if we need engineering accuracy, i.e., 3 digits accurate. Pictures of the American put option values at $t = 0$ and the time-dependent early exercise boundaries for different values of parameter α are shown in Figure 2. Increasing α means an increase of the intensity of small jumps. Loosely speaking, the prices behave similarly to a volatility increase in the classical Black–Scholes framework.

In a third experiment, European and American butterfly options are considered (Table 4 and Figure 3). A butterfly spread is the result of buying call options with strikes K_1 and K_2, respectively, and selling two call options with strike price $K_3 = (K_1 + K_2)/2$. That is, the payoff is of the form

$$\psi(x) = (e^x - K_1)^+ - 2\left(e^x - \frac{K_1 + K_2}{2}\right)^+ + (e^x - K_2)^+.$$

A European style butterfly is appropriate when the investor thinks large moves in the price of the underlying asset are unlikely. Usually the middle strike K_3 is close to today's spot price S_0, so that it pays off when the underlying asset price stays close to S_0. In Figure 3 we show American style butterflies (in the S coordinate) with different values of α and in Table 4 the second-order convergence of a European style butterfly is shown experimentally. The reason for including a butterfly option example is to show that the method discussed in this paper is rather general and provides accurate numerical results for other kinds of payoffs (nonconvex payoffs, for example), without any further complications.

A final remark is concerning the so-called smooth-fit or smooth-paste principle. This principle essentially states that the derivative of the Black–Scholes American

TABLE 2

Computation of European options for different values of α. Other parameters: $r = 0.1, q = 0, K = 1, C = 1, M = G = 5, T = 1$.

α	N	N_t	Error at $x = 0$	Rate	Ref. value
0.1	100	20	9.73E-4		0.06353404
	200	40	1.82E-4	5.3	
	400	80	3.18E-5	5.7	
	800	160	5.70E-6	5.5	
$\alpha = 0.5$	100	40	1.58E-3		0.10296690
	200	80	3.26E-4	4.8	
	400	160	6.38E-5	5.1	
	800	320	1.21E-5	5.2	
$\alpha = 0.8$	100	200	8.19E-3		0.14789424
	200	400	1.96E-3	4.1	
	400	800	4.47E-4	4.3	
	800	1600	9.99E-5	4.4	

TABLE 3

Grid convergence for an American put. Parameters: $r = 0.1, q = 0, K = 1, C = 1,\ M = G = 5, \alpha = 0.5$.

N	N_t	error at $x = 0$	rate	CPU-time (s)
100	40	1.73E-3		0.76
200	80	3.36E-4	5.1	1.3
400	160	4.80E-5	7	5.28
800	320	6.80E-6	7	21.10
ref. value		0.112171		

TABLE 4

Computation of a European butterfly with parameters: $r = 0.1, q = 0, K = 1, T = 1, C = 1, G = 3, M = 5, \alpha = 0.5$.

N	N_t	ℓ_∞-errors	rate
50	30	1.22E-2	
100	60	3.22E-3	3.8
200	120	5.94E-4	5.4
400	240	6.28E-5	9.4

option price is a continuous function, also at the exercise boundary. It is known that this property does not generally hold true when the underlying follows a Lévy process (for a partial proof see [4] and for a complete characterization for perpetual options see [2]). The butterfly payoff has the feature of combining both the put and the call properties. We observe on the left-hand side of the "hat payoff" in Figure 3 (the call part) that the solution enters the payoff smoothly, whereas, for the right-hand side the smooth-fit is not observable. In this example two free-boundaries appear and despite the continuous dividend yield $q = 0$. This is in contrast to the well-known Black–Scholes situation regarding the call option for which early exercise is not favorable with $q = 0$.

5. Conclusions. In this paper we have proposed a numerical procedure to solve integro-differential scalar equations that arise in finance, when pricing options for which the underlying asset follows a Lévy process. The main idea is to integrate by parts in order to transform the integral operator into a sum of Volterra operators with weakly singular kernels. Known numerical techniques like the collocation method for Volterra equations with weakly singular kernels prove useful to set up a high-order

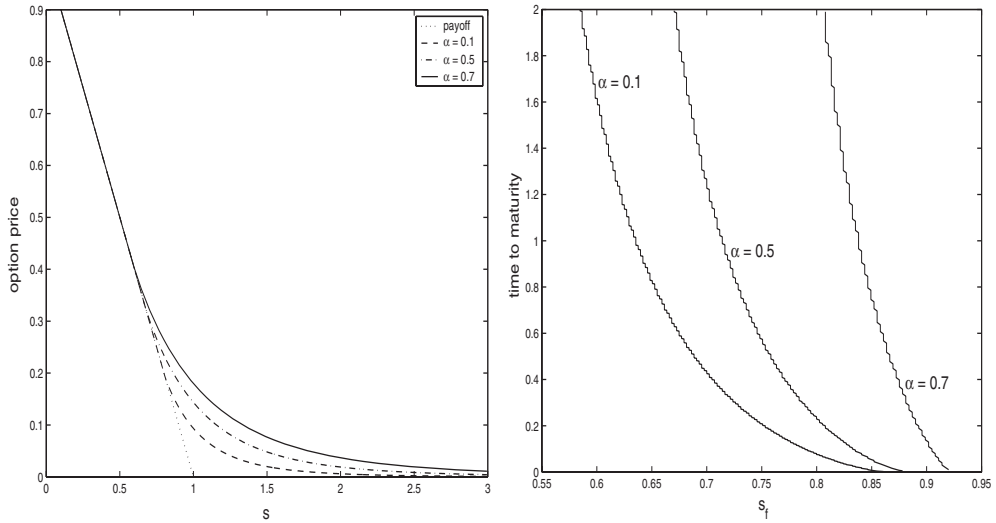


FIG. 2. *American put option prices at $t = 0$ (left) and early exercise boundaries (right) for different values of α. Other parameters: $r = 0.1$, $q = 0$, $K = 1$, $T = 2$, $C = 1$, $G = 5$, $M = 5$.*

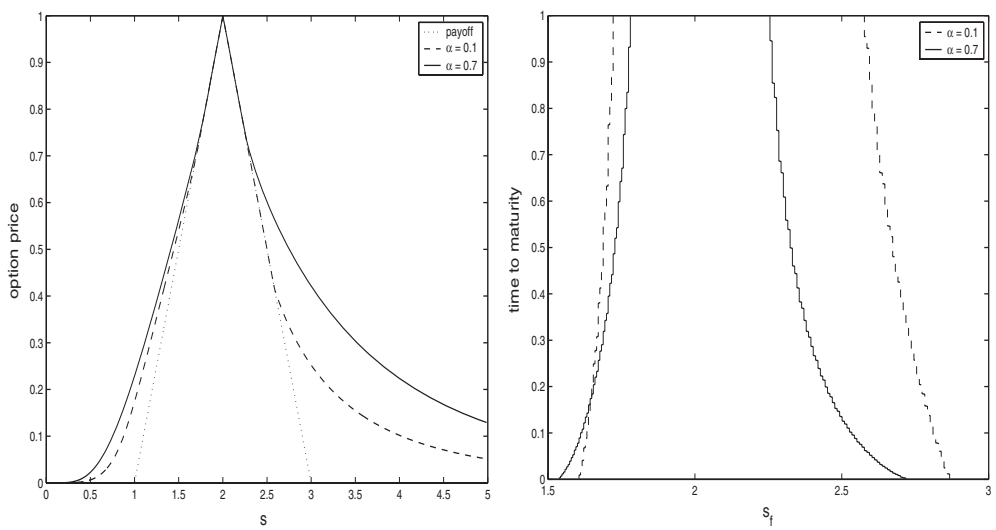


FIG. 3. *American butterfly prices at $t = 0$ (left) and early exercise boundaries for $\alpha = 0.1$ and $\alpha = 0.7$. Other parameters: $r = 0.1$, $q = 0$, $K_1 = 1$, $K_2 = 3$, $T = 1$, $C = 1$, $G = 3$, $M = 5$.*

discretization. Throughout a series of experiments it has been shown that an explicit method is a natural choice for these kinds of problems and for parameter $\alpha \leq 0.7$, as they essentially behave like transport equations. The method proposed can deal with complicated payoffs in a natural way. Due to the FFT algorithm, the complexity of the method is close to linear. With a fast, second-order method at hand, the calibration problem for American options under Lévy processes is now within reach.

Appendix A. Analytic evaluation of integral equation (13). In the numerical experiments in section 3.1.4 we make use of the analytic expression for

$$f(x) = \int_0^\infty (u(x+y) - u(x)) \frac{e^{-My}}{y^{1+\alpha}} dy,$$

where $u(x) = (1 - e^x)^+$. In this paragraph we derive an expression for $f(x)$.

Note that $f(x) = 0$ for $x > 0$. For $x < 0$ we have

$$\begin{aligned} &\int_0^{+\infty} \left[(1 - e^{x+y})^+ - (1 - e^x)^+\right] \frac{e^{-My}}{y^{1+\alpha}} dy \\ &= e^x \int_0^{-x} (1 - e^y) \frac{e^{-My}}{y^{1+\alpha}} dy + (e^x - 1) \int_{-x}^{+\infty} \frac{e^{-My}}{y^{1+\alpha}} dy \\ &= e^x \int_0^{+\infty} (1 - e^y) \frac{e^{-My}}{y^{1+\alpha}} dy + e^x \int_{-x}^{+\infty} \frac{e^{-(M-1)y}}{y^{1+\alpha}} dy - \int_{-x}^{+\infty} \frac{e^{-My}}{y^{1+\alpha}} dy. \end{aligned} \tag{48}$$

To compute the first term, observe that for $\alpha < 0$ the formula

$$\int_0^\infty \frac{e^{-My}}{y^{1+\alpha}} dy = M^\alpha \Gamma(-\alpha)$$

holds. Therefore,

$$\int_0^{+\infty} (1 - e^y) \frac{e^{-My}}{y^{1+\alpha}} dy = \Gamma(-\alpha)[M^\alpha - (M-1)^\alpha]. \tag{49}$$

However, this formula also admits an extension to the complex plane excluding $\alpha \neq 1, 2, \ldots$; see [1]. The second and the third term in (48) can be computed with some quadrature rule, but here we opt for the following approach. These integrals can be written in terms of the so-called upper incomplete gamma function:

$$\Gamma(\beta, y) = \int_y^{+\infty} t^{\beta-1} e^{-t} dt. \tag{50}$$

That is, for instance,

$$\int_{-x}^{+\infty} \frac{e^{-My}}{y^{1+\alpha}} dy = M^\alpha \Gamma(-\alpha, -Mx). \tag{51}$$

An efficient algorithm for computing the incomplete gamma function is discussed in [38].

To summarize, we found that $f(x) = 0$ for $x > 0$ and

$$\begin{aligned} f(x) = e^x \Gamma(-\alpha)[M^\alpha - (M-1)^\alpha] \\ + e^x (M-1)^\alpha \Gamma(-\alpha, -(M-1)x) - M^\alpha \Gamma(-\alpha, -Mx), \quad x < 0. \end{aligned} \tag{52}$$

Appendix B. Collocation method on positive semiaxis. Here, we discretize (22) in the interval $x \in [a, b]$. One looks for a piecewise linear function, defined as V_l on each interval $[x_l, x_{l+1}]$, such that

$$f_{n,j} = \int_{x_{n,j}}^b (z - x_{n,j})^{-\alpha} V(z) \widetilde{k}(z - x_{n,j}) dz. \tag{53}$$

FIG. 4. *Interpolation points.*

The notation $f_{n,j}$ means the values $f(x_{n,j})$.

Splitting the integral term gives the sum

$$f_{n,j} = A_{n,j} + P_{n,j}, \qquad n = 0, \dots, N-1, \qquad j = 1, 2, \tag{54}$$

where the first term is the integral from collocation point $x_{n,j}$ until the next mesh point x_{n+1}:

$$A_{n,j} := \int_{x_{n,j}}^{x_{n+1}} (z - x_{n,j})^{-\alpha} V_n(z) \widetilde{k}(z - x_{n,j}) dz; \qquad j = 1, 2, \tag{55}$$

and the "lag term" $P_{n,j}$ takes into account the intervals $[x_l, x_{l+1}]$:

$$P_{n,j} := \sum_{l=n+1}^{N-1} \int_{x_l}^{x_{l+1}} (z - x_{n,j})^{-\alpha} V_l(z) \widetilde{k}(z - x_{n,j}) dz; \qquad j = 1, 2. \tag{56}$$

With the change of variables $z = x_l + \nu h_l$, both $A_{n,j}$ and $P_{n,j}$ transform, respectively, into

$$A_{n,j} = h_n^{1-\alpha} \int_{c_j}^{1} (\nu - c_j)^{-\alpha} V_n(x_n + \nu h_n) \widetilde{k}(h_n(\nu - c_j)) d\nu, \tag{57}$$

and

$$P_{n,j} = \sum_{l=n+1}^{N-1} h_l^{1-\alpha} \int_0^1 \left[\frac{x_l - x_{n,j}}{h_l} + \nu \right]^{-\alpha} V_l(x_l + \nu h_l) \widetilde{k}(x_l - x_{n,j} + \nu h_l) d\nu. \tag{58}$$

B.1. Product integration formulas on positive semiaxis. Formulas (57) and (58) are not suitable for straightforward evaluation. One would rather prefer some numerical approximation that keeps the order of accuracy. One possibility is to use an interpolatory quadrature rule.

Let us start by fixing two interpolation points belonging to the interval $[c_j, 1)$:

$$c_{j,p} := c_j + (1 - c_j) c_p, \qquad j, p = 1, 2.$$

In other words, the same collocation points c_p are used through a linear mapping of the interval $[0, 1]$ (where they are located) into the interval $[c_j, 1)$; see [11]. Figure 4 illustrates the relative positions of these points. Let $L_{j,p}$ represent the Lagrange interpolation polynomials associated to the points $c_{j,p}$. The product integration rule consists in approximating the integrand (except for the singular part) by a piecewise linear polynomial, i.e., one employs in (57) the following approximation:

$$V_n(x_n + \nu h_n) \widetilde{k}(h_n(\nu - c_j)) \approx \sum_{p=1}^{2} L_{j,p}(\nu) V_n(x_n + c_{j,p} h_n) \widetilde{k}(h_n(c_{j,p} - c_j)) \tag{59}$$

for $\nu \in [c_j, 1]$. This formula requires the values $V_n(x_n + c_{j,p}h_n)$. Let L_q and $V_{n,q}$ denote the Lagrange interpolation polynomials at $x_{n,q}$ for $q = 1, 2$ and the nodal values of V_n at the same collocation points, respectively. It follows that

$$V_n(x_n + c_{j,p}h_n) = \sum_{q=1}^{2} L_q(c_{j,p}) V_{n,q}, \qquad j = 1, 2. \tag{60}$$

The function $V(z)$ need not be continuous. Upon substitution of (60) into (59) and thereafter into (57), one arrives at the formulas suitable for numerical evaluation,

$$\hat{A}_{n,j} = h_n^{1-\alpha} \sum_{p=1}^{2} \sum_{q=1}^{2} \lambda_{j,p} L_q(c_{j,p})\, V_{n,q}\, \widetilde{k}(h_n(1 - c_j)c_p), \tag{61}$$
$$n = 0, \ldots, N-1; \qquad j = 1, 2,$$

where we introduced the (α-dependent) weights

$$\lambda_{j,p} := \int_{c_j}^{1} (\nu - c_j)^{-\alpha} L_{j,p}(\nu) d\nu, \qquad j, p = 1, 2. \tag{62}$$

An analytic expression for these weights will be given in short. For the integrals in the lag term $P_{n,j}$ the analysis goes similarly. There the interpolation nodes are the same as the collocation points c_j, so

$$V_l(x_l + \nu h_l)\widetilde{k}(x_l - x_{n,j} + \nu h_l) \approx \sum_{q=1}^{2} L_q(\nu)\, V_{l,q}\, \widetilde{k}(x_l - x_{n,j} + c_q h_l), \tag{63}$$

which gives the product integration rule for the lag-term

$$\hat{P}_{n,j} = \sum_{l=n+1}^{N-1} h_l^{1-\alpha} \left(\sum_{q=1}^{2} \omega_{l,j,q}\, V_{l,q}\, \widetilde{k}(x_l - x_{n,j} + c_q h_l) \right), \tag{64}$$

and the weights are defined as

$$\omega_{l,j,q} := \int_0^1 \left[\frac{x_l - x_{n,j}}{h_l} + \nu \right]^{-\alpha} L_q(\nu) d\nu, \tag{65}$$
$$j, q = 1, 2; \qquad l = n+1, \ldots, N-1.$$

To conclude, the entries $f_{n,j}$ are approximated as follows:

$$f_{n,j} \approx \hat{P}_{n,j} + \hat{A}_{n,j}, \qquad n = 0, \ldots, N-1, \qquad j = 1, 2. \tag{66}$$

B.2. Weights on positive semiaxis. Formulas for the weights in (62) and (65) are derived next. We start with $\omega_{l,j,q}$ in (65). To simplify the notation let us introduce the numbers

$$d_{l,j} = \frac{x_l - x_{n,j}}{h_l}. \tag{67}$$

This amounts to saying that the following integral is to be evaluated analytically:

$$\int_0^1 (d_{l,j} + \nu)^{-\alpha} L_q(\nu) d\nu. \tag{68}$$

Recall that the Lagrange interpolation polynomials $L_q(\nu)$ are given by the formulas

$$L_1(\nu) = \frac{\nu - c_2}{c_1 - c_2}, \qquad L_2(\nu) = \frac{\nu - c_1}{c_2 - c_1}. \tag{69}$$

We carry out the computation only for L_1, as for L_2 it goes similarly. Adding and subtracting $d_{l,j}$ in the expression $\nu - c_2$ produces two integrals that may be easily evaluated, indeed,

$$\int_0^1 (d_{l,j} + \nu)^{-\alpha}(\nu - c_2)d\nu = \int_0^1 (d_{l,j} + \nu)^{-\alpha+1}d\nu - (d_{l,j} + c_2)\int_0^1 (d_{l,j} + \nu)^{-\alpha}d\nu.$$

Integrating above yields the formulas for the weights

(70)
$$\begin{cases} \omega_{l,j,1} &= \dfrac{(d_{l,j}+1)^{2-\alpha} - (d_{l,j})^{2-\alpha}}{(2-\alpha)(c_1 - c_2)} - (d_{l,j} + c_2)\dfrac{(d_{l,j}+1)^{1-\alpha} - (d_{l,j})^{1-\alpha}}{(1-\alpha)(c_1 - c_2)}, \\ \omega_{l,j,2} &= \dfrac{(d_{l,j}+1)^{2-\alpha} - (d_{l,j})^{2-\alpha}}{(2-\alpha)(c_2 - c_1)} - (d_{l,j} + c_1)\dfrac{(d_{l,j}+1)^{1-\alpha} - (d_{l,j})^{1-\alpha}}{(1-\alpha)(c_2 - c_1)}. \end{cases}$$

To compute $\lambda_{j,p}$, let us change to the variable $\nu = c_j + (1 - c_j)\rho$. This simplifies the Lagrange interpolation polynomials $L_{j,q}(\nu)$ as the following equalities show:

$$L_{j,1}(\nu) = \frac{\nu - c_{j,1}}{c_{j,2} - c_{j,1}} = \frac{v - c_j - (1 - c_j)\rho}{(1 - c_j)(c_2 - c_1)} = \frac{\rho - c_1}{c_2 - c_1} = L_1(\rho).$$

Notice that $c_j \neq 1$ to simplify in this expression. Analogously one finds that $L_{j,2}(\nu) = L_2(\rho)$. Consequently, the expression

$$\lambda_{j,p} = (1 - c_j)^{1-\alpha}\int_0^1 \rho^{-\alpha}L_p(\rho)d\rho$$

gives an alternative way to evaluate these weights. Thus, we may use the previously derived formulas for the ω-weights with $d_{l,j} \equiv 0$ to arrive at the expressions for the λ-weights:

$$\begin{cases} \lambda_{j,1} &= \dfrac{(1 - c_j)^{1-\alpha}}{c_1 - c_2}\left[\dfrac{1}{2-\alpha} - \dfrac{c_2}{1-\alpha}\right], \\ \lambda_{j,2} &= \dfrac{(1 - c_j)^{1-\alpha}}{c_2 - c_1}\left[\dfrac{1}{2-\alpha} - \dfrac{c_1}{1-\alpha}\right]. \end{cases} \tag{71}$$

Appendix C. Integral equation over the negative semiaxis. For the negative semiaxis the discretization is very similar to the one explained in Appendix B. We keep the same notation from the positive situation and summarize the procedure. The integral equation for the negative semiaxis reads

$$g(x) = \int_{-\infty}^0 (u(x + y) - u(x))\frac{e^{-G|y|}}{|y|^{1+\alpha}}dy. \tag{72}$$

This expression reduces, after integration by parts, to

$$g(x) = -\int_{-\infty}^0 v(x + y)\hat{k}(y)dy. \tag{73}$$

Here we have introduced the function

$$\hat{k}(-w) := \int_{-\infty}^{-w} \frac{e^{-G|\zeta|}}{|\zeta|^{1+\alpha}} d\zeta, \qquad w > 0, \tag{74}$$

and v stands for the derivative of u with respect to the space variable. The subscript "-" for the negative kernel is again omitted. An expression in terms of the incomplete gamma function exists, namely, $\hat{k}(-w) = G^{\alpha}\Gamma(-\alpha, Gw)$. The Lipschitz property (17) is now invoked to allow the above integration by parts, along with a growth property for large negative values; i.e.,

- there exist constants $C_3 > 0$ and $0 < C_4 < G$ such that $|u(z)| \leq C_3 e^{C_4|z|}$ for $z \to -\infty$.

Similarly to the situation on the positive semiaxis, one takes into account the singularity of the kernel $\hat{k}$. It is therefore convenient to introduce the kernel

$$\widetilde{k}(w) := w^{\alpha}\hat{k}(-w), \qquad w > 0, \tag{75}$$

and focus on the discretization of the integral equation

$$g(x) = -\int_{-\infty}^{x} (x-z)^{-\alpha} v(z) \widetilde{k}(x-z) dz. \tag{76}$$

Considering the collocation parameters

$$0 < c_1 < c_2 \leq 1, \tag{77}$$

the collocation mesh is defined as in (23). To discretize (76) write

$$g_{n,j} = -\int_{a}^{x_{n,j}} (x_{n,j}-z)^{-\alpha} V(z) \widetilde{k}(x_{n,j}-z) dz. \tag{78}$$

As for the positive semiaxis, one applies the product integration rule with a suitable choice of interpolation points. Summarizing, the quantities $g_{n,j}$ are approximated as follows:

$$g_{n,j} \approx \hat{B}_{n,j} + \hat{Q}_{n,j}, \qquad n = 0, \ldots, N-1, \qquad j = 1, 2. \tag{79}$$

The first component is computed with the formula

$$\hat{B}_{n,j} = -h_n^{1-\alpha} \sum_{p=1}^{2} \sum_{q=1}^{2} \lambda_{j,p} L_q(c_{j,p})\, V_{n,q}\, \widetilde{k}(h_n c_j (1-c_p)), \qquad j = 1, 2, \tag{80}$$

the weights $\lambda_{j,p}$ being

$$\begin{cases} \lambda_{j,1} &= \dfrac{(c_j)^{1-\alpha}}{c_2 - c_1} \left[\dfrac{1}{2-\alpha} - \dfrac{1-c_2}{1-\alpha} \right], \\ \\ \lambda_{j,2} &= \dfrac{(c_j)^{1-\alpha}}{c_1 - c_2} \left[\dfrac{1}{2-\alpha} - \dfrac{1-c_1}{1-\alpha} \right], \end{cases} \tag{81}$$

and $c_{j,p} = c_j c_p$. The second component in this sum corresponds to an approximation of the lag-term,

$$\hat{Q}_{n,j} = -\sum_{l=0}^{n-1} h_l^{1-\alpha} \left[\sum_{q=1}^{2} \omega_{l,j,q}\, V_{l,q}\, \widetilde{k}(x_{n,j} - x_l - c_q h_l) \right], \tag{82}$$

where the weights for the negative semiaxis are

(83)
$$\begin{cases} \omega_{l,j,1} &= -\dfrac{(d_{l,j})^{2-\alpha}-(d_{l,j}-1)^{2-\alpha}}{(2-\alpha)(c_1-c_2)}+(d_{l,j}-c_2)\dfrac{(d_{l,j})^{1-\alpha}-(d_{l,j}-1)^{1-\alpha}}{(1-\alpha)(c_1-c_2)}, \\ \omega_{l,j,2} &= -\dfrac{(d_{l,j})^{2-\alpha}-(d_{l,j}-1)^{2-\alpha}}{(2-\alpha)(c_2-c_1)}+(d_{l,j}-c_1)\dfrac{(d_{l,j})^{1-\alpha}-(d_{l,j}-1)^{1-\alpha}}{(1-\alpha)(c_2-c_1)}. \end{cases}$$

As for the positive integral, a compensation constant is necessary to smooth the kernel. The constant in this situation is

$$\theta_- = \frac{G^\alpha \Gamma(2-\alpha)}{\alpha(1-\alpha)}, \tag{84}$$

so that

$$k_-(x) := \widetilde{k}(x) + \theta_- x^\alpha,$$

and the splitting of the integral equation with respect to the new kernel reads

$$g(x) = -\int_{-\infty}^{x} (x-z)^{-\alpha} v(z) k_-(z-x)dz + \theta_- \int_{-\infty}^{x} v(z)dz. \tag{85}$$

Appendix D. Stability analysis. In order to gain some insight into the stability of the resulting discretization of the PIDE, we perform here a Von Neumann stability analysis; however, we do this for a "model" version of the time-dependent problem under consideration.

Let $\alpha \in (0,1)$ and consider the equation

$$u_t = -\int_0^x (x-z)^{-\alpha} u_z(z,t)dz, \quad x \in [0,1]. \tag{86}$$

To simplify the analysis, we concentrate on a special case of the collocation method, namely, the grid is assumed equidistant, i.e., $h_n = h$, and the collocation parameters are $c_1 = 0.5$ and $c_2 = 1$. These choices give the collocation points $x_{n,1} = (n+1/2)h$ and $x_{n,2} = (n+1)h$.

Let us analyze the Euler scheme for the midpoints $x_{n,1}$ (for the other collocation points the analysis goes similarly). By $u_{n+\frac{1}{2}}$ we denote the value of the solution at the midpoints. With the notation in Appendix C, (80), (81), (82), and (83), we may write the explicit collocation scheme as

$$u^m_{n+\frac{1}{2}} - u^{m-1}_{n+\frac{1}{2}} = -\gamma h \left[\sum_{p=1}^{2} \lambda_{1,p} (L_1(c_{1,p}) V^{m-1}_{n,1} + L_2(c_{1,p}) V^{m-1}_{n,2}) + \sum_{l=0}^{n-1} (\omega_{l,1,1} V^{m-1}_{l,1} + \omega_{l,1,2} V^{m-1}_{l,2}) \right]. \tag{87}$$

Here we have introduced the parameter

$$\gamma := \frac{k}{h^\alpha}. \tag{88}$$

Our objective is to show that by controlling the parameter γ one attains stability of the Euler scheme.

For the collocation parameters under consideration the corresponding collocation derivatives in (87) are

$$V_{n,1}^{m-1} = \frac{u_{n+1}^{m-1} - u_n^{m-1}}{h}, \quad V_{n,2}^{m-1} = \frac{3u_{n+1}^{m-1} - 4u_{n+\frac{1}{2}}^{m-1} + u_n^{m-1}}{h}, \tag{89}$$

and the Lagrange interpolations are

$$L_1(c_{1,1}) = 3/2, \quad L_1(c_{1,2}) = 1, \quad L_2(c_{1,1}) = -1/2, \quad L_2(c_{1,2}) = 0. \tag{90}$$

Substituting (90) and (89) into (87) and collecting terms we arrive at the scheme

$$u_{n+\frac{1}{2}}^{m} = u_{n+\frac{1}{2}}^{m-1} - \gamma \sum_{l=0}^{n} \left[(A_l + 3B_l) u_{l+1}^{m-1} - 4B_l u_{l+\frac{1}{2}}^{m-1} + (B_l - A_l) u_l^{m-1} \right], \tag{91}$$

where $A_n = \frac{3}{2}\lambda_{1,1} + \lambda_{1,2}$, $B_n = -\frac{1}{2}\lambda_{1,1}$, $A_l = \omega_{l,1,1}$, and $B_l = \omega_{l,1,2}$ $(l = 0 \dots n-1)$. The λ-weights are straightforwardly computed using (71), yielding

$$\lambda_{1,1} = \frac{2^\alpha}{2-\alpha}, \quad \lambda_{1,2} = \frac{2^\alpha}{2-\alpha} - \frac{2^{\alpha-1}}{1-\alpha}. \tag{92}$$

The ω-weights are computed using (70) with

$$d_{l,1} = n - l + 1/2, \quad d_{l,2} = n - l + 1. \tag{93}$$

A method to obtain a necessary condition for stability is to substitute the Fourier modes $u_l^m = g^m e^{il\theta}$ into the scheme and thereafter perform the necessary simplifications. This procedure is commonly known as discrete Von Neumann stability analysis. Applied to our case it produces the so-called amplification factor

$$g(\theta) = 1 - \gamma \sum_{l=0}^{n} \left[(A_l + 3B_l) e^{i(l-n+1/2)\theta} - 4B_l e^{i(l-n)\theta} + (B_l - A_l) e^{i(l-n-1/2)\theta} \right]. \tag{94}$$

If the scheme is stable, it is also stable on a particular solution. Hence, the amplification factor must satisfy the requirement

$$|g(\theta)| \leq 1. \tag{95}$$

We refer the reader to [35] for details on the amplification factor in connection with the Von Neumann stability analysis. We are looking for a condition of the form $\gamma \leq C(\alpha)$ such that (95) holds. Instead of deriving this condition for model equation (86), we present in Table 5 some sharp values of $C(\alpha)$ for distinct choices of α, and two representative examples are displayed in Figure 5.

A limiting case is distinguished: For $\alpha \to 0$ one notices that $C(\alpha) \to 1$. This behavior is intuitively correct as for $\alpha = 0$, equation (86) reduces to the ODE $u_t = -u + u(0)$. For $\alpha = 0$ the condition says $k \leq 1$, which is the necessary condition for explicit Euler, since the amplification factor for $u_t = -u + u(0)$ is

$$g(\theta) = 1 - k + ke^{-im\theta}.$$

TABLE 5
Values of $C(\alpha)$ as a function of α.

α	0.05	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.95
$C(\alpha)$	0.99	0.97	0.92	0.85	0.76	0.67	0.57	0.46	0.2	0.03	0.007

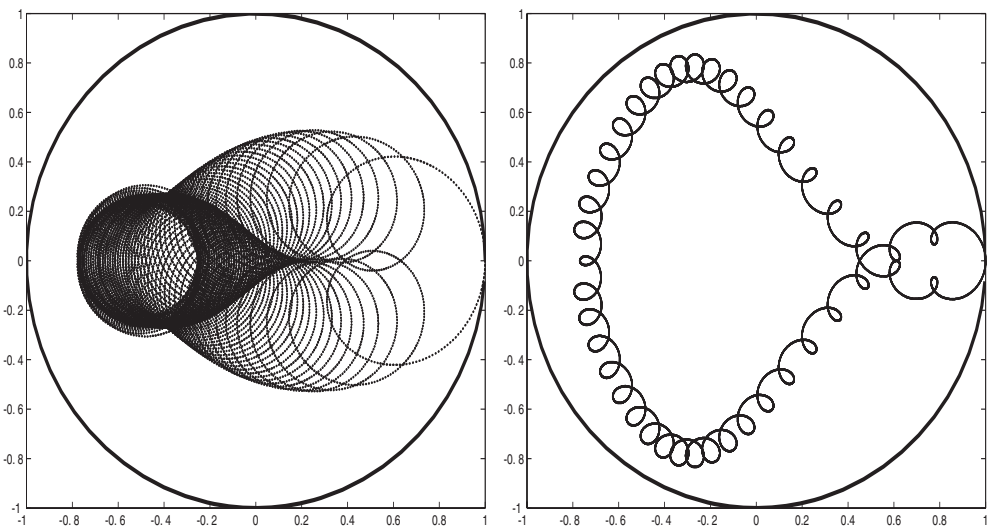


FIG. 5. *Values of $g(\theta)$ inside unit circle, $k = C(\alpha)h^\alpha$. Left: $\alpha = 0.3, N = 50$. Right: $\alpha = 0.7, N = 20$, with $C(0.3) = 0.85$ and $C(0.7) = 0.46$ as in Table 5.*

Then, $|g| \leq 1$ implies $|\operatorname{Re} g| \leq 1$, which in turn implies $k \leq 1$.

The insight gained here from model equation (86) with explicit Euler, i.e., the α-dependent constant $C(\alpha)$ so that (95) holds, is in full agreement with the observations regarding stability for our target discrete PIDE, discretized with explicit BDF2, in the numerical experiments of section 4. We include here a numerical experiment with the test equation (86). Three values of α are chosen to illustrate that the values in Table 5 are sharp; see Table 6. This table shows that

$$k \leq C(\alpha)h^\alpha \tag{96}$$

is in fact a necessary condition for stability. Namely, if (96) is not satisfied, the scheme is not stable. It should be mentioned that $C(\alpha)$ depends weakly on N, and the values in Table 5 are conservative in the sense that they are found with $N = 20$. For larger N, the pictures in Figure 5, for example, tend to shrink.

It can be observed in Table 6 that the number of time steps needed for stability grows with increasing α. For $\alpha \leq 0.7$ the number of time steps remains reasonable, and explicit time-stepping can efficiently be used. For α closer to 1, however, one may need to consider an implicit time discretization. For $\alpha = 0.9$ we see a difference of 200 time steps for a consistency between the stability analysis and the numerical experiment. The reason for this large difference lies in the assumption that a satisfactory stability criterion is $|g| \leq 1$ (95). In fact, a weaker stability criterion that is both necessary and sufficient is given by $|g| \leq 1 + \tilde{C}k$, for $\tilde{C}$ independent of h and k. This typically gives larger circles, so $C(\alpha)$ in the table represents a conservative estimate. The tendency with respect to stability is, however, clear.

TABLE 6
Stability experiments on test equation (86) *with* $T = 1$. *The symbol "v" means "holds" and "X" means "does not hold." The initial condition is taken as* $u(x, 0) = \exp(x)$.

α	N	N_t	Stability	Equation (96)
0.05	100	2	v	v
0.3	100	6	v	v
	100	4	X	X
0.5	100	30	v	v
	100	10	X	X
0.7	100	140	v	v
	100	80	X	v
	100	40	X	X
0.9	20	400	v	v
	20	300	X	v
	20	200	X	X

Acknowledgment. We would like to thank the referees for valuable comments.

REFERENCES

[1] M. ABRAMOWITZ AND I. A. STEGUN, EDS., *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, Dover, New York, 1992. Reprint of the 1972 edition.

[2] L. ALILI AND A. E. KYPRIANOU, *Some remarks on first passage of Lévy processes, the American put and pasting principles*, Ann. Appl. Probab., 15 (2005), pp. 2062–2080.

[3] A. ALMENDRAL, *Numerical valuation of American options under the CGMY process*, in Exotic Option Pricing and Advanced Lévy Models, W. Schoutens, A. Kyprianou, and P. Wilmott, eds., Wiley, Chichester, UK, 2005.

[4] A. ALMENDRAL AND C. W. OOSTERLEE, *On American Options under the Variance Gamma Process*, Tech. report, Delft University of Technology, Delft, The Netherlands, 2004.

[5] A. ALMENDRAL AND C. W. OOSTERLEE, *Numerical valuation of options with jumps in the underlying*, Appl. Numer. Math., 53 (2005), pp. 1–18.

[6] L. ANDERSEN AND J. ANDREASEN, *Jump-diffusion processes: Volatility smile fitting and numerical methods for option pricing*, Review of Derivatives Res., 4 (2000), pp. 231–262.

[7] O. E. BARNDORFF, *Processes of normal inverse Gaussian type*, Finance Stoch., 2 (1998), pp. 41–68.

[8] F. BLACK AND M. S. SCHOLES, *The pricing of options and corporate liabilities*, J. Political Economy, 7 (1973), pp. 637–54.

[9] S. I. BOYARCHENKO AND S. Z. LEVENDORSKIĬ, *Non-Gaussian Merton-Black-Scholes Theory*, Adv. Ser. Stat. Sci. Appl. Probab. 9, World Scientific, River Edge, NJ, 2002.

[10] H. BRUNNER, *Collocation Methods for Volterra Integral and Related Functional Differential Equations*, Cambridge Monogr. Appl. Comput. Math. 15, Cambridge University Press, Cambridge, UK, 2004.

[11] H. BRUNNER AND P. J. VAN DER HOUWEN, *The Numerical Solution of Volterra Equations*, CWI Monogr. 3, North–Holland, Amsterdam, 1986.

[12] J. Y. CAMPBELL, A. W. LO, AND A. C. MACKINLAY, *The Econometrics of Financial Markets*, Princeton University Press, Princeton, NJ, 1997.

[13] P. P. CARR, H. GEMAN, D. B. MADAN, AND M. YOR, *The fine structure of asset returns: An empirical investigation*, J. Business, 75 (2002), pp. 305–332.

[14] P. P. CARR AND D. B. MADAN, *Option valuation using the Fast Fourier Transform*, J. Comp. Finance, 2 (1999), pp. 61–73.

[15] P. P. CARR, D. B. MADAN, AND E. C. CHANG, *The variance Gamma process and option pricing*, European Finance Review, 2 (1998), pp. 79–105.

[16] R. CONT AND P. TANKOV, *Financial Modelling with Jump Processes*, Chapman & Hall, Boca Raton, FL, 2004.

[17] J. C. COX AND S. A. ROSS, *The valuation of options for alternative stochastic processes*, J. Financial Economics, 3 (1976), pp. 145–166.

[18] S. R. DAS AND S. FORESI, *Exact solutions for bond and option prices with systematic jump risk*, Review of Derivatives Res., 1 (1996), pp. 7–24.

[19] Y. d'Halluin, P. A. Forsyth, and K. R. Vetzal, *Robust numerical methods for contingent claims under jump diffusion processes*, IMA J. Numer. Anal., 43 (2005), pp. 1596–1626.
[20] B. Dupire, *Pricing with a smile*, RISK Magazine, 1 (1999), pp. 18–20.
[21] E. Eberlein, *Application of generalized hyperbolic Lévy motions to finance*, in Lévy Processes, Birkhäuser Boston, Boston, 2001, pp. 319–336.
[22] P. Eggermont, *Stability and robustness of collocation methods for Abel-type integral equations*, Numer. Math., 45 (1984), pp. 431–445.
[23] E. Hairer, C. Lubich, and M. Schlichte, *Fast numerical solution of weakly singular Volterra integral equations*, J. Comput. Appl. Math., 23 (1988), pp. 87–98.
[24] D. Heath and M. Schweizer, *Martingales versus PDEs in finance: An equivalence result with examples*, J. Appl. Probab., 37 (2000), pp. 947–957.
[25] S. Heston, *A closed-form solution for options with stochastic volatility with applications to bond and currency options*, Rev. Financ. Stud., 6 (1993), pp. 327–343.
[26] D. J. Higham, *An Introduction to Financial Option Valuation*, Cambridge University Press, Cambridge, UK, 2004.
[27] A. Hirsa and D. B. Madan, *Pricing American options under variance Gamma*, J. Comp. Finance, 7 (2004), pp. 63–80.
[28] J. Hull and A. White, *The pricing of options with stochastic volatilities*, J. Finance, 42 (1987), pp. 281–300.
[29] S. G. Kou, *A jump diffusion model for option pricing*, Management Science, 48 (2002), pp. 1086–1101.
[30] Y. K. Kwok, *Mathematical Models of Financial Derivatives*, 2nd ed., Springer Finance, Springer-Verlag, Singapore, 1998.
[31] P. Linz, *Analytical and Numerical Methods for Volterra Equations*, SIAM Stud. Appl. Math. 7, SIAM, Philadelphia, 1985.
[32] A. M. Matache, P. A. Nitsche, and C. Schwab, *Wavelet Galerkin pricing of American options on Lévy driven assets*, Quant. Finance, 5 (2005), pp. 403–424.
[33] R. C. Merton, *Option pricing when the underlying stocks are discontinuous*, J. Financ. Econ., 5 (1976), pp. 125–144.
[34] W. Schoutens, *Lévy Processes in Finance: Pricing Financial Derivatives*, John Wiley & Sons, Chichester, UK, 2003.
[35] J. C. Strikwerda, *Finite Difference Schemes and Partial Differential Equations*, The Wadsworth & Brooks/Cole Mathematics Series, Wadsworth & Brooks/Cole Advanced Books & Software, Pacific Grove, CA, 1989.
[36] C. Van Loan, *Computational Frameworks for the Fast Fourier Transform*, Frontiers Appl. Math. 10, SIAM, Philadelphia, 1992.
[37] P. Wilmott, J. Dewynne, and S. Howison, *Option Pricing*, Oxford Financial Press, Oxford, UK, 1993.
[38] S. Zhang and J. Jin, *Computation of Special Functions*, John Wiley & Sons, New York, 1996.

SIAM J. SCI. COMPUT.
Vol. 29, No. 1, pp. 118–138
© 2007 Society for Industrial and Applied Mathematics

AN ADAPTIVE MULTILEVEL METHOD FOR TIME-HARMONIC MAXWELL EQUATIONS WITH SINGULARITIES*

ZHIMING CHEN†, LONG WANG†, AND WEIYING ZHENG†

Abstract. We develop an adaptive edge finite element method based on reliable and efficient residual-based a posteriori error estimates for low-frequency time-harmonic Maxwell equations with singularities. The resulting discrete problem is solved by the multigrid preconditioned MINRES (minimum residual) iteration algorithm. We demonstrate the efficiency and robustness of the proposed method by extensive numerical experiments for cavity problems with singular solutions which include, in particular, scattering over screens.

Key words. Maxwell equations, singularities of solutions, adaptive finite element method, multigrid method

AMS subject classifications. 65N30, 65N55, 78A25

DOI. 10.1137/050636012

1. Introduction.

Let $\Omega \subset \mathbb{R}^3$ be a bounded polygonal domain with two disjoint connected boundaries Γ and Σ. Given a current density $\mathbf{f}$, we seek a time-harmonic electric field $\mathbf{E}$ subject to the perfectly conducting boundary condition on Γ and the impedance boundary condition on Σ:

$$\mathbf{curl}\,(\mu_r^{-1}\,\mathbf{curl}\,\mathbf{E}) - \kappa^2\varepsilon_r\,\mathbf{E} = \mathbf{f} \quad \text{in} \quad \Omega, \tag{1.1}$$

$$\mu_r^{-1}\,\mathbf{curl}\,\mathbf{E} \times \nu - \mathbf{i}\,\kappa\lambda\mathbf{E}_t = \mathbf{g} \quad \text{on} \quad \Sigma, \tag{1.2}$$

$$\mathbf{E} \times \nu = 0 \quad \text{on} \quad \Gamma, \tag{1.3}$$

where $\mathbf{i}$ is the imaginary unit, ν is the unit outer normal of the boundary, $\mathbf{E}_t := (\nu \times \mathbf{E}|_\Sigma) \times \nu$, ε_r is the complex relative dielectric coefficient, $\mu_r > 0$ is the relative magnetic permeability of the material in Ω, $\kappa > 0$ is the wave number, and $\lambda > 0$ is the impedance on Σ. We allow Σ to be empty, in which case (1.1)–(1.3) models electromagnetic wave propagation in a cavity with a perfectly conducting wall. For an absorbing boundary condition approximation of a scatting problem, $\mu_r = \lambda = 1$ on Σ and $\varepsilon_r = \mu_r = 1$ in a neighborhood of Σ, $\mathbf{g}$ can be computed from an incident field $\mathbf{E}_i$ ($\mathbf{g} = \mathbf{curl}\,\mathbf{E}_i \times \nu - \mathbf{i}\,\kappa\mathbf{E}_{i,t}$). In this paper, we focus on low-frequency problems; i.e., κ is not very large.

It is now well known that the solution of time-harmonic Maxwell equations can have much stronger singularities than the corresponding Dirichlet and Neumann singular functions of the Laplace operator when the computational domain is non-convex or the coefficients of the equations are discontinuous. For example, for the domains that have "screen" or "crack" parts as indicated in Figure 1.1, the regularity of the solution is only in $\mathbf{H}^s$ with $s < 1/2$. In this case the $\mathbf{H}^1$-conforming discretization cannot be used directly to solve the time-harmonic cavity problem (1.1)–(1.3). One

*Received by the editors July 14, 2005; accepted for publication (in revised form) June 27, 2006; published electronically January 26, 2007.

http://www.siam.org/journals/sisc/29-1/63601.html

†LSEC, Institute of Computational Mathematics, Academy of Mathematics and System Sciences, Chinese Academy of Sciences, Beijing, 100080, People's Republic of China (zmchen@lsec.cc.ac.cn, wangl@math.pku.edu.cn, zwy@lsec.cc.ac.cn). The work of the first author was supported in part by China NSF under grants 10025102 and 10428105, and by China MOST under grant G1999032802. The work of the third author was supported in part by China NSF under grant 10401040.

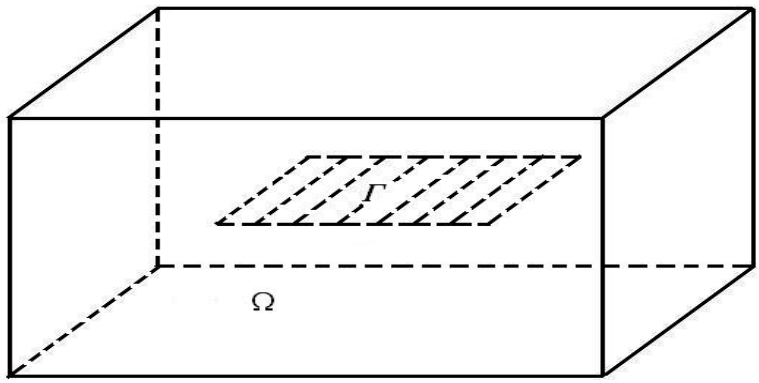


FIG. 1.1. *A domain with screen* Γ.

way to overcome this difficulty is to use the so-called singular field method, which decomposes the solution into a regular part that can be treated by $\mathbf{H}^1$-conforming Lagrangian finite elements and an explicit singular part [2], [10], [18], [19]. For the mathematical analysis of the singularities of the solutions of Maxwell equations, we refer to [8], [9], [16], [17], [18], and the references therein.

The first objective of this paper is to explore the possibility of extending the general framework of adaptive finite element methods based on a posteriori error estimates initiated in [3] to the time-harmonic Maxwell equations. A posteriori error estimates are computable quantities in terms of the discrete solution and known data that measure the actual discrete errors without knowledge of exact solutions. They are essential in designing algorithms for mesh modification which equidistribute the computational effort and optimize the computation. The ability of error control and the asymptotically optimal approximation property (see, e.g., [11] and [27]) make the adaptive finite element methods attractive for complicated physical and industrial processes (cf., e.g., [12], [14], and [15]).

A posteriori error estimates for Nédélec $\mathbf{H}(\mathbf{curl})$-conforming edge elements are obtained in [25] for Maxwell scattering problems and in [6] for eddy current problems. The key ingredient in the analysis is the orthogonal Helmholtz decomposition $\mathbf{v} = \nabla\varphi + \Psi$, where for any $\mathbf{v} \in \mathbf{H}(\mathbf{curl}; \Omega)$, $\varphi \in H^1(\Omega)$ and $\Psi \in \mathbf{H}(\mathbf{curl}; \Omega)$. Since a stable edge element interpolation operator is not available for functions in $\mathbf{H}(\mathbf{curl}; \Omega)$, some kind of regularity result for $\Psi \in \mathbf{H}(\mathbf{curl}; \Omega)$ is required. This regularity result is proved in [25] for domains with smooth boundary and in [6] for convex polyhedral domains. The key observation in our analysis is that if one removes the orthogonality requirement in the Helmholtz decomposition, the regularity $\Psi \in \mathbf{H}^1(\Omega)$ can be proved in the decomposition $\mathbf{v} = \nabla\varphi + \Psi$ for a large class of nonconvex polygonal domains or domains having screens [8], [9]; see also [19]. Our extensive numerical experiments for the lowest order edge element indicate that for the cavity problem (1.1)–(1.3) with very strong singularities $\mathbf{H}^s$ ($s < 1/2$), the adaptive methods based on our a posteriori error estimates have the very desirable quasi-optimality property

$$\|\mathbf{E} - \mathbf{E}_k\|_{\mathbf{H}(\mathbf{curl};\,\Omega)} \leq C\, N_k^{-1/3},$$

where N_k is the number of elements of the kth adaptive mesh $\mathcal{T}_k$, and $\mathbf{E}_k$ is the finite element solution over $\mathcal{T}_k$.

The second objective of this paper concerns the efficient solution of the large linear system of equations resulting from the edge element discretization of (1.1)–(1.3)

which is in general non-Hermitian and indefinite. We propose to use preconditioned MINRES (minimum residual) to solve the linear system of equations where the preconditioner is constructed by the multigrid solver of the stiffness matrix corresponding to the discretization of the following problem:

$$\mathbf{curl}\,(\mu_r^{-1}\,\mathbf{curl}\,\mathbf{u}) + \kappa^2|\varepsilon_r|\,\mathbf{u} = \mathbf{f} \quad \text{in} \quad \Omega, \tag{1.4}$$

$$\mu_r^{-1}\,\mathbf{curl}\,\mathbf{u} \times \nu + \kappa\lambda\mathbf{u}_t = \mathbf{g} \quad \text{on} \quad \Sigma, \tag{1.5}$$

$$\mathbf{u} \times \nu = 0 \quad \text{on} \quad \Gamma. \tag{1.6}$$

Multigrid methods for the edge element discretization of Maxwell equations have been studied in [1], [21], [23], and the references therein. An important discovery in [23] was that for the $\mathbf{H}(\mathbf{curl})$-elliptic problems, multigrid relaxations must be performed on both edges and vertices in order to guarantee the uniform convergence of the multigrid method. We also refer to [5] for the multigrid preconditioned iterative scheme for the Helmholtz equation.

The distinct feature of applying multigrid methods on adaptively refined finite element meshes is that the number of degrees of freedom may not grow exponentially with respect to the number of mesh refinements k. Following the idea of "local" multigrid in [24] for discrete H^1-elliptic problems and [7] for discrete $\mathbf{H}(\mathbf{curl})$-elliptic problems, at each level k of our multigrid algorithm for discrete Maxwell problems, we perform Gauß–Seidel relaxations only on new edges, new vertices, and their immediate neighboring edges and vertices (see (4.3) and Algorithm 4.2 below). Our extensive numerical experiments indicate that our multigrid preconditioned MINRES algorithm has the very desirable property that for low-frequency time-harmonic Maxwell equations with very strong singularity, the number of iterations to reduce the initial residual by a factor of 10^{-8} remains nearly fixed on different levels of adaptively refined meshes. We also test the influence of the frequency to the performance of the preconditioner. It indicates that with the frequency of the problem increasing, the number of preconditioned MINRES iterations increases. But for a fixed frequency, it varies very slightly as the number of degrees of freedom increases. We also refer to [33] for the proof of uniform convergence of the "local" multigrid method for discrete H^1-elliptic problems on adaptively refined meshes.

The rest of the paper is organized as follows. In section 2, we prove Helmholtz-type decompositions of $\mathbf{H}(\mathbf{curl};\,\Omega)$ and introduce conforming finite element approximations to (1.1)–(1.3). In section 3, we derive reliable and efficient residual-based a posteriori error estimates. In section 4, we describe the preconditioned MINRES algorithm. In section 5, we report several numerical experiments to show the competitive performance of the methods proposed in this paper.

2. Hemlholtz-type decompositions and finite element approximations. We start by introducing the definition of a "screen."

DEFINITION 2.1. F [illegible] C^2 [illegible] ∂F [illegible] F [illegible] ∂F

Let Ω be a polyhedral domain in $\mathbb{R}^3$ which satisfies one of the following assumptions.

HYPOTHESIS 2.2.

(i) Ω [illegible] $\Gamma = \partial D$ [illegible] D [illegible] $\bar{D} \cup \Omega$ [illegible] 2.1

(ii) Γ [illegible] $\Omega \cup \Gamma$ [illegible] 1.1

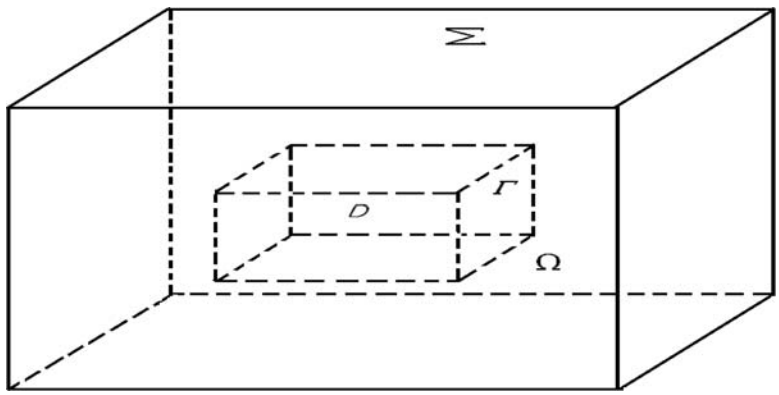


FIG. 2.1. *A domain with an inner scatter D.*

(iii) $\Sigma = \emptyset$ [illegible] $\Gamma = \partial\Omega = \Gamma_{in} \cup \Gamma_{out}$ [illegible] Γ_{in} [illegible] Γ

We remark that in each case Ω need not be simply connected, and the domain in the second case is not even Lipschitz. In the third case, we actually solve a time-harmonic problem with a Dirichlet boundary condition.

We introduce some notations and Sobolev spaces used in this paper. $L^2(\Omega)$ is the usual Hilbert space of square integrable functions equipped with the following inner product and norm:

$$(u,\, v) := \int_\Omega u(\mathbf{x})\, \overline{v(\mathbf{x})}\, d\mathbf{x} \quad \text{and} \quad \|u\|_{0,\Omega} := (u,\, u)^{1/2}.$$

$H^m(\Omega) := \{v \in L^2(\Omega) : D^\xi v \in L^2(\Omega),\ |\xi| \le m\}$ equipped with the norm and seminorm

$$\|u\|_{m,\Omega} := \left(\sum_{|\xi| \le m} \|D^\xi u\|_{0,\Omega}^2 \right)^{1/2} \quad \text{and} \quad |u|_{m,\Omega} := \left(\sum_{|\xi| = m} \|D^\xi u\|_{0,\Omega}^2 \right)^{1/2},$$

where ξ represents a nonnegative triple index. $H^1_\Gamma(\Omega)$ is the subspace of $H^1(\Omega)$ whose functions have zero traces on Γ. We use boldface notation for vectors, such as $\mathbf{L}^2(\Omega) := (L^2(\Omega))^3$ and so on. The following Sobolev spaces are used in the paper:

$$\begin{aligned}
\mathbf{H}(\mathrm{div};\Omega) &:= \{\mathbf{v} \in \mathbf{L}^2(\Omega) \,:\, \mathrm{div}\,\mathbf{v} \in \mathbf{L}^2(\Omega)\},\\
\mathbf{H}(\mathbf{curl};\Omega) &:= \{\mathbf{v} \in \mathbf{L}^2(\Omega) \,:\, \mathbf{curl}\,\mathbf{v} \in \mathbf{L}^2(\Omega)\},\\
\mathbf{H}_\Gamma(\mathbf{curl};\Omega) &:= \{\mathbf{v} \in \mathbf{H}(\mathbf{curl};\,\Omega) \mid \mathbf{v} \times \nu = 0 \ \text{ on } \ \Gamma \ \text{ and } \ \mathbf{v}_t \in \mathbf{L}^2(\Sigma)\}.
\end{aligned}$$

As usual, we denote $H^1_0(\Omega) := H^1_{\partial\Omega}(\Omega)$ and $\mathbf{H}_0(\mathbf{curl};\Omega) := \mathbf{H}_{\partial\Omega}(\mathbf{curl};\Omega)$ for $\Sigma = \emptyset$. $\mathbf{H}(\mathrm{div};\,\Omega)$, $\mathbf{H}(\mathbf{curl};\Omega)$, and $\mathbf{H}_\Gamma(\mathbf{curl};\Omega)$ are equipped, respectively, with the following norms:

$$\begin{aligned}
\|\mathbf{v}\|_{\mathbf{H}(\mathrm{div};\,\Omega)} &:= \left(\|\mathbf{v}\|_{0,\Omega}^2 + \|\mathrm{div}\,\mathbf{v}\|_{0,\Omega}^2\right)^{1/2},\\
\|\mathbf{v}\|_{\mathbf{H}(\mathbf{curl};\,\Omega)} &:= \left(\|\mathbf{v}\|_{0,\Omega}^2 + \|\mathbf{curl}\,\mathbf{v}\|_{0,\Omega}^2\right)^{1/2},\\
\|\mathbf{v}\|_{\mathbf{H}_\Gamma(\mathbf{curl};\,\Omega)} &:= \left(\|\mathbf{v}\|_{0,\Omega}^2 + \|\mathbf{curl}\,\mathbf{v}\|_{0,\Omega}^2 + \|\mathbf{v}_t\|_{0,\Sigma}^2\right)^{1/2}.
\end{aligned}$$

Let $\mathbf{f} \in \mathbf{L}^2(\Omega)$ and $\mathbf{g} \in \mathbf{L}^2(\Sigma)$ satisfying $\mathbf{g} \cdot \nu = 0$ on Σ. The equivalent weak formulation of (1.1)–(1.3) is to find $\mathbf{E} \in \mathbf{H}_\Gamma(\mathbf{curl}; \Omega)$ such that

$$a(\mathbf{E}, \mathbf{v}) = \int_\Omega \mathbf{f} \cdot \overline{\mathbf{v}} + \int_\Sigma \mathbf{g} \cdot \overline{\mathbf{v}}_t \quad \forall\, \mathbf{v} \in \mathbf{H}_\Gamma(\mathbf{curl}; \Omega), \tag{2.1}$$

where

$$a(\mathbf{E}, \mathbf{v}) := (\mu_r^{-1}\, \mathbf{curl}\, \mathbf{E}, \mathbf{curl}\, \mathbf{v}) - (\kappa^2 \varepsilon_r \mathbf{E}, \mathbf{v}) - \mathbf{i} \int_\Sigma \kappa\, \lambda\, \mathbf{E}_t \cdot \overline{\mathbf{v}}_t\,. \tag{2.2}$$

The existence and uniqueness of the solution of problem (2.1) under various conditions on the domain Ω and the coefficients ε_r, μ_r have been studied in [26]. Here for the sake of simplicity we simply assume that problem (2.1) has a unique solution. Thus there exists a constant $\beta > 0$ depending only on Ω, ε_r, μ_r, λ, and the wave number κ such that [4, Chapter 5]

$$\sup_{0 \neq \mathbf{v} \in \mathbf{H}_\Gamma(\mathbf{curl}; \Omega)} \frac{a(\mathbf{E}, \mathbf{v})}{\|\mathbf{v}\|_{\mathbf{H}_\Gamma(\mathbf{curl}; \Omega)}} \geq \beta \|\mathbf{E}\|_{\mathbf{H}_\Gamma(\mathbf{curl}; \Omega)}. \tag{2.3}$$

Furthermore, it follows from (2.3) that there exists a constant $C > 0$ independent of $\mathbf{E}$, $\mathbf{f}$, and $\mathbf{g}$ such that

$$\|\mathbf{E}\|_{\mathbf{H}_\Gamma(\mathbf{curl}; \Omega)} \leq C(\|\mathbf{f}\|_{0,\Omega} + \|\mathbf{g}\|_{0,\Sigma}). \tag{2.4}$$

The following Helmholtz-type decomposition theorem is applicable to polyhedral domains with Lipschitz screens F.

THEOREM 2.3. $\mathcal{D}$

- $\mathcal{D}$
- $\mathcal{D}$ F $\partial\mathcal{D} = \Gamma \cup F$ $\Gamma \cap F = \emptyset$ $\mathcal{D} \cup F$

1.1

$\mathbf{v} \in \mathbf{H}_0(\mathbf{curl}; \mathcal{D})$ $\psi \in H_0^1(\mathcal{D})$ $\mathbf{v}_s \in \mathbf{H}^1(\mathcal{D}) \cap \mathbf{H}_0(\mathbf{curl}; \mathcal{D})$ $\mathbf{v}_s \in \mathbf{H}_0^1(\mathcal{D})$ $\mathcal{D}$

$$\mathbf{v} = \nabla\psi + \mathbf{v}_s \quad \mathcal{D}, \tag{2.5}$$

$$\|\psi\|_{1,\mathcal{D}} + \|\mathbf{v}_s\|_{1,\mathcal{D}} \leq C\|\mathbf{v}\|_{\mathbf{H}(\mathbf{curl}; \mathcal{D})}, \tag{2.6}$$

C $\mathcal{D}$

. The proof for the case when D is a Lipschitz domain is contained in [19, Proposition 5.1]. For completeness we sketch the proof here. The proof below for the case when D has a Lipschitz inner screen F simplifies the argument in [9].

Let $\mathcal{O} := B(0, R)$ be a ball containing D. We extend $\mathbf{v}$ by zero to the exterior of $\mathcal{D}$ and denote the extension by $\tilde{\mathbf{v}}$. Clearly $\tilde{\mathbf{v}} \in \mathbf{H}_0(\mathbf{curl}; \mathcal{O})$. By Theorem 3.4 of [20, p. 45], there exists a $\mathbf{w} \in \mathbf{H}^1(\mathcal{O})$ such that

$$\mathbf{curl}\, \mathbf{w} = \mathbf{curl}\, \tilde{\mathbf{v}}, \quad \mathrm{div}\, \mathbf{w} = 0 \quad \text{in } \mathcal{O}, \tag{2.7}$$

$$\|\mathbf{w}\|_{1,\mathcal{O}} \leq C(\|\mathbf{curl}\, \mathbf{w}\|_{0,\mathcal{O}} + \|\mathrm{div}\, \mathbf{w}\|_{0,\mathcal{O}}) = C\|\mathbf{curl}\, \mathbf{v}\|_{0,\mathcal{D}}. \tag{2.8}$$

Moreover, by (2.7) and Theorem 2.9 of [20, p. 31], there exists a $\varphi \in H^1(\mathcal{O})/\mathbb{R}$ such that

$$\tilde{\mathbf{v}} = \mathbf{w} + \nabla\varphi \quad \text{in } \mathcal{O}, \tag{2.9}$$

$$\|\varphi\|_{1,\mathcal{O}} \leq C|\varphi|_{1,\mathcal{O}} \leq C\|\mathbf{v}\|_{\mathbf{H}(\mathbf{curl}; \mathcal{D})}, \tag{2.10}$$

$$|\varphi|_{2,\mathcal{O}\setminus\bar{\mathcal{D}}} \leq \|\mathbf{w}\|_{1,\mathcal{O}} \leq C\|\mathbf{curl}\, \mathbf{v}\|_{0,\mathcal{D}}. \tag{2.11}$$

Since $\mathcal{O}\setminus\bar{\mathcal{D}}$ is a Lipschitz domain, by the extension theorem [22, Theorem 1.4.3.1, p. 25], there exists an extension of $\varphi|_{\mathcal{O}\setminus\bar{\mathcal{D}}}$ denoted by $\tilde{\varphi}\in H^2(\mathbb{R}^3)$ such that

$$\tilde{\varphi}=\varphi \quad \text{in } \mathcal{O}\setminus\bar{\mathcal{D}} \quad \text{and} \quad \|\tilde{\varphi}\|_{2,\mathbb{R}^3}\le C\|\varphi\|_{2,\mathcal{O}\setminus\bar{\mathcal{D}}}\le C\|\mathbf{v}\|_{\mathbf{H}(\mathbf{curl};\,\mathcal{D})}. \tag{2.12}$$

This completes the proof for the case when D is a Lipschitz domain by setting $\psi=\varphi-\tilde{\varphi}\in H_0^1(D)$ and $\mathbf{v}_s=\mathbf{w}+\nabla\tilde{\varphi}\in\mathbf{H}_0^1(\mathcal{D})$, since $\mathbf{v}_s=\tilde{\mathbf{v}}=\mathbf{0}$ in $\mathcal{O}\setminus\bar{\mathcal{D}}$.

If $\mathcal{D}$ has a Lipschitz screen $\mathcal{F}\subset D$ such that $\partial\mathcal{D}=\Gamma\cup\mathcal{F}$ and $\Gamma\cap\mathcal{F}=\emptyset$, we choose a closed C^2-smooth surface $\mathcal{F}_0\supset\mathcal{F}$ such that $\mathcal{F}_0\cap\partial\bar{\mathcal{D}}=\emptyset$. In view of (2.9), we know that $\nabla_{\mathcal{F}}\varphi=(\nu\times\nabla\varphi)\times\nu=\nu\times(\nu\times\mathbf{w})\in\mathbf{H}^{1/2}(\mathcal{F})$, and thus $\varphi|_{\mathcal{F}}\in H^{3/2}(\mathcal{F})$, where [22, p. 20]

$$H^{m+1/2}(\mathcal{F})=\{v\in L^2(\mathcal{F})\,:\,\|v\|_{m+1/2,\mathcal{F}}<\infty\},\quad m=0,1,$$
$$\|v\|^2_{m+1/2,\mathcal{F}}=\|v\|^2_{m,\mathcal{F}}+\int_{\mathcal{F}}\int_{\mathcal{F}}\frac{|\nabla_{\mathcal{F}}v(x)-\nabla_{\mathcal{F}}v(y)|^2}{|x-y|^3}ds_xds_y,\quad m=0,1.$$

Here $H^{3/2}(\mathcal{F}_0)$ is defined similarly. Since any function in $H^{3/2}(D_1)$ can be continuously extended to be a function in $H^{3/2}(\mathbb{R}^3)$ for any Lipschitz bounded domain $D_1\subset\mathbb{R}^3$ [22, Theorem 1.4.3.1, p. 25], we know that $\varphi|_{\mathcal{F}}$ can be extended continuously to be a function in $H^{3/2}(\mathcal{F}_0)$ by working on the atlas of the C^2-smooth manifold $\mathcal{F}_0$. We denote the extension by $\varphi_{\mathcal{F}}\in H^{3/2}(\mathcal{F}_0)$ which satisfies, by (2.8) and (2.11),

$$\varphi_{\mathcal{F}}=\varphi \quad \text{on } \mathcal{F}, \tag{2.13}$$
$$\|\varphi_{\mathcal{F}}\|_{3/2,\mathcal{F}_0}\le C\|\varphi\|_{3/2,\mathcal{F}}\le C(\|\varphi\|_{1,\mathcal{O}}+\|\mathbf{w}\|_{1,\mathcal{O}})\le C\|\mathbf{v}\|_{\mathbf{H}(\mathbf{curl};\,\mathcal{D})}. \tag{2.14}$$

Since $\tilde{\varphi}\in H^2(\mathbb{R}^3)$, we have $\tilde{\varphi}\in H^{3/2}(\mathcal{F}_0)$. Thus $\varphi_{\mathcal{F}}-\tilde{\varphi}\in H^{3/2}(\mathcal{F}_0)$, which we may extend to be a function $\varphi_0\in H_0^2(D)$ satisfying

$$\varphi_0=\varphi_{\mathcal{F}}-\tilde{\varphi} \quad \text{on } \mathcal{F} \quad \text{and} \quad \|\varphi_0\|_{2,D}\le C\|\varphi_{\mathcal{F}}-\tilde{\varphi}\|_{3/2,\mathcal{F}_0}\le C\|\mathbf{v}\|_{\mathbf{H}(\mathbf{curl};\,\mathcal{D})}, \tag{2.15}$$

where we have used (2.12)–(2.14). Define $\psi:=\varphi-\tilde{\varphi}-\varphi_0\in H_0^1(\mathcal{D})$ and $\mathbf{v}_s:=\mathbf{w}+\nabla\tilde{\varphi}+\nabla\varphi_0\in\mathbf{H}^1(\mathcal{D})\cap\mathbf{H}_0(\mathbf{curl};\,\mathcal{D})$. Combining (2.8), (2.12), and (2.15) leads to (2.5) and (2.6). ∎

THEOREM 2.4. Ω … $\partial\Omega=\Sigma\cup\Gamma$ … (i)–(iii) … 2.2 … $\mathbf{v}\in\mathbf{H}(\mathbf{curl};\,\Omega)$ … $\mathbf{v}\times\nu=0$ … Γ … $\mathbf{v}_s\in\mathbf{H}^1(\Omega)$ … $\mathbf{v}_s\times\nu=0$ … Γ … $\varphi\in H^1_\Gamma(\Omega)$ …

$$\mathbf{v}=\nabla\varphi+\mathbf{v}_s \quad \text{in } \Omega, \tag{2.16}$$
$$\|\mathbf{v}_s\|_{1,\Omega}+\|\varphi\|_{1,\Omega}\le C\,\|\mathbf{v}\|_{\mathbf{H}(\mathbf{curl};\,\Omega)}. \tag{2.17}$$

… For case (iii) in Hypothesis 2.2, Theorem 2.4 follows directly from Theorem 2.3. For case (i), $\Gamma=\partial D$ with D being a bounded Lipschitz domain and $\bar{D}$ embedded in the interior of $\Omega\cup\bar{D}$. Let $\mathcal{O}$ be a ball containing Ω. Extend $\mathbf{v}$ by zero to D such that the extension $\tilde{\mathbf{v}}\in\mathbf{H}(\mathbf{curl};\,\Omega\cup\bar{D})$. By Lemma 2.2 of [13], we may extend $\tilde{\mathbf{v}}$ to the exterior of $\Omega\cup\bar{D}$ such that the extension $E\,\tilde{\mathbf{v}}\in\mathbf{H}_0(\mathbf{curl};\,\mathcal{O})$ and

$$E\,\tilde{\mathbf{v}}=\tilde{\mathbf{v}} \quad \text{in } \Omega\cup\bar{D},$$
$$\|E\,\tilde{\mathbf{v}}\|_{\mathbf{H}(\mathbf{curl};\,\mathcal{O})}\le C\,\|\mathbf{v}\|_{\mathbf{H}(\mathbf{curl};\,\Omega)},$$

where C depends only on Ω. Since $E\,\tilde{\mathbf{v}}\in\mathbf{H}_0(\mathbf{curl};\,\mathcal{O}\setminus\bar{D})$, by Theorem 2.3, there exist $\varphi\in H_0^1(\mathcal{O}\setminus\bar{D})$ and $\mathbf{v}_s\in\mathbf{H}^1(\mathcal{O}\setminus\bar{D})\cap\mathbf{H}_0(\mathbf{curl};\mathcal{O}\setminus\bar{D})$ such that

$$E\,\tilde{\mathbf{v}}=\nabla\varphi+\mathbf{v}_s \quad \text{in } \mathcal{O}\setminus\bar{D},$$
$$\|\varphi\|_{1,\mathcal{O}\setminus\bar{D}}+\|\mathbf{v}_s\|_{1,\mathcal{O}\setminus\bar{D}}\le C\|E\,\tilde{\mathbf{v}}\|_{\mathbf{H}(\mathbf{curl};\,\mathcal{O}\setminus\bar{D})}\le C\,\|\mathbf{v}\|_{\mathbf{H}(\mathbf{curl};\,\Omega)},$$

where the constant C depends only on Ω. Clearly, φ and $\mathbf{v}_s$ match all requirements in Theorem 2.4. Case (ii) can be proved similarly using Theorem 2.3. □

REMARK 2.5. The decompositions in Theorems 2.3 and 2.4 extend the so-called Birman–Solomyak decomposition of $H_0(\mathbf{curl};\Omega)$ in [8], [19], and [29].

Let $\mathcal{T}_k$ be a sequence of tetrahedral triangulations of Ω, and let $\mathcal{F}_k$ be the set of faces not lying on Γ, $k \geq 0$. The finite element space $\mathbf{U}_k$ over $\mathcal{T}_k$ is defined by [28]

$$\begin{aligned}\mathbf{U}_k :=\{&\mathbf{u} \in \mathbf{H}(\mathbf{curl};\Omega) \ : \ \mathbf{u}\times\nu|_\Gamma = \mathbf{0} \quad \text{and} \\ &\mathbf{u}|_T = \mathbf{a}_T + \mathbf{b}_T \times \mathbf{x} \quad \text{with} \quad \mathbf{a}_T,\, \mathbf{b}_T \in \mathbb{R}^3 \quad \forall\, T \in \mathcal{T}_k\big\}.\end{aligned}$$

Degrees of freedom on every $T \in \mathcal{T}_k$ are $\int_{E_i} \mathbf{u}\cdot d\,\mathbf{l}$, $i = 1,\ldots,6$, where $E_1,\ldots,E_6$ are six edges of T. For any $T \in \mathcal{T}_k$ and $F \in \mathcal{F}_k$, we denote the diameters of T and F by h_T and h_F, respectively.

The finite element approximation to (2.1) is to find $\mathbf{E}_k \in \mathbf{U}_k$ such that

$$a(\mathbf{E}_k,\,\mathbf{v}) = \int_\Omega \mathbf{f}\cdot\overline{\mathbf{v}} + \int_\Sigma \mathbf{g}\cdot\overline{\mathbf{v}}_t \quad \forall\,\mathbf{v} \in \mathbf{U}_k. \tag{2.18}$$

3. Residual-based a posteriori error estimates. Let $\mathbf{E}$ and $\mathbf{E}_k$ be the solutions of (2.1) and (2.18), respectively. Define the total error function by $\mathbf{e}_k := \mathbf{E} - \mathbf{E}_k$. By (2.3), we know that

$$\|\mathbf{e}_k\|_{\mathbf{H}_\Gamma(\mathbf{curl};\Omega)} \leq \beta^{-1} \sup_{\mathbf{v}\in\mathbf{H}_\Gamma(\mathbf{curl};\Omega)} \frac{a(\mathbf{e}_k,\,\mathbf{v})}{\|\mathbf{v}\|_{\mathbf{H}_\Gamma(\mathbf{curl};\Omega)}}. \tag{3.1}$$

To derive a posteriori error estimates, we introduce the Scott–Zhang operator $\mathcal{I}_k : H^1_\Gamma(\Omega) \to V_k$ [32] and the Beck–Hiptmair–Hoppe–Wohlmuth operator $\Pi_k : \mathbf{H}^1(\Omega) \cap \mathbf{H}_\Gamma(\mathbf{curl};\Omega) \to \mathbf{U}_k$ [6], where V_k is the piecewise linear H^1_Γ-conforming finite element space over $\mathcal{T}_k$ defined by

$$V_k := \big\{v \in H^1_\Gamma(\Omega) \ : \ v|_T = a_T + \mathbf{b}_T\cdot\mathbf{x} \text{ with } a_T \in \mathbb{R}^1 \text{ and } \mathbf{b}_T \in \mathbb{R}^3 \ \forall\, T \in \mathcal{T}_k\big\}. \tag{3.2}$$

$\mathcal{I}_k$ and Π_k satisfy the following approximation and stability properties: For any $T \in \mathcal{T}_k$, $F \in \mathcal{F}_k$,

$$\begin{cases} \mathcal{I}_k\phi_h = \phi_h & \forall\,\phi_h \in V_k, \\ \|\nabla\mathcal{I}_k\phi\|_{0,T} \ \leq \ C\,|\phi|_{1,D_T}, \\ \|\phi - \mathcal{I}_k\phi\|_{0,T} \ \leq \ C\,h_T\,|\phi|_{1,D_T}, \\ \|\phi - \mathcal{I}_k\phi\|_{0,F} \ \leq \ C\,h_F^{1/2}\,|\phi|_{1,D_F} \end{cases} \tag{3.3}$$

and

$$\begin{cases} \Pi_k\mathbf{w}_h = \mathbf{w}_h & \forall\,\mathbf{w}_h \in \mathbf{U}_k, \\ \|\Pi_k\mathbf{w}\|_{\mathbf{H}(\mathbf{curl};T)} \ \leq \ C\,\|\mathbf{w}\|_{1,D_T}, \\ \|\mathbf{w} - \Pi_k\mathbf{w}\|_{0,T} \ \leq \ C\,h_T\,|\mathbf{w}|_{1,D_T}, \\ \|\mathbf{w} - \Pi_k\mathbf{w}\|_{0,F} \ \leq \ C\,h_F^{1/2}\,|\mathbf{w}|_{1,D_F}, \end{cases} \tag{3.4}$$

where D_A is the union of elements in $\mathcal{T}_k$ with nonempty intersection with A, $A = T$ or F. In (3.3) and (3.4), the constant C depends on the ratio of the diameter of T to the diameter of the maximal ball contained in T. In the procedure of mesh refinements used in ALBERT [30], [31], each element is similar to one of a small

number of reference tetrahedra in shape. Thus C in (3.3)–(3.4) is bounded in the refinement procedure.

By Theorem 2.4, for any $\mathbf{v} \in \mathbf{H}_\Gamma(\mathbf{curl};\,\Omega)$ there exist a $\varphi \in H^1_\Gamma(\Omega)$ and a $\mathbf{v}_s \in \mathbf{H}^1(\Omega) \cap \mathbf{H}_\Gamma(\mathbf{curl};\,\Omega)$ such that

$$\mathbf{v} = \nabla\varphi + \mathbf{v}_s, \tag{3.5}$$

$$\|\varphi\|_{1,\Omega} + \|\mathbf{v}_s\|_{1,\Omega} \le C\,\|\mathbf{v}\|_{\mathbf{H}(\mathbf{curl};\,\Omega)}, \tag{3.6}$$

where the constant C depends only on Ω. Since $\nabla\mathcal{I}_k\varphi$ and $\Pi_k\mathbf{v}_s$ belong to $\mathbf{U}_k$, by the Galerkin orthogonality, we have

$$a(\mathbf{e}_k,\,\mathbf{v}) = a(\mathbf{e}_k,\,\nabla\varphi - \nabla\mathcal{I}_k\varphi) + a(\mathbf{e}_k,\,\mathbf{v}_s - \Pi_k\mathbf{v}_s) \quad \forall\,\mathbf{v} \in \mathbf{H}_\Gamma(\mathbf{curl};\,\Omega). \tag{3.7}$$

For any face $F \in \mathcal{F}_k$, assuming that $F = T_1 \cap T_2,\ T_1,\ T_2 \in \mathcal{T}_k$, and the unit normal ν points from T_2 to T_1, we denote the jump of a function v across F by $[v]_F := v|_{T_1} - v|_{T_2}$.

LEMMA 3.1. [illegible] $\mathbf{g} \in \mathbf{L}^2(\Sigma)$ [illegible] $\operatorname{div}_\Sigma \mathbf{g} \in L^2(\Sigma)$ [illegible] $\mathbf{g}\cdot\nu = 0$ [illegible] Σ [illegible] [illegible] C_0 [illegible] [illegible] κ [illegible] [illegible] $\mathcal{T}_k$ [illegible]

$$a(\mathbf{e}_k,\,\nabla\varphi - \nabla\mathcal{I}_k\varphi) \le C_0 \left(\sum_{T\in\mathcal{T}_k} \eta_{0,T}^2 + \sum_{F\in\mathcal{F}_k} \eta_{0,F}^2 + \sum_{F\subset\Sigma} \eta_{0,\Sigma,F}^2 \right)^{1/2} \|\mathbf{v}\|_{\mathbf{H}(\mathbf{curl};\,\Omega)}, \tag{3.8}$$

[illegible] $\operatorname{div}_\Sigma$ [illegible] Σ [illegible] [illegible] L^2 [illegible] [illegible] $\nabla_\Sigma := -\nu\times(\nu\times\nabla)$ [illegible]

$$\eta_{0,T} := h_T\,\|\operatorname{div}(\kappa^2\varepsilon_r\,\mathbf{E}_k)\|_{0,T},$$

$$\eta_{0,F} := h_F^{1/2}\,\|[\kappa^2\varepsilon_r\,\mathbf{E}_k\cdot\nu]_F\|_{0,F},$$

$$\eta_{0,\Sigma,F} := h_F^{1/2}\|\operatorname{div}_\Sigma(\mathbf{g} + \mathbf{i}\,\kappa\,\lambda\,\mathbf{E}_{k,t})\|_{0,F}.$$

[illegible] Since the tangential field $\mathbf{E}_{k,t}$ is continuous on Σ and piecewise linear, $\mathbf{E}_{k,t} \in \mathbf{H}(\operatorname{div}_\Sigma;\,\Sigma)$. In view of $\mathbf{f}$ being divergence-free, by (2.1) and the formula of integration by parts, we deduce that

$$\begin{aligned}
&a(\mathbf{e}_k,\,\nabla\,\varphi - \nabla\,\mathcal{I}_k\varphi)\\
&\quad= (\mathbf{f} + \kappa^2\varepsilon_r\,\mathbf{E}_k,\,\nabla\,\varphi - \nabla\,\mathcal{I}_k\varphi) + \int_\Sigma (\mathbf{g} + \mathbf{i}\,\kappa\,\lambda\,\mathbf{E}_{k,t})\cdot\nabla_\Sigma\overline{(\varphi - \mathcal{I}_k\varphi)}\\
&\quad= -\sum_{T\in\mathcal{T}_k}\int_T \operatorname{div}(\kappa^2\varepsilon_r\,\mathbf{E}_k)\overline{(\varphi - \mathcal{I}_k\varphi)} + \sum_{F\in\mathcal{F}_k}\int_F [\kappa^2\varepsilon_r\,\mathbf{E}_k\cdot\nu]_F\,\overline{(\varphi - \mathcal{I}_k\varphi)}\\
&\qquad\qquad - \sum_{F\subset\Sigma}\int_F \operatorname{div}_\Sigma(\mathbf{g} + \mathbf{i}\,\kappa\,\lambda\,\mathbf{E}_{k,t})\,\overline{(\varphi - \mathcal{I}_k\varphi)}\\
&\quad\le \sum_{T\in\mathcal{T}_k} \|\operatorname{div}(\kappa^2\varepsilon_r\,\mathbf{E}_k)\|_{0,T}\|\varphi - \mathcal{I}_k\varphi\|_{0,T} + \sum_{F\in\mathcal{F}_k} \|[\kappa^2\varepsilon_r\,\mathbf{E}_k\cdot\nu]_F\|_{0,F}\|\varphi - \mathcal{I}_k\varphi\|_{0,F}\\
&\qquad\qquad + \sum_{F\subset\Sigma} \|\operatorname{div}_\Sigma(\mathbf{g} + \mathbf{i}\,\kappa\,\lambda\,\mathbf{E}_{k,t})\|_{0,F}\|\varphi - \mathcal{I}_k\varphi\|_{0,F}.
\end{aligned}$$

We reach (3.8) by virtue of Schwartz's inequality, (3.3), and (3.6). □

LEMMA 3.2. [illegible] [illegible] C_1 [illegible] [illegible] κ [illegible] [illegible] $\mathcal{T}_k$ [illegible]

$$a(\mathbf{e}_k,\,\mathbf{v}_s - \Pi_k\mathbf{v}_s) \le C_1 \left(\sum_{T\in\mathcal{T}_k} \eta_{1,T}^2 + \sum_{F\in\mathcal{F}_k} \eta_{1,F}^2 + \sum_{F\subset\Sigma} \eta_{1,\Sigma,F}^2 \right)^{1/2} \|\mathbf{v}\|_{\mathbf{H}(\mathbf{curl};\,\Omega)}, \tag{3.9}$$

$$\eta_{1,T} = h_T\|\mathbf{f} + \kappa^2\varepsilon_r\,\mathbf{E}_k - \mathbf{curl}(\mu_r^{-1}\mathbf{curl}\,\mathbf{E}_k)\|_{0,T},$$
$$\eta_{1,F} = h_F^{1/2}\|[\mu_r^{-1}\,\mathbf{curl}\,\mathbf{E}_k \times \nu]_F\|_{0,F},$$
$$\eta_{1,\Sigma,F} = h_F^{1/2}\|\mathbf{g} + \mathbf{i}\,\kappa\,\lambda\,\mathbf{E}_{k,t} + \nu \times \mu_r^{-1}\,\mathbf{curl}\,\mathbf{E}_k\|_{0,F}.$$

. By (2.1) and the formula of integration by parts, we deduce that

$$\begin{aligned}
&a(\mathbf{e}_k,\, \mathbf{v}_s - \Pi_k\mathbf{v}_s)\\
&\quad= \sum_{T\in\mathcal{T}_k}\int_T (\mathbf{f} + \kappa^2\varepsilon_r\,\mathbf{E}_k - \mathbf{curl}\mu_r^{-1}\mathbf{curl}\,\mathbf{E}_k)\cdot\overline{(\mathbf{v}_s - \Pi_k\mathbf{v}_s)}\\
&\qquad+ \sum_{F\in\mathcal{F}_k}\int_F [\nu\times\mu_r^{-1}\,\mathbf{curl}\,\mathbf{E}_k]_F\cdot\overline{(\mathbf{v}_s - \Pi_k\mathbf{v}_s)}\\
&\qquad+ \sum_{F\subset\Sigma}\int_F (\mathbf{g} + \mathbf{i}\,\kappa\,\lambda\,\mathbf{E}_{k,t} + \nu\times\mu_r^{-1}\,\mathbf{curl}\,\mathbf{E}_k)\cdot\overline{(\mathbf{v}_s - \Pi_k\mathbf{v}_s)}\\
&\quad\le \sum_{T\in\mathcal{T}_k}\|\mathbf{f} + \kappa^2\varepsilon_r\,\mathbf{E}_k - \mathbf{curl}\mu_r^{-1}\mathbf{curl}\,\mathbf{E}_k\|_{0,T}\|\mathbf{v}_s - \Pi_k\mathbf{v}_s\|_{0,T}\\
&\qquad+ \sum_{F\in\mathcal{F}_k}\|[\mu_r^{-1}\,\mathbf{curl}\,\mathbf{E}_k\times\nu]_F\|_{0,F}\|\mathbf{v}_s - \Pi_k\mathbf{v}_s\|_{0,F}\\
&\qquad+ \sum_{F\subset\Sigma}\int_F \|\mathbf{g} + \mathbf{i}\,\kappa\,\lambda\,\mathbf{E}_{k,t} + \nu\times\mu_r^{-1}\,\mathbf{curl}\,\mathbf{E}_k\|_{0,F}\,\|\mathbf{v}_s - \Pi_k\mathbf{v}_s\|_{0,F}.
\end{aligned}$$

We reach (3.9) by Schwartz's inequality, (3.4), and (3.6). □

Combining (3.1) and (3.7)–(3.9) leads to the following theorem.

THEOREM 3.3. . $\mathbf{g}\in\mathbf{L}^2(\Sigma)$, $\mathrm{div}_\Sigma\,\mathbf{g}\in L^2(\Sigma)$, $\mathbf{g}\cdot\nu = 0$ Σ .

$$\|\mathbf{e}_k\|^2_{\mathbf{H}_\Gamma(\mathbf{curl};\,\Omega)} \le \frac{2(C_0^2 + C_1^2)}{\beta^2}\sum_{i=0}^{1}\left\{\sum_{T\in\mathcal{T}_k}\eta_{i,T}^2 + \sum_{F\in\mathcal{F}_k}\eta_{i,F}^2 + \sum_{F\subset\Sigma}\eta_{i,\Sigma,F}^2\right\},$$

. β C_0 , C_1 . . , (2.3) (3.8) , (3.9)

Similar to the argument in [6, section 5], we can prove the following lower bound estimates.

THEOREM 3.4. . , μ_r ε_r , λ . , , . C , , μ_r ε_r λ , κ . , , . $\mathcal{T}_k$.

$$\begin{aligned}
(3.10)\qquad &\sum_{T\in\mathcal{T}_k}\left(\eta_{0,T}^2 + \eta_{1,T}^2\right) + \sum_{F\in\mathcal{F}_k}\left(\eta_{0,F}^2 + \eta_{1,F}^2\right) + \sum_{F\subset\Sigma}\left(\eta_{0,\Sigma,F}^2 + \eta_{1,\Sigma,F}^2\right)\\
&\le C\,\|\mathbf{e}_k\|^2_{\mathbf{H}(\mathbf{curl};\,\Omega)} + C\sum_{T\in\mathcal{T}_k}h_T^2\|\mathbf{f} - \mathcal{P}_T\mathbf{f}\|_{0,T}^2 + C\sum_{F\subset\Sigma}h_F\|\mathrm{div}_\Sigma(\mathbf{g} - \mathcal{P}^F_{\mathrm{div}}\,\mathbf{g})\|_{0,F}^2\\
&\qquad+ C\sum_{F\subset\Sigma}h_F\|\mathbf{g} - \mathcal{P}_F\mathbf{g}\|_{0,F}^2,
\end{aligned}$$

. $\mathcal{P}_T$ $\mathbf{L}^2(T)\to\mathbf{P}_2(T)$, $\mathcal{P}_F$ $\mathbf{L}^2(F)\to\mathbf{P}_2(F)$, $\mathbf{L}^2$, , , , $\mathcal{P}^F_{\mathrm{div}}$ $\mathbf{H}(\mathrm{div}_F;\,F)\to\mathbf{P}_1(F)$, , $\mathbf{H}(\mathrm{div}_F)$, , , $\mathbf{P}_k(D)$, , , ,

4. Multigrid preconditioned MINRES algorithm. In this section we introduce our preconditioned MINRES (PMINRES) algorithm for solving (2.18). Our preconditioner is a multigrid solver of the stiffness matrix corresponding to the following coercive and Hermitian sesquilinear form on $\mathbf{U}_k$:

$$a_{mg}(\mathbf{u},\,\mathbf{v}) := (\mu_r^{-1}\,\mathbf{curl}\,\mathbf{u},\,\mathbf{curl}\,\mathbf{v}) + (\kappa^2\,|\varepsilon_r|\,\mathbf{u},\,\mathbf{v}) + \int_\Sigma \kappa\,\lambda\,\mathbf{u}_t\cdot\bar{\mathbf{v}}_t\,. \tag{4.1}$$

It is easy to see that $a_{mg}(\cdot,\,\cdot)$ is the sesquilinear form corresponding to the variational formulation of (1.4)–(1.6).

Let $\mathcal{T}_0,\cdots,\mathcal{T}_J$ be a sequence of nested triangulations by repeated adaptive refinements. Let $\mathbf{U}_k$ be the finite element space in (2.18) and V_k be the linear Lagrangian finite element space in (3.2) over $\mathcal{T}_k$. Denote the canonical basis of $\mathbf{U}_k$ by $\left\{\mathbf{w}_1^{(k)},\cdots,\mathbf{w}_{n_k}^{(k)}\right\}$ and the canonical basis of V_k by $\left\{\phi_1^{(k)},\cdots,\phi_{\bar{n}_k}^{(k)}\right\}$, where n_k and $\bar{n}_k$ are, respectively, the numbers of edges and vertices not on Γ.

We now specify the local multigrid algorithm for the solution of the following algebraic system of equations:

$$A_k\,X_k = F_k, \tag{4.2}$$

where $A_k = \left(a_{mg}\big(\mathbf{w}_i^{(k)},\,\mathbf{w}_j^{(k)}\big)\right)_{n_k\times n_k}$. The algorithm is based on the following space decomposition proposed in [7]:

$$\mathbf{U}_J = \mathbf{U}_0 + \sum_{k=1}^{J}\sum_{E\in\mathcal{E}_k^{\rm new}} \mathrm{span}\,\{\mathbf{w}_E\} + \sum_{k=1}^{J}\sum_{a\in\mathcal{V}_k^{\rm new}} \mathrm{span}\,\{\nabla\phi_a\}\,, \tag{4.3}$$

where $\mathcal{E}_k^{\rm new}$ and $\mathcal{V}_k^{\rm new}$ are, respectively, the set of edges and the set of vertices in new elements of $\mathcal{T}_k$. None of the edges and vertices in (4.3) is on Γ. The algorithm of multigrid V-cycle is defined recursively as follows.

ALGORITHM 4.1. Multigrid V-cycle:

MGsolver(A_J, F_J, m)
{
  Given initial guess $\mathcal{X}_0$
  For $l = 0\ :\ m-1$
    $\mathcal{X}_{l+1} \leftarrow$ MG(J, A_J, F_J, $\mathcal{X}_l$)

  return $\mathcal{X}_m$
}
MG(j, A_j, f, x), $j = 0,\cdots,J$ recursively defined by
{
  if ($j = 0$)
    return $A_0^{-1}f$
  else
  {
    $x \leftarrow$ GSsmooth(A_j, f, x) [Pre-smoothing]
    $y \leftarrow$ MG($j-1$, A_{j-1}, $\mathcal{R}_j^{j-1}(f - A_j\,x)$, 0)
    $x \leftarrow x + \mathcal{I}_{j-1}^{j}\,y$
    $x \leftarrow$ GSsmooth(A_j, f, x) [Post-smoothing]
  }
}

In Algorithm 4.1, $\mathcal{I}_{j-1}^{j}$ is an $n_j \times n_{j-1}$ matrix reflecting the identical embedding $\mathbf{U}_{j-1} \hookrightarrow \mathbf{U}_j$ such that

$$\sum_{i=1}^{n_{j-1}} y_i \mathbf{w}_i^{(j-1)} = \sum_{i=1}^{n_j} \left(\mathcal{I}_{j-1}^{j}\, y\right)_i \mathbf{w}_i^{(j)} .$$

We choose the restriction matrix $\mathcal{R}_j^{j-1}$ to be the transpose of $\mathcal{I}_{j-1}^{j}$, that is, $\mathcal{R}_j^{j-1} = \left(\mathcal{I}_{j-1}^{j}\right)^T$. Define $A_{j,\nabla} = \left(a_{mg}(\nabla\phi_m^{(j)}, \nabla\phi_n^{(j)})\right)_{\bar{n}_j \times \bar{n}_j}$ and $\mathcal{I}_V^E$ to be an $n_j \times \bar{n}_j$ matrix reflecting the identical embedding $\nabla V_j \hookrightarrow \mathbf{U}_j$ such that

$$\sum_{i=1}^{\bar{n}_j} y_i \nabla\phi_i^{(j)} = \sum_{i=1}^{n_j} \left(\mathcal{I}_V^E\, y\right)_i \mathbf{w}_i^{(j)} .$$

GSsmooth(A_j, f, x) is the Gauß–Seidel iterations on level j with initial guess x.

ALGORITHM 4.2. Gauß–Seidel sweeping:

GSsmooth(A_j, f, x)
{
  Gauß–Seidel sweep on DOFs of $A_j\, x = f$ related to $\mathcal{E}_j^{\text{new}}$
  $y \leftarrow f - A_j\, x$
  $y_\nabla \leftarrow (\mathcal{I}_V^E)^T\, y$
  $x_\nabla \leftarrow 0$
  Gauß–Seidel sweep on DOFs of $A_{j,\nabla}\, x_\nabla = y_\nabla$ related to $\mathcal{V}_j^{\text{new}}$
  return $x + \mathcal{I}_V^E\, x_\nabla$
}

5. Adaptive algorithm and numerical results. The implementation of our adaptive algorithm is based on the adaptive finite element package ALBERT [30], [31] and is carried out on Origin 3800. We define the local a posteriori error estimator over an element $T \in \mathcal{T}_h$ by

$$\eta_T := \left\{ \eta_{0,T}^2 + \eta_{1,T}^2 + \frac{1}{2} \sum_{F \subset \partial T} (\eta_{0,F}^2 + \eta_{1,F}^2 + \eta_{0,\Sigma,F}^2 + \eta_{1,\Sigma,F}^2) \right\}^{1/2}$$

and define the global a posteriori error estimate and the maximal element error estimate over $\mathcal{T}_h$, respectively, by

$$\eta_h := \left(\sum_{T \in \mathcal{T}_h} \eta_T^2 \right)^{1/2}, \qquad \eta_{\max} = \max_{T \in \mathcal{T}_h} \eta_T. \tag{5.1}$$

Now we describe the adaptive algorithm used in this paper.

ALGORITHM 5.1. Given a tolerance $Tol > 0$ and the initial mesh $\mathcal{T}_0$, set $\mathcal{T}_h = \mathcal{T}_0$.

(I) Solve the discrete problem (2.18) on $\mathcal{T}_0$.

(II) Compute the local error estimator η_T on each $T \in \mathcal{T}_0$, the global error estimate η_h, and the maximal element error estimate $\eta_{\max}$.

(III) While $\eta_h > Tol$ do

- Refine the mesh $\mathcal{T}_h$ according to the following strategy:

  if $\eta_T > \frac{3}{5}\eta_{\max}$, refine the element $T \in \mathcal{T}_h$.

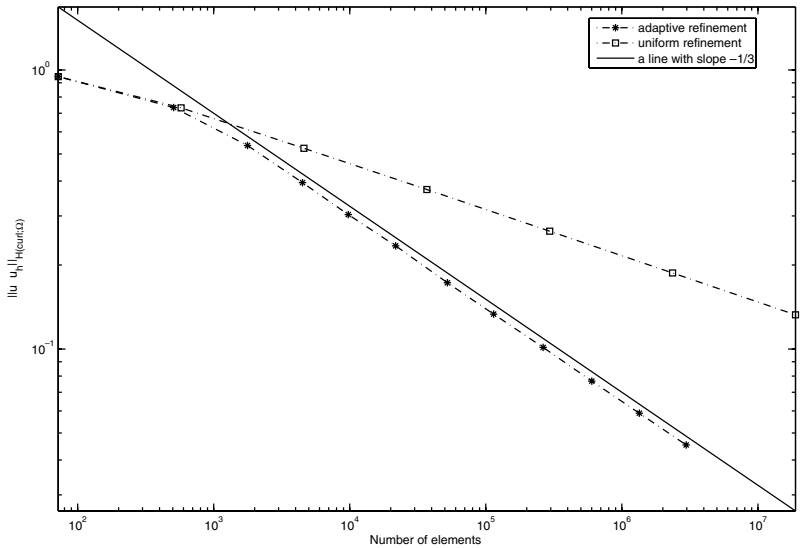


FIG. 5.1. *Quasi optimality of the adaptive mesh refinements of the error* $\|\mathbf{u}-\mathbf{u}_h\|_{\mathbf{H(curl;}\,\Omega)}$ (*Example* 5.1).

- Solve the discrete problem (2.18) on $\mathcal{T}_h$.
- Compute the local error estimator η_T on each $T \in \mathcal{T}_h$, the global error estimate η_h, and the maximal element error estimate $\eta_{\max}$.

end while.

In the following, we report several numerical experiments to demonstrate the competitive behavior of the proposed algorithm. In our PMINRES solver, we use only one step of local multigrid iteration and one Gauß–Seidel sweep for pre- and postsmoothing.

5.1. We consider the Maxwell equation (1.1) on the three-dimensional "L-shaped" domain $\Omega = (-1,\,1)^3 \setminus \{(0,\,1)\times(-1,\,0)\times(-1,\,1)\}$. Let $\mu_r = \kappa^2\varepsilon_r = 1$ and $\Gamma = \partial\Omega$. The nondimensional wavelength is $2\pi/(\kappa \mathrm{diam}(\Omega)) = \pi/\sqrt{3}$, since the diameter of Ω is $2\sqrt{3}$. The Dirichlet boundary condition and the source $\mathbf{f}$ are so chosen that the exact solution is $\mathbf{E} := \nabla\left\{r^{1/2}\sin(\phi/2)\right\}$ in cylindrical coordinates.

Figure 5.1 shows the curves of $\log\|\mathbf{E}-\mathbf{E}_k\|_{\mathbf{H(curl;}\,\Omega)}$ versus $\log N_k$, where N_k is the number of elements and $\mathbf{E}_k$ is the finite element solution of (2.18) over the mesh $\mathcal{T}_k$. Figure 5.2 shows the $\log\eta_k$–$\log N_k$ curves, where η_k is the associated a posteriori error estimate over $\mathcal{T}_k$ defined in Theorem 3.3. They indicate that the adaptive meshes and the associated numerical complexity are quasi-optimal; i.e.,

$$\eta_k = C\,N_k^{-1/3} \qquad \text{and} \tag{5.2}$$

$$\|\mathbf{E}-\mathbf{E}_k\|_{\mathbf{H(curl;}\,\Omega)} = C\,N_k^{-1/3} \tag{5.3}$$

are valid asymptotically. They also clearly show the advantage of the adaptive method compared with the uniform refinements.

Table 5.1 shows the numbers of PMINRES iterations required to reduce the initial residual by a factor of 10^{-8} on different levels. We observe that the PMINRES algorithm converges in very few steps with the number of degrees of freedom varying from 722 to 1,616,983. Figure 5.3 shows the CPU time versus the number of degrees of freedom on different adaptive meshes.

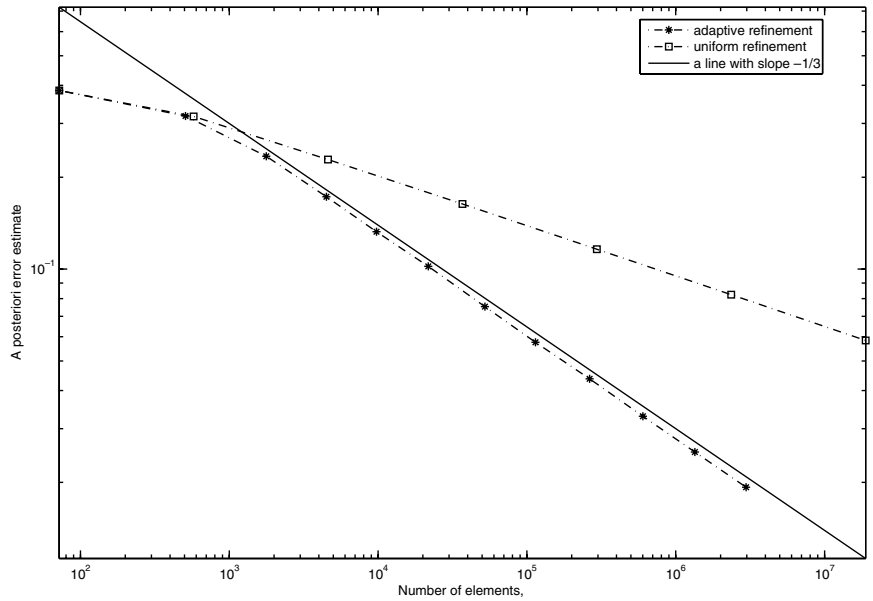


FIG. 5.2. *Quasi optimality of the adaptive mesh refinements of the a posteriori error estimate* (*Example* 5.1).

TABLE 5.1
The number of PMINRES iterations (N_{itrs}) *required to reduce the initial residual by a factor* 10^{-8} (*Example* 5.1).

Level	1	3	5	7	8	9	10
DOFs	722	5668	26810	138667	319360	725217	1616983
N_{itrs}	5	6	6	7	7	8	8

Figure 5.4 shows an adaptive mesh of 18,874,368 elements after 12 adaptive iterations. We observe that the mesh is locally refined near the corner line $x_1 = x_2 = 0$, where the solution is singular.

5.2. We consider a time-harmonic problem containing an inner screen $\Gamma := \{(x, y, z) : -0.5 \le x, z \le 0.5,\ y = 0\}$. We define $\Omega = (-1, 1)^3 \setminus \Gamma$ and $\Sigma = \partial\Omega \setminus \Gamma$. We set $\mu_r = \varepsilon_r = \lambda = 1$ in (1.1) and (1.2) and define

$$\mathbf{f} := \mathbf{0}, \qquad \mathbf{g} := \mathbf{curl}\,\mathbf{E}_i \times \nu - \mathbf{i}\,\kappa\,\mathbf{E}_{i,t},$$

where $\mathbf{E}_i = (e^{\mathbf{i}\,y}, 0, e^{\mathbf{i}\,y})^T/\sqrt{2}$ is perpendicular to the perfect conducting "screen." Thus (1.1)–(1.3) model the scattering by Γ under the incident field $\mathbf{E}_i$. In this case, only $\mathbf{H}^s$-regularity ($s < 1/2$) of the solution is guaranteed.

In this experiment, we test the performance of the a posteriori error estimate and the preconditioned MINRES algorithm under different frequencies, i.e., $\kappa^2 = 1$, 1.5, 10, 20, 36, 64, 100 in (1.1). The nondimensional wavelength here is $\pi/(\sqrt{3}\,\pi)$. Figures 5.5–5.7 show the results for $\kappa = 1$ in (1.1) and (1.2), while Figures 5.8 and 5.9 show the result for $\kappa > 1$.

Figure 5.5 shows the $\log \eta_k$–$\log N_k$ curves and indicates that the adaptive meshes and the associated numerical complexity are quasi-optimal and (5.2) is valid asymptotically. It also clearly shows the advantage of the adaptive method compared with the uniform refinements.

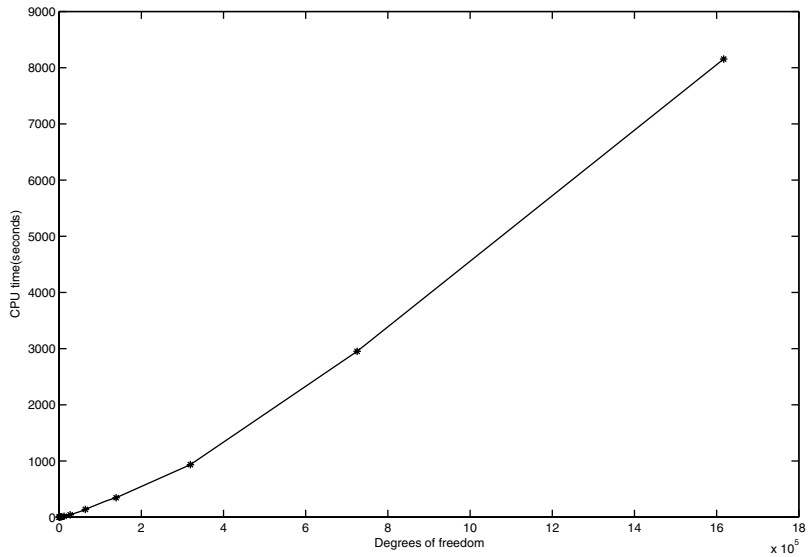


FIG. 5.3. *CPU time for PMINRES iterations on adaptively refined grids* (*Example* 5.1).

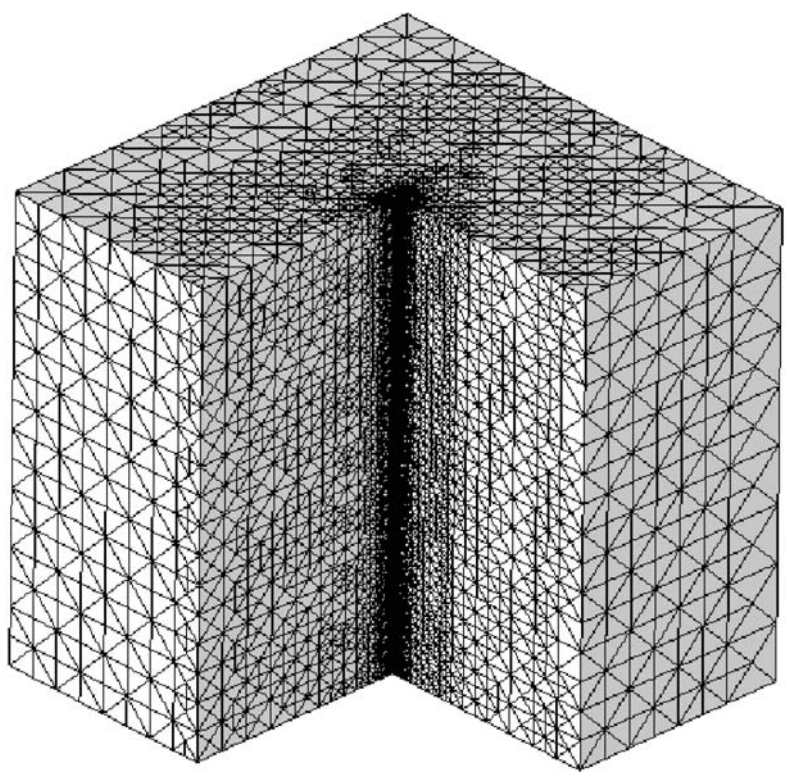


FIG. 5.4. *An adaptively refined mesh of* 18,874,368 *elements after* 12 *adaptive iterations* (*Example* 5.1).

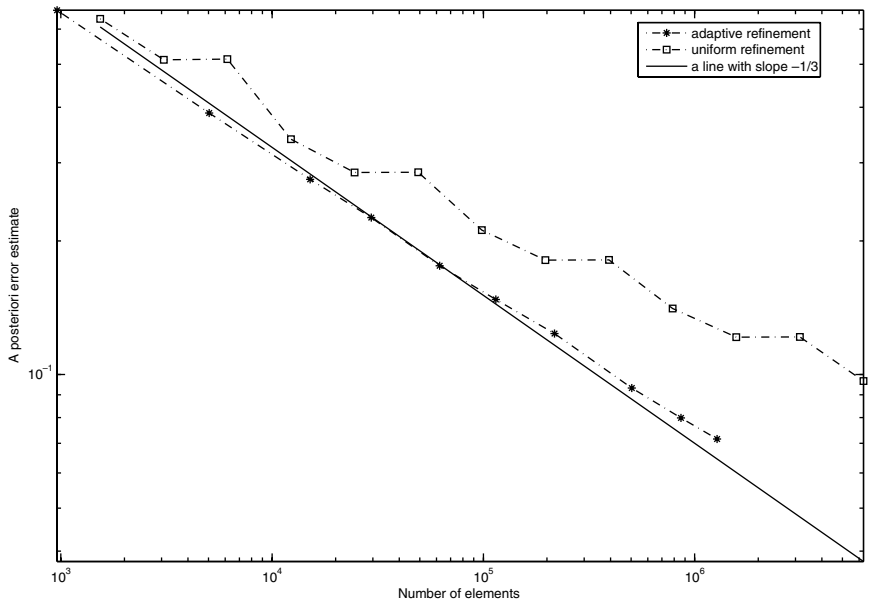


FIG. 5.5. *Quasi optimality of the adaptive mesh refinements of the a posteriori error estimate* (*Example* 5.2).

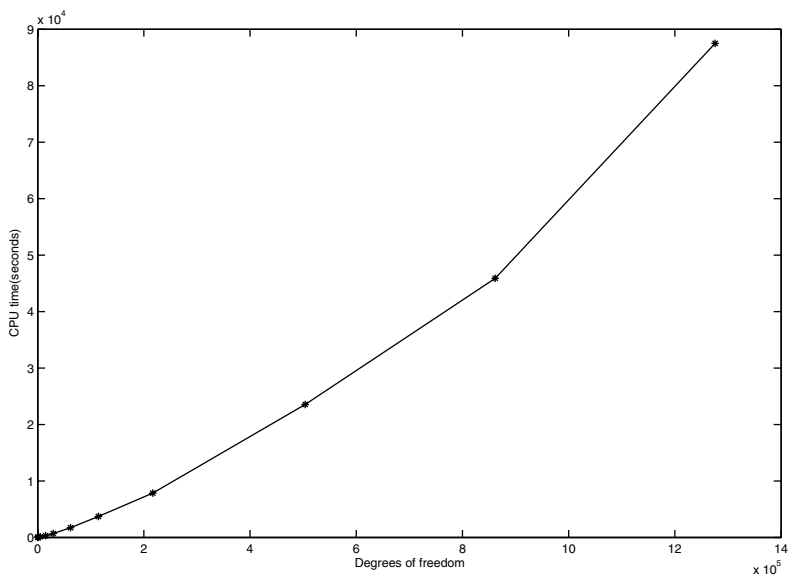


FIG. 5.6. *CPU time for PMINRES iterations on adaptively refined meshes* (*Example* 5.2).

Figure 5.6 shows the CPU time versus the number of degrees of freedom on different adaptive meshes.

Figure 5.7 shows an adaptive mesh of 2,947,848 elements after 11 adaptive iterations. We observe that the mesh is locally refined near the boundary of the "screen."

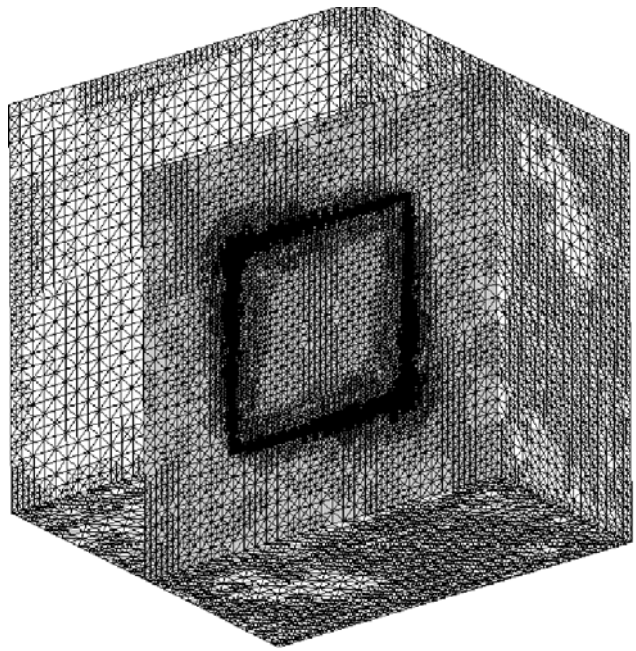


FIG. 5.7. *An adaptively refined mesh of* 2,947,848 *elements after* 11 *adaptive iterations* (*Example* 5.2).

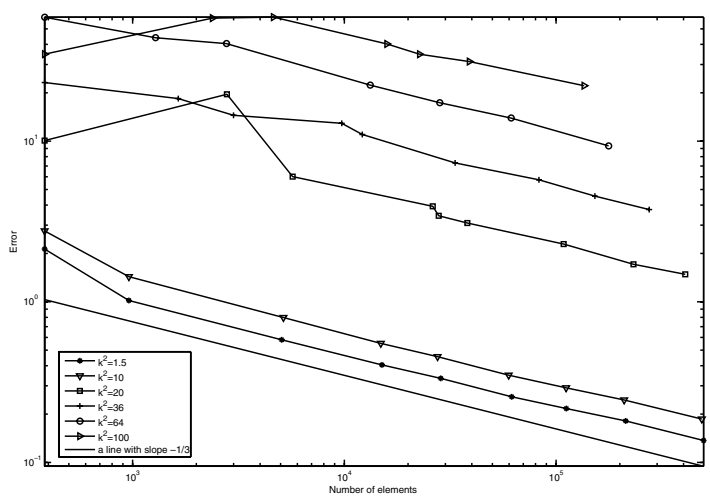


FIG. 5.8. *Quasi optimality of the adaptive mesh refinements of the a posteriori error estimate for* $\kappa^2 = 1.5,\ 10,\ 20,\ 36,\ 64,\ 100$ (*Example* 5.2).

Figure 5.8 shows the $\log \eta_k$–$\log N_k$ curves for $\kappa^2 = 1.5,\ 10,\ 20,\ 36,\ 64,\ 100$. It indicates that the adaptive meshes and the associated numerical complexity are quasi-optimal and (5.2) is valid asymptotically for all theses values of κ.

Figure 5.9 shows the CPU time versus the number of degrees of freedom on different adaptive meshes for $\kappa^2 = 1.5,\ 10,\ 20,\ 36,\ 64,\ 100$. All lines show a quasi-linear increasing of the CPU time of the solution of the algebraic system with respect

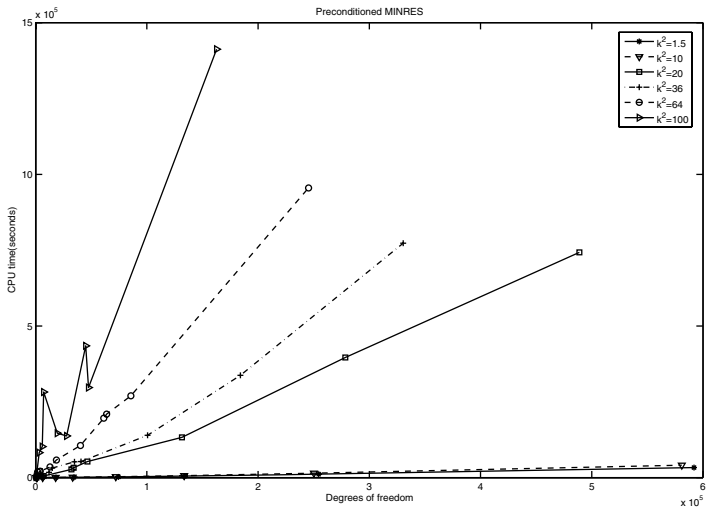


FIG. 5.9. *CPU time for PMINRES iterations on adaptively refined meshes for* $\kappa^2 = 1.5, 10, 20, 36, 64, 100$ (*Example* 5.2).

TABLE 5.2

The number of PMINRES iterations (N_{itrs}) *required to reduce the initial residual by a factor* 10^{-8} (*Example* 5.2).

$\kappa^2 = 1$	Level	1	3	5	6	7	8	9
	DOFs	1322	18376	74655	137309	258991	599640	1022905
	N_{itrs}	36	32	32	32	31	31	31
$\kappa^2 = 1.5$	Level	1	3	5	6	7	8	9
	DOFs	604	6252	34471	74415	133960	254531	591946
	N_{itrs}	36	44	44	44	46	46	46
$\kappa^2 = 10$	Level	1	4	5	6	7	8	9
	DOFs	604	18040	33302	72094	133515	250203	580971
	N_{itrs}	44	56	58	56	58	58	58
$\kappa^2 = 20$	Level	1	4	5	6	7	8	9
	DOFs	604	32163	34225	46611	131532	278454	488868
	N_{itrs}	272	1091	1035	1068	879	854	833
$\kappa^2 = 36$	Level	1	3	5	7	8	9	10
	DOFs	604	3938	14956	40788	100672	184125	330495
	N_{itrs}	845	1917	3358	2166	2167	2231	2314
$\kappa^2 = 64$	Level	1	3	5	7	8	9	10
	DOFs	604	3603	16212	53104	55350	74246	212931
	N_{itrs}	2413	7012	6852	6757	6424	6547	7302
$\kappa^2 = 100$	Level	1	4	5	6	7	8	9
	DOFs	604	7224	19516	27888	45018	47534	162770
	N_{itrs}	2464	55223	15916	9437	17278	10479	15775

to the number of degrees of freedom. However, when $\kappa^2 \geq 20$, the CPU time increases dramatically compared with the time for small κ.

Table 5.2 shows the numbers of PMINRES iterations required to reduce the initial residual by a factor of 10^{-8} for $\kappa^2 = 1$, 1.5, 10, 20, 36, 64, 100 on different levels. We observe that it remains nearly fixed for $\kappa^2 \leq 10$ with the number of degrees of freedom varying from 1,322 to 1,513,049. When $\kappa^2 \geq 20$, the number of iterations

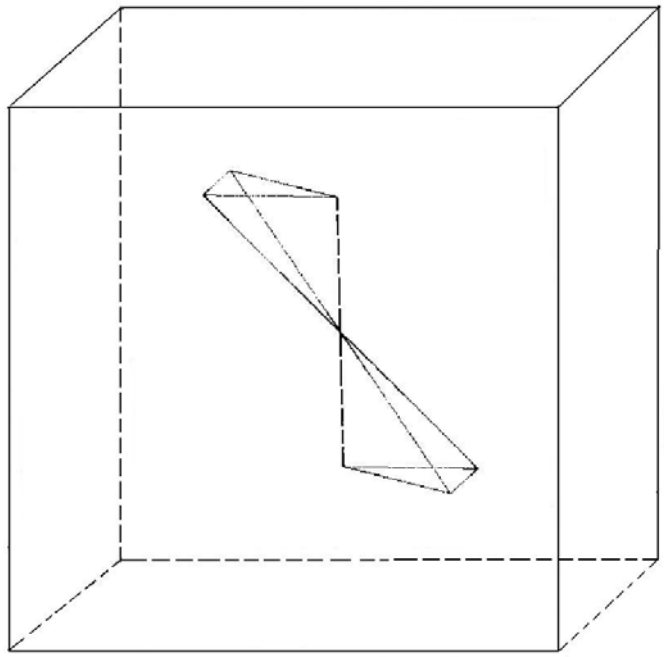


FIG. 5.10. *Two tetrahedral conductors with a common vertex* (*Example* 5.3).

increases rapidly as κ increases, but it varies slightly for all κ with the number of degrees of freedom increasing.

We remark that the frequency of the problem influences the total value of the a posteriori error estimate, but does not influence its reduction rate. These phenomena coincide with Theorems 3.3 and 3.4, where the frequency influences only the constants and the estimates in the theorem, but does not influence the convergence rate. Furthermore, the frequency makes a strong impact on the performance of the preconditioned MINRES algorithm.

5.3. This experiment is to test the robustness of our method for cavity problems on non-Lipschitz domains. The scatter consists of two perfect tetrahedral conductors S_1 and S_2 with vertices

$$\begin{aligned} S_1: &\quad (0,\,0,\,0),\quad (0.5,\,0.5,\,-0.5),\quad (0,\,0.5,\,-0.5),\quad (0,\,0,\,-0.5) \quad \text{and}\\ S_2: &\quad (0,\,0,\,0),\quad (-0.5,\,-0.5,\,0.5),\quad (0,\,-0.5,\,0.5),\quad (0,\,0,\,0.5). \end{aligned}$$

The computational domain is defined by $\Omega = (-1,\ 1)^3 \setminus (S_1 \cup S_2)$ (see Figure 5.10). We set $\mu_r = \epsilon_r = \lambda = \kappa = 1$ in (1.1) and (1.2) and define

$$\mathbf{f} := \mathbf{0}, \quad \mathbf{g} := \mathbf{curl}\,\mathbf{E}_i \times \nu - \mathbf{i}\kappa\,\mathbf{E}_{i,t},$$

where $\mathbf{E}_i = (e^{\mathbf{i}y},\, 0,\, e^{\mathbf{i}y})^T/\sqrt{2}$. The nondimensional wavelength here is $\pi/\sqrt{3}$.

Figure 5.11 shows the $\log \eta_k$–$\log N_k$ curves and indicates that the adaptive meshes and the associated numerical complexity are quasi-optimal and (5.2) is valid asymptotically. It also clearly shows the advantage of adaptive method compared with the uniform refinements.

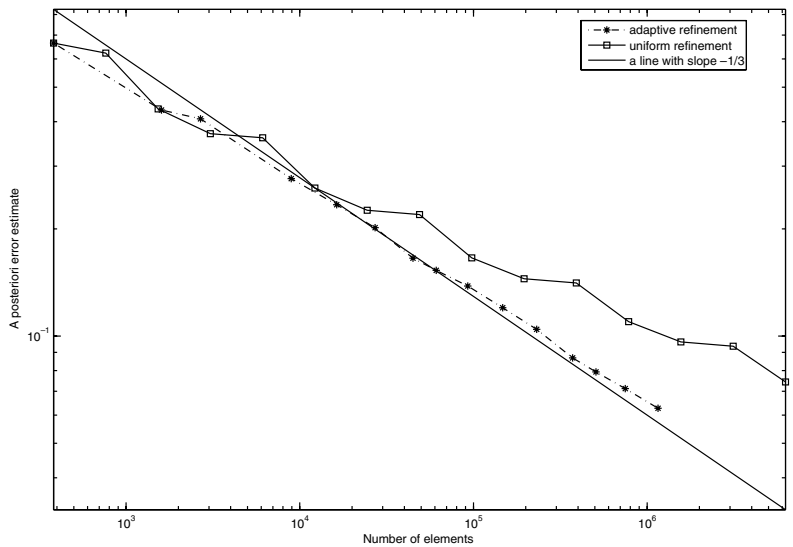


FIG. 5.11. *Quasi optimality of the adaptive mesh refinements of the a posteriori error estimate* (*Example* 5.3).

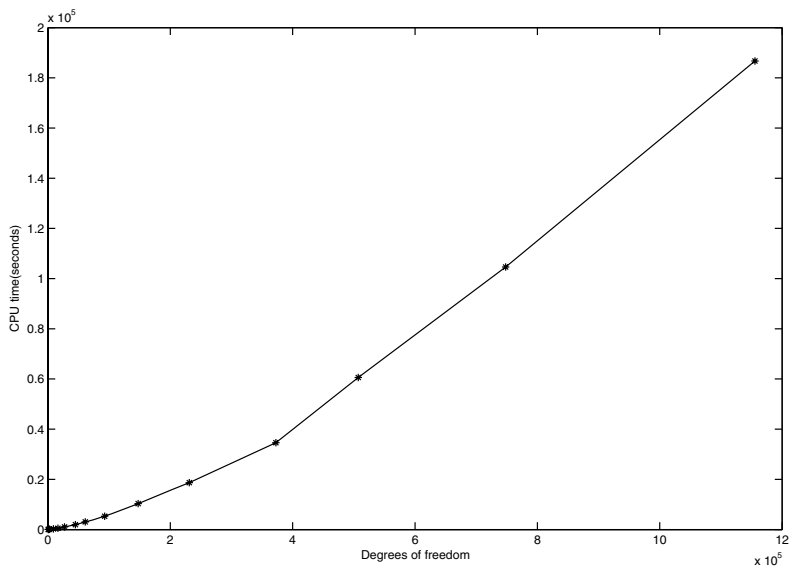


FIG. 5.12. *CPU time for PMINRES iterations on adaptively refined meshes* (*Example* 5.3).

Figure 5.12 shows the CPU time versus the number of degrees of freedom on different adaptive meshes.

Figure 5.13 shows an adaptive mesh of 1,856,117 elements after 15 adaptive iterations. We observe that the mesh is locally refined near the boundary of the scatter.

Table 5.3 shows the number of PMINRES iterations required to reduce the initial residual by a factor of 10^{-8} on different levels. We observe that it remains nearly fixed with the number of degrees of freedom varying from 2,102 to 1,394,075.

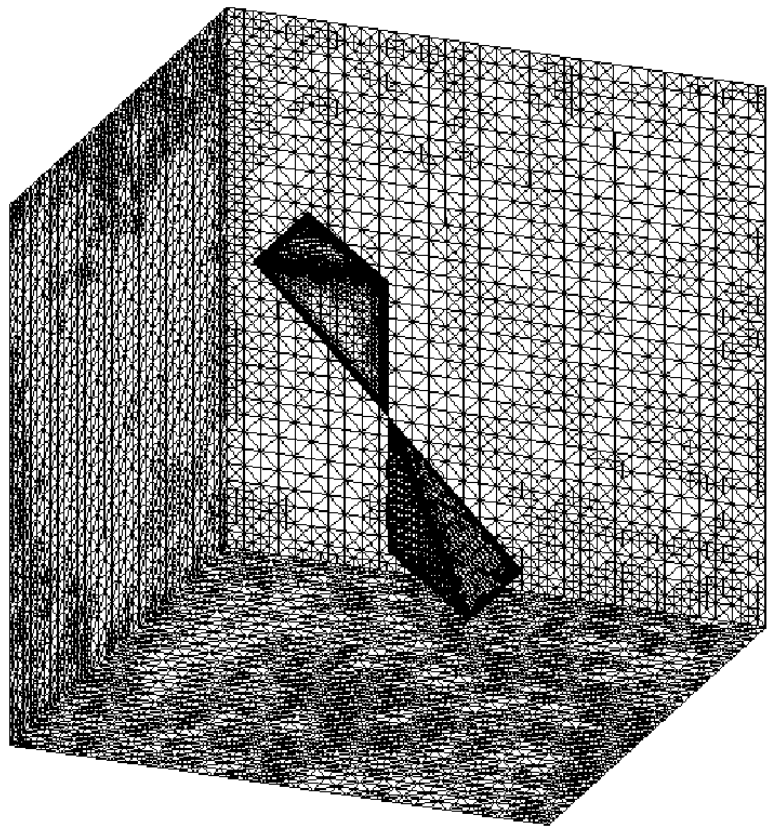


FIG. 5.13. *An adaptively refined mesh of* 1,856,117 *elements after* 15 *adaptive iterations* (*Example* 5.3).

TABLE 5.3
The number of PMINRES iterations (N_{itrs}) *required to reduce the initial residual by a factor of* 10^{-8} (*Example* 5.3).

Level	1	3	5	7	9	11	13	14
DOFs	2102	11134	33437	74735	179603	450971	903251	1394075
N_{itrs}	42	42	42	40	41	41	42	42

REFERENCES

[1] D. N. ARNOLD, R. S. FALK, AND R. WINTHER, *Multigrid in H*(div) *and H*(curl), Numer. Math., 85 (2000), pp. 197–217.

[2] F. ASSOUS, P. CIARLET, JR., AND E. SONNENDRÜCKER, *Resolution of the Maxwell equations in a domain with reentrant corners*, RAIRO Modél. Math. Anal. Numér., 32 (1998), pp. 359–389.

[3] I. BABUŠKA AND W. C. RHEINBOLDT, *Error estimates for adaptive finite element computations*, SIAM J. Numer. Anal., 15 (1978), pp. 736–754.

[4] I. BABUŠKA AND A. AZIZ, *Survey lectures on mathematical foundations of the finite element method*, in The Mathematical Foundations of the Finite Element Method with Application to Partial Differential Equations, A. Aziz, ed., Academic Press, New York, 1972, pp. 1–359.

[5] A. BAYLISS, C. I. GOLDSTEIN, AND E. TURKEL, *On accuracy conditions for the numerical computation of waves*, J. Comput. Phys., 59 (1985), pp. 396–404.

[6] R. BECK, R. HIPTMAIR, R. H. W. HOPPE, AND B. WOHLMUTH, *Residual based a posteriori error estimators for eddy current computation*, M2AN Math. Model. Numer. Anal., 34 (2000), pp. 159–182.

[7] R. Beck, P. Deuflhard, R. Hiptmair, R. H. W. Hoppe, and B. Wohlmuth, *Adaptive multilevel methods for edge element discretizations of Maxwell's equations*, Surveys Math. Indust., 8 (1999), pp. 271–312.

[8] M. Sh. Birman and M. Z. Solomyak, *L_2-theory of the Maxwell operator in arbitrary domains*, Uspekhi Mat. Nauk, 42 (1987), pp. 61–76 (in Russian); Russian Math. Surveys, 43 (1987), pp. 75–96 (in English).

[9] M. Sh. Birman and M. Z. Solomyak, *Principal singularities of the electric component of an electromagnetic field in regions with screens*, St. Petersburg Math. J., 5 (1994), pp. 125–139.

[10] H. Blum and M. Dobrowolski, *On finite element methods for elliptic equations on domains with corners*, Computing, 28 (1982), pp. 53–63.

[11] Z. Chen and S. Dai, *On the efficiency of adaptive finite element methods for elliptic problems with discontinuous coefficients*, SIAM J. Sci. Comput. 24 (2002), pp. 443–462.

[12] Z. Chen and S. Dai, *Adaptive Galerkin methods with error control for a dynamical Ginzburg–Landau model in superconductivity*, SIAM J. Numer. Anal. 38 (2001), pp. 1961–1985.

[13] Z. Chen, Q. Du, and J. Zou, *Finite element methods with matching and nonmatching meshes for Maxwell equations with discontinuous coefficients*, SIAM J. Numer. Anal., 37 (2000), pp. 1542–1570.

[14] Z. Chen and G. H. Ji, *Sharp L^1 a posteriori error analysis for nonlinear convection-diffusion problems*, Math. Comp., 75 (2006), pp. 43–71.

[15] Z. Chen and H. Wu, *An adaptive finite element method with perfectly matched absorbing layers for the wave scattering by periodic structures*, SIAM J. Numer. Anal. 41 (2003), pp. 799–826.

[16] M. Costabel, *A remark on the regularity of solutions of Maxwell's equations on Lipschitz domains*, Math. Methods Appl. Sci., 12 (1990), pp. 365–268.

[17] M. Costabel and M. Dauge, *Singularities of electromagnetic fields in polyhedral domains*, Arch. Ration. Mech. Anal., 151 (2000), pp. 221–276.

[18] M. Costabel and M. Dauge, *Computation of resonance frequencies for Maxwell equations in non-smooth domains*, in Topics in Computational Wave Propagation, M. Ainsworth et al., eds., Springer-Verlag, Berlin, 2003, pp. 125–161.

[19] A.-S. Bonnet-Ben Dhia, C. Hazard, and S. Lohrengel, *A singular field method for the solution of Maxwell's equations in polyhedral domains*, SIAM J. Appl. Math., 59 (1999), pp. 2028–2044.

[20] V. Girault and P. A. Raviart, *Finite Element Methods for Navier-Stokes Equations*, Springer-Verlag, Berlin, 1986.

[21] J. Gopalakrishnan, J. E. Pasciak, and L. F. Demkowicz, *Analysis of a multigrid algorithm for time harmonic Maxwell equations*, SIAM J. Numer. Anal., 42 (2004), pp. 90–108.

[22] P. Grisvard, *Elliptic Problems in Nonsmooth Domains*, Pitman, Boston, 1985.

[23] R. Hiptmair, *Multigrid method for Maxwell's equations*, SIAM J. Numer. Anal., 36 (1998), pp. 204–225.

[24] W. F. Mitchell, *Optimal multilevel iterative methods for adaptive grids*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 146–167.

[25] P. Monk, *A posteriori error indicators for Maxwell's equations*, J. Comput. Appl. Math., 100 (1998), pp. 173–190.

[26] P. Monk, *Finite Element Methods for Maxwell's Equations*, Clarendon Press, Oxford, UK, 2003.

[27] P. Morin, R. H. Nochetto, and K. G. Siebert, *Data oscillation and convergence of adaptive FEM*, SIAM J. Numer. Anal., 38 (2000), pp. 466–488.

[28] J. Nédélec, *Mixed finite elements in R^3*, Numer. Math., 35 (1980), pp. 315–341.

[29] J. E. Pasciak and J. Zhao, *Overlapping Schwarz methods in H(curl) on polyhedral domains*, J. Numer. Math., 10 (2002), pp. 221–234.

[30] A. Schmidt and K. G. Siebert, *ALBERT—An Adaptive Hierarchical Finite Element Toolbox*, http://www.alberta-fem.de.

[31] A. Schmidt and K. G. Siebert, *Design of Adaptive Finite Element Software: The Finite Element Toolbox ALBERTA*, Lecture Notes in Computational Science and Engineering 2, Springer-Verlag, Berlin, Heidelberg, 2005.

[32] L. R. Scott and Z. Zhang, *Finite element interpolation of nonsmooth functions satisfying boundary conditions*, Math. Comp., 54 (1994), pp. 483–493.

[33] H. Wu and Z. Chen, *Uniform convergence of multigrid V-cycle on adaptively refined finite element meshes for second order elliptic problems*, Science in China: Ser. A Mathematics, 49 (2006), pp. 1–28.

SIAM J. SCI. COMPUT.
Vol. 29, No. 1, pp. 139–181
© 2007 Society for Industrial and Applied Mathematics

USING HIGH-ORDER METHODS ON ADAPTIVELY REFINED BLOCK-STRUCTURED MESHES: DERIVATIVES, INTERPOLATIONS, AND FILTERS*

JAIDEEP RAY†, CHRISTOPHER A. KENNEDY‡, SOPHIA LEFANTZI‡, AND HABIB N. NAJM‡

Abstract. Block-structured adaptive mesh refinement (AMR) is used for efficient resolution of partial differential equations (PDEs) solved on large computational domains by clustering mesh points only where required by large gradients. Previous work has indicated that fourth-order convergence can be achieved on such meshes by using a suitable combination of high-order derivative, interpolation, and filter stencils and can deliver significant computational savings over conventional second-order methods at engineering error tolerances. In this paper, we explore the interactions between the errors introduced by the discretized operators for derivatives, interpolations, and filters. We develop general expressions for high-order derivative, interpolation, and filter stencils applicable in multiple dimensions, using a Fourier approach, to facilitate high-order block-structured AMR implementations. These stencils are derived under the assumption that the fields that they are applied to are smooth. For a given derivative stencil (and thus a given order of accuracy), the necessary interpolation order is found to be dependent on the highest spatial derivative in the PDE being solved. This is demonstrated empirically, using one- and two-dimensional model equations, by observing the increase in delivered accuracy as the order of accuracy of the interpolation stencil is increased. We also examine the empirically observed order of convergence, as the effective resolution of the mesh is increased by successively adding levels of refinement, with different orders of derivative, interpolation, and filter stencils. The procedure devised here is modular and its various pieces, i.e., the derivative, interpolation, and filter stencils, can be extended independently to higher orders and dimensions (in the case of interpolation stencils). The application of these methods often requires nontrivial logic, especially near domain boundaries; this logic, along with the stencils used in this study, are available as a freely downloadable software library.

Key words. high order, AMR, SAMR, derivative, interpolation, filter

AMS subject classifications. 65M06, 65Y99, 65N12

DOI. 10.1137/050647256

1. Introduction.

Space-time partial differential equations (PDEs) are routinely solved numerically in scientific and engineering studies. Using the method of lines, and possibly finite-difference techniques applied to the spatial operators, one solves a large system of ordinary differential equations (ODEs). A critical aspect of solving these sometimes enormous problems is the efficiency of the spatial discretization. Ideally, one first establishes an error tolerance requirement for the simulations and then

*Received by the editors December 12, 2005; accepted for publication (in revised form) July 25, 2006; published electronically January 26, 2007. This work was supported by the U.S. Department of Energy (DOE), Office of Basic Energy Sciences, Division of Chemical Sciences, Geosciences, and Biosciences, SciDAC (Scientific Discovery through Advanced Computing) Computational Chemistry Program. Support was also provided by the DOE Office of Advanced Scientific Computing Research SciDAC program. Sandia National Laboratories is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94-AL85000. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution or allow others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by the these rights.

http://www.siam.org/journals/sisc/29-1/64725.html

†Corresponding author. Sandia National Laboratories, MS 9159, P.O. Box 969, Livermore, CA 94551-0969 (jairay@somnet.sandia.gov).

‡Sandia National Laboratories, Livermore, CA 94551-0969 (cakenne@speakeasy.net, lefantzi@comcast.net, hnnajm@ca.sandia.gov).

sets about finding the method that will reliably deliver this tolerance at minimum cost. In general, there are two simple guiding ideas to consider: high-order methods become relatively more efficient as error tolerances are reduced, and grid points should be allocated more generously to less smooth parts of the solution. Combining these two ideas in simple geometries, we will consider high-order spatial discretizations in conjunction with adaptive mesh refinement (AMR) for solution of PDEs having substantial spatial variation in smoothness but lacking spatial discontinuities. In particular, we will use block-structured AMR [4, 5], where a layered hierarchy of rectangular uniform-mesh patches is used to discretize the domain. While this approach resolves a domain efficiently by refining a grid only where required, one would also like to minimize the number of mesh levels, and this is facilitated by high-order methods. Large multiphysics problems are typically solved on distributed memory parallel computers. These require that the domain of the solution be decomposed among processors in a load-balanced manner. If one tries to achieve a given spatial error tolerance by refining repeatedly, one gets localized regions of very high mesh density, distributed within an otherwise coarse mesh. This strongly inhomogeneous grid poses a challenge to domain-decomposition parallel load-balancing algorithms, which frequently fail to achieve a scalable decomposition. Thus, while AMR does allow one to concentrate mesh density in regions of interest, a certain moderation in its use adds tremendously to the ease and speed of computation.

High-order (i.e., > 2) discretizations on adaptively refined meshes are not very common. If one forgoes domains with intricate geometries, one may use block-structured adaptive meshes and exploit the meshs' regularity to consider high-order finite-difference methods. Barad and Colella [3] address the problem of solving the Poisson equation to fourth-order accuracy on block-structured adaptively refined meshes. Starting with a classical Mehrstellen method, the authors develop and test a fourth-order solution methodology for the Poisson equation in two and three dimensions. Tests were done on an adaptive mesh (a coarse base mesh with one extra level of refinement), and fourth-order convergence was predicted theoretically and demonstrated empirically. Interpolations at coarse-fine boundaries were done using a sequence of high-order one-dimensional interpolations, some fourth-order accurate and others sixth-order accurate, depending upon the configuration of patches. Their method requires that the refinement ratio between successive levels be four. Huang and Zhang [15] used fourth-order dispersion relation preserving (DRP) [35, 34] and prefactored compact schemes [2, 13] to solve a coupled set of three linear hyperbolic equations. A "halo" of cells, set up around the fine mesh block, is filled via symmetric one-dimensional interpolations of order 4. Tests showed that a Gaussian pulse propagating across a coarse-fine boundary showed far smaller oscillations when fourth-order interpolants were used; the authors used filters and artificial viscosity in separate tests to remove them. A preference for artificial viscosity was indicated, since it can be controlled in a spatially variable manner. Tests were done in two dimensions with a 3-level block-structured adaptive mesh. Rectangular overset grids may be considered to be block-structured adaptive meshes which are adaptive in space (i.e., have variable resolution in space) but do not change in time. Henshaw [12] studied fourth-order solutions of the Navier–Stokes equations on cylindrical and spherical meshes overset on a Cartesian mesh. Fifth-order one-dimensional interpolations were used between overset grids. Many of the numerical results used in the study were developed in [7], which also contains recommendations for the order of interpolations to be used when an arbitrary PDE is solved to arbitrary order of accuracy on overset grids. A proof for the order of the interpolation was developed for $\partial^{2n}\phi/\partial x^{2n} = F$ for a second-

order discretization only. Alternative approaches to high-order AMR are discussed by Holmström in [14], where wavelets are often central to the AMR procedure.

The use of block-structured AMR to perform PDE-based simulations requires the resolution of a host of issues concerning efficiency, accuracy, and ease of use/implementation of a particular numerical strategy within a block-structured AMR context. In our study, we will follow the approach of [16, 20, 19, 18, 9], where differentiation and interpolations are performed in physical space, as opposed to being done via wavelets. From a computational efficiency point of view, [19] points out that block-structured AMR is preferable to a traditional single grid method only when the higher wavenumber content of the flowfield is relatively nonuniform and resides within smaller regions of the domain. A key element of typical block-structured AMR constructions is that the more accurate solution from the finest grids is periodically interpolated onto the coarser ones, and solutions from the coarse mesh are used at coarse-fine interfaces. The coarse-fine interface takes the form of a "halo" of cells/grid points around fine patches, where the solution is interpolated from the underlying coarse mesh. Since integration variables on each grid are eventually the result of interpolations based on local polynomials, one cannot differentiate this interpolated data indefinitely. Interpolated data based on an order p_I interpolant, which is differentiated k times (differentiation order), result in a field with a maximum differentiation order of $p_I - k + 1$, i.e., C^{p_I-k+1} (see Appendix C). For example, the viscous terms in the Navier–Stokes equations are expressed as second derivatives in space. Consider a point A where a second derivative has to be evaluated. If the evaluation of this derivative requires data to be interpolated in the vicinity of A, prior to being acted on by a (p_D)th-order stencil, one may observe an order reduction in the solution to $p_D - 1$ unless interpolations of the correct order are used. Note, however, that this will be observed only if the viscous terms are of significant magnitude; otherwise the errors in the viscous terms (of order $p_D - 1$) could be swamped by (p_D)th-order derivative errors contributed by other (e.g., convective) terms.

Another important issue in high-order AMR is dealing with the potential for the [illegible] associated with interpolation on uniform grids. A key result of [36] is that for spatial modes of the grid variables, the error in interpolation of modes, $\exp(i\xi x)$, decreases to zero as the grid density tends to infinity if and only if ξ, the wavenumber [32], is small enough to provide six grid points per wavelength. This implies that high-order interpolation will succeed only if there is active control of the spatial wavenumbers that live on the computational grid. Central to this are robust refinement/coarsening criteria combined with careful addition of numerical dissipation for an otherwise low-dissipation numerical method. This robust refinement/coarsening strategy might act to ensure that the energy content of a grid variable above some chosen cut-off wavenumber is essentially zero. This also establishes the requirement that reliable spatial error estimates be available. Wavelets are one of the means for doing this [29]. One could also carry on two simultaneous computations, one on an "actual" grid and another on a "ghost" grid with a factor-of-two coarser resolution. The difference between the two solutions gives an estimate of the spatial discretization error [4]. However, approximate error estimation techniques based on first and second derivatives are more commonly used [30]. To facilitate the interpolations required in both refinement and coarsening, several approaches to high-wavenumber dissipation may be utilized. Filtering of grid variables may be done directly using stand-alone filters. Indirectly, this may be accomplished by using upwinded derivative operators rather than centered-difference approximations in evaluating the RHS, or a hyperviscosity term may be added. A robust refine-

ment/coarsening strategy along with control of numerical dissipation is essential to an efficient high-order AMR method, but one does not appear to fully exist currently.

Several other outstanding issues must be considered when contemplating high-order AMR. One is the proper specification of interpolant boundary closures. It is well known that elaborate boundary closures must be used if one seeks uniformly high-order, finite-difference, first-derivative operators and time-stable integration [6, 26, 27, 31]. Less clear is what, if anything, is required of finite-difference interpolant boundary closures above and beyond satisfying accuracy requirements. Do interpolant boundary closures affect time stability of the overall method? The literature offers no guidance on this matter. Another topic that may affect the efficiency of an AMR procedure is the selection of the Level0 grid spacing. Large initial grid spacings imply more grid levels, more interpolations, but fewer total grid points. A lower grid point count can be advantageous for expensive source terms like chemical reaction rates—more so if the chemical mechanism is stiff. If the refinement/coarsening apparatus is of high quality, interpolations are relatively cheap, and the domain decomposition is good; then a relatively larger Level0 grid spacing seems appropriate. On the other hand, to the degree to which these three items are not met, a smaller Level0 grid spacing seems more prudent. When determining the number of mesh levels, one must also remember that the smallest acceptable grid spacing on any grid level is a function not just of the governing equations but also of the numerical dissipation.

Block-structured AMR algorithms allow variables to be declared at the center of the rectangular or cuboid cells of a Cartesian mesh ("cell-centered" collocation), as well as at the cell corners ("vertex-centered") or at the center of the faces ("face-centered"). One of the major challenges in high-order block-structured AMR is in cell-centered settings, where discrete numerical conservation of the stencils is also demanded. In second-order methods, where a field variable is assumed to vary linearly across a cell, the cell-centered value is also the cell-averaged one. In high-order methods, where the intracell variation is not linear, a distinction has to be made between cell-centered and cell-averaged values, as indicated by Barad and Colella in [3]. They also ensure discrete conservation by imposing continuity of fluxes across coarse-fine interfaces. Their basic approach is the same as the one outlined in [4] (where discrepancies between coarse and fine mesh fluxes are stored in "registers" to "fix up" coarse cells abutting a coarse-fine interface after a timestep), but the fourth-order approach is considerably more involved.

Although AMR with high-order discretizations is appealing and potentially useful, it complicates the time integration strategy. Each mesh level has its own grid spacing, and consequently the stiffness of the convective and diffusive eigenvalues vary inversely and inversely squared with the grid spacing, respectively. If convection and diffusion are treated explicitly in time, then there will be a large range in stepsizes corresponding to the stability limit based upon convection and diffusion on each grid. One may approach this by integrating all grids with a uniform stepsize. In this way, the formal order of traditional ODE integrators is maintained; however, the global timestep is restricted by the stability constraint on the finest grid. Alternatively, a "time-refined" approach can be used, where different timestep sizes are employed for each mesh level [5]. Typically, this is approached with a [illegible] integration on the various mesh levels, starting with the coarsest level (at a timestep determined by the stability constraint of the mesh level itself), followed recursively by the integration of its children mesh levels, which are processed more often but at their (smaller) stability-constrained timesteps. For example, in a viscous CFL dominated problem on a 4-level block-structured AMR mesh, the finest grid takes 2^8 steps for every one on the coarsest. Generally, this approach constitutes what is known as [illegible],

but may also be approached more rigorously using partitioned multirate [10] methods. This approach constitutes partitioning of the RHS at the mesh level. Applying a nonmultirate integrator in a multirate context may often return less than the formal rate of convergence. One may further optimize the integration strategy by treating convection, diffusion, and reaction using different integration methods. A partitioning like this is often referred to as an additive partitioning [23]. Alternatively, A- and L-stable [11] implicit schemes may ignore stability restrictions and select stepsizes based exclusively on accuracy and iteration concerns. Such methods usually result in linear systems which are solved by iterative techniques.

In [24] we demonstrated, using a one-dimensional FitzHugh–Nagumo (FN) equation, that fourth-order convergence could be achieved on block-structured adaptively refined meshes. We motivated the use of high-order AMR by showing, analytically, the savings in mesh resolution (for a given error) and the simplifications in the configuration of the adaptive mesh that could be realized by using high-order discretizations within the context of AMR. This was verified in computational experiments, where actual floating point operations were measured for second- and fourth-order approaches and compared for given error tolerances. For low error tolerances (e.g., 10^{-1}) fourth-order methods were marginally more economical compared to second-order approaches; however, as higher error tolerances were demanded of the computation (e.g., 10^{-4}) the computational saving, as measured in terms of floating point operations, became significant (the fourth-order simulations achieved the same error tolerance as the second-order methods but at a tenth of the floating point operations). These savings in computational work can be directly attributed to the fewer grid points which were needed by the fourth-order simulation, vis-à-vis the second-order simulation, to achieve a given error tolerance. These savings in computational work were also sufficiently large to compensate for the more involved high-order stencils used for the fourth-order simulations. Details of the choice of interpolations and filters, and the interactions between them, were omitted; instead we extended our method to more realistic problems by solving (on 28 processors) a coupled system of 10 reaction-diffusion equations modeling the ignition of H_2 in air.

The goal of this paper is to demonstrate high-order spatial convergence in multidimensions on adaptively refined Cartesian grids by using finite-difference differentiation, interpolation, and dissipation. It is beyond the scope of the paper to address all of the topics relevant to an efficient, high-order spatial scheme; for example, we will not address high-order temporal discretizations or time stable boundary closures for interpolant operators. Instead, we focus on demonstrating methods that give high-order convergence, and begin to address effective strategies for retaining high order but avoiding Runge phenomena. Runge phenomena are avoided by using dealiasing filters. In section 2 we discuss specific numerical and computational issues associated with block-structured AMR constructions and establish the stencil coefficients of the derivatives, interpolants, and filters. In section 3 we use these tools in different problems (the Korteweg–de Vries (KdV) and the Kuramoto–Sivashinsky (KS) equations in one dimension and a FitzHugh–Nagumo (FN) equation in two dimensions) and analyze their efficacy. The KdV and KS equations were solved to gauge the variation in interpolation order as the highest spatial derivative in the equation increased. The two-dimensional FN equation was addressed to study the use of filters to suppress Runge phenomena while preserving the solution order. We draw our conclusions in section 4.

2. Structured AMR. Structured AMR (SAMR; sometimes also referred to as block-structured AMR) [4, 5] is a particularly appealing approach to solving PDEs

within geometrically simple spatial domains. The starting point for the method consists of laying a relatively coarse Cartesian mesh over a rectangular domain. Based on some suitable metric, regions requiring further refinement are identified, and the grid points are flagged and collated into . . . . . children patches on which a denser Cartesian mesh is defined. The refinement factor between parent and child meshes is usually kept constant for a given problem. In this work, it is always specified as 2. The procedure is done recursively, so that one ultimately obtains a hierarchy of patches with different grid densities, with the finest patches overlaying a small part of the domain. This hierarchy will be referred to henceforth as a . . . . . (GH) or the . . ; individual patches will be termed . . . or . . . . Individual patches may be viewed as a wireframe laid over a rectangular domain with the dependent variables of a PDE defined at the intersection of the "wires" (referred to as "vertex-centered grids") or as a division of a domain into small cells, with the variables stored at cell centers ("cell-centered grids"). We will study only . . . . . grids in this paper. Such grids have the property that the finer mesh grid points lie either on top of the coarse mesh grid points or at their geometric midpoints. This may be seen in Figure 2.1, where large circles correspond to the coarse mesh, and fine mesh points are denoted by small triangles, squares, and circles requiring one-, two-, and three-dimensional interpolations, respectively. The solution at these edge-centered (triangles), face-centered (squares), and body-centered (small circle) points, at coarse-fine interfaces/halos, are interpolated from the underlying coarse mesh (the large circles) using a variety of interpolations techniques.

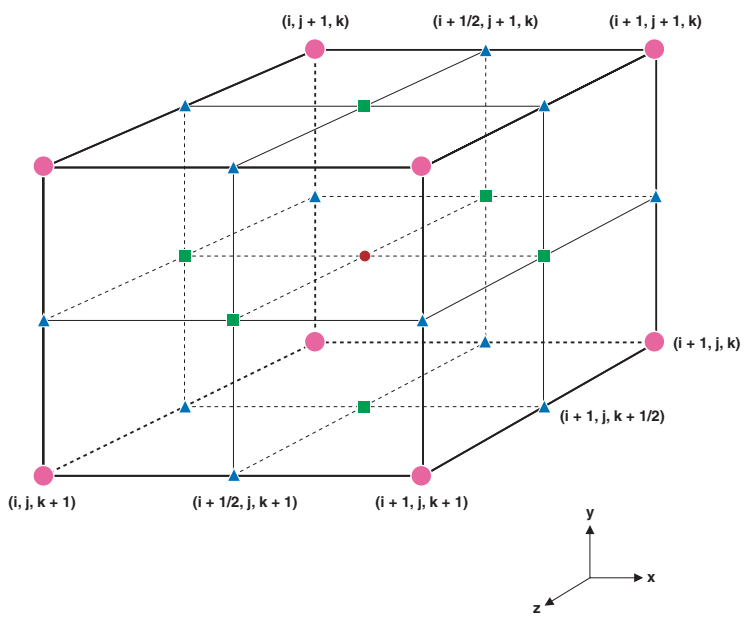


FIG. 2.1. *A vertex-centered grid. The large circles are the coarse mesh, and the fine mesh points at edge centers, face centers, and body centers are represented by triangles, squares, and a (small) circle, respectively.*

If a constant refinement factor is used (e.g., 2), the number of grid points rises rapidly as one refines. When used with time-refined explicit integration, SAMR-based simulations spend almost all their time in the finest levels. Further, in parallel simu-

lations, the requirement of keeping parents and children on the same processor results in poor domain-partitioning by conventional partitioners. Thus, in practice, time-refined simulations usually have shallow hierarchies (e.g., 4 deep) with a fairly dense "coarse" grid. The resulting timestep size (for an explicit integrator) determined by the stability constraints of the fine "coarse" grid is relatively small. To date, second-order spatial discretizations have been typical for SAMR simulations on vertex- and cell-centered grids. Accuracy has been achieved mainly by increasing mesh density, which has exacerbated the problem of a rather fine starting mesh (henceforth called Level0 mesh) or resulted in deep hierarchies. The high-order spatial discretizations employed in the present work will remedy this situation, reducing the requisite resolution requirement on the Level0 mesh.

High-order SAMR has a tremendous potential to reduce resolution requirements to achieve a given level of accuracy. Jameson [17] has attempted to quantify the relative efficiencies of different orders of accuracy in one dimension. Table 1 in [24] contains a listing of the number of grid points needed by centered derivative operators of various orders to resolve a particular wavenumber mode to a given error tolerance; a similar list for high-order interpolants is in Table 2.1. In both cases, we see that a modest increase in order (from second to fourth) reduces the resolution requirements by a factor of 3 at 1% accuracy; at tighter tolerances a factor of 20 is achieved, and at higher orders, one achieves almost 2 orders of magnitude. These savings are magnified as one proceeds to higher spatial dimensions.

TABLE 2.1

Number of grid points per wavelength required to obtain a chosen relative error tolerance ε using explicit, centered-difference, midpoint, interpolant operators of orders 2 *through* 10. *p_I is the order of accuracy of the interpolation operator. See Table* 2.8 *for stencil coefficients.*

ε	$p_I = 2$	$p_I = 4$	$p_I = 6$	$p_I = 8$	$p_I = 10$
$10^{-2.0}$	22.20	7.666	5.381	4.486	4.006
$10^{-2.5}$	39.49	10.29	6.597	5.262	4.577
$10^{-3.0}$	70.24	13.76	8.055	6.144	5.205
$10^{-3.5}$	124.9	18.39	9.810	7.153	5.900
$10^{-4.0}$	222.1	24.55	11.93	8.308	6.671
$10^{-4.5}$	395.0	32.76	14.48	9.635	7.530
$10^{-5.0}$	702.5	43.70	17.57	11.16	8.488

However, the use of high-order discretizations and interpolants is not without its drawbacks. SAMR interpolant operators are evaluated using a linear combination of as many as p_I^d (where p_I is the order of accuracy and d is the dimensionality) values and their respective coefficients. While this renders high-order interpolants more expensive compared to bilinear or biquadratic (trilinear and triquadratic for three dimensions) interpolations on a per-gridpoint basis, results in [24] indicate that the computational savings from the sparser grids may far outweigh the heavier per-gridpoint computational requirement in high-order methods. Nevertheless, high-order derivative and interpolation stencils do lead to certain complications. They may need to be closed to lower order at domain boundaries to preserve time-stability when coupled with explicit time-marching algorithms. Further, they may require smaller timesteps compared to second-order stencils for stability while holding the grid spacing fixed, though this reduction is often not too consequential. In addition, the reduction of timesteps has to be viewed within the context of solutions of PDEs. One usually sets a spatial error tolerance and strives to find a grid spacing and a stencil that delivers it most economically. Since high-order methods can deliver the same accuracy at

far coarser discretization (i.e., Δx) than second-order methods, solutions of PDEs with high-order methods tend to be performed on meshes which are less refined than those required when using second-order methods. These larger Δx's then translate to larger stable timestep sizes (Δt) when using explicit integrators. Thus, in practice, it is frequently possible to use a larger Δt when using high-order discretizations (as opposed to second-order approaches) to solve PDEs.

In this study we examine empirically high-order spatial convergence on a multiple-level SAMR mesh to identify correct derivative-interpolation-dissipation stencil sets. As a first step in that direction, we develop the expression for high-order derivative, interpolant, and filter stencils which will be used in section 3. Adaptive mesh refinement also requires a mesh adaptation criterion, but a discussion of such refinement/derefinement criteria is beyond the scope of this paper. For the purposes of this study, we will not adapt the grid in time. This simplification will allow us to investigate the interaction between various errors (introduced by interpolations and derivative stencils) in the absence of spatial discretization changes introduced by abrupt refinement/derefinement of the grid. In what follows, the Fourier analysis of finite-difference methods is used extensively. The interested reader is referred to the book by Strikwerda [32].

2.1. Derivatives. Finite-difference differentiation operators are constructed for the purposes of providing high-order derivatives as well as, potentially, adding dissipation via upwinding. In the case of arbitrary skewed stencils representing the p_Dth-order accurate approximation to the kth-derivative on a uniform stencil of width Δx, we may write the derivative at grid point i as [21]

$$\begin{aligned}(2.1)\qquad &\cdots + \beta_L f_{i-2}^{(k)} + \alpha_L f_{i-1}^{(k)} + f_i^{(k)} + \alpha_R f_{i+1}^{(k)} + \beta_R f_{i+2}^{(k)} + \cdots \\ &= \cdots + \frac{c_L f_{i-3}}{(\Delta x)^k} + \frac{b_L f_{i-2}}{(\Delta x)^k} + \frac{a_L f_{i-1}}{(\Delta x)^k} + \frac{\Upsilon f_i}{(\Delta x)^k} + \frac{a_R f_{i+1}}{(\Delta x)^k} + \frac{b_R f_{i+2}}{(\Delta x)^k} + \frac{c_R f_{i+3}}{(\Delta x)^k} + \cdots,\end{aligned}$$

where f_i is the value of the function at point i, and $f_i^{(k)}$ is the value of the kth-derivative at that point. The Greek letters α, β, ... are used to denote coefficients of the derivative terms $f_i^{(k)}$ that are one, two, ... grid points on either side of point i, with R and L subscripts denoting the right and left directions on either side of point i, respectively. We use a, b, ... similarly for the coefficients of the function terms f_j for $j = i \pm 1, i \pm 2, \ldots$, and Υ for the coefficient of f_i. The Fourier image of this discrete derivative is given by

$$(2.2)\qquad \Psi = \frac{\left\{\begin{array}{l}[\Upsilon + (a_R + a_L)\cos(\xi) + (b_R + b_L)\cos(2\xi) + (c_R + c_L)\cos(3\xi) + \cdots] \\ \quad + i\,[(a_R - a_L)\sin(\xi) + (b_R - b_L)\sin(2\xi) + (c_R - c_L)\sin(3\xi) + \cdots]\end{array}\right\}}{\left\{\begin{array}{l}[1 + (\alpha_R + \alpha_L)\cos(\xi) + (\beta_R + \beta_L)\cos(2\xi) + \cdots] \\ \quad + i\,[(\alpha_R - \alpha_L)\sin(\xi) + (\beta_R - \beta_L)\sin(2\xi) + \cdots]\end{array}\right\}},$$

where $i = \sqrt{-1}$, or, after expanding the sine and cosine functions as a Taylor series,

$$(2.3)\qquad \Psi = \psi_0 + \psi_1 \xi + \psi_2 \xi^2 + \psi_3 \xi^3 + \cdots + \psi_\infty \xi^\infty,$$

where ξ is the Fourier dual variable, and $\Psi(\xi)$ is the approximation of the derivative in Fourier space [21, 32].

For the kth-derivative, which is p_Dth-order accurate, we must now solve $(p_D + k)$ simultaneous equations; $\psi_k = (i)^k$ and $\psi_l = 0, l = 0, 1, \ldots, (p_D + k - 1), l \neq k$, in

$(p_D + k)$ unknowns; $\{\ldots, \beta_L, \alpha_L, \alpha_R, \beta_R, \ldots, c_L, b_L, a_L, \Upsilon, a_R, b_R, c_R, \ldots\}$. Nonzero values of $\{\ldots, \beta_L, \alpha_L, \alpha_R, \beta_R, \ldots\}$ imply an implicit derivative operator. While implicitness can increase the accuracy of the finite-difference operator, it does this at a significantly higher cost. Throughout this work, we consider only explicit finite-difference derivative, filters, and interpolants, but this is not required for a high-order SAMR method.

2.1.1. Centered-difference explicit derivative operators: Interior. In the interior of a computational domain, one often would like to approximate $\partial^k/\partial x^k$ to some order, p_D, in such a way to preserve the essential character of $\partial^k/\partial x^k$. For k odd, this means a dispersive operator, and for k even, a dissipative operator. To do this, one can choose finite-difference stencils that are either symmetric or antisymmetric about the point where the derivative is being sought. Specific instances of (2.2), where all implicit terms have been set to zero, are listed in Table 2.2 for k odd and Table 2.3 for k even. Orders of accuracy of 2, 4, 6, and 8 are listed, as well as the leading-order truncation error term, $\Psi - (i\xi)^k$. This term is then proportional to ξ^{k+p_D}.

TABLE 2.2

Coefficients of centered, explicit $\partial^{2m-1}/\partial x^{2m-1}$ operators for $m = 1, 2, 3$ at orders of $p_D = 2, 4, 6$, and 8 on uniform grids. All stencils are antisymmetric; $a_L = -a_R$, $b_L = -b_R$, $c_L = -c_R$, $d_L = -d_R$, $e_L = -e_R$, and $f_L = -f_R$ with $\Upsilon = 0$.

Term	p_D	a_L	b_L	c_L	d_L	e_L	f_L	Ψ
$\frac{\partial}{\partial x}$	2	$\frac{-1}{2}$	0	0	0	0	0	$i\xi - \frac{i}{6}\xi^3 + \cdots$
$\frac{\partial}{\partial x}$	4	$\frac{-8}{12}$	$\frac{1}{12}$	0	0	0	0	$i\xi - \frac{i}{30}\xi^5 + \cdots$
$\frac{\partial}{\partial x}$	6	$\frac{-45}{60}$	$\frac{9}{60}$	$\frac{-1}{60}$	0	0	0	$i\xi - \frac{i}{140}\xi^7 + \cdots$
$\frac{\partial}{\partial x}$	8	$\frac{-672}{840}$	$\frac{168}{840}$	$\frac{-32}{840}$	$\frac{3}{840}$	0	0	$i\xi - \frac{i}{630}\xi^9 + \cdots$
$\frac{\partial^3}{\partial x^3}$	2	$\frac{2}{2}$	$\frac{-1}{2}$	0	0	0	0	$-i\xi^3 + \frac{i}{4}\xi^5 + \cdots$
$\frac{\partial^3}{\partial x^3}$	4	$\frac{13}{8}$	$\frac{-8}{8}$	$\frac{1}{8}$	0	0	0	$-i\xi^3 + \frac{7i}{120}\xi^7 + \cdots$
$\frac{\partial^3}{\partial x^3}$	6	$\frac{488}{240}$	$\frac{-338}{240}$	$\frac{72}{240}$	$\frac{-7}{240}$	0	0	$-i\xi^3 + \frac{41i}{3024}\xi^9 + \cdots$
$\frac{\partial^3}{\partial x^3}$	8	$\frac{70098}{30240}$	$\frac{-52428}{30240}$	$\frac{14607}{30240}$	$\frac{-2522}{30240}$	$\frac{205}{30240}$	0	$-i\xi^3 + \frac{479i}{151200}\xi^{11} + \cdots$
$\frac{\partial^5}{\partial x^5}$	2	$\frac{-5}{2}$	$\frac{4}{2}$	$\frac{-1}{2}$	0	0	0	$i\xi^5 - \frac{i}{3}\xi^7 + \cdots$
$\frac{\partial^5}{\partial x^5}$	4	$\frac{-29}{6}$	$\frac{26}{6}$	$\frac{-9}{6}$	$\frac{1}{6}$	0	0	$i\xi^5 - \frac{13i}{144}\xi^9 + \cdots$
$\frac{\partial^5}{\partial x^5}$	6	$\frac{-1938}{288}$	$\frac{1872}{288}$	$\frac{-783}{288}$	$\frac{152}{288}$	$\frac{-13}{288}$	0	$i\xi^5 - \frac{139i}{6048}\xi^{11} + \cdots$
$\frac{\partial^5}{\partial x^5}$	8	$\frac{-99744}{12096}$	$\frac{101559}{12096}$	$\frac{-48176}{12096}$	$\frac{12500}{12096}$	$\frac{-1936}{12096}$	$\frac{139}{12096}$	$i\xi^5 - \frac{37i}{6480}\xi^{13} + \cdots$

2.1.2. Noncentered explicit first-derivative operators: Interior. In the interest of adding numerical dissipation to a numerical method, it is a common practice to skew a first-derivative, finite-difference operator such that its leading-order error term is dissipative. Stencils like this are referred to as upwind or upwinded stencils. This could be generalized beyond first-derivative operators to those representing $\partial^{2m-1}/\partial x^{2m-1}$; however, we do not attempt this here. Table 2.4 lists various values obtained from solving these equations for $m = 1$, i.e., $\partial/\partial x$ along with their leading-order truncation errors. To obtain order of accuracy p_D for the $k = 1$ derivative operator, at least $p_D + 1$ grid points are needed. One may allocate these points at the grid point where the derivative is sought and to either side of this point. The suffix U is used to denote the most minimal upwind biasing, while UU is for a stronger biasing. It is interesting to inspect these stencils further. On uniform grids, the difference operators may be considered as linear combinations of centered first-derivative differ-

TABLE 2.3
Coefficients of centered, explicit $\partial^{2m}/\partial x^{2m}$ operators for $m = 1, 2$ at orders of $p_D = 2, 4, 6$, and 8 on uniform grids. All stencils are symmetric; $a_L = a_R$, $b_L = b_R$, $c_L = c_R$, $d_L = d_R$, and $e_L = e_R$.

Term	p_D	Υ	a_L	b_L	c_L	d_L	e_L	Ψ
$\frac{\partial^2}{\partial x^2}$	2	-2	1	0	0	0	0	$-\xi^2 + \frac{1}{12}\xi^4 + \cdots$
$\frac{\partial^2}{\partial x^2}$	4	$\frac{-30}{12}$	$\frac{16}{12}$	$\frac{-1}{12}$	0	0	0	$-\xi^2 + \frac{1}{90}\xi^6 + \cdots$
$\frac{\partial^2}{\partial x^2}$	6	$\frac{-490}{180}$	$\frac{270}{180}$	$\frac{-27}{180}$	$\frac{2}{180}$	0	0	$-\xi^2 + \frac{1}{560}\xi^8 + \cdots$
$\frac{\partial^2}{\partial x^2}$	8	$\frac{-14350}{5040}$	$\frac{8064}{5040}$	$\frac{-1008}{5040}$	$\frac{128}{5040}$	$\frac{-9}{5040}$	0	$-\xi^2 + \frac{1}{3150}\xi^{10} + \cdots$
$\frac{\partial^4}{\partial x^4}$	2	6	-4	1	0	0	0	$\xi^4 - \frac{1}{6}\xi^6 + \cdots$
$\frac{\partial^4}{\partial x^4}$	4	$\frac{56}{6}$	$\frac{-39}{6}$	$\frac{12}{6}$	$\frac{-1}{6}$	0	0	$\xi^4 - \frac{7}{240}\xi^8 + \cdots$
$\frac{\partial^4}{\partial x^4}$	6	$\frac{2730}{240}$	$\frac{-1952}{240}$	$\frac{676}{240}$	$\frac{-96}{240}$	$\frac{7}{240}$	0	$\xi^4 - \frac{41}{7560}\xi^{10} + \cdots$
$\frac{\partial^4}{\partial x^4}$	8	$\frac{192654}{15120}$	$\frac{-140196}{15120}$	$\frac{52428}{15120}$	$\frac{-9738}{15120}$	$\frac{1261}{15120}$	$\frac{-82}{15120}$	$\xi^4 - \frac{479}{453600}\xi^{12} + \cdots$

TABLE 2.4
Stencil coefficients for upwinded $\partial/\partial x$ operators on uniform grids.

Name	p_D	c_L	b_L	a_L	Υ	a_R	b_R	c_R	d_R	Ψ
3U	3	0	0	$\frac{-2}{6}$	$\frac{-1}{2}$	1	$\frac{-1}{6}$	0	0	$i\xi - \frac{i}{12}\xi^4$
3UU	3	0	0	0	$\frac{-11}{6}$	3	$\frac{-3}{2}$	$\frac{1}{3}$	0	$i\xi + \frac{i}{4}\xi^6$
4U	4	0	0	$\frac{-1}{4}$	$\frac{-5}{6}$	$\frac{3}{2}$	$\frac{-1}{2}$	$\frac{1}{12}$	0	$i\xi + \frac{i}{20}\xi^5$
4UU	4	0	0	0	$\frac{-25}{12}$	4	-3	$\frac{4}{3}$	$\frac{-1}{4}$	$i\xi - \frac{i}{5}\xi^5$
5U	5	0	$\frac{1}{20}$	$\frac{-4}{8}$	$\frac{-1}{3}$	1	$\frac{-2}{8}$	$\frac{1}{30}$	0	$i\xi - \frac{i}{60}\xi^6$
5UU	5	0	0	$\frac{-1}{5}$	$\frac{-13}{12}$	2	-1	$\frac{1}{3}$	$\frac{-1}{20}$	$i\xi + \frac{i}{30}\xi^6$
6U	6	0	$\frac{1}{30}$	$\frac{-2}{5}$	$\frac{-7}{12}$	$\frac{4}{3}$	$\frac{-1}{2}$	$\frac{2}{15}$	$\frac{-1}{60}$	$i\xi + \frac{i}{105}\xi^7$
7U	7	$\frac{-1}{105}$	$\frac{1}{10}$	$\frac{-6}{10}$	$\frac{-1}{4}$	1	$\frac{-3}{10}$	$\frac{1}{15}$	$\frac{-1}{140}$	$i\xi - \frac{i}{280}\xi^8$

ence operators plus low-order approximations to higher derivatives. Singly upwinded stencils may be readily decomposed as

$$\left({}^{(1)}\Delta^{+}_{(2n-1)}\right)_i = \left({}^{(1)}\Delta^{c}_{(2n)}\right)_i - (-1)^n \frac{(n-1)!n!}{(2n)!}\left({}^{(2n)}\Delta^{c}_{(2)}\right)_i, \tag{2.4}$$

$$\left({}^{(1)}\Delta^{+}_{(2n)}\right)_i = \left({}^{(1)}\Delta^{c}_{(2n)}\right)_i + (-1)^n \frac{(n-1)!n!}{(2n)!}\left({}^{(2n+1)}\Delta^{+}_{(1)}\right)_i, \tag{2.5}$$

where Δ denotes the derivative operator, the northwest superscript is the order of the derivative, the southeast subscript is the order of the accuracy, the northeast superscript denotes centered (c), upwind ($+$), or downwind ($-$), and the subscript outside of the parentheses denotes the grid point at which the difference operator is acting. We illustrate this with an example. Consider stencil 3U in Table 2.4. By (2.4) ($n = 2$), its stencil coefficients can be obtained from those of a fourth-order centered stencil listed in Table 2.2 and by scaling those of a second-order approximation to the fourth derivative (Table 2.3) by $-1/12$. We can verify this by simply reading the stencil coefficients from the tables, performing the scaling, and subtracting similar coefficients. This is done in Table 2.5. The first row contains the fourth-order stencil for the first derivative operator from Table 2.2. The second row contains the second-order approximation for the fourth derivative (Table 2.3), scaled by $-1/12$, as per (2.4) ($n = 2$). The result, in the third row, reproduces the 3U stencil in Table 2.4.

TABLE 2.5

An example of how an upwinded stencil for a derivative (3U from Table 2.4) may be derived from a higher order centered stencil for the same operator ($\partial_x, p_D = 4$ from Table 2.2) augmented by a dissipation operator ∂_{xxxx} discretized to low order (2, in this case). The second-order discretized form of ∂_{xxxx} is scaled by $\frac{-1}{12}$ and added to the fourth-order centered stencil for ∂_x to obtain the third-order upwinded stencil for ∂_x denoted as 3U in Table 2.4.

Stencil	b_L	a_L	Υ	a_R	b_R	Table	Operator, order
$\left({}^{(1)}\Delta^c_{(4)}\right)_i$	$\frac{1}{12}$	$\frac{-8}{12}$	0	$\frac{8}{12}$	$\frac{-1}{12}$	Table 2.2	$\frac{\partial}{\partial x}, p_D = 4$
$\frac{-1}{12}\left({}^{(4)}\Delta^c_{(2)}\right)_i$	$\frac{-1}{12}$	$\frac{-4}{12}$	$\frac{6}{12}$	$\frac{-4}{12}$	$\frac{-1}{12}$	Table 2.3	$\frac{\partial^4}{\partial x^4}, p_D = 2$
$\left({}^{(1)}\Delta^+_{(3)}\right)_i$	0	$\frac{-4}{12}$	$\frac{-6}{12}$	$\frac{12}{12}$	$\frac{-2}{12}$	Table 2.4	$3U$

More elaborate upwinding stencils may be decomposed as

$$\left({}^{(1)}\Delta^{++}_{(4)}\right)_i = \left({}^{(1)}\Delta^c_{(4)}\right)_i + \frac{1}{12}\left({}^{(5)}\Delta^+_{(1)}\right)_i - \frac{1}{4}\left({}^{(5)}\Delta^+_{(1)}\right)_{i+1}, \tag{2.6}$$

$$\left({}^{(1)}\Delta^{++}_{(3)}\right)_i = \left({}^{(1)}\Delta^c_{(6)}\right)_i + \frac{1}{60}\left({}^{(4)}\Delta^c_{(2)}\right)_{i-1} - \frac{1}{12}\left({}^{(4)}\Delta^c_{(2)}\right)_i + \frac{19}{60}\left({}^{(4)}\Delta^c_{(2)}\right)_{i+1}, \tag{2.7}$$

$$\left({}^{(1)}\Delta^{++}_{(5)}\right)_i = \left({}^{(1)}\Delta^c_{(8)}\right)_i - \frac{1}{280}\left({}^{(6)}\Delta^c_{(2)}\right)_{i-1} + \frac{1}{60}\left({}^{(6)}\Delta^c_{(2)}\right)_i - \frac{13}{280}\left({}^{(6)}\Delta^c_{(2)}\right)_{i+1}. \tag{2.8}$$

It may be seen that the even-order, noncentered operators include dispersive components (e.g., (2.6); 4UU in Table 2.4). Other operators have only dissipative components but of both signs (e.g., (2.7); 3UU in Table 2.4) which can amplify errors. It thus seems prudent to consider only odd-order, singly-upwinded derivative operators to add dissipation (i.e., 3U, 5U, 7U, etc.). Note that the reason for the alternating sign in front of the dissipative term is related to the fact that $({}^{(2n)}\Delta^c_{(2)}) \approx (i\xi)^{2n} = (-1)^n\xi^{2n}$. Similarly, for dispersive terms one finds $({}^{(2n+1)}\Delta^c_{(1)}) \approx (i\xi)^{2n+1} = (-1)^n i\xi^{2n+1}$.

2.1.3. Noncentered explicit derivative operators: Boundary. In the absence of periodic boundary conditions, the computational domain must be truncated. Differentiation operators must then be designed to work at all grid points, including those on or adjacent to the boundary. As the centered interior stencils outlined above have widths of $k + p_D$ and $k + p_D - 1$ for odd and even values of k, respectively, there will be a problem obtaining derivatives at the nearest $(k + p_D - 1)/2$ and $(k + p_D)/2$ points to the wall, respectively.

Two important goals in deriving boundary stencils for finite-difference operators are to preserve the order of accuracy of the interior operator and to avoid a temporal instability. In [33], the authors have shown that if one would like to close the boundaries to a $\partial^k/\partial x^k$ operator whose interior operator has order p_D, then the boundary stencils must be at least order $p_D - k$. While it is a simple matter to derive stencils of the appropriate order at these near wall points, this does not necessarily mean that these new boundary stencils will lead to a stable overall numerical method. In the case of first-derivative operators, one can generally no longer take time-stability for granted for boundary closures above third-order accuracy. It may be speculated that this number will decrease as k increases. For first-derivative operators, there is a significant body of literature on constructing time-stable boundary closures [6, 26, 27, 31] but less so for second-derivatives [25] or higher [33].

Given the nontrivial nature of generating high-order, time-stable boundary treatments to all of the derivative operators discussed in this work, lower-order closures are used. While this lowers the formal order of accuracy of the derivative operators, it also likely allows for time-stable integration. No attempt has been made to determine the time-stability characteristics of operators for $\partial^k/\partial x^k$ at $k = 2, 3, 4, 5$. However, since the boundaries must be closed, simple closure stencils are offered. For the stencils listed, each will be given as it would be used at the left boundary. For application to the right boundary, each stencil transforms from its appearance on the left side as

(2.9)
$$f_i^{(k)} = \cdots + \frac{c_L f_{i-3}}{(\Delta x)^k} + \frac{b_L f_{i-2}}{(\Delta x)^k} + \frac{a_L f_{i-1}}{(\Delta x)^k} + \frac{\Upsilon f_i}{(\Delta x)^k} + \frac{a_R f_{i+1}}{(\Delta x)^k} + \frac{b_R f_{i+2}}{(\Delta x)^k} + \frac{c_R f_{i+3}}{(\Delta x)^k} + \cdots$$

into

(2.10)
$$f_i^{(k)} = \cdots - \frac{c_R f_{i-3}}{(\Delta x)^k} - \frac{b_R f_{i-2}}{(\Delta x)^k} - \frac{a_R f_{i-1}}{(\Delta x)^k} - \frac{\Upsilon f_i}{(\Delta x)^k} - \frac{a_L f_{i+1}}{(\Delta x)^k} - \frac{b_L f_{i+2}}{(\Delta x)^k} - \frac{c_L f_{i+3}}{(\Delta x)^k} - \cdots$$

for $\partial^k/\partial x^k$ at $k = 1, 3, 5, \ldots$, and from

(2.11)
$$f_i^{(k)} = \cdots + \frac{c_L f_{i-3}}{(\Delta x)^k} + \frac{b_L f_{i-2}}{(\Delta x)^k} + \frac{a_L f_{i-1}}{(\Delta x)^k} + \frac{\Upsilon f_i}{(\Delta x)^k} + \frac{a_R f_{i+1}}{(\Delta x)^k} + \frac{b_R f_{i+2}}{(\Delta x)^k} + \frac{c_R f_{i+3}}{(\Delta x)^k} + \cdots$$

into

(2.12)
$$f_i^{(k)} = \cdots + \frac{c_R f_{i-3}}{(\Delta x)^k} + \frac{b_R f_{i-2}}{(\Delta x)^k} + \frac{a_R f_{i-1}}{(\Delta x)^k} + \frac{\Upsilon f_i}{(\Delta x)^k} + \frac{a_L f_{i+1}}{(\Delta x)^k} + \frac{b_L f_{i+2}}{(\Delta x)^k} + \frac{c_L f_{i+3}}{(\Delta x)^k} + \cdots$$

for $\partial^k/\partial x^k$ at $k = 2, 4, \ldots$.

To close the boundaries for the first- through fifth-derivative operators when insufficient grid points are available for centered-difference operators, Table 2.6 lists lower-order closures. Expressions for the third and fourth derivatives are used in section 3 to investigate the KdV and KS equations [28]; fifth derivatives could be used to perform a similar investigation for the Kawahara equation [28].

We note that the method developed above to generate centered stencils and their skewed boundary closures is readily generalized to arbitrary orders.

2.2. Filters. Finite-difference filters are a useful adjunct to a numerical method lacking spectral accuracy [21, 22]. In a non-AMR context, they are used simply to remove high-wavenumber spatial information that is not resolvable by the numerical method. However, in an AMR context, they may additionally be used to ensure that the high-order interpolants do not encounter the Runge phenomenon. Several criteria exist for a useful filter. Low-wavenumber information that is resolved should be virtually untouched, while the relatively unresolved high-wavenumber information should be removed. This constitutes dissipation. Dispersion, which is often introduced by the filter boundary closures, is to be avoided if possible. The filter user must estimate which wavenumbers are resolvable by their numerical methods and choose an appropriate filter profile.

TABLE 2.6
Coefficients of $\partial^k/\partial x^k$ operators near left boundary points for uniform grids.

Term	p_D	c_L	b_L	a_L	Υ	a_R	b_R	c_R	d_R	e_R	f_R	g_R	Ψ
$\frac{\partial}{\partial x}$	1	0	0	0	-1	1	0	0	0	0	0	0	$\xi - \frac{1}{2}\xi^2$
$\frac{\partial}{\partial x}$	3	0	0	0	$\frac{-11}{6}$	3	$\frac{-3}{2}$	$\frac{1}{3}$	0	0	0	0	$\xi + \frac{1}{4}\xi^4$
$\frac{\partial}{\partial x}$	3	0	0	$\frac{-1}{3}$	$\frac{-1}{2}$	1	$\frac{-1}{6}$	0	0	0	0	0	$\xi - \frac{1}{12}\xi^4$
$\frac{\partial^2}{\partial x^2}$	1	0	0	0	1	-2	1	0	0	0	0	0	$-\xi^2 - \xi^3$
$\frac{\partial^2}{\partial x^2}$	3	0	0	0	$\frac{35}{12}$	$\frac{-26}{3}$	$\frac{19}{2}$	$\frac{-14}{3}$	$\frac{11}{12}$	0	0	0	$-\xi^2 + \frac{5}{6}\xi^5$
$\frac{\partial^2}{\partial x^2}$	3	0	0	$\frac{11}{12}$	$\frac{-5}{3}$	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{-1}{12}$	0	0	0	0	$-\xi^2 - \frac{1}{12}\xi^5$
$\frac{\partial^3}{\partial x^3}$	1	0	0	0	-1	3	-3	1	0	0	0	0	$-i\xi^3 + \frac{3}{2}\xi^4$
$\frac{\partial^3}{\partial x^3}$	1	0	0	-1	3	-3	1	0	0	0	0	0	$-i\xi^3 + \frac{1}{2}\xi^4$
$\frac{\partial^3}{\partial x^3}$	3	0	0	0	$\frac{-17}{4}$	$\frac{71}{4}$	$\frac{-118}{4}$	$\frac{98}{4}$	$\frac{-41}{4}$	$\frac{7}{4}$	0	0	$-i\xi^3 - \frac{15}{8}\xi^6$
$\frac{\partial^3}{\partial x^3}$	3	0	0	$\frac{-7}{4}$	$\frac{25}{4}$	$\frac{-34}{4}$	$\frac{22}{4}$	$\frac{-7}{4}$	$\frac{1}{4}$	0	0	0	$-i\xi^3 - \frac{1}{8}\xi^6$
$\frac{\partial^3}{\partial x^3}$	3	0	$\frac{-1}{4}$	$\frac{-1}{4}$	$\frac{10}{4}$	$\frac{-14}{4}$	$\frac{7}{4}$	$\frac{-1}{4}$	0	0	0	0	$-i\xi^3 + \frac{1}{8}\xi^6$
$\frac{\partial^4}{\partial x^4}$	1	0	0	0	1	-4	6	-4	1	0	0	0	$\xi^4 + 2i\xi^5$
$\frac{\partial^4}{\partial x^4}$	1	0	0	1	-4	6	-4	1	0	0	0	0	$\xi^4 + i\xi^5$
$\frac{\partial^4}{\partial x^4}$	3	0	0	0	$\frac{35}{6}$	$\frac{-186}{6}$	$\frac{411}{6}$	$\frac{-484}{6}$	$\frac{321}{6}$	$\frac{-114}{6}$	$\frac{17}{6}$	0	$\xi^4 - \frac{7i}{2}\xi^7$
$\frac{\partial^4}{\partial x^4}$	3	0	0	$\frac{17}{6}$	$\frac{-84}{6}$	$\frac{171}{6}$	$\frac{-184}{6}$	$\frac{111}{6}$	$\frac{-36}{6}$	$\frac{5}{6}$	0	0	$\xi^4 - \frac{2i}{3}\xi^7$
$\frac{\partial^4}{\partial x^4}$	3	0	$\frac{5}{6}$	$\frac{-18}{6}$	$\frac{21}{6}$	$\frac{-4}{6}$	$\frac{-9}{6}$	$\frac{6}{6}$	$\frac{-1}{6}$	0	0	0	$\xi^4 - \frac{i}{6}\xi^7$
$\frac{\partial^5}{\partial x^5}$	1	0	0	0	-1	5	-10	10	-5	1	0	0	$i\xi^5 - \frac{5}{2}\xi^6$
$\frac{\partial^5}{\partial x^5}$	1	0	0	-1	5	-10	10	-5	1	0	0	0	$i\xi^5 - \frac{3}{2}\xi^6$
$\frac{\partial^5}{\partial x^5}$	1	0	-1	5	-10	10	-5	1	0	0	0	0	$i\xi^5 - \frac{1}{2}\xi^6$
$\frac{\partial^5}{\partial x^5}$	3	0	0	0	$\frac{-46}{6}$	$\frac{295}{6}$	$\frac{-810}{6}$	$\frac{1235}{6}$	$\frac{-1130}{6}$	$\frac{621}{6}$	$\frac{-190}{6}$	$\frac{25}{6}$	$i\xi^5 + \frac{35}{6}\xi^8$
$\frac{\partial^5}{\partial x^5}$	3	0	0	$\frac{-25}{6}$	$\frac{154}{6}$	$\frac{-405}{6}$	$\frac{590}{6}$	$\frac{-515}{6}$	$\frac{270}{6}$	$\frac{-79}{6}$	$\frac{10}{6}$	0	$i\xi^5 + \frac{5}{3}\xi^8$
$\frac{\partial^5}{\partial x^5}$	3	0	$\frac{-10}{6}$	$\frac{55}{6}$	$\frac{-126}{6}$	$\frac{155}{6}$	$\frac{-110}{6}$	$\frac{45}{6}$	$\frac{-10}{6}$	$\frac{1}{6}$	0	0	$i\xi^5 + \frac{11}{144}\xi^8$
$\frac{\partial^5}{\partial x^5}$	3	$\frac{-1}{6}$	$\frac{-2}{6}$	$\frac{27}{6}$	$\frac{-70}{6}$	$\frac{85}{6}$	$\frac{-54}{6}$	$\frac{17}{6}$	$\frac{-2}{6}$	0	0	0	$i\xi^5 - \frac{1}{6}\xi^8$

TABLE 2.7
Stencil coefficients for the centered $\partial^{2n}/\partial x^{2n}$ operators on uniform grids given in (2.13). *These are used to create the dissipation matrix* $\mathbf{D}$.

Operator	Υ	a	b	c	d	e	f	g	Ψ
$\partial^2/\partial x^2$	-2	1	0	0	0	0	0	0	$-\xi^2 + \frac{\xi^4}{12} + \cdots$
$\partial^4/\partial x^4$	6	-4	1	0	0	0	0	0	$\xi^4 - \frac{2\xi^6}{12} + \cdots$
$\partial^6/\partial x^6$	-20	15	-6	1	0	0	0	0	$-\xi^6 + \frac{3\xi^8}{12} + \cdots$
$\partial^8/\partial x^8$	70	-56	28	-8	1	0	0	0	$\xi^8 - \frac{4\xi^{10}}{12} + \cdots$
$\partial^{10}/\partial x^{10}$	-252	210	-120	45	-10	1	0	0	$-\xi^{10} + \frac{5\xi^{12}}{12} + \cdots$
$\partial^{12}/\partial x^{12}$	924	-792	495	-220	66	-12	1	0	$\xi^{12} - \frac{6\xi^{14}}{12} + \cdots$
$\partial^{14}/\partial x^{14}$	-3432	3003	-2002	1001	-364	91	-14	1	$-\xi^{14} + \frac{7\xi^{16}}{12} + \cdots$

As a filtering function in Fourier space, we seek a function that is equal to 0 at $\xi = \pi$ and is equal to 1 at $\xi = 0$. A simple function to satisfy this is $\left[1 - \sin^{2n}(\xi/2)\right]$. To create this Fourier image using centered explicit finite-difference stencils, one may utilize scaled values of the symmetric $(2n)$th-order derivative for a function f,

$$f_i^{(2n)} = \frac{\Upsilon f_i}{(\Delta x)^{(2n)}} + a\frac{f_{i+1} + f_{i-1}}{(\Delta x)^{(2n)}} + b\frac{f_{i+2} + f_{i-2}}{(\Delta x)^{(2n)}} + c\frac{f_{i+3} + f_{i-3}}{(\Delta x)^{(2n)}} + \cdots, \tag{2.13}$$

where $a = a_L = a_R$, $b = b_L = b_R$, ..., etc. with $\Psi = \Upsilon + 2a\cos(\xi) + 2b\cos(2\xi) + 2c\cos(3\xi) + \cdots$. Solving for the stencil coefficients for the second-order accurate version of this operator, $\partial^{2n} f/\partial x^{2n}$, i.e., $\Psi = (-1)^n \xi^{2n} + O(\xi^{2n+2})$, the Fourier image is given by $\Psi = (-4)^n \sin^{2n}(\xi/2)$. Hence, the forms of Ψ for the desired filters and $f^{(2n)}$ are

$$\Psi_{f^{(2n)}} = (-4)^n \sin^{2n}(\xi/2), \qquad \Psi_{\text{filter}} = 1 - \sin^{2n}(\xi/2), \tag{2.14}$$

and the filter function may then be expressed using the derivative operator as

$$\Psi_{\text{filter}} = 1 + (-1)^{n+1}\left(\frac{1}{2}\right)^{2n} \Psi_{f^{(2n)}}. \tag{2.15}$$

Actual stencil coefficients for the $\partial^{2n}/\partial x^{2n}$ operator, up to fourteenth order, are given in Table 2.7. Alternatively, one could develop these second-order accurate stencils for arbitrary n using Pascal's triangle. Interior stencil coefficients are quickly generated with this small Mathematica code:

```
FiltCoeff = Join[{ups},CharacterRange["a", "z"]];
Do[ Do[ Print[Part[FiltCoeff,(ii-nn+1)]," = ",
  ((-1)^(ii)*(2*nn)!)/((2*nn-ii)!*ii!)]; ,{ii,nn,2*nn}];
Print[""];,{nn,1,26}];
```

The user chooses `nn` as an integer between 1 and 26. It may be determined that the leading-order truncation error is $(n/12)\xi^{2n+2}$. These symmetric stencil coefficients are the interior elements of a dissipation matrix, $\mathbf{D}$, the discretized form of $\partial^{2n}/\partial x^{2n}$ scaled by $(\Delta x)^{2n}$, as applied to the entire solution vector U. If the filters are applied

over periodic domains, no one-sided boundary stencils need to be included in the dissipation matrix. Stencils of order $2n$ give rise to a filter of order $p_F = 2n$. By a slight rescaling of the dissipation matrix, the filtered value of U may be seen to be

$$\hat{U} = (\mathbf{I} + (-1)^{n+1} 2^{-2n} \mathbf{D}) U = \mathbf{F} U, \tag{2.16}$$

where $\hat{U}$ is the filtered vector and $\mathbf{F}$ is the complete finite-difference filter.

One essential ingredient to have in a filter is strictly real eigenvalues of $\mathbf{F}$. Practically, this requires that $\mathbf{F}$ and $\mathbf{D}$ be symmetric. Imaginary eigenvalues will confer a dispersive element to the filter. At the boundaries, where $\partial^{2n}/\partial x^{2n}$ cannot be approximated using symmetric stencils, one must use skewed stencils. To maintain the overall symmetry of $\mathbf{F}$ and $\mathbf{D}$, one may implement stencils representing $\partial^n/\partial x^n$ rather than $\partial^{2n}/\partial x^{2n}$. For any given order filter (the interior operator), $p_F = 2n$, the interior and boundary operators are given by

$$\Psi_{\mathrm{F,Interior}} = (i\xi)^{2n} + O(\xi)^{2n+2} \qquad \Psi_{\mathrm{F,Boundary}} = (i\xi)^n + O(\xi)^{n+1}. \tag{2.17}$$

Boundary blocks are included, if necessary, in the upper-left and lower-right portions of $\mathbf{D}$ with dimension n^2. These boundary blocks reduce the order of the filter at boundary points and can potentially diminish the overall spatial order of the numerical method. Filter boundary coefficients and other details may be found in the literature [21, 22].

A degree of care is required during simulations where filters are applied. For example, in an explicit Runge–Kutta time-integration method, filtering the stage value will destroy the temporal accuracy of the scheme because of the low stage-order of these methods. Therefore, filtering can be applied only to the step value of the solution vector. Further, filtering changes the solution by removing the high-wavenumber spectrum of the solution; thus high-order filters are to be preferred since their effective cut-off wavenumber is high, and small amounts of energy are removed from the variables. If the amount of energy being removed from the solution vector is nontrivial, then users should re-evaluate their numerical method along with the spatial and temporal grids. High-order filters have wide stencils that may pose a practical difficulty in adaptive mesh schemes since this requires the patches to be at least as wide as the filter stencil. In our studies, we impose a "halo" of grid points n wide around each patch. This ensures that all the "valid" grid points on the patch (i.e., the grid points whose refinement is dictated by the requirements of the physics being simulated) are considered "interior" points by the filter matrix. This is not satisfied at the domain boundaries, and fine patches abutting the domain boundary incur filter errors of $O\left((\Delta x)^{n+1}\right)$ at certain points. However, if n is chosen appropriately, this does not adversely affect the desired convergence rate of the overall spatial discretization.

2.3. Hyperviscosity. While not used here, we present the following brief comments on hyperviscosity. Beyond using upwinding to apply high-order dissipation during construction of the RHS of the governing equations or applying stand-alone filters to remove high wavenumbers from the U-vector itself, one can add a hyperviscosity term to the RHS to both dealias the numerical method and to forestall Runge phenomena. The term would appear as $\mu_H(\nabla^2)^n U$ and may be readily constructed using the symmetric dissipation matrices, $\mathbf{D}^{2n}$, used in the filters. To avoid interfering with the order of the overall method, p, it is necessary to use $2n = p_H < p$ in periodic directions and $n < p$ in aperiodic directions.

One may then write $\nabla^{2n}U = (\mathbf{D}_x^{(2)} + \mathbf{D}_y^{(2)} + \mathbf{D}_z^{(2)})^n U$. It is not clear that, for the purposes in mind, this much effort needs to be expended. One could effectively assume that dissipation matrices commute and reduce workload. This would transform

$$\nabla^4 U = \Big(\mathbf{D}_x^{(4)} + \mathbf{D}_y^{(4)} + \mathbf{D}_z^{(4)} + \mathbf{D}_x^{(2)}\mathbf{D}_y^{(2)} + \mathbf{D}_x^{(2)}\mathbf{D}_z^{(2)} + \mathbf{D}_y^{(2)}\mathbf{D}_x^{(2)} + \mathbf{D}_y^{(2)}\mathbf{D}_z^{(2)} + \mathbf{D}_z^{(2)}\mathbf{D}_x^{(2)} + \mathbf{D}_z^{(2)}\mathbf{D}_y^{(2)}\Big)U$$

into

$$\nabla^4 U \approx \left(\mathbf{D}_x^{(4)} + \mathbf{D}_y^{(4)} + \mathbf{D}_z^{(4)} + 2\mathbf{D}_x^{(2)}\mathbf{D}_y^{(2)} + 2\mathbf{D}_y^{(2)}\mathbf{D}_z^{(2)} + 2\mathbf{D}_z^{(2)}\mathbf{D}_x^{(2)}\right) U.$$

To further reduce work, one could ignore cross-terms and simply approximate $\nabla^{2n}U \approx (\mathbf{D}_x^{(2n)} + \mathbf{D}_y^{(2n)} + \mathbf{D}_z^{(2n)})U$. The remaining difficulty is to select an appropriate value for μ_H. It should be remembered that the intent of this construct is to provide a high-order dissipation term to the governing equations. This is why the seemingly inconsistent removal of noncommutativity, cross-terms, and inappropriate boundary closures is immaterial to the purpose at hand.

The use of hyperviscous terms in explicit time-marched schemes can be problematic. Stable integration of the governing equations requires that all eigenvalues from the RHS of the Navier–Stokes equations times the timestep fit with the linear stability envelope of the explicit method. Eigenvalues associated with convection and diffusion scale with the grid spacing as $(\Delta x)^{-1}$ and $(\Delta x)^{-2}$, respectively. If hyperviscosity is included, then new eigenvalues will scale as $(\Delta x)^{-2n}$. This is quite likely to force unacceptably small time steps. While this could be fixed by using an implicit integration method, the procedure would be extremely computationally expensive. Thus, while hyperviscosity appears to be an elegant means of controlling high-wavenumber oscillations, practical considerations make filters more attractive.

2.4. Interpolants. Interpolations, in a vertex-centered AMR context, are used in two instances: (1) to initialize a fine patch (from coarse mesh data) when it is first created, and (2) to fill up the coarse-fine interface or "halo" that is kept around all patches. This "halo" enables the use of symmetric stencils for differentiation and filter operators uniformly on all patches, except near the domain boundaries. As the order of the filters and differentiation operators are increased, the width of the "halo" is increased accordingly. The solution in this halo region is not calculated using the equations being solved; rather they are interpolated from the underlying coarse mesh. Since the halo is additive, i.e., it is added around the "valid" points whose refinement is dictated by resolution requirements, these interpolations are obtained from coarse grid points where the solution is deemed sufficiently resolved.

Interpolation requirements for structured AMR using a vertex-centered grid arrangement include as many as seven distinct finite-difference subinterpolations for up to three Cartesian coordinate directions. All necessary coarse-to-fine interpolations may be accomplished by using one-dimensional interpolants in the x-, y-, and z-directions, two-dimensional interpolants in the xy-, yz-, and zx-directions, and three-dimensional interpolants in the xyz-direction. Two-dimensional simulations require three, say, x-, y-, and xy-interpolations. Three-dimensional simulations require all seven to interpolate the $3^3 - 2^3 = 19$ points inside a unit cube that are unique to the fine mesh. Twelve interpolations require one-dimensional interpolants (the triangles in Figure 2.1), six require two-dimensional interpolants (the squares in Figure 2.1), and one requires a three-dimensional interpolant (the small circle).

TABLE 2.8
Coefficients of one-dimensional, vertex-centered, midpoint interpolants in terms of ξ.

Stencil	2^{nd}-order	4^{th}-order	6^{th}-order	8^{th}-order	10^{th}-order
a	1/2	9/16	150/256	1225/2048	39690/65536
b	0	−1/16	−25/256	−245/2048	−8820/65536
c	0	0	3/256	49/2048	2268/65536
d	0	0	0	−5/2048	−405/65536
e	0	0	0	0	35/65536
$\psi_{p_I}\xi^{p_I}$	$-(1/8)\xi^2$	$-(3/128)\xi^4$	$-(5/1024)\xi^6$	$-(35/32768)\xi^8$	$-(63/262144)\xi^{10}$

Since all $2^3 = 8$ coarse grid points (the large circles in Figure 2.1) are also fine grids points, interpolation from fine-to-coarse is merely an injection and requires no special finite difference stencils. Coarse-to-fine interpolations, however, do need specific stencils. Each of these stencils, whether it be for one, two, or three dimensions, interpolates from the coarse grid data to its geometric midpoint where the fine grid point lies. Although implicitness in finite-difference derivative and interpolant operators provides better high-wavenumber resolution, their added cost makes them ultimately less efficient than explicit methods unless their implicit solves can be made quite economical. Hence, all interpolant operators considered here will be explicit.

Two concerns that should be kept in mind while performing interpolations are, first, what order of accuracy is needed at boundaries to preserve the order of an interior interpolant of order p_I, and second, are there time-stability issues with various interpolant boundary closures. We proceed by closing the boundary points to the same order as the interior and assume that any time-stability issues that may exist with interpolant boundary closures are secondary to more immediate problems.

2.4.1. One dimension: Interior. The stencils in the interior of the domain can symmetrically utilize equal numbers of grid points on either side of the point being interpolated. Hence, the stencil coefficients are identical on each side,

$$f_i^{(0)} = a(f_{i+1/2} + f_{i-1/2}) + b(f_{i+3/2} + f_{i-3/2}) + c(f_{i+5/2} + f_{i-5/2}) + \cdots,$$

and give its Fourier image by

$$\Psi = 2a\cos(\xi/2) + 2b\cos(3\xi/2) + 2c\cos(5\xi/2) + \cdots = \sum_{l=0}^{\infty} \psi_l \xi^l. \tag{2.18}$$

To achieve order p_I, we require $\psi_0 = 1$ and $\psi_l = 0$, with $l = 1, 2, \ldots, (p_I - 1)$. Symmetric interior stencils are listed in Table 2.8 along with their leading-order truncation errors. It will be useful to also cast these stencils in Fourier space as

$$\Psi_I^{2(x)} = \cos\left(\frac{\xi}{2}\right), \tag{2.19}$$

$$\Psi_I^{4(x)} = \frac{-1}{4}\cos\left(\frac{\xi}{2}\right)[-5 + \cos(\xi)], \tag{2.20}$$

$$\Psi_I^{6(x)} = \frac{+1}{64}\cos\left(\frac{\xi}{2}\right)[89 - 28\cos(\xi) + 3\cos(2\xi)], \tag{2.21}$$

$$\Psi_I^{8(x)} = \frac{-1}{512}\cos\left(\frac{\xi}{2}\right)[-762 + 299\cos(\xi) - 54\cos(2\xi) + 5\cos(3\xi)], \tag{2.22}$$

$$\Psi_I^{10(x)} = \frac{+1}{16384}\cos\left(\frac{\xi}{2}\right)[+25609 - 11528\cos(\xi) + 2708\cos(2\xi) - 440\cos(3\xi) + 35\cos(4\xi)], \tag{2.23}$$

for the x-direction. In terms of the original stencil coefficients, the stencil is also expressible as

$$\Psi_I^{2n(x)} = 2\cos(\xi/2)[c_0 + c_1\cos(\xi) + c_2\cos(2\xi) + c_3\cos(3\xi) + c_4\cos(4\xi) + c_5\cos(5\xi) + \cdots], \tag{2.24}$$

where

$$\begin{array}{rcl} c_0 &=& a - b + c - d + e - f + g - \cdots, \\ c_1 &=& 2(b - c + d - e + f - g + \cdots), \\ c_2 &=& 2(c - d + e - f + g - \cdots), \\ c_3 &=& 2(d - e + f - g + \cdots), \\ c_4 &=& 2(e - f + g - \cdots), \\ c_5 &=& 2(f - g + \cdots). \end{array} \tag{2.25}$$

In the y- or z-directions, these expressions are modified by simply replacing ξ with $\eta(y)$ or $\zeta(z)$.

2.4.2. One dimension: Boundaries. As boundaries are approached, interpolant stencils must become asymmetrical if higher order is desired. Stencils are now considered as

$$f_i^{(0)} = \cdots + Cf_{i-5/2} + Bf_{i-3/2} + Af_{i-1/2} + af_{i+1/2} + bf_{i+3/2} + cf_{i+5/2} + \cdots$$

with a Fourier image given by

$$\begin{aligned} \Psi &= \left\{ \begin{array}{c} [(a+A)\ \cos(\xi/2)\ +\ (b+B)\ \cos(3\xi/2)\ +\ (c+C)\ \cos(5\xi/2)\ +\ \cdots] + \\ i\,[(a-A)\ \sin(\xi/2)\ +\ (b-B)\ \sin(3\xi/2)\ +\ (c-C)\ \sin(5\xi/2)\ +\ \cdots] \end{array} \right\} \\ &= \sum_{l=0}^{\infty} \psi_l \xi^l. \end{aligned} \tag{2.26}$$

Again, to achieve order p_I, we require $\psi_0 = 1$ and $\psi_l = 0, l = 1, 2, \ldots, (p_I - 1)$. As the stencils are skewed, we use $\Psi_{\mathsf{a}}^{4(x)}$ to denote the fourth-order interpolant in the x-direction that is shifted one grid point to the right; i.e., the nonzero stencil coefficients are $\{A, a, b, c\}$. Shifting two grid points to the right at sixth order produces $\Psi_{\mathsf{b}}^{6(x)}$ with a stencil determined by $\{A, a, b, c, d, e\}$. To fully close the interpolant boundaries at orders 4, 6, 8, and 10, the stencils as computed from the Mathematica script provided in Appendix A are required. In the y- or z-direction, these expressions are modified by simply replacing ξ with either η or ζ. For stencils shifted to the left rather than the right, the Fourier image expressions should have each occurrence of I replaced with $-I$, and the lowercase letter used to denote the degree of skewness should be replaced with the uppercase version of that letter.

2.4.3. Two dimensions: Interior. Before discussing multidimensional stencils, it will be useful to develop a reasonably clear and compact notation in which to convey both stencils and their coefficients. Let

$$\{\Gamma_\alpha | \alpha = -p_I^*, -p_I^*+1, \ldots, p_I^*-1, p_I^*\} = \{\ldots, C, B, A, a, b, c, \ldots\}$$

be the stencil coefficients associated with grid points $\{\ldots, i-(5/2), i-(3/2), i-(1/2), i+(1/2), i+(3/2), i+(5/2), \ldots\}$. Using this, the stencil coefficient associated with the point $(i-(1/2), j+(1/2))$ would be Aa. With this, the preceding symmetric, one-dimensional stencil may be written as

(2.27)

$$\begin{aligned} f_i^{(0)} &= \sum_{\alpha=-p_I^*}^{p_I^*} \Gamma_\alpha f_{i+\alpha} \\ &= \cdots + C f_{i-5/2} + B f_{i-3/2} + A f_{i-1/2} + a f_{i+1/2} + b f_{i+3/2} + c f_{i+5/2} + \cdots , \end{aligned}$$

where $p_I^* = (p_I - 1)/2$. For a two-dimensional stencil, we have

(2.28)

$$\begin{aligned} f_{i,j}^{(0)} &= \sum_{\alpha=-p_I^*}^{p_I^*} \sum_{\beta=-p_I^*}^{p_I^*} \Gamma_\alpha \Gamma_\beta f_{i+\alpha, j+\beta} \\ &= \cdots + AA\, f_{i-1/2,j-1/2} + Aa\, f_{i-1/2,j+1/2} + aA\, f_{i+1/2,j-1/2} + aa\, f_{i+1/2,j+1/2} + \cdots \end{aligned}$$

and for its Fourier image, we have

$$\Psi = \sum_{\alpha=-p_I^*}^{p_I^*} \sum_{\beta=-p_I^*}^{p_I^*} \Gamma_\alpha \Gamma_\beta \exp(i\alpha\xi) \exp(i\beta\eta) \;=\; \sum_{l=0}^{\infty} \sum_{m=0}^{\infty} \psi_{l,m} \xi^l \eta^m, \tag{2.29}$$

where we require $\psi_{0,0} = 1$ and $\psi_{l,m} = 0$, with $l, m = 1, 2, \ldots, (p_I - 1)$ to achieve order p_I. Symmetry reduces the number of independent stencil coefficients from p_I^2 to $p_I(p_I+2)/8$. For instance, certain off-diagonal stencil coefficients satisfy the relations $ce = cE = Ce = CE = ec = Ec = eC = EC$, while diagonal coefficients satisfy $cc = cC = Cc = CC$. These interior stencils may be cast in Fourier space quite simply as

$$\Psi_I^{p_I(xy)} = \Psi_I^{p_I(x)} \Psi_I^{p_I(y)}, \tag{2.30}$$

where the one-dimensional stencils are given in (2.19)–(2.23). Values for the unique stencil coefficients are listed in Table 2.9 for even orders from 2 to 10.

2.4.4. Two Dimensions: Boundaries. Boundary stencils to two-dimensional interpolants may be constructed in Fourier space by simply multiplying together the proper one-dimensional interpolants. At orders $p_I = \{2, 4, 6, 8, 10\}$, one needs to close the interior method with $\{0, 2, 5, 9, 14\} = p_I(p_I+2)/8 - 1$ unique boundary stencils if one wishes to preserve overall order. At fourth order, when x and y are chosen as the two directions, these are given by

$$\Psi_{I,\mathsf{a}}^{4(xy)} = \Psi_I^{4(x)} \Psi_{\mathsf{a}}^{4(y)}, \qquad \Psi_{\mathsf{a},\mathsf{a}}^{4(xy)} = \Psi_{\mathsf{a}}^{4(x)} \Psi_{\mathsf{a}}^{4(y)}. \tag{2.31}$$

Note that the other stencils, $\Psi_{\mathsf{A},\mathsf{A}}^{4(xy)}$, $\Psi_{\mathsf{a},\mathsf{A}}^{4(xy)}$, and $\Psi_{\mathsf{a},\mathsf{A}}^{4(xy)}$, are simply rotations of $\Psi_{\mathsf{a},\mathsf{a}}^{4(xy)}$, while $\Psi_{\mathsf{a},I}^{4(xy)}$, $\Psi_{\mathsf{A},I}^{4(xy)}$, and $\Psi_{I,\mathsf{A}}^{4(xy)}$ are rotations of $\Psi_{I,\mathsf{a}}^{4(xy)}$. Above fourth order, all needed

TABLE 2.9
Coefficients and truncation errors of two-dimensional, vertex-centered interpolants in terms of ξ and η.

Stencil	2^{nd}-order	4^{th}-order	6^{th}-order	8^{th}-order	10^{th}-order
aa	1/4	81/256	22500/65536	$1500625/2^{22}$	$1575296100/2^{32}$
ba	0	−9/256	−3750/65536	$-300125/2^{22}$	$-350065800/2^{32}$
bb	0	+1/256	625/65536	$60025/2^{22}$	$77792400/2^{32}$
ca	0	0	450/65536	$60025/2^{22}$	$90016920/2^{32}$
cb	0	0	−75/65536	$-12005/2^{22}$	$-20003760/2^{32}$
cc	0	0	9/65536	$2401/2^{22}$	$5143824/2^{32}$
da	0	0	0	$-6125/2^{22}$	$-16074450/2^{32}$
db	0	0	0	$1225/2^{22}$	$3572100/2^{32}$
dc	0	0	0	$-245/2^{22}$	$-918540/2^{32}$
dd	0	0	0	$25/2^{22}$	$164025/2^{32}$
ea	0	0	0	0	$1389150/2^{32}$
eb	0	0	0	0	$-308700/2^{32}$
ec	0	0	0	0	$79380/2^{32}$
ed	0	0	0	0	$-14175/2^{32}$
ee	0	0	0	0	$1225/2^{32}$
$\psi_{p_I,p_I}\xi^{p_I}\eta^{p_I}$	$-(1/8)\xi^2$ $-(1/8)\eta^2$	$-(3/128)\xi^4$ $-(3/128)\eta^4$	$-(5/1024)\xi^6$ $-(5/1024)\eta^6$	$-(35/32768)\xi^8$ $-(35/32768)\eta^8$	$-(63/262144)\xi^{10}$ $-(63/262144)\eta^{10}$

stencils may be created in a similar manner by considering all permutations of the relevant one-dimensional stencils, i.e.,

$$\Psi^{2n(xy)}_{\alpha,\beta} = \Psi^{2n(x)}_{\alpha}\Psi^{2n(y)}_{\beta}, \tag{2.32}$$

where α and β each represent one of the following subscripts: I, a, b, c, ... The principal difference at higher order is that there appear additional, different boundary stencils. For instance, at sixth order the unique stencils consist of $\Psi^{6(xy)}_{I,\mathsf{a}}$, $\Psi^{6(xy)}_{I,\mathsf{b}}$, $\Psi^{6(xy)}_{\mathsf{a},\mathsf{a}}$, $\Psi^{6(xy)}_{\mathsf{a},\mathsf{b}}$, and $\Psi^{6(xy)}_{\mathsf{b},\mathsf{b}}$. At eighth order, one must add "c" stencils.

2.4.5. Three dimensions: Interior. Similar to the derivation of two-dimensional stencils, for three dimensions we use

$$f^{(0)}_{i,j,k} = \sum_{\alpha=-p^*_I}^{p^*_I}\sum_{\beta=-p^*_I}^{p^*_I}\sum_{\gamma=-p^*_I}^{p^*_I}\Gamma_\alpha\Gamma_\beta\Gamma_\gamma f_{i+\alpha,j+\beta,k+\gamma}, \tag{2.33}$$

along with its Fourier image,

$$\Psi = \sum_{\alpha=-p^*_I}^{p^*_I}\sum_{\beta=-p^*_I}^{p^*_I}\sum_{\gamma=-p^*_I}^{p^*_I}\Gamma_\alpha\Gamma_\beta\Gamma_\gamma\exp(i\alpha\xi)\exp(i\beta\eta)\exp(i\gamma\zeta) = \sum_{l=0}^{\infty}\sum_{m=0}^{\infty}\sum_{n=0}^{\infty}\psi_{l,m,n}\xi^l\eta^m\zeta^n, \tag{2.34}$$

where we require $\psi_{0,0,0} = 1$ and $\psi_{l,m,n} = 0, l, m, n = 1, 2, \ldots, (p_I - 1)$ to achieve order p_I. Stencil coefficients may be computed by either solving for the many conditions placed on $\psi_{l,m,n}$ or by constructing the stencils in Fourier space and then converting the trigonometric functions to exponential functions using

$$\Psi^{p_I(xyz)}_I = \Psi^{p_I(x)}_I\Psi^{p_I(y)}_I\Psi^{p_I(z)}_I. \tag{2.35}$$

The $p_I(p_I+2)(p_I+4)/48$ unique stencil coefficient values are listed in Table B.1 in the appendix for orders $p_I = \{2, 4, 6, 8, 10\}$.

2.4.6. Three dimensions: Boundaries. Boundary stencils to three-dimensional interpolants may be constructed, by a slight extension of the two-dimensional results, by simply multiplying together the proper three one-dimensional interpolants rather than two. At orders $\{2, 4, 6, 8, 10\}$ one needs to close the interior method with $\{0, 3, 9, 19, 34\} = p_I(p_I + 2)(p_I + 4)/48 - 1$ unique boundary stencils if one wishes to preserve overall order, where each stencil has p_I^3 stencil coefficients that may or may not be unique. At fourth order the unique stencils are given by

(2.36)
$$\Psi^{4(xyz)}_{I,I,\mathsf{a}} = \Psi^{4(x)}_{I}\Psi^{4(y)}_{I}\Psi^{4(z)}_{\mathsf{a}}, \quad \Psi^{4(xyz)}_{I,\mathsf{a},\mathsf{a}} = \Psi^{4(x)}_{I}\Psi^{4(y)}_{\mathsf{a}}\Psi^{4(z)}_{\mathsf{a}}, \quad \Psi^{4(xyz)}_{\mathsf{a},\mathsf{a},\mathsf{a}} = \Psi^{4(x)}_{\mathsf{a}}\Psi^{4(x)}_{\mathsf{a}}\Psi^{4(z)}_{\mathsf{a}}.$$

Other combinations using the subscripts I and a are merely rotations of the stencils just listed but with identical stencil coefficients. Similar to two-dimensional boundary stencils, all needed stencils may be created in the same manner by considering all permutations of the relevant one-dimensional stencils, i.e.,

$$\Psi^{2n(xyz)}_{\alpha,\beta,\gamma} = \Psi^{2n(x)}_{\alpha}\Psi^{2n(y)}_{\beta}\Psi^{2n(z)}_{\gamma}, \tag{2.37}$$

where α, β, and γ each represent one of the following subscripts: I, a, b, c, ...

3. Test problems. In this section we will address the following two issues:

1. Given a PDE to solve on a block-structured adaptive mesh, with a derivative stencil of order p_D, what is the order of the interpolations (p_I) that needs to be used for prolongations and restrictions in order to obtain (p_D)th-order spatial convergence?
2. Once an appropriate (p_D, p_I) pair has been identified, which filter order p_F should be used in case Runge phenomenon is observed?

For the purposes of this paper, we will specify a refinement pattern and [illegible] the mesh. The issues regarding the criteria for refining-derefining a mesh point will not be addressed here.

3.1. Appropriate interpolation orders. As described in sections 1 and 2, the solution on any patch in a GH is the result of interpolations from finer patches (restrictions) and/or interpolations from the solution on coarser patches at a coarse-fine boundary (prolongations). In this paper, since we address vertex-centered meshes, restrictions are mere injections and do not involve interpolations; however, prolongations do involve interpolations at coarse-fine boundaries.

The position of the coarse-fine boundary is determined by the refining-derefining criteria. These are usually based on a threshold on the solution error [4, 5] or a threshold on the gradient [30]. Thus, there exists a small but nonzero gradient at a coarse-fine boundary and an interpolation error is incurred there. Ideally the gradient should be small enough that the interpolation error is negligible vis-à-vis the differentiation error regardless of the interpolation chosen, but this is difficult to achieve in practice and still have an efficient grid (we run the risk of refining large areas of the domain and losing the advantage of concentrating resolution only where needed). Since these interpolated values cannot be differentiated indefinitely and still preserve order, it becomes critical to choose an appropriate interpolation order.

In [24], we conducted a study of the appropriate interpolation order p_I given a derivative stencil of order $p_D = 4$. The tests were done with the FN equation in one dimension (where the highest spatial derivative is ∂_{xx}) and we concluded that $p_I \geq (p_D + 2)$ was sufficient. In Appendix C, we present an argument for the requisite

interpolation order p_I for a given order of discretization and arbitrary derivative order. In this analysis, we find that the "correct" interpolation order $p_I = p_I(p_D, H_D)$, where H_D is the highest spatial derivative in the equation. We verify this empirically by using the KdV equation (where highest spatial derivative is ∂_{xxx}) and the KS equation (∂_{xxxx}).

The tests are performed on GHs of different depths. The KdV and KS equations are time-advanced using the classical fourth-order Runge–Kutta method; the FN equation employs a second-order version (Heun's method). Time stepping on a GH, as mentioned in section 1, is done in a time-refined manner; see [4, 24] for a complete description. Briefly, for every timestep on Level0, we take 2 on Level1, 4 on Level2, etc., in a recursive manner, i.e., the sequence is Level0 - Level1 - Level2 - Level2 - Level1 - Level2 - Level2. Each patch has a halo (whose width is dependent on the equation being solved and the stencil used) of points around it so that central difference stencils (for derivatives) may be used. However, at domain boundaries, the width of the halo is restricted to 1, which necessitates the use of skewed, lower-order stencils. Dirichlet boundary conditions are imposed. On each patch, data are stored in a multidimensional FORTRAN77 (F77) array, which is passed into a subroutine, along with a stencil accuracy order, to compute the spatial derivative needed.

The halo of points around a refined (Level1 or higher) patch lies over a valid region of the patch's coarser parent patch and is filled by interpolating (prolonging) the data from the parent. Symmetric interpolation stencils are used, except near domain boundaries where the stencil is skewed; however, the order is preserved. This is practically achieved by passing the parent's data array to an F77 subroutine, along with an index interval (corresponding to the patch's halo) where the interpolation has to be done, along with the order of accuracy desired. The subroutine detects boundaries and skews the stencil accordingly. Further, since the different levels have different Δt, the filling up of "halos" may also involve temporal interpolation of the data of the parent mesh. This temporal interpolation is linear. At the end of a timestep on a given level, its data is restricted down to its coarser parent. Since we use vertex-centered meshes, the coarse and fine mesh points are aligned, and the restriction is a pure injection.

At the end of a Level0 timestep, all the patches on all the levels are processed by a filter (as described in section 2.2). These too are F77 subroutines that accept the multidimensional data array associated with a patch, along with an order of the filter.

These numerical tools, i.e., F77 subroutines for evaluating high-order derivatives, filters, and interpolations, along with a user's guide and code examples, are available online in the form of a software library from [1]. These tools accept multidimensional F77 arrays as inputs and are not tied to any particular data structure; the rest of our adaptive mesh code is in C++. See Appendix D for details regarding the library.

3.1.1. KdV equation. The KdV equation is written as

$$U_t + 6UU_x + U_{xxx} = 0. \tag{3.1}$$

Under the assumption that $U_x \to 0$ as $x \to \pm\infty$, we get a traveling wave solution $U = 2k^2(\cosh\{k(x - x_0 - 4k^2t)\})^{-2}$, where k is a wavenumber and the amplitude and velocity of the traveling wave are $2k^2$ and $4k^2$, respectively.

We solved (3.1) on a 2-level GH (i.e., with a Level0 and Level1 mesh, abbreviated as an $\{L_0, L_1\}$ mesh), with a factor of refinement of 2. The domain was $0 \le x \le 100$ with $x_0 = 50$, i.e., the initial condition was a soliton placed at the center of the

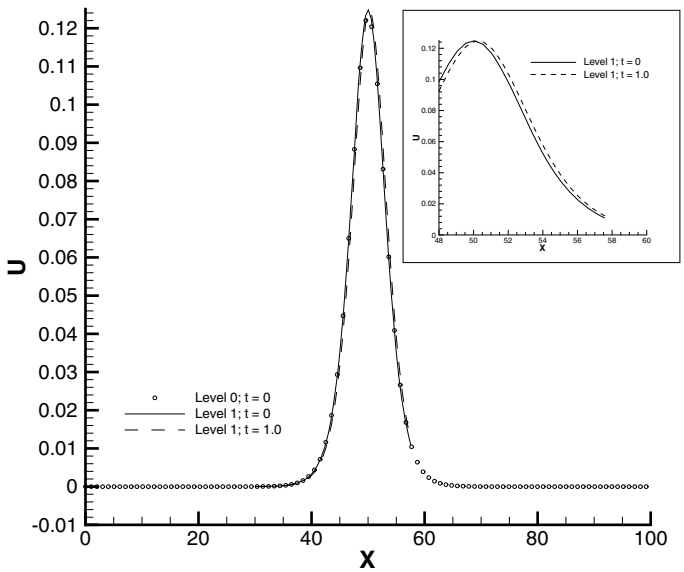


FIG. 3.1. *A soliton solution of* (3.1) *at* $t = 0$ *and* 1. *Symbols are used to plot the solution on* L_0 *and lines on* L_1. *The dashed line shows the solution at* $t = 1.0$. *Inset: A detail of the* L_1 *solution. We see that the movement of the soliton during the simulation is small vis-à-vis its size.*

domain. The region $30 \leq x \leq 56$ was refined to an L_1 mesh. We used $p_D = 4$. Both the first (∂_x) and third (∂_{xxx}) derivative stencils were closed to third order at the two points nearest the domain boundary. We refer to this as the 3-3-4-3-3 discretization strategy. The width of the halo around a fine patch was kept at 6. Runge phenomenon was not observed, and we did not use filters. Figure 3.1 shows the initial condition on L_0 in symbols and the L_1 solution as a solid line. The solution is marched explicitly in a "time-refined" manner using a classic fourth-order explicit Runge–Kutta integrator up to time $t = 1.0$ using a timestep (on the coarse mesh) of $\Delta t_0 = 1.0 \times 10^{-3}$. Such a small timestep was used to keep the temporal content small in the discrepancy between the numerical and exact solutions compared to the spatial content. Figure 3.1 (inset) shows the movement of the profile. We see that there is a nonzero gradient at the coarse-fine boundary. Such a situation might arise in practice if one is not extremely rigorous about the refinement-derefinement criteria. The solution at $t = 1.0$ is also shown.

In Figure 3.2 we show the RMS (root mean square) error in the RHS, i.e., $-(6UU_x + U_{xxx})$ with respect to the analytical solution on L_1. The x-axis is the resolution of the L_0 mesh. The choice of the RHS for convergence analysis is made in order to bring out clearly the effect of interpolation errors, which are best seen in the higher derivatives. We plot solutions obtained with $p_I = 4, 6$, and 8. A uniform mesh solution, i.e., without using interpolations, is also plotted as a guide for an ideal fourth-order convergence slope. We see that with $p_I = 4$ and 6 the slope is shallower than fourth order, while $p_I = 8$ achieves a greater slope. Thus, using $p_I = 8$ should enable (p_D)th-order convergence.

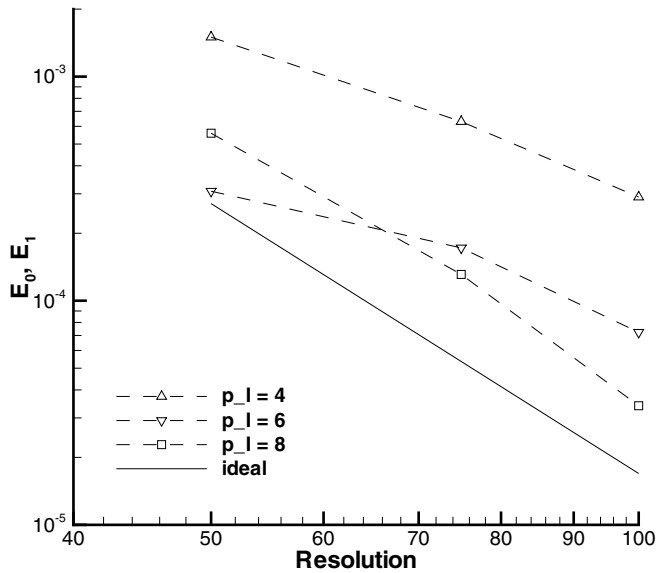


FIG. 3.2. *Convergence of the RHS of the KdV equation on L_1 with $p_D = 4$ and $p_I = 4, 6,$ and 8. The time is $t = 1.0$. A uniform mesh run is plotted as a guide for the convergence slope. $p_I = 8$ shows a convergence steeper than fourth order, while the others do not. Note that the resolution here is L_0 resolution; L_1 is a factor of 2 finer. The uniform mesh run was done on a grid of 100, 150, and 200 grid points. E_1 is the RMS (root mean square) error (with respect to the exact solution) on Level1; E_0 is the error, also of the RHS, for the uniform mesh run.*

Note that an $\{L_0, L_1\}$ mesh has an [illegible] resolution that is twice the L_0 resolution and should be compared with a uniform mesh solution that is equally resolved. This is done in Figure 3.2, where the uniform mesh error plotted against an L_0 resolution of 50 was obtained on a mesh of 100 grid points. Further, the uniform mesh run was uniformly more accurate since the entire domain was well resolved; in the adaptive mesh case, we purposely limited the refinement to $30 \leq x \leq 56$ to see the effect of the interpolants.

3.1.2. Kuramoto–Sivashinsky equation. The KS equation can be written as

$$U_t + UU_x + U_{xx} + U_{xxxx} = 0. \tag{3.2}$$

Under the assumption that $U_x \to 0$ as $x \to \pm\infty$, we get a traveling wave solution

$$\begin{aligned} U &= \frac{15}{19}k(11H^3 - 9H^2 + 2), \qquad k = \sqrt{\frac{11}{19}}, \\ H &= \tanh\left\{\frac{k}{2}\left(x - x_0 - \frac{30}{19}kt\right)\right\}. \end{aligned} \tag{3.3}$$

As in [24], we solved (3.2) on a 2-level GH with a factor of refinement of 2. The domain was $0 \leq x \leq 50$ with $x_0 = 29$. The region $20 \leq x \leq 29$ was refined to an L_1 mesh. A derivative stencil of order $p_D = 4$ was used. The first and second

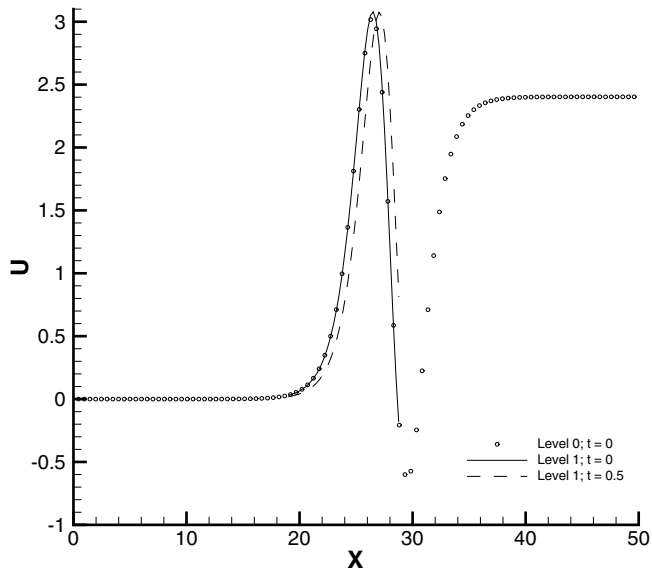


FIG. 3.3. *A traveling wave solution of* (3.2) *at* $t = 0$ *and* 0.5. *Symbols are used to plot the solution on* L_0 *and lines on* L_1. *The dashed line shows the solution at* $t = 0.5$.

derivative stencils were closed in a 3-3-4-3-3 manner, while the fourth derivative stencil was treated in a 3-3-3-4-3-3-3 fashion. The width of the halo was kept at 6. Runge phenomenon was not observed and we did not use filters. Figure 3.3 shows the initial condition on L_0 in symbols and the L_1 solution as a solid line. The solution was marched explicitly using a fourth-order Runge–Kutta integrator to time $t = 0.5$ using a timestep (on the coarse mesh) of $\Delta t_0 = 5.0 \times 10^{-4}$. This solution is also shown in Figure 3.3.

In Figure 3.4 we show the error in the RHS, i.e., $-(UU_x + U_{xx} + U_{xxxx})$ with respect to the analytical solution on L_1. The x-axis is the resolution of the L_0 mesh. We plot solutions obtained with $p_I = 6$ and 8. An ideal fourth-order convergence plot is provided as a guide. We see that with $p_I = 6$ the slope is shallower than fourth order, while $p_I = 8$ and 10 achieve a slope similar to the ideal. Thus, using $p_I = 8$ or 10 should enable (p_D)th-order convergence. However, the choice of interpolant order is also dependent on the position of the coarse-fine boundary. We repeated these runs with a refinement region of $20 \le x \le 38$. In this case, the gradient at the coarse-fine boundaries was small and both $p_I = 6$ and 8 provided fourth-order convergence.

Combining the results obtained above with those in [24], we observe that the results derived in Appendix C hold, i.e., they ensure (p_D)th-order spatial accuracy in block-structured adaptive meshes, even when there exists a substantial gradient at a coarse-fine boundary. This expression is seen to hold true at least for $\partial_{xx}, \partial_{xxx}$ and ∂_{xxxx} using fourth-order discretizations. We will use it to achieve high-order convergence in two dimensions below. A similar study, performed using the Kawahara equation [28], could verify the relationship between p_I, p_D, and H_D for $H_D = 5$.

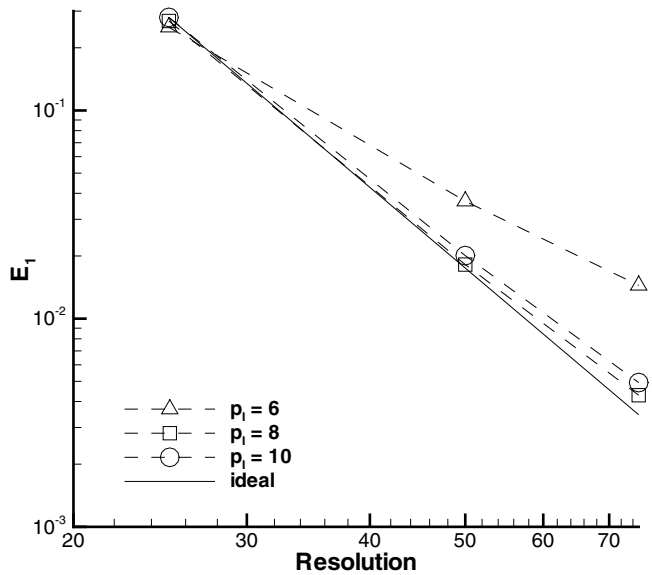


FIG. 3.4. *Convergence of the RHS of the KS equation on L_1 with $p_D = 4$ and $p_I = 6, 8,$ and 10. The time is $t = 0.5$. An ideal fourth-order convergence plot is provided for guidance. Both $p_I = 8$ and 10 show convergence similar to fourth order while the $p_I = 6$ does not. Note that the resolution here is L_0 resolution; L_1 is a factor of 2 finer. E_1 is the RMS error on Level*1.

3.2. Achieving high-order convergence. In this section we combine the tools developed in section 2 to empirically achieve high-order convergence on block-structured adaptive meshes in two dimensions. Tests were carried out in two dimensions using the FN equation. We chose a Level0 resolution that led to Runge phenomenon (because of its coarseness), thus necessitating the use of filters.

The FN equation in two dimensions is

$$U_t = D\nabla^2 U + AU(1-U)(U-\alpha). \tag{3.4}$$

We look for traveling wave solutions to (3.4), similar to the one-dimensional case [8]. We assume that a solution of the form $U = U(\xi + st)$ exists, where $\xi = x + \beta y - c$. We can derive the expression for $\nabla^2 U$ and U_t in terms of U' and U''. Under the assumption that

$$U' = aU(1-U), \tag{3.5}$$

where $U \to 0$ as $\xi \to -\infty$, and $U \to 1$ as $\xi \to \infty$, we get

$$a = \left(\frac{A}{2(1+\beta^2)D}\right)^{1/2}, \qquad s = \left(\frac{AD(1+\beta^2)}{2}\right)^{1/2}(1-2\alpha),$$

and

$$U = \frac{1}{2}\left(1 + \tanh\left(\frac{a\xi}{2}\right)\right) = \frac{1}{2}\left(1 + \tanh\left(\frac{\xi}{2\delta_f}\right)\right). \tag{3.6}$$

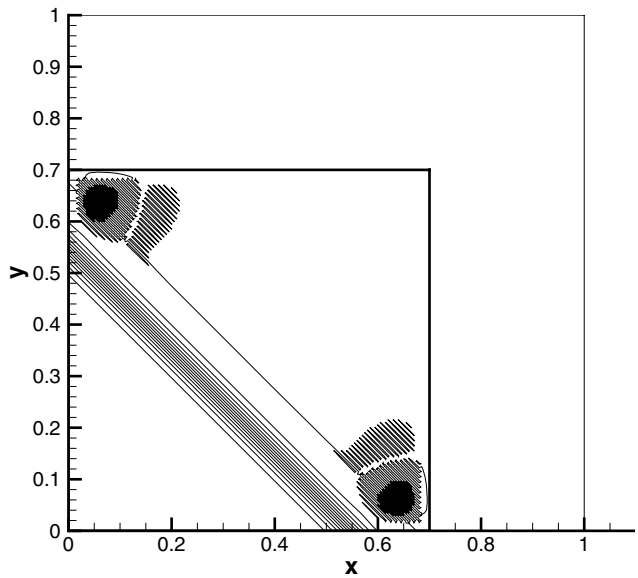


FIG. 3.5. *The Runge phenomenon on a* 2*-level block-structured mesh. The patch outlined in black is the Level*1 *patch. Level*0 *is a* 100×100 *mesh on a unit square. The solution is at* $t = 5 \times 10^{-5}$ *and was computed with* $p_D = 4$ *and* $p_I = 6$*. The correct solution should not contain the structure/oscillations one observes at the southeast and northwest corners of the Level*1 *patch.*

The last expression was obtained by solving (3.5) and completes the solution of (3.4). δ_f, a measure of the front thickness, is defined as $\delta_f = 1/a$.

We solved (3.4) on a unit square. We specified $D = 1, c = 0.5$, and $\beta = 1$. A was set such that $\delta_f = 0.02$. Equation (3.6) evaluated at $t = 0$ provided the initial condition and was also used to specify Dirichlet boundary conditions on the unit square. The analytical solution, (3.6), was also used to evaluate the numerical error (the difference between the exact and the numerical solutions) for convergence tests. The region $0 \leq x \leq 0.7, 0 \leq y \leq 0.7$ was refined (see Figure 3.5). The meshes on successive levels were refined by a factor of 2.

3.2.1. Fourth-order approach. Solving (3.4) with fourth-order derivative stencils ($p_D = 4$) and sixth-order interpolants ($p_I = 6$) led to Runge phenomenon (see Figure 3.5). This was not observed in a one-dimensional context in [24]. The particular choices of $p_D = 4$ and $p_I = 6$ were also obtained from [24]. The solution was filtered after every timestep to remove the Runge phenomenon. The $p_D = 4$ derivative stencils for ∂_{xx} and ∂_{yy} were evaluated in a 3-3-4-3-3 fashion per the nomenclature defined in section 3.1.1.

In order to choose a filter of the correct order p_F, we conducted a series of tests (see Figure 3.6). We fixed a derivative stencil ($p_D = 4$, centered) and experimented with filters of different orders p_F. A (p_F)th-order filter is a (p_F)th-spatial derivative in the interior points of the mesh implemented with a centered stencil that is of second-order accuracy. It is closed at the domain boundaries to first order. The Level0 mesh was kept at 100×100. The solution was advanced to time $t = 4 \times 10^{-4}$ using a

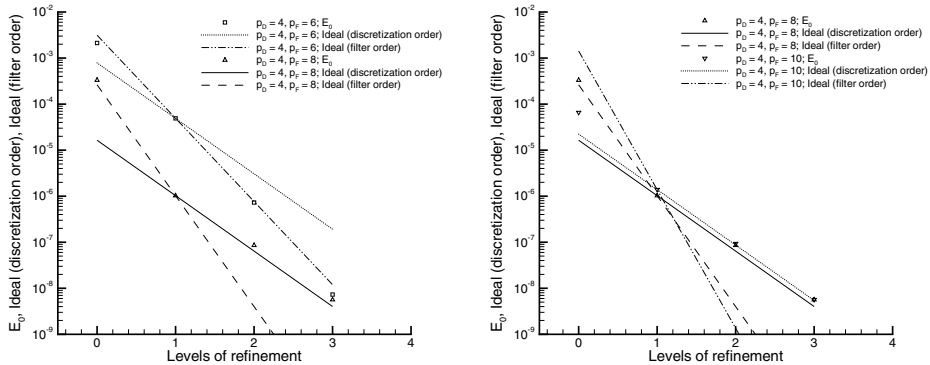


FIG. 3.6. *Numerical errors (calculated using the exact solution) for the FN equations solved using fourth-order discretizations. Numerical solutions computed with $p_F = 6, 8$, and 10 on GHs with up to 3 levels of refinement are plotted with $\square$, $\triangle$, and $\triangledown$, respectively. On the left, we compare results obtained with $p_F = 6$ and 8, while on the right, we compare $p_F = 8$ and 10. (p_D)th-order convergence is denoted by a solid or dotted line, while (p_F)th-order convergence is denoted by a dashed or dash-dotted line. For $p_F \geq 8$, we see fourth-order convergence as errors introduced by filtering are smaller than the derivative stencil's errors. The y-axis has the Level0 RMS error (E_0) and the "ideal" errors plotted as along a (p_D)th- and (p_F)th-order convergence slopes.*

Level0 $\Delta t = 10^{-6}$. The width of the halo was kept at 6. The temporal component in the RMS difference between the numerical and the exact solution was expected to be negligible with respect to the spatial component.

In Figure 3.6, we plot the Level0 RMS error using symbols, though tests were done with zero, one, two, and three levels of refinement (a zero level of refinement is a uniform mesh run). Lines are used to plot the ideal convergence (for the purpose of comparison) at both (p_D)th and (p_F)th order. For $p_F = 6$ (Figure 3.6, left), we see that the RMS errors (square symbols) lie along the dash-dotted line, which has a slope equivalent to sixth-order convergence. We also plot a dotted line, with a slope equivalent to fourth-order convergence (the order of the derivative stencil) as a guide; it is clear that the observed RMS errors do not follow this slope. The RMS errors are also far higher (between one and two orders of magnitude) than a similar run done with a $p_F = 8$ filter (triangles). The errors encountered when using a $p_F = 8$ filter are seen to have a slope close to that denoted by the solid line which is the ideal fourth-order convergence slope. Thus the error convergence is the same as the order of accuracy of the derivative stencil. We also plot a dashed line with a slope equivalent to eighth-order convergence; it is clear that when we have at least one level of refinement, this convergence slope is not observed. The only exception is while going from a uniform mesh to one level of refinement when $p_F = 8$; the two points lie on the dashed line (eighth-order convergence).

In Figure 3.6 (right) we compare the convergence of the RMS error for $p_F = 8$ and 10. The observed results (triangles for $p_F = 8$ and inverted triangles for $p_F = 10$) lie close to each other; further they fall directly on the solid and dotted lines, respectively, which are the ideal fourth-order convergence slopes. The only exceptions are the results when there were no levels of refinement. For the eighth-order filter, the convergence of the RMS error from a uniform mesh to the one-level-of-refinement result follow an eighth-order slope as denoted by the dashed line. For the tenth-order filter, the uniform mesh run shows a behavior that seems to

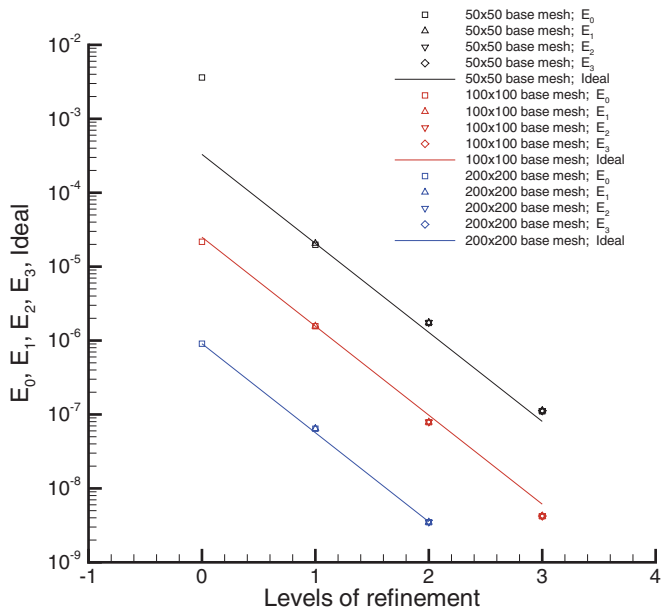


FIG. 3.7. *Numerical (RMS) errors observed when* $p_D = 4, p_F = 8$. *Solutions were done on* 50×50 *(black),* 100×100 *(red), and* 200×200 *(blue) meshes with different levels of refinement. Errors on levels* $0, 1, 2,$ *and* 3 *are plotted with* $\square$, $\triangle$, $\triangledown$, *and* $\diamond$, *respectively. Ideal fourth-order convergence is plotted using solid lines. Apart from the* 50×50 *uniform mesh run, the errors follow the ideal convergence closely.* E_i, $i = 0,\ldots,3$, *refers to the RMS error on level* i.

be intermediate between tenth-order (dash-dotted line) and fourth-order (solid line) convergence. Also, note that once the fourth-order convergence sets it (at one level of refinement), the difference between the $p_F = 8$ and $p_F = 10$ results is very small.

It is clear that using the $p_F = 6$ filter is inappropriate, and the errors introduced by this inappropriate use dominate the derivative stencil error. At higher filter orders, the errors introduced by filtering drop and we recover convergence commensurate with the order of the derivative stencil. That the "correct" filter could depend on the derivative stencil being used is evident in (2.17). In order that the leading-order error due to filters be similar to or smaller than those introduced by the derivative stencil, n (where $2n = p_F$) cannot be chosen at random; a conservative approach is to choose n such that the boundary errors of a filter (caused by the first-order closure of the filter stencil at the boundaries) is smaller than the errors introduced by a derivative stencil in the interior, where it (i.e., the derivative stencil) is full order (as opposed to its own lower-order closure at the boundaries). Further, if the dissipation introduced by the filter is overwhelming, (2.16) indicates that the dissipation will scale as $(\Delta x)^{2n}$.

Based on Figure 3.6 we choose $p_F = 8$ and conduct convergence tests on 50×50, 100×100, and 200×200 Level0 meshes (see Figure 3.7). Numerical errors calculated on each level are plotted, as levels of refinement are allowed. We see that the RMS errors on each of the levels are very similar, as might be expected since the solution is restricted from the finer to the coarser meshes. Further, as more levels of refinement are allowed, fourth-order convergence is observed. The only outlier is the 50×50 uniform mesh run; the grid is too coarse to resolve the traveling front, and the errors

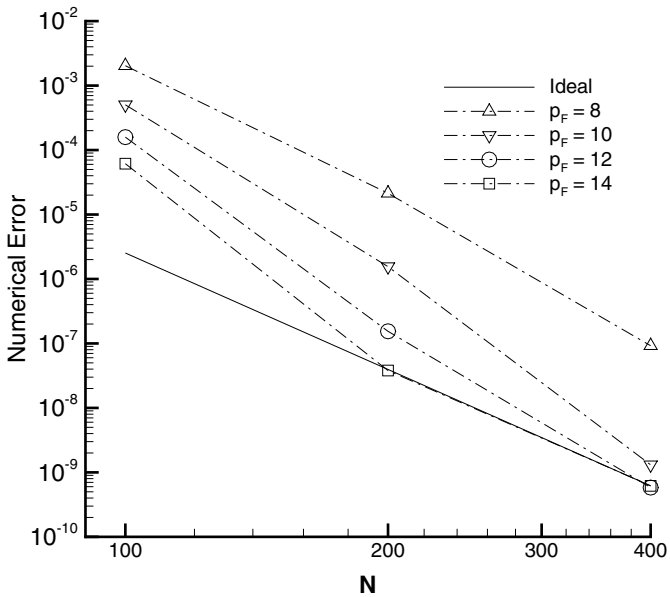


FIG. 3.8. *The RMS difference between numerical and exact solutions for the FN equations solved on an $N \times N$ uniform mesh using $p_D = 6$ with filters of order $p_F = 8, 10, 12,$ and 14. An ideal sixth-order convergence plot is provided for guidance. $\triangle, \triangledown, \bigcirc,$ and $\square$ are used to denote $p_F = 8, 10, 12,$ and 14 results; a solid line is used for runs without filtering.*

are excessive.

3.2.2. Sixth-order approach. In this section, we address the question of using $p_D = 6$ instead of $p_D = 4$ as was done in section 3.2.1. The sixth-order stencils for ∂_{xx} and ∂_{yy} were closed to third order on the two points closest to the domain boundary and to fourth order on one point adjacent to the third-order points, i.e., a 3-3-4-6-4-3-3 discretization strategy was followed. In Figure 3.6, it was seen that (p_D)th-order convergence was observed only with levels of refinement, when errors due to the derivative stencils dominated all other sources of errors (since it is the RMS difference between exact and numerical solutions that we plot). Since sixth-order stencils are expected to be more accurate than fourth-order ones, we first study the interaction between the filters and derivative stencils in a simplified setting, with no interpolations and on a uniform mesh.

In Figure 3.8, we plot the RMS difference between exact solutions and numerical solutions computed with $p_D = 6$ along with filters of order $p_F = 8, 10, 12,$ and 14. An ideal sixth-order convergence is also plotted for comparison. Runs were done on an $N \times N$ grid, where $N = 100, 200,$ and 400 with $\Delta t = 10^{-7}$. The problem was integrated up to time $t = 4 \times 10^{-4}$ as in section 3.2.1. We see that filtering introduces a significant error, and the behavior of the convergence of the errors lies between p_D and p_F. It is only at $N = 200$ that the convergence of the $p_F = 14$ run behaves in a sixth-order manner. Consequently, we will henceforth use the fourteenth-order filter to obtain sixth-order convergence on a GH. Note that the leading-order truncation term at the boundary for a fourteenth-order filter (ξ^8 for $n = 7$; see (2.17)) has the same order of magnitude as the leading-order truncation error term for a sixth-order

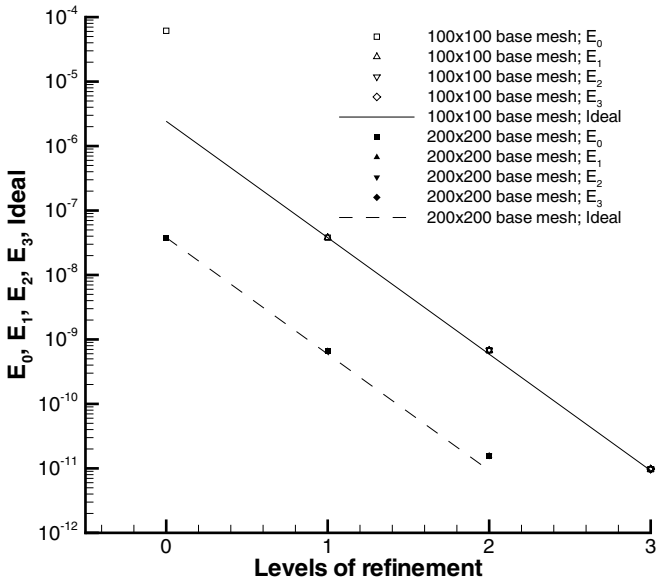


FIG. 3.9. *Numerical (RMS) errors observed when* $p_D = 6, p_I = 8, p_F = 14$ *when solving the FN equation. Solutions were done on a GH with Level*0 *equal to* 100×100 *(open symbols) and* 200×200 *(filled symbols) with different levels of refinement. Errors on levels* $0, 1, 2,$ *and* 3 *are plotted with* $\square$, $\triangle$, $\triangledown$, *and* $\diamond$, *respectively. Ideal sixth-order convergence is plotted using lines. Apart from the* 100×100 *uniform mesh run, the errors follow the ideal convergence closely.* E_i, $i = 0, \ldots, 3$ *refers to the RMS error on level i.*

second derivative stencil (see section 2.1.1).

We performed the analogue of the experiment in section 3.2.1 with $p_D = 6$. A GH with three levels of refinement was used. The interpolation order p_I was kept at 8 along with a halo of 10 cells. In Figure 3.9 we plot the error on each level, as more levels of refinement are enabled in a GH. We performed experiments with Level0 resolutions of 100×100 and 200×200. For the GH with a 200×200 Level0 resolution, we see an ideal sixth-order convergence; for the experiments with a 100×100 Level0, sixth-order convergence sets in only when the effective resolution of the mesh reaches 200×200; i.e., we have a Level1 mesh. This is also consistent with Figure 3.8, where sixth-order convergence sets in only at $N = 200$ with a $p_F = 14$ filter.

4. Conclusions. In this paper we have addressed the problem of the solution of a time-dependent PDE of the form

$$\frac{\partial U}{\partial t} = F\left(U, \frac{\partial U}{\partial x}, \cdots, \frac{\partial^{H_D} U}{\partial x^{H_D}}\right) \tag{4.1}$$

on a block-structured adaptive mesh using an approach that is high-order (> 2) in space. We have developed high-order stencils for derivatives, interpolations, and filters using a Fourier approach, a choice that enables high-order implementations with relative ease. These stencils were tested in one and two dimensions using model equations, and high-order convergence (fourth- and sixth-order) was demonstrated on

"vertex-centered" GHs. Time-instability near the boundaries was not observed, at least not for the fourth- and sixth-order derivative stencils used in this study. The combined use of derivative, interpolation, and filter stencils in these studies required a careful choice of their orders.

The Fourier image of a kth-spatial derivative evaluated using a (p_D)th-order stencil is $\Psi_{\mathrm{D}} = (i\xi)^k + O(\xi^{k+p_D})$, while that of an interpolant of (p_I)th-order accuracy is $\Psi_{\mathrm{I}} = (i\xi)^0 + O(\xi^{p_I})$. In order that the leading-order truncation error introduced by the two discrete operators be similar, one requires that

$$p_I = H_D + p_D, \tag{4.2}$$

where H_D is the highest spatial derivative observed in (4.1). This was verified for $2 \leq H_D \leq 4$ for $p_D = 2, 4$. A Mathematica [37] program that produces the necessary code for all these high-order interpolations is provided in Appendix A.

The use of high-order interpolation stencils to fill "halo" cells can lead to oscillations in situations when the solution is insufficiently resolved on the mesh where the interpolations are being performed. These oscillations (the Runge phenomenon) were observed on Level1 and Level2 and were removed by applying a (p_F)th-order filter to the solution after every timestep. The "correct" filter order p_F can be estimated in a manner similar to that for interpolations. The Fourier image of a filter stencil of order p_F is $\Psi_{\mathrm{F,Interior}} = (i\xi)^{p_F} + O(\xi^{p_F+2})$ in the interior and $\Psi_{\mathrm{F,Boundary}} = (i\xi)^{p_F/2} + O(\xi^{p_F/2+1})$. A conservative estimate of p_F can be obtained by equating the order of magnitudes of the leading-order truncation errors of $\Psi_{\mathrm{F,Boundary}}$ and Ψ_{D} to obtain

$$p_F = 2(p_D + H_D - 1). \tag{4.3}$$

This was shown to hold for $p_D = 4, 6$ for $H_D = 2$.

Combined with [24], this paper presents an alternative approach to the one developed in [3]. Our method, vis-à-vis [3], trades off one complexity versus another. While the high-order one-dimensional interpolations in [3] are far simpler than our multidimensional ones, their application requires bookkeeping, viz., tracking (a) patch configurations and (b) tangential and normal directions to a coarse-fine boundary, whereas we use our interpolations uniformly. Also, unlike [3], we have no requirement, in principle, that the refinement ratio between levels be any particular value, although the practical complexities of deriving and implementing the high-order multidimensional interpolations have led us to use a fixed refinement factor of 2 in the present work. This also ensures that the difference in (resolvable) wavenumber spectra between successive levels of refinement is kept manageable. While this is not much of a problem in vertex-centered grids, where restrictions (interpolating the fine mesh solution to the coarse mesh) are mere injections, cell-centered grids require interpolations to be performed, and care must be taken not to transfer high-wavenumber energies to lower ones while interpolating. This is obviously simpler if the wavenumber spectra are not too different. However, the ramifications of the choice (2 versus 4) of the level-to-level refinement factor, within the context of high-order AMR, need a systematic study. Furthermore, as shown in section 3, our interpolations have consistently provided high-order convergence even when the coarse-fine boundary was placed, on purpose, in a region of high gradient. While such an eventuality is not likely in an AMR simulation, the threshold-based refinement-derefinement criteria do not guarantee against it, and the high-order interpolations serve as a fallback in borderline resolution cases.

High-order methods, coupled with AMR, hold the potential to perform high-accuracy simulations economically. These methods, as presented in this paper, are quite general—stencils for derivatives, filters, and interpolations of any orders, p_D, p_F, and p_I, may be generated using the expressions developed here. There are neither special cases nor assumptions and/or constructions predicated on a specific choice of stencils. Replacements of interpolation, derivative, and filter stencils are affected simply by replacing subroutines. Significant computational savings can be achieved over second-order AMR methods by adopting high-order approaches as documented in [24, 3]. Filtering is used to introduce dissipation to control oscillations, and an inappropriate filter can significantly degrade the accuracy of high-order approaches. In this regard, we have derived and verified the relationship that needs to be preserved between the orders of derivative, filter, and interpolation stencils in order to achieve high-order convergence.

Appendix A. Stencil coefficient computation. Below is a Mathematica [37] program that may be used to generate an F77 file containing all boundary and interior interpolant stencils discussed in this paper. First, write individual stencils as functions.

```
    (* Second-Order *)
Psi2I[a_]:=Cos[a/2];
Psi2a[a_]:=(Cos[a]+I*Sin[a])*(Cos[a/2]+I*(-2*Sin[a/2]));

(* Fourth-Order *)
Psi4I[a_]:=Cos[a/2]*(-5+Cos[a])/4;
Psi4a[a_]:=(Cos[a]+I*Sin[a])*(5*Cos[a/2]+3*Cos[3*a/2]+
I*(-10*Sin[a/2]-2*Sin[3*a/2]))/8;
Psi4b[a_]:=Cos[2*a]+I*Sin[2*a])*(7*Cos[a/2]-15*Cos[3*a/2]+
I*(-28*Sin[a/2]+20*Sin[3*a/2]))/8;

(* Sixth-Order *)
Psi6I[a_]:=Cos[a/2]*(89-28*Cos[a]+3*Cos[2*a])/64;
Psi6a[a_]:=(Cos[a]+I*Sin[a])*(70*Cos[a/2]+63*Cos[3*a/2]-5*Cos[5*a/2]+
I*(-140*Sin[a/2]-42*Sin[3*a/2]+2*Sin[5*a/2]))/128;
Psi6b[a_]:=Cos[2*a]+I*Sin[2*a]*(42*Cos[a/2]-135*Cos[3*a/2]-
35*Cos[5*a/2]+
I*(-168*Sin[a/2]+180*Sin[3*a/2]+28*Sin[5*a/2]))/128;
Psi6c[a_]:=(Cos[3*a]+I*Sin[3*a])*(198*Cos[a/2]-385*Cos[3*a/2]+
315*Cos[5*a/2]+
I*(-1188*Sin[a/2]+770*Sin[3*a/2]-378*Sin[5*a/2]))/128;

(* Eighth-Order *)
Psi8I[a_]:=Cos[a/2]*(-762+299*Cos[a]-54*Cos[2*a]+5*Cos[3*a])/512;
Psi8a[a_]:=(Cos[a]+I*Sin[a])*(525*Cos[a/2]+567*Cos[3*a/2]-
75*Cos[5*a/2]+
7*Cos[7*a/2]+I*(-1050*Sin[a/2]-378*Sin[3*a/2]+30*Sin[5*a/2]-
2*Sin[7*a/2]))/1024;
Psi8b[a_]:=Cos[2*a]+I*Sin[2*a]*
(231*Cos[a/2]-891*Cos[3*a/2]-385*Cos[5*a/2]+21*Cos[7*a/2]+
I*(-924*Sin[a/2]+1188*Sin[3*a/2]+308*Sin[5*a/2]-12*Sin[7*a/2]))/1024;
```

```
Psi8c[a_]:=(Cos[3*a]+I*Sin[3*a])*(429*Cos[a/2]-1001*Cos[3*a/2]+
1365*Cos[5*a/2]+
231*Cos[7*a/2]+I*(-2574*Sin[a/2]+2002*Sin[3*a/2]-1638*Sin[5*a/2]-
198*Sin[7*a/2]))/1024;
Psi8d[a_]:=Cos[4*a]+I*Sin[4*a]*(3575*Cos[a/2]-7371*Cos[3*a/2]+
5775*Cos[5*a/2]-
3003*Cos[7*a/2]+I*(-28600*Sin[a/2]+19656*Sin[3*a/2]-9240*Sin[5*a/2]+
3432*Sin[7*a/2]))/1024;

(* Tenth-Order *)
Psi10I[a_]:=Cos[a/2]*(25609-11528*Cos[a]+2708*Cos[2*a]-440*Cos[3*a]+
35*Cos[4*a])/16384;
Psi10a[a_]:=(Cos[a]+I*Sin[a])*(16170*Cos[a/2]+19404*Cos[3*a/2]-
3300*Cos[5*a/2]+539*Cos[7*a/2]-45*Cos[9*a/2]+I*(-32340*Sin[a/2]-
12936*Sin[3*a/2]+1320*Sin[5*a/2]-154*Sin[7*a/2]+10*Sin[9*a/2]))/32768;
Psi10b[a_]:=-(Cos[2*a]+I*Sin[2*a])*(6006*Cos[a/2]-25740*Cos[3*a/2]-
14300*Cos[5*a/2]+1365*Cos[7*a/2]-99*Cos[9*a/2]+I*(-24024*Sin[a/2]+
34320*Sin[3*a/2]+11440*Sin[5*a/2]-780*Sin[7*a/2]+44*Sin[9*a/2]))/32768;
Psi10c[a_]:=(Cos[3*a]+I*Sin[3*a])*(7722*Cos[a/2]-20020*Cos[3*a/2]+
35100*Cos[5*a/2]+10395*Cos[7*a/2]-429*Cos[9*a/2]+I*(-46332*Sin[a/2]+
40040*Sin[3*a/2]-42120*Sin[5*a/2]-8910*Sin[7*a/2]+
286*Sin[9*a/2]))/32768;
Psi10d[a_]:=-(Cos[4*a]+I*Sin[4*a])*(24310*Cos[a/2]-55692*Cos[3*a/2]+
56100*Cos[5*a/2]-51051*Cos[7*a/2]-6435*Cos[9*a/2]+I*(-194480*Sin[a/2]+
148512*Sin[3*a/2]-89760*Sin[5*a/2]+58344*Sin[7*a/2]+
5720*Sin[9*a/2]))/32768;
Psi10e[a_]:=(Cos[5*a]+I*Sin[5*a])*(293930*Cos[a/2]-639540*Cos[3*a/2]+
554268*Cos[5*a/2]-285285*Cos[7*a/2]+109395*Cos[9*a/2]+
I*(-2939300*Sin[a/2]+
2131800*Sin[3*a/2]-1108536*Sin[5*a/2]+407550*Sin[7*a/2]-
121550*Sin[9*a/2]))/32768;
```

Next, set desired parameters. This is the only section of the code that needs to be modified by the user. In the following example, we wish to compute a fourth-order stencil for a point shifted from the center by zero grid points in x, $\Psi_I^{4(x)}$, one in y, $\Psi_{\mathsf{a}}^{4(y)}$, and two in z, $\Psi_{\mathsf{b}}^{4(z)}$, or $\Psi_{I,\mathsf{a},\mathsf{b}}^{4(xyz)}$.

```
(* Specify stencil *)
order = 4;
Psi = Expand[ TrigToExp[ Psi4I[x]*Psi4a[y]*Psi4b[z] ] ];
ix = 0; iy = 1; iz = 2;
```

Alternatively, we could compute the stencil $\Psi_{\mathsf{b},\mathsf{c},\mathsf{d}}^{8(xyz)}$ by writing

---

```
(* Specify stencil *)
order = 8;
Psi = Expand[ TrigToExp[ Psi8b[x]*Psi8c[y]*Psi8d[z] ] ];
ix = 2; iy = 3; iz = 4;
```

Now the actual stencil coefficients are computed and printed to the screen in F77 format.

```
(* Compute stencil *)
leftx=-(order-1)+2*ix;
lefty=-(order-1)+2*iy;
leftz=-(order-1)+2*iz;
psi = Expand[ Factor[ Normal[ Series[ Psi,{x,0,order},{y,0,order},
{z,0,order}]]]];
Print[" psi = ", psi ];
psilistx = CoefficientList[ psi - 1,x] /.  {y -> 0, z -> 0};
psilisty = CoefficientList[ psi - 1,y] /.  {x -> 0, z -> 0};
psilistz = CoefficientList[ psi - 1,z] /.  {x -> 0, y -> 0};
actualorder = Sum[ (psilistx[[i]]+psilisty[[i]]+psilistz[[i]]),
{i,1,order-1}];

Do[
Print["Do loop counter = ", k];
Do[
Do[ bnd[i,j,k] = ExpToTrig[ Expand[
Psi*Exp[-I*(leftx+2*(i-1))*x/2]*Exp[
-I*(lefty+2*(j-1))*y/2]*Exp[-I*(leftz+2*(k-1))*z/2]]]; ,{i,1,order}];
x = 3; y = 17; z = 9973;
Do[ bnd[i,j,k] = Part[ bnd[i,j,k], 1 ];, {i,1,order}];
,{j,1,order}]; ,{k,1,order}];

denominator = Part[ {8,2^(12),2^(24),2^(33),2^(48),2^(58)}, order/2];
letters = {I,a,b,c,d,e,f};
Do[ bnd[i,j,k] = Factor[ denominator*bnd[i,j,k] ];,
{i,1,order},{j,1,order},{k,1,order}];
Do[ Print[" bnd",order,Part[letters,ix+1],Part[letters,iy+1],
Part[letters,iz+1],
"(",i,",",j,",",k,") = ", bnd[i,j,k],".d0/",denominator,".d0" ];,
{i,1,order},{j,1,order},{k,1,order}];
If[ actualorder == 0, Print["Stencil is correct"],
Print["Stencil is wrong"]];
```

Appendix B. Interior stencils for three-dimensional interpolants.

TABLE B.1
Coefficients of three-dimensional, vertex-centered interpolants in terms of ξ, η, and ζ.

Stencil	2nd-order	4th-order	6th-order	8th-order	10^{th}-order
aaa	1/8	$729/2^{12}$	$3375000/2^{24}$	$1838265625/2^{33}$	$62523502209000/2^{48}$
baa	0	$-81/2^{12}$	$-562500/2^{24}$	$-367653125/2^{33}$	$-13894111602000/2^{48}$
bba	0	$9/2^{12}$	$93750/2^{24}$	$73530625/2^{33}$	$3087580356000/2^{48}$
bbb	0	$-1/2^{12}$	$-15625/2^{24}$	$-14706125/2^{33}$	$-686128968000/2^{48}$
caa	0	0	$67500/2^{24}$	$73530625/2^{33}$	$3572771554800/2^{48}$
cba	0	0	$-11250/2^{24}$	$-14706125/2^{33}$	$-793949234400/2^{48}$
cbb	0	0	$1875/2^{24}$	$2941225/2^{33}$	$176433163200/2^{48}$
cca	0	0	$1350/2^{24}$	$2941225/2^{33}$	$204158374560/2^{48}$
ccb	0	0	$-225/2^{24}$	$-588245/2^{33}$	$-45368527680/2^{48}$
ccc	0	0	$27/2^{24}$	$117649/2^{33}$	$11666192832/2^{48}$
daa	0	0	0	$-7503125/2^{33}$	$-637994920500/2^{48}$
dba	0	0	0	$1500625/2^{33}$	$141776649000/2^{48}$
dbb	0	0	0	$-300125/2^{33}$	$-31505922000/2^{48}$
dca	0	0	0	$-300125/2^{33}$	$-36456852600/2^{48}$
dcb	0	0	0	$60025/2^{33}$	$8101522800/2^{48}$
dcc	0	0	0	$-12005/2^{33}$	$-2083248720/2^{48}$
dda	0	0	0	$30625/2^{33}$	$6510152250/2^{48}$
ddb	0	0	0	$-6125/2^{33}$	$-1446700500/2^{48}$
ddc	0	0	0	$1225/2^{33}$	$372008700/2^{48}$
ddd	0	0	0	$-125/2^{33}$	$-66430125/2^{48}$
eaa	0	0	0	0	$55135363500/2^{48}$
eba	0	0	0	0	$-12252303000/2^{48}$
ebb	0	0	0	0	$2722734000/2^{48}$
eca	0	0	0	0	$3150592200/2^{48}$
ecb	0	0	0	0	$-700131600/2^{48}$
ecc	0	0	0	0	$180033840/2^{48}$
eda	0	0	0	0	$-562605750/2^{48}$
edb	0	0	0	0	$125023500/2^{48}$
edc	0	0	0	0	$-32148900/2^{48}$
edd	0	0	0	0	$5740875/2^{48}$
eea	0	0	0	0	$48620250/2^{48}$
eeb	0	0	0	0	$-10804500/2^{48}$
eec	0	0	0	0	$2778300/2^{48}$
eed	0	0	0	0	$-496125/2^{48}$
eee	0	0	0	0	$42875/2^{48}$
$\psi_{N,N,N}\xi^N\eta^N\zeta^N$	$-(1/8)\xi^2$ $-(1/8)\eta^2$ $-(1/8)\zeta^2$	$-(3/128)\xi^4$ $-(3/128)\eta^4$ $-(3/128)\zeta^4$	$-(5/1024)\xi^6$ $-(5/1024)\eta^6$ $-(5/1024)\zeta^6$	$-(35/32768)\xi^8$ $-(35/32768)\eta^8$ $-(35/32768)\zeta^8$	$-(63/262144)\xi^{10}$ $-(63/262144)\eta^{10}$ $-(63/262144)\zeta^{10}$

Appendix C. Rules for pairing a derivative and an interpolation stencil.

In this appendix we determine a rule to pair a spatial derivative stencil of order P with an interpolation of order Q given that the highest spatial derivative that needs to be evaluated is R.

Let $\phi^{(R)}$ be the Rth derivative of ϕ in the domain $\mathcal{A}$, the patch G2 in Figure C.1. Let $D_P^{(R)}$ be a discrete operator (a stencil) that, when applied at a point in $\mathcal{A}$, evaluates the derivative with an error proportional to $(\Delta x)^P$. We will analyze only in the x-direction, where points are indexed using i; the results will carry over to a higher-dimensional space.

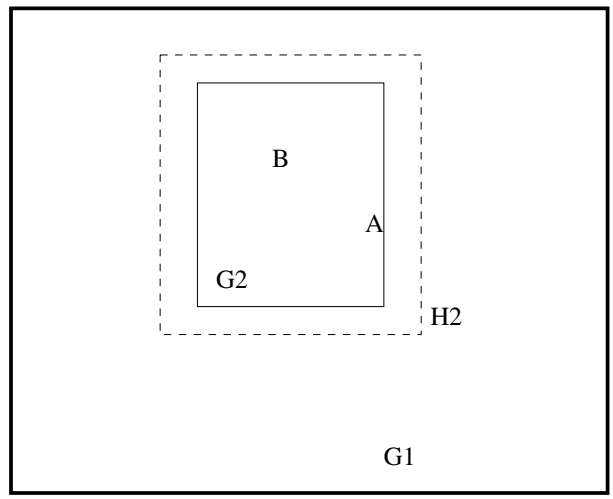


FIG. C.1. *Two overlaid patches* G1 *and* G2. G2 *is finer and has a halo* H2 *around it to facilitate the evaluation of discrete derivatives via stencils. Points* A *and* B *lie on the finer* G2 *patch.*

Applying this at point B in Figure C.1 poses no problems,

$$\begin{aligned}\phi_B^{(R)} &= \frac{D_P^{(R)}(\phi_B)}{(\Delta x)^R} + \varepsilon_1(\Delta x^P) \\ &= \frac{D_P^{(R)}(\ldots, \phi_{i-1}, \phi_i, \phi_{i+1}, \ldots)}{(\Delta x)^R} + \varepsilon_1(\Delta x^P),\end{aligned} \tag{C.1}$$

as long as the stencil has enough points on the left and right. At A, however, the stencil "spills" over into the coarser G1 patch.

Typically, in order to not use skewed stencils, one keeps a halo H2 of points around G2, where data are interpolated from G1 with an interpolant of order Q. Let these interpolated ϕ be called $\tilde{\phi}$, i.e.,

$$\tilde{\phi} = \phi + \varepsilon_2(\Delta x^Q).$$

Thus, we evaluate $\phi_A^{(R)}$ as

$$\begin{aligned}\phi_A^{(R)} &= \frac{D_P^{(R)}(\phi_A)}{(\Delta x)^R} + \varepsilon_1(\Delta x^P) \\ &= \frac{D_P^{(R)}(\ldots, \phi_{i-1}, \phi_i, \tilde{\phi}_{i+1}, \ldots)}{(\Delta x)^R} + \varepsilon_1(\Delta x^P) \\ &= \frac{D_P^{(R)}(\ldots, \phi_{i-1}, \phi_i, \phi_{i+1}, \ldots)}{(\Delta x)^R} + \varepsilon_1(\Delta x^P) + \varepsilon_3(\Delta x^{Q-R}).\end{aligned} \tag{C.2}$$

That $\varepsilon_1(\Delta x^P)$ and $\varepsilon_3(\Delta x^{Q-R})$ be of the same order requires

$$Q = P + R. \tag{C.3}$$

This is sufficient to allow Pth-order convergence of $\phi^{(R)}$. Note that ε_1 exists over $\mathcal{A}$ while ε_3 exists only along $\partial\mathcal{A}$, its boundary, a domain of dimension one less than $\mathcal{A}$.

TABLE C.1

A comparison of the order of interpolation Q for the FN, KdV, and KS equations when solved with fourth- ($P = 4$) and sixth- ($P = 6$) order discretizations as indicated by (C.3) *and* [7]*. The first column also mentions the highest spatial derivative in the equation. The figure in parentheses in the second column indicates the interpolations used in this study—we use only even-order interpolations to avoid dispersive errors.*

Equation	$P = 4$		$P = 6$	
	Eq. (C.3)	[7]	Eq. (C.3)	[7]
FN (∂_{xx})	6	5	8	7
KdV (∂_{xxx})	7 (8)	7	9	10
KS (∂_{xxxx})	8 (8 and 10)	9	10	13

However, the requirement above can be relaxed. Often, errors are measured, integrated, and analyzed in a certain area. This was also the case in this paper. Thus,

$$E = \int_{\mathcal{A}} \varepsilon_1(\Delta x^P) dA + \int_{\partial\mathcal{A}} \varepsilon_3(\Delta x^{Q-R}) ds \\ = E_1(\Delta x^{P-2}) + E_3(\Delta x^{Q-R-1}) \tag{C.4}$$

if $\mathcal{A}$ is two-dimensional. If $\mathcal{A}$ is three-dimensional, the RHS is $E_1(\Delta x^{P-3}) + E_3(\Delta x^{Q-R-2})$. In order that E_1 and E_3 have the same convergence rates,

$$Q = P + R - 1. \tag{C.5}$$

Strictly speaking, (C.5) will ensure Pth-order convergence of $\phi^{(R)}$ in a global sense if $\partial\mathcal{A}$ is one grid point in thickness. For our high-order discretizations, this is not the case; interpolations are done in a thin [illegible] of half the stencil width. Thus, it is unclear whether (C.3) or (C.5) is preferable, but (C.3) should be sufficient.

We contrast this with the recommendation (in [7]) that $Q = \frac{1}{2}PR + 1$. This expression is derived in [7] under the assumption of second-order discretizations ($P = 2$). Empirical studies are performed for $R = 2$ (Laplace and Navier–Stokes equations) for $P = 2$ and 4. In Table C.1 we compare Q as recommended by [7] and (C.3) for the equations used in this study. We find that the recommended values for Q are generally not too far apart.

Appendix D. Description of HODIF: A library of high-order derivatives, interpolations, and filters. In this appendix, we describe HODIF, a library that contains the stencils described in the paper. This library can be downloaded from [1] and comes with a user's manual that explains how this library may be called from FORTRAN or C/C++. Example codes doing so are in the `example/` directory of the distribution.

The library is a collection of F77 subroutines that implement derivative, interpolation, and filter stencils. The subroutines are kept under separate directories `deriv/`, `interp/`, and `filter/`. These subroutines expect double precision (`real*8`) three-dimensional F77 arrays as input, but lower-dimensional arrays may be passed in with the size of the extra dimension set to 1. Multidimensional C/C++ arrays can be used only if the data lie in one contiguous memory segment and are stored in F77 ordering; in such a case, a pointer to this memory segment can be passed in as an argument to the subroutines. The subroutines are called with a set of integers which serve as demarcations of array sections. We explain this further.

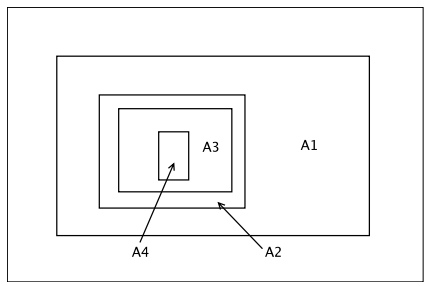


FIG. D.1. *Various array sections and their subsetting.* A1 *is a "distributed" area,* A2 *is a patch, which contains* A3 *which is the valid region/interior points.* A4 *is a "window."*

Our library was developed to be used with frameworks for doing computations on block-structured adaptive meshes, where operations are performed on a patch-by-patch basis. Each patch (A2 in Figure D.1) contains a "valid area"/ "interior points" (A3 in Figure D.1) with a "halo" of points around it. The "halo" is written into during prolongations from the underlying coarse parent patch; alternatively, in parallel computations, they may serve as ghost cells. For the purposes of filter and derivative stencils, the halo is treated as a read-only area of the array, except near domain boundaries, where boundary conditions have to be applied to fill up the halos. However, the data for a patch (interior and halo points) are stored in the same array for ease of access; thus if the data array being passed in is dimensioned `a(fiL:fiR, fjL:fjR)`, the indices `fiL, fiR, fjL, fjR` have to be passed in as arguments. Many parallel software packages provide the notion of a "distributed" array, spread over many processors, with a global indexing convention (A1 in Figure D.1). Each processor owns a section of this global array (in other words, a patch) and accesses are done using the global index. We accommodate this by requiring that indices supplied to the subroutine be either consistently local or global. Inside the subroutine, accesses are done using a local index. We also allow the concept of "windows" (A4 in Figure D.1), the ability to apply an operation to a subsection of the interior points. This is useful during prolongations which only have to be done in a small "window" of a parent patch, encompassing the region around a coarse-fine patch boundary.

By far the most difficult aspect of high-order methods on block-structured adaptive meshes is the application of multidimensional interpolation at corner points. Consider the top left case in Figure D.2. The stencil is symmetric about both the vertical and horizontal axes. For the top right figure, we skew vertically but retain symmetry about the vertical axis. The stencil for the bottom right case is the same as the top right, after a 90° clockwise rotation. The stencil for the bottom right case is completely different. We thus have three unique stencils, applied to four corner points, sometimes with rotation. The task of determining the nature of the points at a corner automatically and identifying the correct skewed stencil, along with the correct rotation, is tedious and error prone; further, as the order of the interpolation stencil increases, the number of corner points increase (as p_I^d, where p_I is the order of interpolation and d is the dimensionality of space). Since we have already gone through this exercise, we are making a validated implementation of this logic available with our library; thus the user calls the library, and the correct stencils are identified and applied without any prompting from the user. It is this enabling algorithm that makes this library far more than a collection of stencils and is integral to the usability of high-order methods.

Another feature of the library is the "piecewise" evaluation of midpoints during

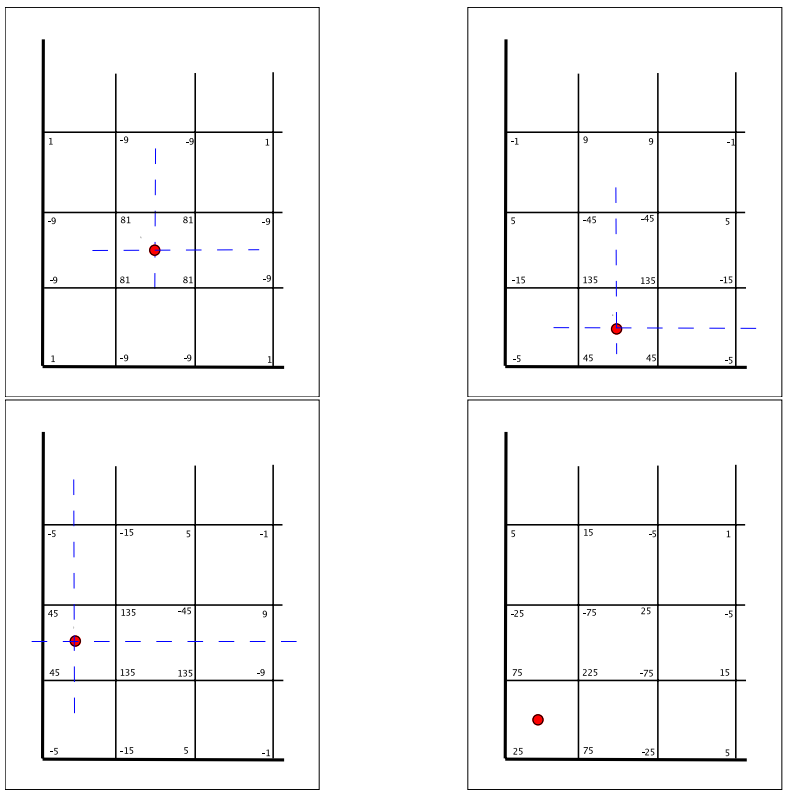


FIG. D.2. *Skewing of stencils near a corner at the domain boundary. At the top left, the value at the circle is obtained by a symmetric interpolation stencil whose coefficients are shown in the figure. For the top right and bottom left cases, the stencils are the same, but differ by a clockwise rotation of* 90°. *The stencil for the bottom right case is completely different but symmetric about the diagonal. All stencil coefficients need to be normalized by* 256.

prolongations. Figure D.3 shows a fine mesh overlayed on a coarse one; the factor of refinement is 2. Data from the large circles have to be interpolated to the small circles and squares. The values at the small circles are obtained by one-dimensional interpolations; further, x- and y-interpolations are performed by different subroutines and have to be called separately. The squares are evaluated by a separate call to a two-dimensional interpolation routine. The interpolated results are stored separately and then inserted by the user into their correct locations on the fine mesh's data array. There are two reasons for not automating the interpolation: (1) The marshaling of the various midpoint values is not difficult, and (2) large temporary arrays, which would be allocated on the stack in F77 subroutines, can lead to overflow. We thus allow the user to reuse any workspace arrays that he/she might have in case large arrays are involved. In adaptive mesh runs, this has not been an issue.

Proficiency in using this library is best achieved by practice. To this effect we have provided examples in F77 and C along with a user's manual. Further, for C/C++ programmers, the interface to the FORTRAN subroutines is in the headers in `include/`; each of the directories `deriv/`, `interp/`, and `filter/` include a C stub that translates the C function into their F77 counterparts. The headers in `include/` are annotated for convenience.

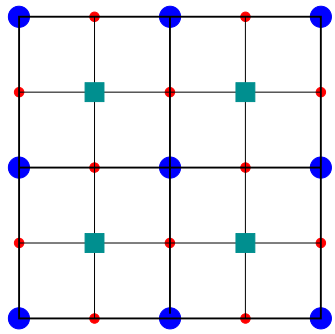


FIG. D.3. *Coarse mesh points (large circles) along with collocation points for the fine child patch (small circles and squares). The values for the small circle points are obtained by one-dimensional interpolations, while for the square points, a two-dimensional interpolation is used.*

We illustrate the general approach to using this library using code snippets from `example/interp_example.f` in the software distribution. The example involves:

1. initializing a field $\sin(2\pi x)\sin(2\pi y)$ on a unit square discretized uniformly by 50 points. This is the coarse mesh.
2. interpolating the field on the coarse mesh onto a fine mesh, twice as fine.
3. taking the root-mean-square difference between the two.

In `MAIN`, we define the array limits and allocate the coarse and fine mesh arrays. We keep the halo width at 2, but this particular aspect does not play a part in this example since the entire fine patch including the halo is filled up during interpolation. The definition and allocation of arrays holding the coarse and fine fields are on lines 48–56 in the example source code.

The coarse mesh is then initialized by a call to a subroutine (`initialize_field`). We next proceed to the process of orchestrating the interpolation. This is done in a separate subroutine called `interp_vc_field`. First, we allocate temporaries that will contain the values interpolated separately at edge-centers in x and y, as well as the values evaluated at cell centers (lines 120–122). We then perform the interpolation in x, i.e., evaluate the field at collocation points that lie at the center of the horizontal edges. In order to do so, we specify the array limits of the incoming (coarse) field, the interior points (which are the same as array limits since we wish to interpolate the entire domain covered by the coarse mesh, including the halo points), and the array limits of the temporary array (which are the same as that of the coarse mesh). Note that all variables `b?L = b?R = 0`, indicating that all points are valid points/interior points. `?` is a wildcard that can stand for `i`, `j`, or `k`. This corresponds to lines 150–189. We make the call to the library on line 203. `iperx, ipery,` and `iperz` are variables that indicate if periodicity exists in any of the directions. `orderi, orderb` indicate which order of interpolation to use for the interior (`i`) and boundary (`b`) and are set to 6. `orderb` is ignored; the library skews the stencil to a lower order by itself. The output is kept in `tempx`.

This is repeated with `tempy` and `tempxy` while calling `y_intp_cf_vc()` and `xy_intp_cf_vc()` for the interpolations in y and the mid-points. Finally, the contents of `tempx, tempy,` and `tempxy` are inserted into their correct places in the "fine" array (lines 238–247).

REFERENCES

[1] *Libraries of High-Order Discretizations, Interpolations, and Filters*, http://www.caip.rutgers.edu/∼jaray/HighOrder/SISC06.html (2006).

[2] G. Ashcroft and X. Zhang, *Optimized prefactored compact schemes*, J. Comput. Phys., 190 (2003), pp. 459–477.

[3] M. Barad and P. Colella, *A fourth-order accurate local refinement method for Poisson equation*, J. Comput. Phys., 209 (2005), pp. 1–18.

[4] M. Berger and P. Colella, *Local adaptive mesh refinement for shock hydrodynamics*, J. Comput. Phys., 82 (1989), pp. 64–84.

[5] M. Berger and J. Oliger, *Adaptive mesh refinement for hyperbolic partial differential equations*, J. Comput. Phys., 53 (1984), pp. 484–512.

[6] M. H. Carpenter, D. Gottlieb, and S. Abarbanel, *Stable and accurate boundary treatments for compact, high-order finite-difference schemes*, Appl. Numer. Math., 12 (1993), pp. 55–87.

[7] G. Chesshire and W. D. Henshaw, *Composite overlapping meshes for the solution of partial differential equations*, J. Comput. Phys., 90 (1990), pp. 1–64.

[8] C. P. Fall, E. S. Marland, J. M. Wagner, and J. J. Tyson, eds., *Computational Cell Biology*, Interdisciplinary Appl. Math. 20, Math. Biology, Springer, New York, 2002.

[9] J. S. Hesthaven, *Adaptive high-order finite difference method for nonlinear wave problems*, J. Sci. Comput., 16 (2001), pp. 47–67.

[10] M. Gunther, A. K. Rno, and P. Rentrop, *Multirate partitioned Runge-Kutta methods*, BIT, 41 (2001), pp. 504–514.

[11] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations* I: *Stiff and Differential Algebraic Equations*, Springer-Verlag, Berlin, 1996.

[12] W. D. Henshaw, *A fourth-order accurate method for the incompressible Navier-Stokes equations on overlapping grids*, J. Comput. Phys., 113 (1994), pp. 13–25.

[13] R. Hixon, *Prefactored small-stencil compact schemes*, J. Comput. Phys., 165 (2000), pp. 522–541.

[14] M. Holmström, *Solving hyperbolic PDEs using interpolating wavelets*, SIAM J. Sci. Comput., 21 (1999), pp. 405–420.

[15] X. Huang and X. Zhang, *Block-structured adaptive mesh refinement for computational aeroacoustics*, Internat. J. Comput. Fluid Mech., submitted.

[16] L. Jameson, *A wavelet-optimized, very high order adaptive grid and order numerical method*, SIAM J. Sci. Comput., 19 (1998), pp. 1980–2013.

[17] L. Jameson, *AMR Versus High-Order Schemes, Wavelets as a Guide*, Technical report UCRL-ID-141537, Lawrence Livermore National Laboratory, Livermore, CA, 2000.

[18] L. Jameson, *High-Order Schemes for RM and RT*, Technical report UCRL-ID-141537, Lawrence Livermore National Laboratory, Livermore, CA, 2000.

[19] L. Jameson, *AMR versus high order schemes*, J. Sci. Comput., 18 (2003), pp. 1–24.

[20] L. Jameson and T. Miyama, *Wavelet analysis and ocean modeling: Dynamically adaptive numerical method WOFD-AHO*, Monthly Weather Review, 128 (2000), pp. 1536–1548.

[21] C. A. Kennedy and M. H. Carpenter, *Several new numerial methods for compressible shear layer simulations*, Appl. Numer. Math., 14 (1994), pp. 397–433.

[22] C. A. Kennedy and M. H. Carpenter, *Comparison of Several Numerical Methods for Simulation of Compressible Shear Layers*, Technical report 3484, National Aeronautical and Space Administration, Langley Research Center, Hampton, VA, 1997.

[23] C. A. Kennedy and M. H. Carpenter, *Additive Runge-Kutta schemes for convection-diffusion-reaction equations*, Appl. Numer. Math., 44 (2003), pp. 139–181.

[24] S. Lefantzi, J. Ray, C. A. Kennedy, and H. N. Najm, *A component-based toolkit for reacting flows with high order spatial discretizations on structured adaptively refined meshes*, Prog. Comput. Fluid Dyn., 5 (2005), pp. 298–315.

[25] K. Mattsson and J. Nordström, *Summation by parts operators for finite difference approximations of second derivatives*, J. Comput. Phys., 199 (2004), pp. 503–540.

[26] P. Olsson, *Summation by parts, projections, and stability.* I, Math. Comput., 64 (1995), pp. 1035–1065.

[27] P. Olsson, *Summation by parts, projections, and stability.* II, Math. Comput., 64 (1995), pp. 1473–1493.

[28] A. D. Polyanin and V. F. Zaitsev, *Handbook of Nonlinear Partial Differential Equations*, Chapman and Hall/CRC, Boca Raton, FL, 2004.

[29] J. Ray and L. Jameson, *Estimation of shock induced vorticity on irregular gaseous interfaces: A wavelet-based approach.*, Shock Waves, 14 (2005), pp. 147–160.

[30] R. Samtaney, *Suppression of the Richtmyer–Meshkov instability in the presence of a magnetic field*, Phys. Fluids, 15 (2003), pp. L53–L56.

[31] B. Strand, *Numerical studies of hyperbolic IBVP with high-order finite difference operators satisfying a summation by parts rule*, Appl. Numer. Math., 26 (1998), pp. 497–521.

[32] J. C. Strikwerda, *Finite Difference Schemes and Partial Differential Equations*, 2nd ed., SIAM, Philadelphia, 2004.

[33] M. Svärd and J. Nordström, *On the Order of Accuracy for Difference Approximations of Initial-Boundary Value Problems*, Technical report 2004-040, Department of Information Technology, Uppsala University, Uppsala, Sweden, 2004.

[34] C. K. W. Tam and K. A. Kurbatskii, *Multi-size-mesh multi-time-step dispersion-relation-preserving scheme for multiple-scales aeroacoustic problems*, Int. J. Comput. Fluid Dyn., 17 (2003), pp. 119–132.

[35] C. K. W. Tam and J. C. Webb, *Dispersion-relation-preserving finite difference schemes for computational acoustics*, J. Comput. Phys., 10 (1993), pp. 262–281.

[36] L. N. Trefethen and J. A. C. Weideman, *Two results on polynomial interpolation in equally spaced points*, J. Approx. Theory, 65 (1991), pp. 247–260.

[37] S. Wolfram, *The Mathematica Book*, Wolfram Media, Inc., Champaign, IL; Cambridge University Press, Cambridge, UK, 1999.

SIAM J. SCI. COMPUT.
Vol. 29, No. 1, pp. 182–196
© 2007 Society for Industrial and Applied Mathematics

PRECONDITIONED ITERATIVE SOLVERS FOR INDUCTANCE EXTRACTION OF VLSI CIRCUITS*

HEMANT MAHAWAR† AND VIVEK SARIN†

Abstract. Parasitic extraction techniques are used to estimate signal delay in VLSI circuits. Inductance extraction is a critical component of the parasitic extraction process that involves estimation of on-chip inductive effects with high accuracy. The problem requires the solution of a large, dense, complex linear system of equations, where the unknown current must satisfy the constraint imposed by Kirchoff's current law. In this paper, we describe a solenoidal basis method to transform the constrained linear system into an unconstrained one. The method uses discrete local solenoidal flows to represent the unknown current, and to obtain a reduced linear system. The paper proposes preconditioning techniques that do not require explicit construction of the reduced system. Numerical experiments are presented to illustrate the effectiveness of the preconditioning approach. Comparisons with a well-known inductance extraction package are provided to highlight the advantages of the proposed scheme.

Key words. iterative methods, preconditioning, solenoidal basis, inductance extraction

AMS subject classifications. 65F10, 65N22

DOI. 10.1137/040609628

1. Introduction.

Signal integrity and timing analyses of modern VLSI circuits depend on accurate estimates of signal delay in various parts of the circuit. A significant portion of this delay can be attributed to the circuit's , which arises from undesirable resistive, capacitive, and inductive coupling among circuit components. The process of determining the parasitic impedance in a VLSI circuit is known as . It is a common practice to compute signal delay in low-frequency circuits using estimates of the capacitive and resistive parasitics only. At high frequencies, however, the effect of the inductive parasitic cannot be ignored. For instance, the inductive coupling tends to dominate the parasitic impedance for circuits operating at frequencies in the gigahertz (GHz) range. Reducing the resistance of circuit components through the use of low resistive materials and wider interconnect segments also increases the effect of the inductive parasitic. It is widely believed that fast and accurate inductance extraction techniques that are capable of handling large complicated layouts are critical to the design of next-generation VLSI circuits.

In a VLSI circuit, changes in current flow create a varying magnetic field that leads to inductive coupling among different components. This effect is more pronounced when these components are in close physical proximity. Chips designed with submicron VLSI technology are prone to parasitic inductive effects because of tightly packed components. The property of an electrical circuit to resist change in its own current due to the presence of an associated magnetic field is called . Self-inductance of a conductor refers to the resistance offered to current flow by the magnetic field induced by its own current. Mutual inductance between a pair of conductors refers to the resistance in one conductor due to current flow in the other. The

*Received by the editors June 7, 2004; accepted for publication (in revised form) July 28, 2006; published electronically January 26, 2007. This work has been supported in part by NSF grants NSF-CCR 9984400 and NSF-CCR 0113668, and by the Texas Advanced Technology Program grant 000512-0266-2001.

http://www.siam.org/journals/sisc/29-1/60962.html

†Department of Computer Science, Texas A&M University, College Station, TX 77843-3112 (mahawarh@cs.tamu.edu, sarin@cs.tamu.edu).

process of estimating the inductive coupling between the components of a layout is called [illegible].

A magneto-quasi-static approach is often used to compute the parasitic inductance for a set of conductors at a particular frequency. The surface of each conductor is discretized by a uniform two-dimensional mesh that represents a network of smaller conductors or filaments, and a linear system of equations is solved to determine the inductive coupling [4]. The linear system is derived from Kirchoff's current and voltage laws that govern the current flow into mesh nodes and the potential drop across mesh edges, respectively. The coefficient matrix is large and complex, with a sparse-dense structure. This matrix consists of a dense submatrix arising from the inductive coupling among filaments and a sparse submatrix arising from the current law. The cost of computing and storing the dense submatrix becomes prohibitive as the problem size grows beyond a few thousand filaments. Matrix-vector products with the dense submatrix are computed approximately via hierarchical multipole-based methods such as the fast multipole method (FMM) [3, 7] and the Barnes–Hut method [1]. These methods do not require construction of a coefficient matrix. Since the dense submatrix is not available explicitly, the linear system must be solved by an iterative solver such as the generalized minimum residual (GMRES) method [10] that relies only on the ability to compute matrix-vector products with the coefficient matrix. In principle, this approach can lead to a matrix-free algorithm that does not require explicit computation of the coefficient matrix. In practice, however, one may be forced to compute "sparse" approximations of the matrix that can then be used to construct preconditioners based on incomplete factorization schemes.

Widely available software for inductance extraction include academic codes such as FastHenry [4], and commercial codes such as Momentum and Columbus. FastHenry uses the approach described earlier to compute accurate estimates of a circuit's parasitic inductance. Although the code is used as a benchmark for accuracy comparison, it has found limited use in the VLSI-CAD (computer-aided design) community due to the high cost of computing sparse approximations and the ineffectiveness of the resulting preconditioners. This paper presents an inductance extraction algorithm that uses a solenoidal basis approach to construct effective preconditioners for the linear systems. Preliminary experiments presented in [5, 6] highlight the performance of the method. This paper analyzes the numerical aspects of the algorithm with particular emphasis on the effectiveness of the preconditioning approach.

The paper is organized as follows. Section 2 presents the mathematical formulation of the inductance extraction problem and the associated linear system of equations. Section 3 outlines the solenoidal basis method that is used to transform the linear system into a reduced system. This is followed by a description of the preconditioning approach in section 4. The inductance extraction algorithm is presented in section 5. This section also includes an overview of the hierarchical multipole-based methods for computing dense matrix-vector products and a comparison with FastHenry. Conclusions are presented in section 6.

2. Mathematical formulation. Inductance extraction of a layout with n_s conductors involves computing an $n_s \times n_s$ impedance matrix $\mathbf{Z}$, where $\mathbf{Z}_{ij}$ represents the inductive coupling between conductors i and j at a given frequency f. To determine the jth column of $\mathbf{Z}$, an alternating current source of unit magnitude and frequency f is applied to the jth conductor. The resulting potential drop across conductor i gives the value of $\mathbf{Z}_{ij}$. The potential drop across all the conductors can be determined simultaneously by solving a linear system of equations with a right-hand side that is

derived from the boundary condition on the jth conductor. One needs to solve n_s instances of the linear system, each with a different right-hand side, to compute the impedance matrix $\mathbf{Z}$.

A number of techniques based on the integral form of Maxwell's equations have been used to model VLSI circuits [2, 8]. Under magneto-quasi-static assumptions, current density $\mathbf{J}(\mathbf{r})$ and potential $\phi(\mathbf{r})$ obey the following equations at steady state:

$$\rho \mathbf{J}(\mathbf{r}) + j\omega \int_V \frac{\mu}{4\pi} \frac{\mathbf{J}(\mathbf{r}')}{\|\mathbf{r} - \mathbf{r}'\|_2} dV' = -\nabla\phi(\mathbf{r}), \tag{2.1}$$

$$\nabla \cdot \mathbf{J}(\mathbf{r}) = 0, \tag{2.2}$$

where $\mathbf{r}$ and $\mathbf{r}'$ denote three-dimensional position vectors, V is the volume of the conductors, dV' is the incremental volume with respect to $\mathbf{r}'$, μ is the magnetic permeability, ρ is the resistivity of the material, ω is the angular frequency that equals $2\pi f$, and $j = \sqrt{-1}$. The magneto-quasi-static assumption ignores the displacement current.

To obtain a numerical solution of (2.1)–(2.2), each conductor surface is discretized by a two-dimensional mesh of filaments that form the edges of the mesh. Within each filament, current density is assumed to be constant. Furthermore, current is assumed to flow along the length of a filament. The vector of unknown filament currents, $\mathbf{I}_f$, is related to the vector of unknown potential differences across filament end-points, $\mathbf{V}_f$, as shown below:

$$[\mathbf{R} + j\omega\mathbf{L}]\,\mathbf{I}_f = \mathbf{V}_f, \tag{2.3}$$

where $\mathbf{R}$ is an $m \times m$ real diagonal matrix of filament resistances for a mesh with m filaments, and $\mathbf{L}$ is an $m \times m$ complex, dense inductance matrix. The kth diagonal element of $\mathbf{R}$ is assigned the value $\rho l_k / a_k$, where l_k and a_k are the length and cross-sectional area, respectively, of the kth filament. The inductance matrix $\mathbf{L}$ is symmetric positive definite with entries:

$$\mathbf{L}_{kl} = \frac{\mu}{4\pi} \frac{1}{a_k a_l} \int_{r_k \in V_k} \int_{r_l \in V_l} \frac{\mathbf{u}_k \cdot \mathbf{u}_l}{\|\mathbf{r}_k - \mathbf{r}_l\|_2} dV_k dV_l,$$

where $\mathbf{u}_k$ denotes the unit vector along the kth filament, and $\mathbf{r}_k$ and $\mathbf{r}_l$ denote position vectors for points in filaments k and l, respectively. The integral is calculated over the volumes of the two filaments. Since $\mathbf{R}$ and $\mathbf{L}$ are symmetric positive definite matrices, the coefficient matrix in (2.3) is nonsingular.

Kirchoff's current law (2.2) imposes an additional set of constraints on the filament current vector:

$$\mathbf{B}^T \mathbf{I}_f = \mathbf{I}_s, \tag{2.4}$$

where $\mathbf{B}^T$ is the branch index matrix and $\mathbf{I}_s$ is the external source current. The branch index matrix for a mesh with n nodes and m filaments is a matrix of size $n \times m$, in which the (k, l) entry has the value -1 if the lth filament originates at node k, 1 if it terminates at k, and 0 otherwise. For example, consider a mesh with an $n_x \times n_y$ array of cells in which the nodes and filaments have a natural ordering starting from the left bottom corner. Horizontal filaments are ordered before vertical filaments. The branch index matrix is the following matrix with $(n_y + 1)$ block rows

and $(2n_y + 1)$ block columns:

$$\mathbf{B}^T = \left[\begin{array}{ccccc|ccccc} T & & & & & I & & & & \\ & T & & & & -I & I & & & \\ & & T & & & & -I & \ddots & & \\ & & & \ddots & & & & \ddots & I & \\ & & & & T & & & & -I & \end{array}\right],$$

where I is the identity matrix of size $(n_x + 1) \times (n_x + 1)$ and T is a matrix of size $(n_x + 1) \times n_x$ given below:

$$T = \begin{bmatrix} 1 & & & \\ -1 & 1 & & \\ & -1 & \ddots & \\ & & \ddots & 1 \\ & & & -1 \end{bmatrix}.$$

The potential difference across the filaments can be expressed in terms of the node potential:

$$\mathbf{V}_f = \mathbf{B}\mathbf{V}_n, \tag{2.5}$$

where $\mathbf{V}_n$ is a vector of node potentials defined with respect to the ground. Equations (2.3), (2.4), and (2.5) give rise to a linear system of equations governing filament currents and node potentials:

$$\begin{bmatrix} \mathbf{R} + j\omega\mathbf{L} & -\mathbf{B} \\ \mathbf{B}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{I}_f \\ \mathbf{V}_n \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{I}_s \end{bmatrix}. \tag{2.6}$$

The coefficient matrix of the linear system (2.6) is a sparse-dense matrix of size $(n+m)\times(n+m)$ in which the first diagonal block is dense and the off-diagonal blocks are sparse. Even for a small problem with a few thousand filaments, it is prohibitively expensive to compute and store $\mathbf{L}$. On the other hand, one can use "fast" hierarchical multipole-based methods to compute approximate matrix-vector products with $\mathbf{L}$. When filaments are uniformly distributed, the FMM requires $O(m)$ operations, while multipole-based variants of the Barnes–Hut method [1] require $O(m \log m)$ operations to compute these matrix-vector products. Higher degree multipoles can be used to reduce the approximation error. However, the computational cost grows proportional to d^4, where d is the multipole degree. Due to the unavailability of the coefficient matrix, the linear system (2.6) has to be solved by iterative techniques such as GMRES that rely solely on the ability to compute matrix-vector products with the coefficient matrix. Robust and effective preconditioners are necessary to accelerate the convergence of the solver. Construction of preconditioners in the absence of the coefficient matrix turns out to be a formidable challenge.

3. A solenoidal basis method for inductance extraction. Solenoidal functions are divergence-free functions that satisfy conservation laws such as Kirchoff's current law in electric circuits and the mass conservation law in fluid mechanics. For a uniform mesh, it is easy to construct a solenoidal basis that consists of divergence-free functions in the discrete sense. Figure 3.1 shows several instances of discrete

solenoidal current flows that can be used to construct a solenoidal basis. Each flow consists of a constant amount of current flowing anticlockwise through the four filaments of a cell in the mesh. These flows satisfy Kirchoff's current law since the net flow of current into any node is zero.

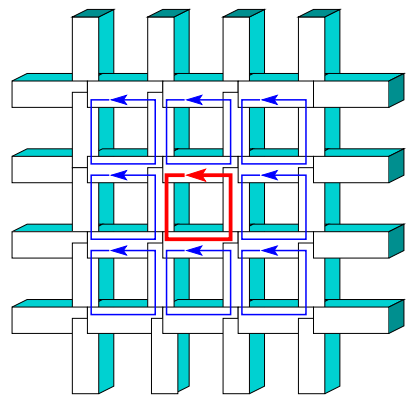


FIG. 3.1. *A set of nine solenoidal current flows in a section of a uniform two-dimensional mesh.*

A basis consisting of such functions can be viewed as a [illegible] in which the basis functions have local support on the mesh. The solenoidal basis matrix $\mathbf{P}$ is an $m \times s$ matrix that is derived from the current flows in mesh cells. Each column corresponds to a cell in the mesh and consists of four nonzero entries that denote the current flow in the cell: 1 indicates a unit current flow along the edge, and -1 indicates a unit current flow opposite the direction of the edge. Construction of the solenoidal basis matrix in this manner ensures that the following condition is satisfied:

$$\mathbf{B}^T\mathbf{P} = 0. \tag{3.1}$$

Again, consider a mesh with an $n_x \times n_y$ array of cells in which the nodes and filaments have a natural ordering starting from the left bottom corner. Horizontal filaments are ordered before vertical filaments. The solenoidal basis matrix is the following matrix with n_y block rows and $(2n_y + 1)$ block columns:

$$\mathbf{P}^T = \left[\begin{array}{cccc|ccc} I & -I & & & W & & \\ & I & -I & & & W & \\ & & \ddots & \ddots & & & \ddots \\ & & & I \quad -I & & & & W \end{array}\right],$$

where I is the identity matrix of size $n_x \times n_x$ and W is a matrix of size $n_x \times (n_x + 1)$ given below:

$$W = \left[\begin{array}{ccccc} 1 & -1 & & & \\ & 1 & -1 & & \\ & & \ddots & \ddots & \\ & & & 1 & -1 \end{array}\right].$$

A local solenoidal basis for a two-dimensional mesh is a complete basis with linearly independent components. The linear independence of the columns of $\mathbf{P}$ is established by observing that the matrix $\mathbf{P}^T\mathbf{P}$ is the standard two-dimensional Laplace

operator matrix. In a uniform two-dimensional mesh consisting of an $n_x \times n_y$ array of cells, the numbers of nodes, edges, and cells are given by $n = (n_x + 1)(n_y + 1)$, $m = 2n - n_x - n_y - 2$, and $s = m - n + 1$, respectively. Since the number of cells equals the dimension of the null space of the discrete divergence matrix $\mathbf{B}^T$, it follows that the local solenoidal basis is complete. It may be noted that $\mathbf{B}$ has a rank-deficiency of 1 to allow the potential to vary by a constant. The rank deficiency can be removed by specifying the value of the potential at a single node.

The linear system (2.6) must be transformed before one can use the solenoidal basis method. The first step is to determine a particular current vector $\mathbf{I}_p$ that satisfies the constraints imposed by the external current source. The vector $\mathbf{I}_p$ represents current flow along an arbitrary path between the nodes, where the external source is connected (see, e.g., Figure 3.2). By splitting the current $\mathbf{I}_f$ into a known component $\mathbf{I}_p$ and an unknown component $\mathbf{I}$, the linear system (2.6) can be transformed into an equivalent system with a different right-hand side:

$$\begin{bmatrix} \mathbf{R} + j\omega\mathbf{L} & -\mathbf{B} \\ \mathbf{B}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{I} \\ \mathbf{V}_n \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ \mathbf{0} \end{bmatrix}, \tag{3.2}$$

where

$$\mathbf{I} = \mathbf{I}_f - \mathbf{I}_p, \qquad \mathbf{F} = -\left[\mathbf{R} + j\omega\mathbf{L}\right]\mathbf{I}_p.$$

The difference between (2.6) and (3.2) is that the first system satisfies current boundary conditions, whereas the second system satisfies voltage boundary conditions.

Note that the linear system in (3.2) includes the constraint $\mathbf{B}^T\mathbf{I} = \mathbf{0}$, which ensures that the net current flowing into every node is zero, as required by Kirchoff's law. Thus, $\mathbf{I} \in Null(\mathbf{B}^T)$. Since $\mathbf{P}$ is a basis for $Null(\mathbf{B}^T)$, the unknown current $\mathbf{I}$ can be represented as a linear combination of the columns of $\mathbf{P}$:

$$\mathbf{I} = \mathbf{P}x,$$

where x is an unknown vector of size s. This is equivalent to representing current as a weighted sum of local solenoidal flows, where the weights are given by x. From (3.1), it follows that $\mathbf{I}$ must satisfy the constraint $\mathbf{B}^T\mathbf{I} = \mathbf{0}$. By restricting the unknown current to the solenoidal subspace in this manner, the linear system (3.2) can be transformed into the following system:

$$\left[\mathbf{R} + j\omega\mathbf{L}\right]\mathbf{P}x - \mathbf{B}\mathbf{V}_n = \mathbf{F}.$$

The vector of unknown node potentials can be eliminated by multiplying the system with $\mathbf{P}^T$ from the left:

$$\mathbf{P}^T\left[\mathbf{R} + j\omega\mathbf{L}\right]\mathbf{P}x = \mathbf{P}^T\mathbf{F}. \tag{3.3}$$

The above system is a linear system of order s that must be solved to determine x.

A physical interpretation of transforming (3.2) into (3.3) can be obtained by noting that $\mathbf{P}$ and $\mathbf{P}^T$ implement discrete curl operators on the conductor surface. The vector x corresponds to magnetic field normal to the surface, whose curl determines the electric field which is proportional to $\mathbf{I}$. The transformation itself can be viewed as a conversion of variables from electric field to magnetic field.

Note that the coefficient matrix in (3.3) is nonsingular because $\mathbf{R}$ and $\mathbf{L}$ are symmetric positive definite matrices and $\mathbf{P}$ is a full rank matrix. Since the coefficient

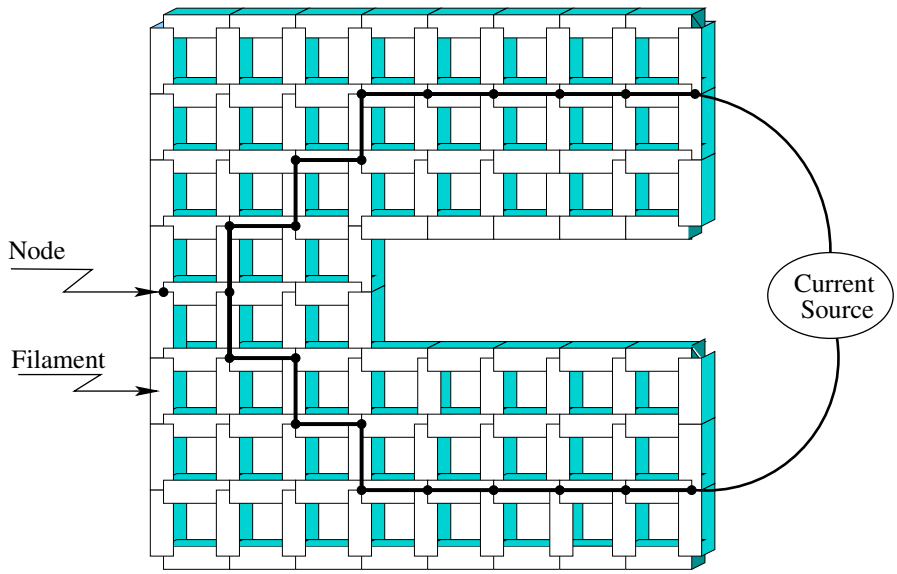


FIG. 3.2. *Current flow along an arbitrary path in the mesh can be used to satisfy the constraints imposed by the external current source.*

TABLE 3.1
The sizes of the original and reduced systems arising in the inductance extraction of a ground plane.

Mesh Size ($n_x \times n_y$)	Nodes (n)	Filaments (m)	Unknowns ($n+m$)	Solenoidal functions (s)
64×64	4225	8320	12545	4096
128×128	16641	33024	49665	16384
256×256	66049	131584	197633	65536
512×512	263169	525312	788481	262144

matrix is never computed explicitly, an iterative method such as GMRES must be used to solve the system. Once x has been computed, the filament current vector, $\mathbf{I}_f$, and the vector of unknown potential differences across the filaments, $\mathbf{V}_f$, can be computed as follows:

$$\mathbf{I}_f = \mathbf{I}_p + \mathbf{P}x, \qquad \mathbf{V}_f = [\mathbf{R} + j\omega\mathbf{L}]\,\mathbf{I}_f.$$

When a unit current is applied by the external source, the impedance between any pair of nodes is equal to the potential difference between the nodes. This potential difference can be calculated by adding the potential differences across the filaments on an arbitrary path between the nodes.

The transformation of the linear system (3.2) into the reduced system (3.3) has several advantages. The number of unknowns reduces considerably whenever two-dimensional discretizations are employed. Table 3.1 shows the number of unknowns in a ground plane problem that involves computing the self-impedance of a square conductor. The use of a local solenoidal basis results in a sparse matrix $\mathbf{P}$ that is amenable to efficient matrix-vector product computations. Matrix-free implementations are also possible since explicit construction of $\mathbf{P}$ is unnecessary. In addition, the sparsity of $\mathbf{P}$ is instrumental in improving the efficiency of parallel implementations.

4. Preconditioning the reduced system. The main challenge in solving the reduced system (3.3) iteratively is posed by the preconditioning step. The ability to precondition the reduced system effectively is critical to the success of the solenoidal basis method. The rate of convergence of the iterative solver depends on the spectral properties of the coefficient matrix. Preconditioning may be considered as a process of transforming a linear system into one that has more favorable spectral properties. For instance, the linear system $\mathbf{A}x = b$ may be preconditioned by postmultiplying $\mathbf{A}$ with a matrix $\mathbf{M}$ as shown below:

$$\mathbf{A}\mathbf{M}y = b, \qquad x = \mathbf{M}y. \tag{4.1}$$

The transformed system is solved by an iterative method. Each iteration now requires an additional preconditioning step that involves a matrix-vector product with $\mathbf{M}$. A preconditioning approach is advantageous only if the overall time to compute the solution is reduced. For this to happen, the preconditioner must be easy to compute, the preconditioning step must be relatively inexpensive, and $\mathbf{M}$ should be an effective preconditioner that reduces the number of iterations considerably. Preconditioning can be done from the left side by premultiplication, from the right side by postmultiplication, or from both sides [9].

A good preconditioner can be characterized in a variety of ways. In general, clustering of eigenvalues of the preconditioned system can lead to rapid convergence. In many cases, a preconditioned system with a small condition number can be solved in a few iterations. A preconditioning approach is considered optimal if the number of iterations required is independent of the discretization parameters.

It is common to use the symbol $\mathbf{M}^{-1}$ instead of $\mathbf{M}$ in (4.1) to indicate that the preconditioner is an approximation of the matrix $\mathbf{A}$ and that the preconditioning step requires the solution of a linear system with the preconditioner as the coefficient matrix. Such approximations are obtained implicitly by computing incomplete factorizations of $\mathbf{A}$. Although the coefficient matrix in the reduced system (3.3) is not available, one can compute a "sparse" approximation by ignoring interactions between distant filament pairs. An incomplete factorization of this sparse matrix yields L and U factors that can be used to define the preconditioner for the reduced system.

FastHenry [4] uses a similar approach in which the matrix is sparsified by several different strategies in order to obtain inexpensive preconditioners. Unfortunately, the sparsification schemes in the package tend to be very expensive from the perspective of computation and memory requirements. This has restricted the use of the code to solving small benchmark problems only.

An effective preconditioning approach can be developed by analyzing the reduced system (3.3) carefully. The use of local solenoidal flows defined on a uniform two-dimensional mesh provides a basis for this analysis. Consider the matrices that form the reduced system: $\mathbf{R}$ is a diagonal matrix with positive entries, $\mathbf{L}$ is a dense symmetric positive definite matrix, and $\mathbf{P}$ is a sparse matrix. The matrices $\mathbf{P}$ and $\mathbf{P}^T$ implement discrete curl operators on the conductor surface, and $\mathbf{P}^T\mathbf{P}$ is a discrete two-dimensional Laplace operator. Table 4.1 provides estimates of the largest and smallest eigenvalues of these matrices for a discretization with filament length h. Discretizations of the conductor surfaces must use filaments with a fixed ratio between length and cross-section area to be consistent with physical laws. As a result, the eigenvalues of $\mathbf{R}$ are independent of h.

At higher frequencies, the reduced system is dominated by the imaginary part, whose condition number is proportional to h^{-1}. The real part dominates at lower

TABLE 4.1
Estimates of the extremal eigenvalues of matrices that form the reduced system.

Matrix	$\lambda_{\min}$	$\lambda_{\max}$
$\mathbf{R}$	c	c
$\mathbf{L}$	$O(h)$	$O(1)$
$\mathbf{P}^T\mathbf{R}\mathbf{P}$	$O(h^2)$	$O(1)$
$\mathbf{P}^T\mathbf{L}\mathbf{P}$	$O(h^2)$	$O(h)$
$Re(\mathbf{P}^T[\mathbf{R}+j\omega\mathbf{L}]\mathbf{P})$	$O(h^2)$	$O(1)$
$Im(\mathbf{P}^T[\mathbf{R}+j\omega\mathbf{L}]\mathbf{P})$	$O(\omega h^2)$	$O(\omega h)$

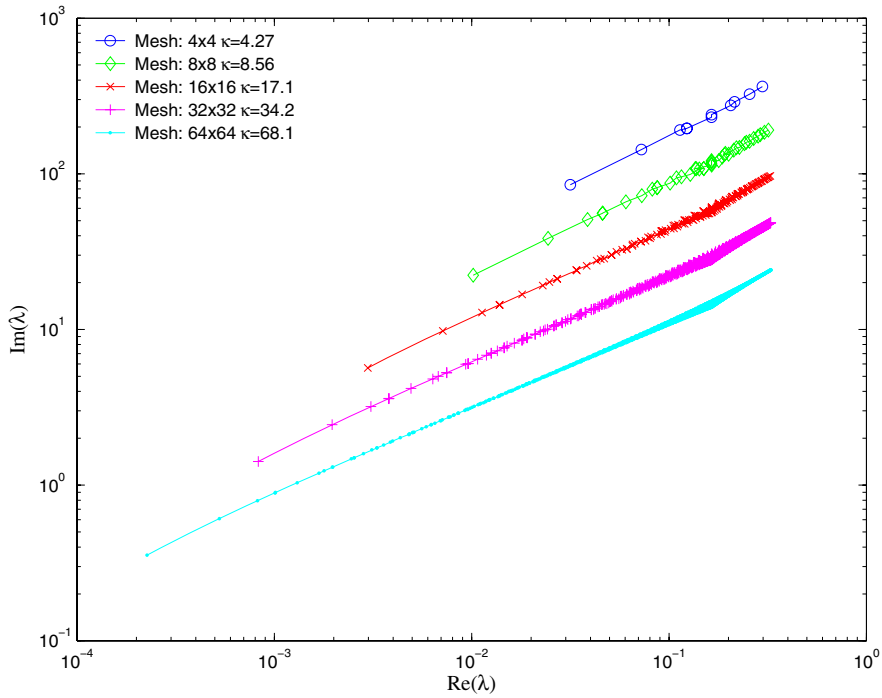


FIG. 4.1. *The spectrum of the reduced system that arises from a uniform discretization of the square ground plane problem ($\omega = 2\pi \times 10$ GHz). The mesh size is $n_x \times n_x$ and the filament length is $h = 1/n_x$. The condition number of the system is denoted by κ.*

frequencies, and the condition number grows proportional to h^{-2}. Figure 4.1 shows the spectrum of the reduced system (3.3) obtained from the ground plane problem with $\omega = 2\pi \times 10$ GHz. A uniform two-dimensional mesh is used to discretize the ground plane of size 1 cm $\times$ 1 cm. The eigenvalues of the matrix lie on a straight line in the complex plane. The condition number of the reduced system doubles every time the mesh is refined.

The entries of $\mathbf{L}$ are derived from Green's function for the three-dimensional Laplace operator. While $\mathbf{P}^T\mathbf{P}$ is a two-dimensional Laplace operator with a condition number proportional to h^{-2}, the matrix $\mathbf{P}^T\mathbf{L}\mathbf{P}$ tends to have a condition number that is proportional to h^{-1} only. Furthermore, as shown in Figure 4.2, the matrix $\mathbf{L}\mathbf{P}$ is a well-conditioned matrix with a condition number that is nearly independent of h. To exploit this fact, we express $\mathbf{L}\mathbf{P}$ as shown below:

$$\mathbf{L}\mathbf{P} = \mathbf{P}\tilde{\mathbf{L}} + \hat{\mathbf{P}}\hat{\mathbf{L}},$$

where $\tilde{\mathbf{L}}$ represents the mutual inductance among hypothetical filaments placed at the center of each mesh cell and oriented in the same direction, $\hat{\mathbf{L}}$ represents the inductive effect of these filaments on the boundary of the conductors, and $\hat{\mathbf{P}}$ updates the boundary filaments. This representation can be viewed as a set of filaments placed at the centers of the mesh cells with an additional set of ghost cells along the boundary. The filaments at the centers of the ghost cells carry no current. Note that $\hat{\mathbf{P}}\hat{\mathbf{L}}$ is a low-rank matrix, whose influence is limited to the boundary. Since it has a marginal effect on the singular values of $\mathbf{LP}$, one can expect $\mathbf{P}\tilde{\mathbf{L}}$ to be a good approximation of $\mathbf{LP}$. Figure 4.2 shows that a large number of singular values of these two matrices are identical. A theoretical estimate of the quality of this approximation remains an open problem.

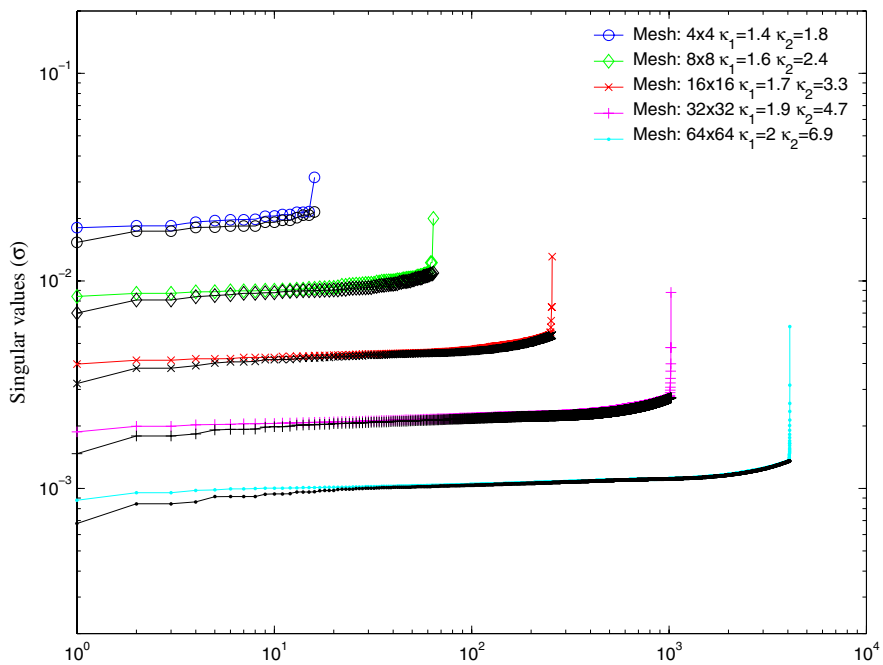


FIG. 4.2. *Singular values of* $\mathbf{LP}$ *(bottom line) and* $\mathbf{P}\tilde{\mathbf{L}}$ *(top line) of the square ground plane problem* ($\omega = 2\pi \times 10$ *GHz) for different meshes. The mesh size is* $n_x \times n_x$ *and the filament length is* $h = 1/n_x$*. The condition numbers of* $\mathbf{LP}$ *and* $\mathbf{P}\tilde{\mathbf{L}}$ *are denoted by* κ_1 *and* κ_2*, respectively.*

The matrix $\tilde{\mathbf{L}}$ is a symmetric positive definite matrix defined as follows:

$$\tilde{\mathbf{L}}_{kl} = \frac{\mu}{4\pi}\frac{1}{a_k a_l}\int_{r_k \in V_k}\int_{r_l \in V_l}\frac{1}{\|\mathbf{r}_k - \mathbf{r}_l\|_2}dV_k dV_l.$$

The matrix element $\tilde{\mathbf{L}}_{kl}$ equals the mutual inductance between a pair of parallel filaments placed at the centers of cells k and l. At high frequencies, when the imaginary part of the reduced system dominates, $\tilde{\mathbf{L}}$ can be used as a preconditioner for the reduced system. Since

$$\mathbf{P}^T\mathbf{LP}\tilde{\mathbf{L}} \approx \mathbf{P}^T\mathbf{L}^T\mathbf{LP},$$

the preconditioner is expected to yield a well-conditioned system. Figure 4.3 shows that the eigenvalues of the preconditioned reduced system are clustered close together,

almost independent of the filament width h. The preconditioned system is solved in a few iterations typically, and the preconditioner appears to be effective for the mid-frequency range as well. For low frequency problems, one should use a two-dimensional Laplace matrix to precondition the real part of the reduced system. It should be noted that accurate inductance extraction may not be needed at low frequencies.

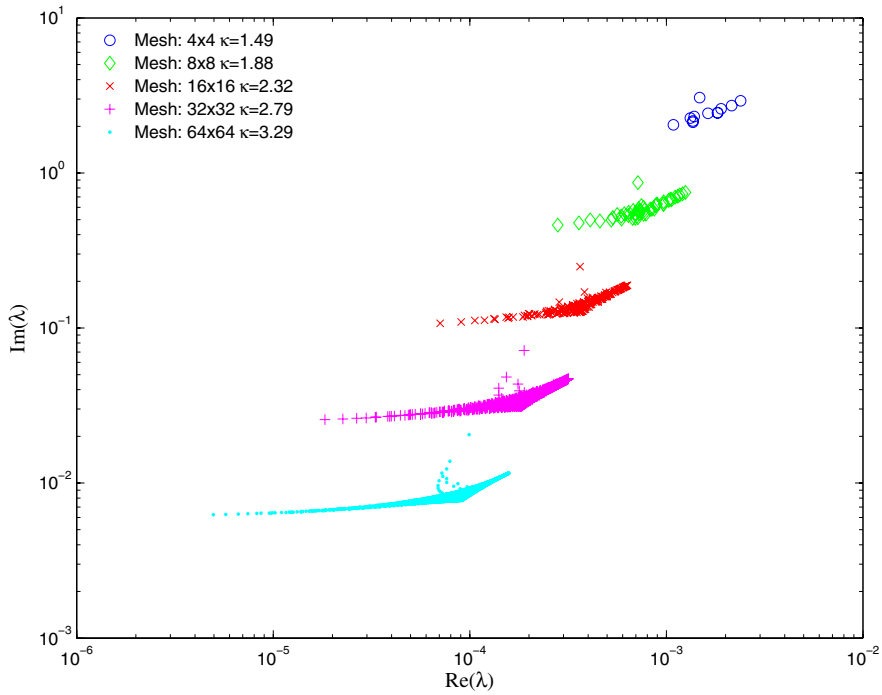


FIG. 4.3. *The spectrum of the preconditioned reduced system for the square ground plane problem ($\omega = 2\pi \times 10$ GHz). The mesh size is $n_x \times n_x$ and the filament length is $h = 1/n_x$. The condition number of the preconditioned system is denoted by κ.*

4.1. Effectiveness of the preconditioning approach. To illustrate the effectiveness of the preconditioning scheme, we consider two benchmark problems. The first problem involves computing the self-inductance of a square ground plane of size 1 cm $\times$ 1 cm. The plane is discretized by a uniform two-dimensional mesh with mesh width h varying from 2^{-5} cm to 2^{-9} cm. The width of each filament is one-third of its length, and the thickness is 2^{-13} cm. A tolerance of 10^{-6} was specified on the relative residual norm. Table 4.2 shows the number of iterations required by the right-preconditioned GMRES method to solve the reduced system. It can be seen that the number of iterations remains nearly constant across a large frequency range. The slow rate of convergence of the GMRES method without any preconditioner can be gauged from the iterations shown in parentheses.

The second benchmark problem is the cross-over problem shown in Figure 4.4. In VLSI circuits, this is a typical layout configuration with interconnect segments crossing each other on different layers. Each conductor is a rectangle of size 2 cm $\times$ 2 mm. The discretization is similar to the first problem. These simulations were con-

TABLE 4.2
Ground plane: Number of iterations required by the preconditioned GMRES method. Unpreconditioned GMRES iterations are shown in parentheses.

Mesh size	Filament length (cm)	Frequency			
		100 Mhz	1GHz	10 GHz	100 GHz
32×32	1/32	10 (34)	8 (30)	8 (30)	8 (30)
64×64	1/64	14 (59)	9 (43)	9 (43)	9 (43)
128×128	1/128	19 (108)	10 (61)	10 (59)	10 (59)
256×256	1/256	26 (206)	12 (90)	12 (82)	11 (82)
512×512	1/512	34 (–)	14 (–)	12 (–)	12 (–)

TABLE 4.3
Cross-over: Number of iterations required by the preconditioned GMRES method ($\omega = 2\pi \times 10$ GHz). Unpreconditioned GMRES iterations are shown in parentheses.

Mesh size	Filament length (cm)	Conductor layout		
		1+1	2+2	4+4
16×160	1/80	10–11 (34–35)	12–13 (37–38)	14–15 (40–41)
32×320	1/160	12 (49–50)	14 (54)	16 (56–58)
64×640	1/320	15 (72–76)	17 (79–81)	19 (83–88)
128×1280	1/640	16–17 (113–114)	18–19 (121–124)	20–21 (–)

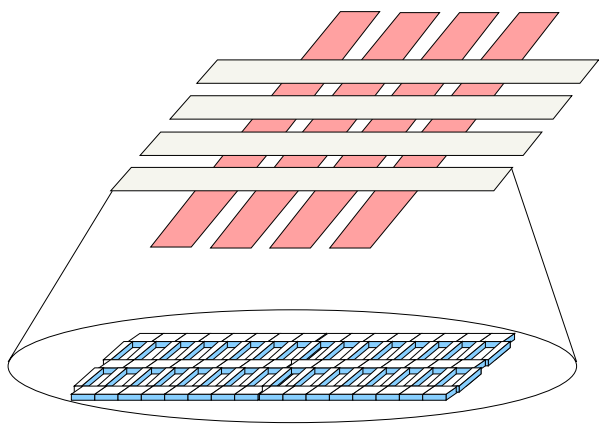


FIG. 4.4. *The cross-over problem with* $4 + 4$ *conductors.*

ducted for 10 GHz frequency. Table 4.3 reports the number of iterations required by the right-preconditioned GMRES method to reduce the relative residual norm below 10^{-6}. In each case, the range of iterations needed to solve n_s instances of the linear system are reported. The number of iterations grows slowly as the number of conductors in the configuration is increased. Furthermore, the growth in iterations is slow as the mesh is refined. These results illustrate the effectiveness of the preconditioning scheme for typical extraction problems.

5. The inductance extraction algorithm. An outline of the preconditioned solenoidal basis method for computing the impedance matrix $\mathbf{Z}$ is given below.

ALGORITHM 1.

1. Discretize the surfaces of the conductors with two-dimensional uniform meshes.
2. Construct an oct-tree for the filaments in each conductor.
3. Compute the solenoidal basis matrix $\mathbf{P}$.
4. For conductor $l = 1, \ldots, n_s$:
   (a) Compute the particular solution $\mathbf{I}_p^{(l)}$ for a unit current flow through conductor l and the corresponding induced potential difference vector $\mathbf{F}^{(l)}$.
   (b) Solve the preconditioned system

$$\mathbf{P}^T \left[\mathbf{R} + j\omega\mathbf{L}\right] \mathbf{P}\tilde{\mathbf{L}}x = \mathbf{P}^T\mathbf{F}^{(l)}, \qquad x^{(l)} = \tilde{\mathbf{L}}x$$

   to determine $x^{(l)}$, and compute filament current and filament potential difference vectors:

$$\mathbf{I}_f^{(l)} = \mathbf{P}x^{(l)} + \mathbf{I}_p^{(l)}, \qquad \mathbf{V}_f^{(l)} = \left[\mathbf{R} + j\omega\mathbf{L}\right] \mathbf{I}_f^{(l)}.$$

   Use the GMRES method to solve the preconditioned reduced system. Use the FMM to compute matrix-vector products with $\mathbf{L}$ and $\tilde{\mathbf{L}}$ in each iteration.
   (c) For conductor $k = 1, \ldots, n_s$, determine $\mathbf{Z}_{kl}$ by adding the potential difference across all filaments along a path between the two ends of conductor k.

In the above algorithm, a majority of the time is spent in step 4(b). The most time-consuming computations involve matrix-vector products with the reduced system matrix $\mathbf{P}^T \left[\mathbf{R} + j\omega\mathbf{L}\right] \mathbf{P}$ and with the preconditioner matrix $\tilde{\mathbf{L}}$. Products with the reduced system are computed as a sequence of three products:

$$u = \mathbf{P}x, \qquad v = \left[\mathbf{R} + j\omega\mathbf{L}\right] u, \qquad y = \mathbf{P}^T v.$$

The cost of multiplying the dense matrix $\mathbf{L}$ with a vector is significantly larger than the cost of matrix-vector products with $\mathbf{P}$ or $\mathbf{P}^T$. It is worthwhile to use multipole-based hierarchical methods to compute matrix-vector products with $\mathbf{L}$. Furthermore, since the structure of the preconditioner $\tilde{\mathbf{L}}$ is similar to $\mathbf{L}$, these fast methods should be used to compute matrix-vector products in the preconditioning step as well.

The number of operations needed to compute accurate matrix-vector products with an $m \times m$ dense matrix is $O(m^2)$. Approximations to these products can be computed efficiently through hierarchical multipole-based methods that exploit the decaying nature of the $1/r$ kernel in the entries of $\mathbf{L}$. These methods include the FMM [3, 7] and variants of the Barnes–Hut method [1]. For uniform filament distributions, the FMM reduces the computational complexity to $O(m)$, and the Barnes–Hut variants achieve $O(m \log m)$ complexity. The Barnes–Hut approach uses filament-cluster interactions, whereas the FMM uses both filament-cluster and cluster-cluster interactions.

We use a variant of the FMM to compute the potential at each filament due to current flow in all the filaments. The algorithm works in two phases: the oct-tree construction phase and the potential evaluation phase. To construct the oct-tree, the domain is divided into eight equal nonoverlapping subdomains. The process is repeated on each subdomain recursively until every subdomain has at most s filaments, where s is a parameter chosen to maximize computational efficiency. A subdomain

is represented by a subtree whose leaf nodes contain the filaments in the subdomain. The potential evaluation phase consists of two traversals of the tree. During the up-traversal, multipole coefficients are computed at each node. These coefficients can be used to compute the potential due to all the filaments within the node's subdomain at a , , , point. During the down-traversal, the multipole coefficients are used to compute local coefficients at each node. These local coefficients can be used to compute the potential at any point inside a node's subdomain due to , , , filaments. Potential due to , , filaments is computed directly.

The implementation based on the preceding approach is a matrix-free code in which neither the system matrix nor the preconditioner matrix is ever computed. This reduces the storage requirement considerably, thereby allowing larger problems to be solved. The overall complexity of Algorithm 1 is $O(i_G m)$, where m is the number of filaments and i_G is the total number of GMRES iterations required to solve n_s instances of the linear system.

5.1. Comparison with FastHenry. It is instructive to compare the performance of the preconditioned solenoidal basis method with that of FastHenry. FastHenry uses mesh currents similar to the local solenoidal flows to generate a reduced system. This system is solved by the preconditioned GMRES method. Matrix-vector products with the dense matrix $\mathbf{L}$ are computed via the FMM. The main difference between the two algorithms lies in the preconditioning step. FastHenry uses preconditioners that are derived from incomplete factorizations of sparsified forms of the reduced system. These sparse approximations are constructed explicitly by restricting the off-diagonal nonzeros in each column to those resulting from coupling between specific filament pairs. These pairs may be in the same box at a particular level in the oct-tree or may be within a fixed distance of each other.

Table 5.1 shows the execution time and the number of iterations needed by the preconditioned GMRES method to solve several instances of the cross-over problem. The column marked "Iter." gives the range of iterations needed to solve n_s instances of the linear system. A tolerance of 10^{-3} was specified on the relative residual norm. Multipoles of degree 2 were used in all the FMM computations. In accordance with standard practice, it was ensured that the error in the entries of the inductance matrix $\mathbf{Z}$ was 2% of the values obtained by a more accurate calculation based on the FMM with a higher multipole degree. These experiments were conducted on a 2.4 GHz Pentium Workstation with 1 GB of memory.

FastHenry requires a significant amount of memory to construct the sparsified matrix and compute its incomplete LU factorization. The entries marked "—" indicate the inability of FastHenry to solve the problem within the available system memory.

TABLE 5.1
Comparison of the preconditioned solenoidal basis method with FastHenry.

Conductor layout	FastHenry		Solenoidal method	
	Iter.	Time (s)	Iter.	Time (s)
Mesh size: 32×320				
1	16	23.9	5	10.0
1 + 1	19–21	72.2	7–8	39.9
2 + 2	20–23	249.9	9	197
Mesh size: 64 × 640				
1	22	217.8	7	56.3
1 + 1	—	—	9–10	205.2
2 + 2	—	—	11–12	1020.4

Furthermore, a growth in the number of iterations with mesh size indicates a suboptimal preconditioning scheme. Efforts to reduce the storage requirements through greater sparsification tend to decrease the effectiveness of the preconditioner, leading to a higher execution time.

6. Conclusions. The dense, complex linear systems of equations arising in inductance extraction of VLSI circuits can be solved efficiently by the preconditioned solenoidal basis method. The method uses local solenoidal flows to convert the system into a reduced system that is solved by an iterative method. Although the coefficient matrix of the reduced system is never computed explicitly, one can construct very effective preconditioners based on the inductive coupling among filaments of the mesh. Hierarchical multipole-based methods should be used to compute matrix-vector products with the dense matrices that form the reduced system and the preconditioner. Numerical experiments indicate that for high frequency simulations, the number of iterations required by the GMRES method to solve the preconditioned system is almost independent of the mesh width and frequency. The preconditioned solenoidal basis method outperforms FastHenry, a well-known inductance extraction package, on typical benchmark problems.

REFERENCES

[1] J. Barnes and P. Hut, *A hierarchical $O(n \log n)$ force calculation algorithm*, Nature, 324 (1986), pp. 446–449.
[2] A. C. Cangellaris, J. L. Prince, and L. P. Vakanas, *Frequency-dependent inductance and resistance calculation for three-dimensional structures in high-speed interconnect systems*, IEEE Trans. Components, Hybrids, and Manufacturing Technology, 13 (1990), pp. 154–159.
[3] L. Greengard, *The Rapid Evaluation of Potential Fields in Particle Systems*, MIT Press, Cambridge, MA, 1988.
[4] M. Kamon, M. J. Tsuk, and J. White, *FASTHENRY: A multipole-accelerated 3D inductance extraction program*, IEEE Trans. Microwave Theory Tech., 42 (1994), pp. 1750–1758.
[5] H. Mahawar and V. Sarin, *Parallel iterative methods for dense linear systems in inductance extraction*, Parallel Comput., 29 (2003), pp. 1219–1235.
[6] H. Mahawar, V. Sarin, and W. Shi, *A solenoidal basis method for efficient inductance extraction*, in Proceedings of the 39th Annual IEEE Conference on Design Automation, New Orleans, LA, 2002, pp. 751–756.
[7] V. Rokhlin, *Rapid solution of integral equations of classical potential theory*, J. Comput. Phys., 60 (1985), pp. 187–207.
[8] A. E. Ruehli and A. Cangellaris, *Progress in the methodologies for the electrical modeling of interconnects and electronic packages*, Proc. IEEE, 89 (2001), pp. 740–771.
[9] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.
[10] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.

SIAM J. SCI. COMPUT.
Vol. 29, No. 1, pp. 197–220
© 2007 Society for Industrial and Applied Mathematics

MOVCOL4: A MOVING MESH CODE FOR FOURTH-ORDER TIME-DEPENDENT PARTIAL DIFFERENTIAL EQUATIONS*

R. D. RUSSELL†, J. F. WILLIAMS†, AND X. XU†

Abstract. In this paper we develop and analyze a moving mesh code for the adaptive simulation of fourth-order PDEs based on collocation. The scheme is shown to enforce discrete conservation for problems written in a generalized conservation form. To demonstrate the breadth of applicability we present examples from both Cahn–Hilliard and thin-film–type equations exhibiting metastable behavior, finite-time solution blow-up, finite-time extinction, and moving interfaces.

Key words. moving mesh methods, collocation discretization, adaptive numerical methods, finite-time singularities, blow-up, pinch-off, thin-film equation, higher-order equations

AMS subject classifications. 65M50, 65M70, 35G25

DOI. 10.1137/050643167

1. Introduction.

Well-chosen model equations play an indispensable role in the study of complex patterns in physical, mechanical, chemical, and biological systems. Classical model equations have typically been partial differential equations (PDEs) which are second-order in space because, even though they are often much simpler than the full equations describing physical processes, they capture the essential features of certain specific problems. Familiar examples are Laplace's equation, the general heat equation, and the wave equation, designed to shed light on physical processes such as diffusion, dispersion, absorption, or wave propagation, and their mutual interaction. Nevertheless, they are ultimately limited in their ability to describe certain complex features. To gain insight into the dynamics of rich spatial and temporal patterns in a wide range of physical and mechanical problems, fourth-order model equations and systems of fourth-order equations have been proposed.

Examples of physical applications of these include problems in thin-film theory, lubrication theory, convection-explosion theory, flame and wave propagation, phase transition at critical Lipschitz points, and bistable systems. In recent years, fourth-order problems have also become important in image processing and for modeling diffusion processes in physics and material sciences. Thus, fourth-order terms are increasingly recognized as being significant in many physical models, and this has led to a burgeoning literature, including the recent book [36], which lists a number of models and references to the development of analytical techniques for the study of certain fourth-order problems.

Unlike studies for second-order equations, the theoretical study of higher-order equations, even semilinear ones, is far from complete. The main reason is that many effective methods used in treating second-order equations, such as those based on maximum principles, variational structure, and self-adjointness properties, are not applicable for higher-order equations. As in other areas, where the analytical theory is underdeveloped, many crucial features have been first explored by careful numerical

*Received by the editors October 19, 2005; accepted for publication (in revised form) August 3, 2006; published electronically February 2, 2007. The first and third authors were supported in part by NSERC (Canada) through grant OGP-0008781. The work of the second author is supported by NSERC (Canada) through grant OGP-327619.

http://www.siam.org/journals/sisc/29-1/64316.html

†Department of Mathematics, Simon Fraser University, Burnaby, BC V5A 1S6, Canada (rdr@math.sfu.ca, jfw@math.sfu.ca, xxub@sfu.ca).

experimentation. Examples include nonuniqueness in thin films [3], finite-time extinction in thin-film equations [10, 6], exact self-similar finite-time blow-up in higher-order semilinear parabolic equations [14], and solitary waves in nonlinear beam equations [22]. Generally, adaptive numerical methods must of necessity be brought to bear for the solution of these problems since important structures for many typical fourth-order problems occur at scales which change over the duration of the time period of interest, making a fixed grid inadequate for the task of efficiently obtaining an accurate final solution. While a number of numerical approaches have been used—e.g., see [7, 20, 15, 40, 4]—in our opinion the need to develop general numerical methods and software specifically for fourth-order problems is clear.

Generally speaking, there are three adaptive approaches for most numerical methods: h-refinement, whereby nodes are added to or removed from an existing mesh in such a way as to improve local grid resolution; p-refinement, whereby higher-order schemes are used to improve local accuracy when the solution is sufficiently smooth; and r-refinement, whereby a fixed number of nodes is moved along with the solution over the domain. These strategies may of course be used singly or in combination.

We consider r-refinement methods [28], which have recently been successfully used for solving a number of differential equations that involve large solution variations like the ones arising in fourth-order problems—e.g., extinction, boundary layers, and blow-up. In particular, in this paper we discuss the construction and implementation of a high resolution moving collocation scheme for the adaptive approximation of fourth-order (in space) evolutionary PDEs. The numerical method used is a generalization of the one developed in [32]. The adaptive strategy is largely based upon the basic moving mesh methods in [30, 31] and found to be particularly effective for scaling invariant problems of a type similar to many important fourth-order problems. With our approach, the numerical schemes are applied directly to the fourth-order differential equations instead of converting the equations to lower-order systems. This direct approach has proven useful in terms of efficiency and effectiveness for solving ordinary differential equations (ODEs) [1]. We believe it to be appropriate in the context of solving time-dependent PDEs in one spatial dimension (1D) as well, for much the same reasons, and because added difficulties arise using lower-order systems for moving mesh methods (cf. section 6).

The collocation-type method we use is based upon a 7th degree Hermite spline basis since this is a natural order of spline approximation for fourth-order problems [2], as is a 3rd degree Hermite basis for second-order problems [32]. The method is analyzed and the error analysis in [32] extended. The layout for this paper is as follows. In sections 2–4 we introduce the moving mesh method and its spatial Gauss point collocation discretization. In section 5 we describe the conservation properties of this method and analyze a related fully conservative strategy. Some properties of the resulting code MOVCOL4 as well as alternative methods for solving fourth-order PDEs are briefly discussed in section 6. Finally, in section 7 several problems are solved with MOVCOL4 which demonstrate the breadth of its applicability and the advantages of this approach.

2. 1D r-adaptivity for PDEs.

2.1. Equidistribution and optimal meshes. The main tool for optimal 1D adaptivity is to require that a local estimate of the error (e.g., the discretization error) be equalized over each computational cell [23], or [illegible]. If we define a [illegible] variable ξ and a monitor function $M(x) > 0$, then the equidistributed mapping satisfies [30]

$$\int_0^{x(\xi)} M(s)ds = \xi \int_0^1 M(s)ds. \tag{2.1}$$

Although not written explicitly here, $M(x)$ is also a function of t and the solution $u(x,t)$ of the underlying physical PDE (and/or its derivatives). Without loss of generality, if M is normalized such that $\int M = 1$, then differentiating with respect to ξ gives the differential form of the equidistribution principle

$$Mx_\xi = 1. \tag{2.2}$$

Note that this specifies a condition on the Jacobian of the mapping $x(\xi)$. For an appropriately chosen monitor function, the equidistributed grid is optimal with respect to a given error norm [23].

Solving (2.2) directly in concert with a PDE leads to a coupled highly nonlinear differential algebraic equation (DAE) system with the mesh equations being algebraic. To solve this difficult numerical problem, various regularizations have been proposed. The one we use here involves solving so-called moving mesh PDEs (MMPDEs), so that (2.2) is solved only approximately in time [31]. While there are many different MMPDEs [30], here we discuss only two. The first, MMPDE6, is

$$x_{t\xi\xi} = -\frac{1}{\tau}(Mx_\xi)_\xi. \tag{2.3}$$

The positive parameter $\tau \ll 1$ is a relaxation time determining the time-scale over which a mesh converges to steady-state (2.2). The second, MMPDE4, is

$$(Mx_{t\xi})_\xi = -\frac{1}{\tau}(Mx_\xi)_\xi, \tag{2.4}$$

which has different scaling properties than MMPDE6, and the suitability of one over another can be problem dependent. One can show that when solved exactly, neither produces mesh crossing, i.e., the mapping is well defined for all time [31].

A key to the success of an MMPDE method is the choice of monitor function. Monitor functions based on error control have been used in [4, 21], as have ones which cluster nodes based on qualitative solution properties, e.g., in [15, 40, 34]. A general methodology for constructing monitor functions to minimize errors in specified norms is presented in [29]. Scaling structure proves essential to many of the example problems listed here and elsewhere. One motivation for using a scale-invariant monitor function is the possibility of computing solutions with uniform error estimates, even for singular problems. For a general discussion of scale-invariant monitor functions and their role when using MMPDEs, see [15, 16, 18].

3. Moving collocation method. In the remainder of this paper, U and X will denote the numerical approximations for the spline solution parameters and the moving grid. The independent variables are the time t and the computational spatial variable ξ.

To solve time-dependent problems, we use a moving method of lines approach, first discretizing the physical PDE in space using Hermite collocation and the MMPDE using finite differences. Then the resulting system of ODEs in time is solved with an ODE solver. Our adaptive method is motivated by the desire to handle a variety of singular solution features for the problems under consideration. Because we are considering only problems in 1D, solving the MMPDE and physical PDE simultaneously for both the transformation $x(\xi,t)$ and physical solution $u(x(\xi,t),t)$ is

computationally realistic. This avoids the need to interpolate the solution between different grids, and the method of lines discretization for the entire system may be integrated with standard software. An alternate strategy is mesh decoupling for which the physical and mesh PDEs are solved alternately—e.g., see [4]. Using Hermite collocation for the physical PDE and only central finite differences for the mesh is also the approach taken in [32] with MOVCOL, a spline collocation code designed for second-order problems. Its success is predicated on the fact that, as experience indicates, the mesh usually need not be computed as accurately as the solution to the physical PDE. This can be motivated by noting that the physical solution can, in theory, be solved on any grid. As long as the mesh nodes do not cross, errors in the computation of the grid do not directly correspond to errors in the approximation of the physical PDE.

The code MOVCOL has proven flexible and robust in solving a large class of second-order problems, including ones with finite time blow-up [13, 17, 19]. Our objective is to develop a method which is an extension of that approach for fourth-order problems. In fact, many of the strengths of the coupling of collocation and MMPDEs are greatly enhanced for higher-order problems.

Spline collocation has several advantages over standard finite difference approximation: it affords a continuous representation of the solution and its first three spatial derivatives, provides a higher order of convergence, is simple and flexible to program, easily handles boundary conditions involving high derivatives, and gives errors independent of local mesh grading. This last point is one of the most important for a moving mesh method, particularly for higher-order problems. By using collocation, we are able to avoid the problem of approximating high order derivatives via differences over a wide nonuniform stencil (cf. [39]).

When considering a discretization with a coordinate transformation, one can either write the PDE in terms of the physical variable x and discretize on a nonuniform mesh, or solve a transformed PDE in terms of the computational variable ξ on a uniform mesh. The latter approach is typical of moving finite difference approximations; however, with the flexibility of the collocation method it is simpler in 1D to discretize the PDE directly with respect to the physical variable x.

Specifically, suppose that at a time t the mesh

$$x_L(t) = X_1(t) < X_2(t) < \cdots < X_{N+1}(t) = x_R(t)$$

is the solution of the discretized MMPDE. The physical solution $u(x,t)$ is approximated by the piecewise 7th-order Hermite polynomial

$$\begin{aligned} U(x,t) = {} & U_i(t)L_{0,0}(s) + U_{x,i}(t)H_i(t)L_{0,1}(s) + U_{xx,i}(t)H_i^2(t)L_{0,2}(s) \\ & + U_{xxx,i}(t)H_i^3(t)L_{0,3}(s) + U_{i+1}(t)L_{1,0}(s) + U_{x,i+1}(t)H_i(t)L_{1,1}(s) \\ & + U_{xx,i+1}(t)H_i^2(t)L_{1,2}(s) + U_{xxx,i+1}(t)H_i^3(t)L_{1,3}(s) \end{aligned} \tag{3.1}$$

for $x \in [X_i(t), X_{i+1}(t)]$, $i = 1, \ldots, N$, where $U_i(t), U_{x,i}(t), U_{xx,i}(t)$, and $U_{xxx,i}(t)$ denote $U(X_i(t),t), U_x(X_i(t),t), U_{xx}(X_i(t),t)$, and $U_{xxx}(X_i(t),t)$, respectively. The local coordinate s is defined by

$$s = \frac{x - X_i(t)}{H_i(t)} \in [0,1], \qquad H_i(t) = X_{i+1}(t) - X_i(t).$$

The functions L_{ik} satisfy the relations

$$\frac{d^p}{dx^p}L_{ik}(x_j) = \begin{cases} 1 & \text{if } i=j \text{ and } k=p, \\ 0 & \text{otherwise,} \end{cases} \tag{3.2}$$

where for completeness we give

(3.3)

$$\begin{aligned} L_{0,0}(s) &= \left(20s^3+10s^2+4s+1\right)(s-1)^4, & L_{0,1}(s) &= s\left(10s^2+4s+1\right)(s-1)^4, \\ L_{0,2}(s) &= \frac{s^2}{2}(4s+1)(s-1)^4, & L_{0,3}(s) &= \frac{s^3}{6}(s-1)^4, \\ L_{1,0}(s) &= -\left(20s^3-70s^2+84s-35\right)s^4, & L_{1,1}(s) &= s^4(s-1)\left(10s^2-24s+15\right), \\ L_{1,2}(s) &= -\frac{s^4}{2}(s-1)^2(4s-5), & L_{1,3}(s) &= \frac{s^4}{6}(s-1)^3. \end{aligned}$$

On $x \in [X_i(t), X_{i+1}(t)]$, derivatives of U are determined by direct differentiation of (3.1), e.g.,

$$\begin{aligned} U_x(x,t) = \frac{1}{H_i}\Bigg(& U_i(t)\frac{dL_{0,0}}{ds} + U_{x,i}(t)H_i(t)\frac{dL_{0,1}}{ds} + U_{xx,i}(t)H_i^2(t)\frac{dL_{0,2}}{ds} \\ & + U_{xxx,i}(t)H_i^3(t)\frac{dL_{0,3}}{ds} + U_{i+1}(t)\frac{dL_{1,0}}{ds} + U_{x,i+1}(t)H_i(t)\frac{dL_{1,1}}{ds} \\ & + U_{xx,i+1}(t)H_i^2(t)\frac{dL_{1,2}}{ds} + U_{xxx,i+1}(t)H_i^3(t)\frac{dL_{1,3}}{ds}\Bigg) \end{aligned} \tag{3.4}$$

and

(3.5)

$$\begin{aligned} U_t(x,t) = {} & \frac{dU_i}{dt}L_{0,0} + \left(\frac{dU_{x,i}}{dt}H_i + U_{x,i}\frac{dH_i}{dt}\right)L_{0,1} + \left(\frac{dU_{xx,i}}{dt}H_i^2 + 2U_{xx,i}H_i\frac{dH_i}{dt}\right)L_{0,2} \\ & + \left(\frac{dU_{xxx,i}}{dt}H_i^3 + 3U_{xxx,i}H_i^2\frac{dH_i}{dt}\right)L_{0,3} + \frac{dU_{i+1}}{dt}L_{1,0} \\ & + \left(\frac{dU_{x,i+1}}{dt}H_i + U_{x,i+1}\frac{dH_i}{dt}\right)L_{1,1} + \left(\frac{dU_{xx,i+1}}{dt}H_i^2 + 2U_{xx,i+1}H_i\frac{dH_i}{dt}\right)L_{1,2} \\ & + \left(\frac{dU_{xxx,i+1}}{dt}H_i^3 + 3U_{xxx,i+1}H_i^2\frac{dH_i}{dt}\right)L_{1,3} - U_x(x,t)\left(\frac{dX_i}{dt} + s^{(i)}\frac{dH_i}{dt}\right). \end{aligned}$$

There are several natural ways to derive a system of ODEs for determining the unknowns $U_i(t)$, $U_{x,i}(t)$, $U_{xx,i}(t)$, and $U_{xxx,i}(t)$. The first we consider is collocation at the Gauss points. The second scheme is a conservative method, which is a natural alternative choice for many physical problems.

4. Gauss point collocation. Consider the PDE

$$u_t = \mathcal{N}(t,x,u,u_x,u_{xx},u_{xxx},u_{xxxx}), \quad t>0,\ x\in I=(a,b), \tag{4.1}$$

supplemented with the initial condition

$$u(x,0) = u_0(x), \quad x\in(a,b),$$

and separated boundary conditions

$$B_{a,1} = 0,\ B_{a,2} = 0,\ B_{b,1} = 0,\ B_{b,2} = 0, \tag{4.2}$$

where $B_{y,i} = B_{y,i}(t, y, U_t(y), U(y), U_x(y), U_{xx}(y), U_{xxx}(y))$. The standard collocation approach would be to solve for the $(4+1)(N+1)$ unknowns consisting of the solution U and its first three spatial derivatives at the mesh points and the $(N+1)$ mesh points $X_i(t)$ themselves. One enforces the collocation condition

$$U_t - \mathcal{N}(t, x, U, U_x, U_{xx}, U_{xxx}, U_{xxxx}) = 0 \tag{4.3}$$

at the four Gauss points in each interval. The MMPDE is approximated with central differences at the $(N-1)$ interior points, e.g., for $X_{t,\xi\xi} = -\frac{1}{\tau}\left(MX_\xi\right)_\xi$ use

$$X_{t,i-1} - 2X_{t,i} + X_{t,i+1} = -\frac{1}{\tau}\left(M_{i+1/2}(X_{i+1} - X_i) - M_{i-1/2}(X_i - X_{i-1})\right).$$

Here $M_{i+1/2}$ and $M_{i-1/2}$ are approximations to the average of the monitor function over cells $I_i = (x_i, x_{i+1})$ and I_{i-1}, respectively. Satisfying the boundary conditions (4.2) and requiring $X_1 = a$, $X_{N+1} = b$, this gives a system of $5(N+1)$ equations in $5(N+1)$ unknowns. From the standard ODE error analysis for a spline collocation solution U to (4.1) at a fixed time level t [2], the spatial error satisfies

$$|u(x) - U(x)| = \mathcal{O}(h^8) \quad \text{for all } x \in I, \quad \text{where } h = \max_i h_i.$$

Frequently in practice a PDE occurs in the form

$$u_t = \left(g(t, x, u, u_x, u_{xx}, u_{xxx})\right)_x, \tag{4.4}$$

and on a physical basis one is often interested in preserving the conservation property

$$\frac{d}{dt}\int_I u = g(b) - g(a). \tag{4.5}$$

The above Gauss collocation method does not do so in general, but it does in special cases such as the following.

LEMMA 4.1. [illegible] (4.5) [illegible] (4.4) [illegible]

$$g(u, u_x, u_{xx}, u_{xxx}) = \alpha_1 u + \alpha_2 u_x + \alpha_3 u_{xx} + \alpha_4 u_{xxx}, \tag{4.6}$$

[illegible] α_i, $i = 1, \ldots, 4$ [illegible] Denoting $U_i^{(0)} = U_i$, $U_i^{(1)} = U_{x,i}$, $U_i^{(2)} = U_{xx,i}$, $U_i^{(3)} = U_{xxx,i}$, we have that

$$\begin{aligned}
\frac{d}{dt}\int_I U dx &= \int_I \alpha_1 U_x + \alpha_2 U_{xx} + \alpha_3 U_{xxx} + \alpha_4 U_{xxxx} dx \\
&= \sum_i \sum_j \alpha_j \left(\sum_{k=0}^{3} (H_i)^{k-j} \left(U_i^{(k)} \int_0^1 \frac{d^j}{ds^j} L_{0,k}(s) ds + U_{i+1}^{(k)} \int_0^1 \frac{d^j}{ds^j} L_{1,k}(s) ds \right) \right) \\
&= \sum_i \sum_j \alpha_j (U_{i+1}^{(j-1)} - U_i^{(j-1)}) \qquad \text{(from (3.2))} \\
&= \sum_j \alpha_j (U_{N+1}^{(j-1)} - U_1^{(j-1)}) \\
&= G(X_{N+1}) - G(X_1). \qquad \square
\end{aligned}$$

5. Conservative method. While Gauss point collocation performs well for many problems, here we construct another method which has the advantage of preserving general conservation properties. For this, we consider a more general PDE of the form

$$f(t,x,u_t,u,u_x,u_{xx},u_{xxx}) = (g(t,x,u,u_x,u_{xx},u_{xxx}))_x \tag{5.1}$$

and derive an approximation method for it which satisfies the generalized conservation property

$$\int_I f dx = g|_{x=b} - g|_{x=a}. \tag{5.2}$$

An additional advantage of treating the PDE (5.1) directly is that when solutions develop singularities, each successive spatial derivative becomes larger, and we can avoid the explicit computation of u_{xxxx} by using a cell-averaged form. As a result, one can often solve the problem more efficiently by reducing the stiffness of the ODE system compared to the standard Gauss scheme which requires fourth derivatives. Note, however, that a PDE (4.1) can be written in the form (5.1) in a number of ways.

To generate an approximation to u, we again approximate u by its Hermite interpolant U as defined in (3.1). To eliminate any dependence on U_{xxxx} in our system of ODEs and to enforce the conservation property (5.2) we use a cell-averaged approach for solving (5.1). We begin by approximating f by its Lagrange interpolant

$$F_i = \sum_{j=0}^{3} f_{ij}\tilde{L}_j, \tag{5.3}$$

where the canonical points are the four Gauss points on the unit interval

$$\rho_1 = \frac{1}{2} - \frac{\sqrt{525+70\sqrt{30}}}{70},\ \rho_2 = \frac{1}{2} - \frac{\sqrt{525-70\sqrt{30}}}{70},\ \rho_3 = 1-\rho_1,\ \rho_4 = 1-\rho_2, \tag{5.4}$$

and $f_{ij} = f(x_i + H_i\rho_j) = f(x_{ij})$. Equation (5.1) is integrated approximately using the five Lobatto points, which on the unit interval are

$$\tilde{\rho}_1 = 0,\ \ \tilde{\rho}_2 = \frac{1}{2} - \frac{\sqrt{21}}{14},\ \ \tilde{\rho}_3 = \frac{1}{2},\ \ \tilde{\rho}_4 = \frac{1}{2} + \frac{\sqrt{21}}{14},\ \ \tilde{\rho}_5 = 1. \tag{5.5}$$

Integrating the approximation (5.3) to f over each cell gives the system

$$A\boldsymbol{F}_i = B\boldsymbol{G}_i/H_i \tag{5.6}$$

in each interval I_i, where the matrices A, B are defined by

$$A_{jk} = \int_{\tilde{\rho}_j}^{\tilde{\rho}_j+1} \tilde{L}_k dt, \quad j,\, k = 1,2,3,4,$$

$$B_{jk} = \begin{cases} 1 \text{ if } j = k, \\ -1 \text{ if } k = j+1 \\ 0 \text{ else,} \end{cases} \qquad \text{for } j = 1,\dots,4 \text{ and } k = 1,\dots,5,$$

and the vectors $\boldsymbol{F}_i$, $\boldsymbol{G}_i$ are

$$(\boldsymbol{F}_i)_j = F_i(x_i + H_i\rho_j) = F_i(x_{ij}), \quad j = 1, 2, 3, 4,$$

$$(\boldsymbol{G}_i)_j = G_i(x_i + H_i\tilde{\rho}_j) = G_i(\tilde{x}_{ij}), \quad j = 1, 2, 3, 4, 5\ .$$

By direct construction we have enforcement of cell-averaging, i.e.,

$$\int_{I_i} F_i = G(X_{i+1}) - G(X_i),$$

and we have the following lemma.

LEMMA 5.1. [illegible] (5.6) [illegible]

$$\int_I F dx = \sum_i \int_{I_i} F_i dx = \sum_i G(X_{i+1}) - G(X_i) = G(X_{N+1}) - G(X_1). \qquad \square$$

The Hermite interpolant U is defined in (3.1) and is specified by choosing the coefficients $U_i(t), U_{x,i}(t), U_{xx,i}(t), U_{xxx,i}(t)$ such that in each interval,

$$H_i\boldsymbol{F}_i = W\boldsymbol{G}_i,$$

where the differentiation matrix $W = A^{-1}B$. This gives four equations in each cell, as in the standard Gauss collocation method, but they do not involve U_{xxxx}. Using the fact that the two extreme Lobatto points $\tilde{\rho}_1 = 0$ and $\tilde{\rho}_5 = 1$ coincide at successive intervals, an efficient implementation requires $8N+1$ function evaluations, twice that of the Gauss point formulation with $2N+1$ ODE function evaluations. However, in many cases the structure of g is much simpler than g_x, leading to fewer total floating point operations.

The spline collocation error analysis in [2] can, when suitably modified, be extended to give error bounds for the conservative scheme. We can find the local truncation error for our equations by considering the problem

$$g_x = f \tag{5.7}$$

as an equation for g. This approach extends the error estimate derived in [32], which is valid only at the Gauss points, to a spatially global one. Henceforth, we shall refer to this discretization of (5.7) as a "conservative collocation" scheme, although technically speaking it is not a collocation scheme, as we discuss below.

THEOREM 5.2. [illegible] (5.6) [illegible]. The results of Theorem 5.147 from [2] can be used to estimate the leading order term in $G_x - g_x$. There are $k = 5$ Lobatto points, with precision $p = 2k - 2 = 8$ to solve the first-order ($m = 1$) problem (5.7)

$$G_x - g_x \simeq H_i^5 \frac{g^{(6)}}{6!} \prod_{j=1}^{5} (s - \tilde{\rho}_j).$$

The error in the Lagrange approximation to f is given by

$$F - f \simeq \frac{f^{(4)}}{4!} \prod_{j=1}^{4} (s - \rho_j)$$

and thus the error is

$$e = |(G_x - F)| = |(G_x - g_x) - (F - f)|$$
$$\simeq \left| H_i^5 \frac{g^{(6)}}{6!} \prod_{j=1}^{5} (s - \tilde{\rho}_j) - H_i^4 \frac{g^{(5)}}{4!} \prod_{j=1}^{4} (s - \rho_j) \right| \tag{5.8}$$

for all points $x_i \le x \le x_i$, and $s = (x - x_i)/H_i$. □

Here we have used the fact that the exact solution satisfies $g_x = f$. At first glance this estimate may seem slightly strange—it is an estimate of the residual, something normally set to be identically zero in a collocation method. However, recall that there are two sources of error in the conservative method: one is an interpolation error for f, and the other is from the approximate solution to $g_x = f$. These errors vanish at different points in the domain, so there is no point where the residual is necessarily zero. This estimate may also be recovered by a standard Taylor series expansion, but that is far more cumbersome. This same construction can be used to extend the error estimate in [32] from the Gauss points to the entire solution domain.

For the standard Gauss collocation scheme we have explicit error estimates for $e_G = |u - U|$, whereas for the conservative scheme we have estimates only for the residual which, in the general case, requires information about the Green function to estimate $|u - U|$. We expect a similar estimate for $e = |u - U|$ to hold for both but leave a proof for future work and content ourselves now with a numerical test of the methods.

6. Implementation. MOVCOL4 is a Fortran code with the capability of using either the Gauss collocation or the conservative scheme described in the two previous sections. The user is provided the option of selecting either one and of solving the MMPDE in the form of either MMPDE4 or MMPDE6. The resulting equations are integrated with the software DASSL [37], a backward differentiation code designed to solve stiff DAEs. Other inputs provided by the user are the absolute error tolerance *atol* and relative error tolerance *rtol*, which are needed for DASSL, the number of mesh points *npts* and initial mesh $X_i : i = 1, \ldots, npts$, and the definition of the monitor function to be used in the MMPDE. The onus is on the user to ensure that the PDE is well-posed (e.g., not $u_t = u_{xxxx} + r(u)$, which is unstable like the backward heat equation). The code is designed to deal with very general systems of PDEs of the form

$$\boldsymbol{f}(t, x, \boldsymbol{u}, \boldsymbol{u}_t, \boldsymbol{u}_x, \boldsymbol{u}_{xx}, \boldsymbol{u}_{xxx}, \boldsymbol{u}_{xxxx}) = (\boldsymbol{g}(t, x, \boldsymbol{u}, \boldsymbol{u}_x, \boldsymbol{u}_{xx}, \boldsymbol{u}_{xxx}))_x, \tag{6.1}$$

although the form chosen can significantly affect the code's efficiency. One of the most important input parameters is the initial mesh. For many problems, it must be close to equidistributing in order for the integrator DASSL to start. To help alleviate this common difficulty, MOVCOL4, like MOVCOL, has an option to compute an equidistributing mesh corresponding to the initial data $u_0(x)$ before solving the PDE.

The user also has the option to choose different tolerances for the solution components. This is particularly effective for the higher derivatives which, because they appear in the solution definition multiplied by powers of the local grid spacing, have a lesser effect on the error in reconstructing the entire solution. Typically, one can take the most stringent tolerances for the solution only and demand less accuracy on the derivatives with no noticeable loss in the accuracy of the final solution.

A common approach for solving high order PDEs is to first convert them into lower-order systems and use a code designed for these lower-order problems. For

example, the fourth-order PDE

$$v_t = f(t, x, v, v_x, v_{xx}, v_{xxx}, v_{xxxx})$$

can be written as a second-order system in the variables $\boldsymbol{u} = (u_1, u_2)^T = (v, v_{xx})^T$. The equivalent system

$$(u_1)_t = f(t, x, u_1, (u_1)_x, u_2, (u_2)_x, (u_2)_{xx}),\ (u_1)_{xx} = u_2$$

is then solved. Discretization on a fixed grid gives a DAE system of index 1, where time differentiation of the algebraic equation once gives an ODE system with the second equation replaced by $(u_2)_t = (u_1)_{xxt}$. One could then employ a method of lines code designed for second-order problems which uses a time integration code such as DASSL, which is designed to solve index 1 DAEs under reasonable restrictions on the rate of change of step size [26]. However, when this approach is used for a moving grid we obtain

$$(u_2)_t = (u_1)_{xxt} + [(u_1)_{xxx} - (u_2)_x]\dot{x}(t).$$

Since u_2 and u_1 are approximated with different spline functions, U_1 and U_2 $(u_1)_{xx} = u_2$ does not imply that $(u_1)_{xxx} = (u_2)_x$. Indeed, experience has shown that the error can easily grow if this term is dropped. Consequently, both the third derivative term $(u_1)_{xxx}$ and the mesh speed $\dot{x}(t)$ must be dealt with, and a code for second-order problems like MOVCOL cannot be used directly. One can convert the PDE into the second-order system

$$\begin{aligned}
(u_1)_t &= f(t, x, u_1, (u_1)_x, u_2, (u_2)_x, (u_3)_x, (u_3)_{xx}),\\
(u_2)_t &= (u_1)_{xt} + [(u_1)_{xx} - (u_2)_x]\dot{x}(t),\\
(u_3)_t &= (u_2)_{xt} + [(u_2)_{xx} - (u_3)_x]\dot{x}(t),
\end{aligned}$$

although this form is clearly more awkward. In addition, this can be quite inefficient due to the need to handle the mesh speed term and the large number of additional unknowns as compared to the method applied directly to the high order equation. In our experience, applying the moving mesh code MOVCOL for lower-order systems has not generally warranted the associated cost nor achieved comparable accuracy to that with MOVCOL4 for fourth-order problems. Nevertheless, it remains an interesting question, both in the fixed grid and moving grid contexts, as to whether or not there are reformulations of high order problems as systems of lower order ones which are sufficiently stable (and well-posed) to warrant their use with codes designed for such low order problems.

7. Numerical examples. In this section we demonstrate the efficiency, accuracy, and robustness of MOVCOL4 by solving a variety of problems. Example 1 (equation (7.1)) has been chosen to illustrate the rate of convergence for a simple test case. The remaining examples have been chosen from applications to show the breadth of applicability of our approach. The first is the semilinear Cahn–Hilliard equation and the second concerns some quasi-linear thin-film equations. For both examples two different solution regimes are investigated.

One of the most difficult choices for the novice when using moving mesh methods is that of the monitor function. There are several competing strategies, and the different problems in this section motivate some of those different choices. The first problem is trivial and not particularly sensitive to the precise choice of monitor function.

In the examples we use monitor functions determined either by properties of the physical equations or by information about the particular solution features. Two cases exhibit finite-time blow-up, and for such problems preserving scale invariance in the moving mesh method works particularly well [15, 18]. One case has very delicate finite-time extinction, and the monitor function is constructed based loosely on asymptotic properties of the solution. For this case MMPDE6 is most appropriate as it allows adaptivity on the same time-scale as the monitor function, which is key for finite-time behavior. The remaining problem exhibits metastable fronts. Because solution features emerge on an $O(1)$ time-scale we use MMPDE4 and a monitor function prepared to place more points in any emergent layers. In this scenario, the time stepping is determined solely by the physical PDE and not by changes in magnitude of the monitor function.

7.1. A simple test problem. We begin with a simple test problem,

$$\text{Ex. 1} \quad \begin{cases} u_t = -\dfrac{\sin t}{\cos t + 3} u_{xxxx}, & x \in (0,\pi),\ 0 < t < \pi, \\ u(x,0) = 4c\cos(x), \\ u_x(0,t) = u_{xxx}(0,t) = 0, \\ u_x(\pi,t) = u_{xxx}(\pi,t) = 0, \end{cases} \tag{7.1}$$

which has the exact solution $u(x,t) = c(\cos(t)+3)\cos(x)$. We investigate the convergence properties of MOVCOL4 for this example.

The high order of the spatial discretization necessitates that the problem be designed very carefully in order for one to check the actual convergence rate. An advantage of the problem in this form is that there is only the fourth derivative term, which simplifies the form of the discretization error over mathematically equivalent, more complicated forms, although some care must be taken since this problem is ill-posed for $\sin t < 0$. Also, it is very hard to discern the rate of convergence before the time integration error and machine error become involved and play a significant role, so the absolute error tolerance and relative error tolerance for the time integration must be chosen to be extremely small—we choose $atol = 10^{-15}$, $rtol = 10^{-13}$. We solve the problem with 2 subintervals (3 mesh points), 4 subintervals (5 mesh points), and finally, 8 subintervals (9 mesh points). In order to see the global rate of convergence, we first compute the solution on $[0,\pi]$ using 3, 5, or 9 mesh points and then construct the solution on the whole interval using Hermite spline interpolation.

It is easiest to start the integration with a fixed grid using MMPDE6 and $\tau = 1.d-3$, where the smoothness of the solution and the monitor function $M(u) = 1+|u|$ prevent the mesh from becoming too skew and the temporal discretization error from strongly affecting the total error. In our tests we choose $c = 0.3$.

In the log-log plot of the L^∞ norm of the error versus the mesh size given in Figure 1, the slope of the straight line fit, that is, the rate of convergence, is approximately 8. In Table 1, we show computed rates of convergence

$$\mathrm{ROC}N_1N_2 = \frac{\log\left(\frac{\|E_{N_1}\|_\infty}{\|E_{N_2}\|_\infty}\right)}{\log\left(\frac{\pi/N_1}{\pi/N_2}\right)}.$$

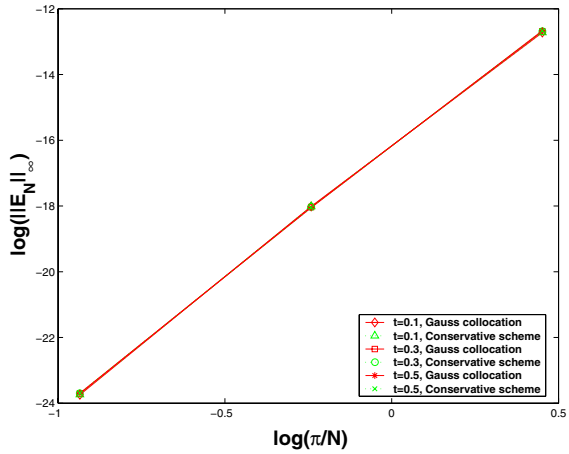


FIG. 1. *Rates of convergence, Gauss collocation method, and conservative collocation method. The errors for both methods are indistinguishable.*

TABLE 1
Rates of convergence.

Time	M	$\|E_2\|_\infty$	$\|E_4\|_\infty$	$\|E_8\|_\infty$	ROC42	ROC84	ROC82
$t = 0.1$	G	2.982353E-06	1.501874E-08	4.862577E-11	7.6335	8.2708	7.9522
	C	2.982353E-06	1.501874E-08	4.870682E-11	7.6335	8.2684	7.9510
$t = 0.3$	G	3.099078E-06	1.475483E-08	5.055389E-11	7.7145	8.1891	7.9518
	C	3.099078E-06	1.475483E-08	5.062472E-11	7.7145	8.1871	7.9508
$t = 0.5$	G	3.150803E-06	1.446814E-08	5.111500E-11	7.7667	8.1445	7.9558
	C	3.150803E-06	1.446855E-08	5.113077E-11	7.7667	8.1445	7.9556

For example,

$$\mathrm{ROC42} = \frac{\log\left(\frac{\|E_2\|_\infty}{\|E_4\|_\infty}\right)}{\log\left(\frac{\pi/2}{\pi/4}\right)},$$

where $\|E_2\|_\infty = \max_{i=1}^{101} |u_{\mathrm{comp}} - u_{\mathrm{exact}}|$ is measured by sampling the reconstructed and the exact solutions at 101 equally spaced points on $[0, \pi]$.

7.2. The Cahn–Hilliard equation. The Cahn–Hilliard equation is a classical semilinear fourth-order equation modeling phase separation. Many diffusive processes with a natural length scale, such as phase separation in binary alloys, growth and dispersal in population, or spreading of an oil film over a solid surface, can be described by this equation. The special form for the Cahn–Hilliard equation which we investigate here is

$$u_t = -(\gamma_1 u_{xx} - \gamma_2 u^3 + \gamma_3 u^2 + \gamma_4 u)_{xx}. \tag{7.2}$$

We consider two cases with strikingly different behaviors, both arising in various application areas.

7.2.1. Finite-time blow-up in the unstable regime. For the case of $\gamma_1 = 1, \gamma_2 = -1$, and $\gamma_3 = \gamma_4 = 0$ [25], the equation has a combination of a fourth-order diffusion operator and the classical second-order porous medium operator but posed

in time. This competition of effects can lead to finite time blow-up, where there exists $T < \infty$ such that

$$\lim_{t \to T} \|u(\cdot,t)\|_\infty = \infty \quad \text{and} \quad \|u(\cdot,t)\|_\infty < \infty \ \text{ for all } \ 0 \le t < T.$$

The complete problem is

$$\text{Ex. 2} \quad \begin{cases} u_t = -(u_{xx} + u^3)_{xx}, & x \in (0,6),\ t > 0, \\ u(x,0) = 2e^{-\frac{x^2}{2}}, \\ u_x(0,t) = u_{xxx}(0,t) = 0, \\ u_x(6,t) = u_{xxx}(6,t) = 0. \end{cases} \tag{7.3}$$

The problem is conservative, and conservation with blow-up implies that the blow-up can occur only at an isolated point (or set of isolated points). In this case the blow-up is self-similar in that there is a balance between the magnitude of and natural length-scale of the solution. The use of scaling invariant adaptivity is natural for problems such as this one, since as we see below the intrinsic length-scale of the solution is then automatically determined.

We take $npts = 41$, $rtol = atol = 10^{-5}$, and $\tau = 1e-4$. To resolve the blow-up solution we choose the monitor function

$$M(x,t) = u(x,t)^4 + \frac{1}{12}\int_0^6 u(x,t)^4\,dx$$

because it is scale invariant in the blow-up region; i.e., the MMPDE is invariant under the same scaling group as the physical PDE. The role of the second term for the monitor function is to ensure that there is an even distribution of mesh points inside and outside the blow-up region as blow-up is approached. The factor 1/12 is chosen to place approximately two-thirds of the grid points inside the blow-up region and one-third outside—for further discussion of scaling in constructing monitor functions, see [15] and [16]. This idea, first introduced by Beckett and Mackenzie [5], is analyzed theoretically for PDEs with finite-time singularities in [18].

Typical blow-up solutions are presented in Figure 2. (A comprehensive treatment of this problem is given in [25].) We compare the effectiveness of the two spatial discretizations in Table 2. Note that the conservative discretization is more efficient and can integrate marginally further into the blow-up regime. A comparison of the accuracy of conservation for the two approaches is presented for a more difficult problem in section 7.3.2.

7.2.2. Metastable fronts in the stable regime. While the Cahn–Hilliard equation can also give exponential decay (like the heat equation) or blow-up, we now concentrate on the metastable case [42]. The typical underlying physical context involves a melted binary alloy with a given concentration of components that is quenched to a temperature at which exactly two different concentration phases can coexist. For $\gamma_1 \to 0$, the qualitative features of the dynamics associated with (7.2) are as follows: Through a very intricate transient process, a pattern of internal layers forms from the initial data over an $O(1)$ time interval. Once this pattern is formed, the subsequent motion of the internal layers is exponentially slow and hence becomes metastable. For a study of a two-layer pattern in [38], the internal layers collapse at a time of order 10^{11} and the interesting dynamics then occur rapidly on an $O(1)$ timescale.

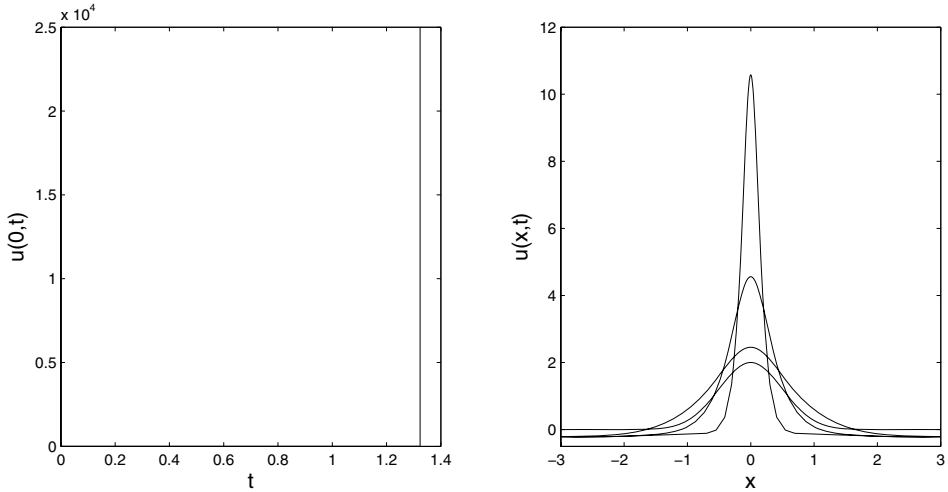


FIG. 2. *Blow-up in the unstable Cahn–Hilliard equation* (7.3). *Left: The solution magnitude. Right: Sample solutions.*

TABLE 2

Comparison of spatial discretizations for (7.2). *The "Steps" column lists the number of time steps required, "Fcn." the number of function evaluations, and "Jac." the number of Jacobian evaluations required by DASSL.*

	$u(0,t)$	Steps	Fcn.	Jac.
Conservative collocation	2.51e4	1748	2934	217
	1000	1103	1998	167
	500	999	1729	161
	100	821	1279	131
	50	677	1199	121
Gauss point collocation	2.31e4	2731	4491	751
	1000	1499	3149	489
	500	1401	2691	390
	100	1121	1713	243
	50	720	1299	117

The form of the Cahn–Hilliard equation we consider is

$$\text{(7.4)} \qquad \text{Ex. 3} \quad \begin{cases} u_t = -(0.001u_{xx} + u - u^3)_{xx}, & x \in (-1,1),\ t > 0, \\ u(x,0) = 0.1\cos(2\pi x) + 0.02\cos(10\pi x), \\ u_x(-1,t) = u_{xxx}(-1,t) = 0, \\ u_x(1,t) = u_{xxx}(1,t) = 0. \end{cases}$$

Since the nonlinear term in (7.2), together with the fourth derivative term, make the Cahn–Hilliard equation very stiff, most finite difference schemes need to use time steps many orders of magnitude smaller than Δx. As a consequence, it is numerically expensive to reach the time scales where the interesting dynamics occur. In Figure 3 we show our numerical results for a finite difference scheme provided in [35]. With this approach, the resulting ODE system is linear and easily solved directly in MATLAB. We have observed that if the ODE from the finite difference scheme is not carefully

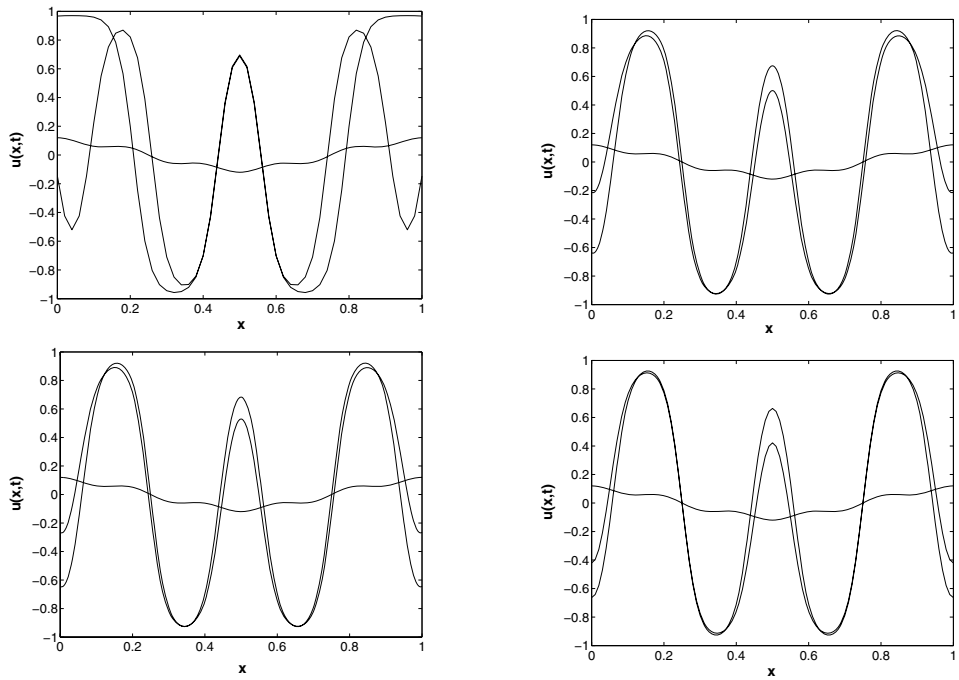


FIG. 3. *Solutions at* $t = 0, 0.05, 0.1$. *Finite difference scheme versus moving collocation scheme. Upper left: Finite difference scheme,* $\Delta x = 0.02, \Delta t = 1.d{-}6$. *Upper right: Finite difference scheme,* $\Delta x = 0.002, \Delta t = 1.d - 4$. *Lower left: Finite difference scheme,* $\Delta x = 0.002, \Delta t = 1.d - 8$. *Lower right: Moving collocation scheme,* $\Delta x = 0.05$, *variable time step size by DDASSL.*

resolved by using a large stencil and very small time step size, a two-layer pattern will be formed much faster than actually occurs. As we increase our stencil and decrease the time step size, the numerical solution obtained by the finite difference scheme converges to the numerical solution obtained by using the moving collocation scheme with only 21 points (but a system of 105 unknowns). If the number of points is increased from 21 to 101, the numerical solution for the moving collocation scheme remains almost the same, suggesting that a small number of mesh points can be used to investigate the metastable behavior of the solution efficiently. Generally speaking, a very accurate discretization is required for the resolution of metastable behavior; otherwise the dynamics will be driven by numerical error rather than the very slow analytical instability.

Figure 4 shows the solutions of (7.4) starting with a uniform mesh of 51 points. The solution is computed until $t = 1 \times 10^{10}$ with the monitor function chosen as $M = \sqrt{1 + u_x^2}$ to place more mesh points in the internal layer and $atol = rtol = 1 \times 10^{-6}$. The solution at $t = 1 \times 10^{10}$, shown in Figure 4, is indistinguishable from the one at $t = 1$.

Our computational tests have shown that the time at which the two metastable interfaces collapse is highly affected by the error tolerances, making it difficult to capture the precise time of collapse, particularly since it is usually of order 10^{11}. Furthermore, the role of adaptivity in computing metastable solutions is at best problematic, since a suitable monitor would need to incorporate information about the solution's sensitivity to the boundary conditions. Further investigation of this issue is

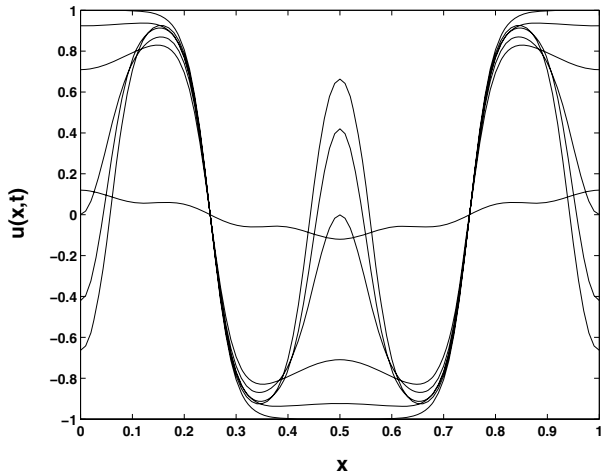


FIG. 4. *Solutions to* (7.4) *at different times:* $t = 0, 0.05, 0.1, 0.3, 0.5, 0.7, 1,\ 1 \times 10^{10}$.

needed before a code such as MOVCOL4 could be recommended over other adaptive (or nonadaptive) ones for a general study of metastability for such problems.

7.3. Equations of thin-film type. A fourth-order equation of wide interest is the thin-film equation

$$u_t = -(f(u)u_{xxx} + g(u)u_x)_x. \tag{7.5}$$

In most applications u represents the thickness of a thin fluid on a substrate. This equation has garnered much analytical and computational interest over the past 15 years (e.g., see [3, 6, 8, 10, 9, 12]), although many open problems remain. Typically, the nonlinearities are taken to be

$$f(u) = u^n \quad \text{and} \quad g(u) = u^m$$

for positive integers n, m. We consider the cases

$$\begin{aligned} f(u) &= u^{1/2} \quad \text{with} \quad g(u) = 0 \quad \text{and} \\ f(u) &= u \quad \text{with} \quad g(u) = 4u^3. \end{aligned}$$

For suitable initial boundary and free boundary conditions, (7.5) has compactly supported solutions which may spread or contract over time. In both of our examples we use (for both analysis and computation) a standard regularization of f,

$$f_\varepsilon(u_\varepsilon) = \frac{u_\varepsilon^4}{\varepsilon + u_\varepsilon^{4-n}}. \tag{7.6}$$

Because of this regularization one expects the solution to be bounded away from zero with

$$\min u_\varepsilon \sim O(\varepsilon^{1/(4-n)}),$$

where $f_\varepsilon(u) \sim O(u^n)$ as $u \to 0$ for $n < 4$. For $n \geq 4$ no such regularization is required. Moreover, in this regime the dynamics are dominated by the regularization, since if

$u_\varepsilon \ll u_\varepsilon^{4-n}$, then

$$f_\varepsilon \sim \frac{u_\varepsilon^4}{\varepsilon}.$$

With the appropriate boundary conditions, the solution to (7.5) satisfies the condition (5.2), or

$$\frac{d}{dt}\int_I u(x,t)\,dx = 0.$$

7.3.1. Moving contact lines in thin liquid films. For the form of the thin-film equation first studied,

$$u_t = -(u^n u_{xxx})_x,$$

the solutions do not blow up, but initially uniformly positive solutions can become extinct at some finite time. We consider this particular case because it is one where there is conjectured to be approximate self-similar behavior and because its numerical solution has been computed elsewhere, making it a reliable test problem.

For $n = 1/2$, the problem

$$\text{Ex. 4} \quad \begin{cases} u_t + (u^{1/2}u_{xxx})_x = 0, & x \in (-1,1),\ t > 0, \\ u(x,0) = 0.8 - \cos(\pi x) + 0.25\cos(2\pi x), \\ u_x(-1,t) = u_x(1,t) = 0, \\ u_{xxx}(-1,t) = u_{xxx}(1,t) = 0 \end{cases} \tag{7.7}$$

is considered in [10, 9, 43]. It is shown computationally in [9] that the solution exhibits finite-time extinction. Specifically, the solution becomes zero at an isolated point for some finite time, and there is simultaneous blow-up of higher derivatives there. Near the singular point the solution has a leading order asymptotic form [10]

$$u(x,t) \approx c(t_c - t) + \frac{p(x - x_c)^2}{2}, \tag{7.8}$$

where t_c is the time of extinction, x_c is the "pinch-point," and the constant p is the curvature of the interface at this time. The solution and its first two spatial derivatives are smooth here, while blow-up in higher derivatives is locally described by

$$u_{xxx} \approx \frac{cx}{\sqrt{c(t_c - t) + p(x - x_c)^2/2}}. \tag{7.9}$$

It is proven in [11, 9] that after the initial singularity time, there exists a nonnegative weak solution for (7.7) in the sense of distributions, and there exists a second critical time after which the solution is again uniformly positive and strong, decaying to its mean in the infinite time limit. Introducing the regularization (7.6), the regularized problem has global, positive, and smooth solutions [9]. The weak solution for (7.7) can then be obtained by finding the solution of the regularized problem and taking the regularization parameter ε to zero [12, 6, 8]. Many thin-film problems have been solved with finite element discretizations [3], but in [9] a careful finite difference static regridding computation starting with a fine initial grid of $2^{10} = 1024$ uniform intervals on [0,1] is used. Since the initial data are symmetric about $x = 0$ and the equation preserves the symmetry, the solution is computed only on $[0, 1]$, using reflection symmetry at the boundary. In order to resolve the smallest length-scale and keep the

numerical solution positive, regridding is done by adding 64 new mesh points whenever $u_{\min}^{1/2} \leq 64/1024$. At the end of the simulation, 55 levels of regridding have been done, with 3520 new points produced and the smallest grid spacing being 2^{-65}. We use M to automatically monitor quantities of interest and relocate the mesh points, making it a good test problem for the adaptive features of our high order moving mesh code MOVCOL4.

Our computation consists of two parts: computing the solution of the nonregularized problem very close to the singularity time to examine the formation of the third derivative jump discontinuity, and computing the regularized solution until it decays to its mean as dictated by the theory in [9]. In either case, we start with an initially uniform grid with $2^7 = 128$ intervals on $[0, 1]$, and hence a system of 640 equations for all time. For the monitor function, we basically choose $M(x, t) = 1/u$. It has been chosen to resolve the singularity as $\min_x u(x, t)$ approaches zero. The exact structure is based on the asymptotic form (7.8) which shows that the solution touches down proportionally to $(t - t_c)$. Hence, for MMPDE6, our monitor function scales in time the same way as the singularity. In other cases different powers of u may be required; a generic example is

$$M = \frac{1}{u^{1/p}} \qquad \text{when} \qquad u \simeq (t - t_c)^p.$$

By comparing (2.3) and (2.4), we can see that if the monitor function varies dramatically on the whole interval of interest, then the mesh evolves faster when MMPDE4 is used, but MMPDE6 is better at forcing the mesh to concentrate in the area where the monitor function has larger values. For the nonregularized problem, in order to resolve the spatial structure and solve the problem very close to the singularity time we use MMPDE6 with $\tau = 1 \times 10^{-3}$. To make it easier for the mesh to evolve, we modify the monitor function to be $M(x, t) = c/(u + d)$. Here, $c < 1$ rescales the basic monitor function and helps the mesh to evolve faster while d determines the "radius" of the area centered at the one-point singularity $u = 0$, where the mesh points concentrate. In the results shown in Figure 5, we use $c = 0.1$ and $d = 1 \times 10^{-8}$. The error tolerances for the time integrator are $atol = rtol = 1 \times 10^{-6}$.

Figure 5 displays the corresponding solutions and third derivatives at specified times close to the singularity time. These results closely agree with those found in [9, 10]. Figure 5 also shows that, by carefully choosing a suitable monitor function and an MMPDE, the mesh can in turn evolve along with the singularity and hence capture or resolve the spatial structure.

For the regularized system, in order to sufficiently resolve the spatial structure at all times, we use MMPDE4 with $\tau = 1 \times 10^{-3}$ and the basic monitor function $M(x, t) = 1/u$. The error tolerances for the time integrator are $atol = rtol = 1 \times 10^{-8}$ for the solution and $atol = rtol = 1 \times 10^{-4}$ for all its derivatives.

Figure 6 shows the corresponding solutions at specified times which are typical for this problem. These results are also in close agreement with those found in [9, 10]. With our methods, only 128 points on $[0, 1]$ are used. Furthermore, we are able to easily evaluate other potential quantities of interest. In particular, higher derivatives may be easily and much more reliably evaluated than with a moving finite difference approach. Also, because we are using the inverse of the solution as the monitor function, it is easy to track the movement of the minimum of the solution. This makes it very simple to test asymptotic hypotheses.

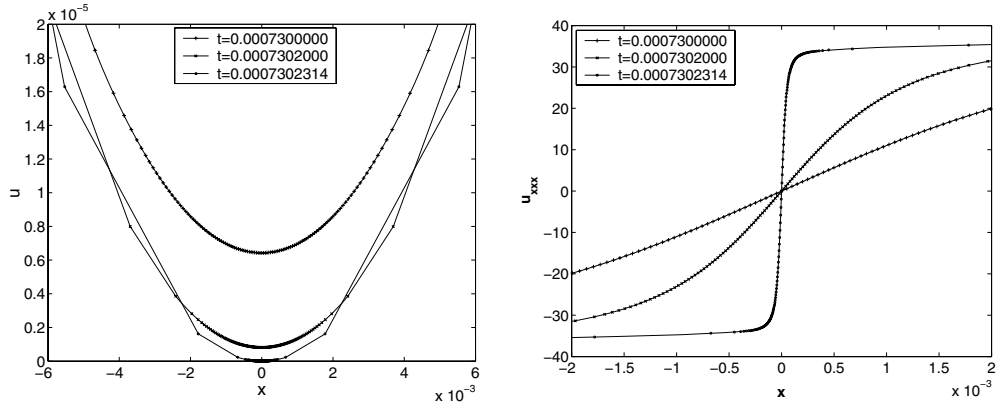


FIG. 5. *Solutions at times very close to the singularity time. Left: Solutions near the pinch-point. Right: Onset of initial singularity, formation of a jump discontinuity in the third derivative.*

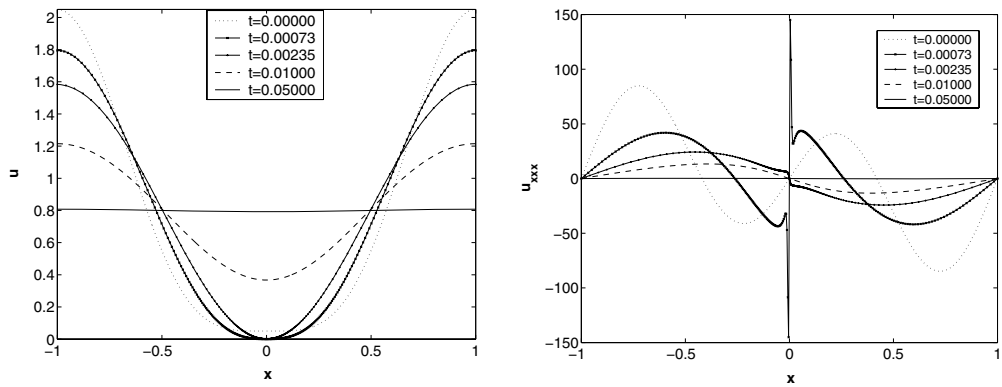


FIG. 6. *Solutions of the regularized system at typical times: $t = 0$, the initial profile, $t = 0.00073$, roughly the time when the pinch-off singularity occurs; $t = 0.002345$, roughly the time when the weak solution becomes strong and positive again; $t = 0.01$, strong solution decays to its mean; $t = 0.05$, solution almost decays to its mean $\bar{u} = 0.8$. Left: Typical solutions. Right: The third derivatives.*

7.3.2. Finite-time blow-up. Given the specific case when $f(u) = u$ and $g(u) = 4u^3$, (7.5) takes the form

$$u_t = -(uu_{xxx} + 4u^3 u_x)_x.$$

As for the unstable Cahn–Hilliard equation, the balance between a stable fourth-order operator and an unstable second-order one can lead to finite-time blow-up. This is particularly delicate here, as it is a critical case for blow-up [12, 41]. Given $f(u) = u$ and $g(u) = u^p$ we never have blow-up for $p < 3$, always have blow-up for $p > 3$, and sometimes have blow-up for $p = 3$. Moreover, the long time dynamics of solutions to this equation are very strongly determined by the mass of the initial data; for sufficient mass there is always blow-up, while for insufficient mass there is never blow-up. Not only that, but the set of allowable final-time blow-up profiles is also restricted by the initial mass [41].

Because of the combination of conservation of mass and finite-time blow-up, we immediately see that the singularity must form at a point. This problem is another

excellent test for our adaptive method, and in particular the adaptive [illegible] method.

We use both the conservative and Gauss collation methods when solving the problem

$$\text{Ex. 5} \quad \begin{cases} u_t = -(uu_{xxx} + 4u^3u_x)_x, \quad x \in (0,6),\ t > 0, \\ u_x(0,t) = u_{xxx}(0,t) = u_x(6,t) = u_{xxx}(6,t) = 0, \\ u(x,0) = \frac{m}{\sqrt{2\pi}} e^{-.5x^2}. \end{cases} \tag{7.10}$$

It is solved with $N = 41$ grid points, $atol = rtol = 1e-3$, and the monitor function

$$\tilde{M}(x,t) = u(x,t)^5 + |x^2 u_{xx}(x,t)|^5, \qquad M(x,t) = \tilde{M}(x,t) + \frac{1}{12}\int_0^6 \tilde{M}(s,t)ds.$$

Again, this particular monitor function has been chosen to be scale invariant in the blow-up region. One of the difficulties of this problem is that the solution approaches a blow-up profile which is compactly supported in the similarity variables, i.e.,

$$\lim_{t\to T}(T-t)^{1/5}u(x,t) = f(y) \quad \text{with} \quad y = \frac{x}{(T-t)^{1/5}},$$

where $f(y)$ is supported on a compact interval [41]. Moreover, there are only certain masses allowable for the similarity solution, so as in the previous example, a portion of the initial data must pinch-off if the initial data are sufficiently large. Depending on the initial data, this pinch-off may be before the blow-up time or else possibly precisely at the blow-up time. We have chosen the parameter $m = 2.65$ to illustrate this. Because of the pinch-off behavior near the boundary of the blow-up region, we cannot simply use the solution u for the monitor function but need the second-derivative as well to ensure that the numerical solution remains positive in the region where the curvature is very high. We have added the scaled curvature to place additional points near the edge of the support of the emerging similarity solution. This behavior is evident in Figure 7, where the spike forming at $x = 0$ is seen to detach from a small droplet. Eventually this droplet is lost due to lack of resolution. Figure 8 shows the grid on which these solutions were computed.

8. Conclusions. We have discussed the implementation of a high-resolution collocation method to integrate nonlinear fourth-order evolutionary equations on adaptive grids. By using a cell-averaged discretization we are able to satisfy a general discrete conservation property and also reduce stiffness in the associated ODE problem. By carefully choosing the monitor function we are also able to resolve very delicate solution features.

The resulting software, MOVCOL4, has been designed with a simple easily expandable interface making it appropriate for solving a general class of fourth-order problems which are of interest to an increasingly wide range of researchers working on PDEs involving high order equations. Advantages of MOVCOL4 include its ability to satisfy conservation equations, its high order accuracy, the local stencil (versus finite differences, which can cause difficulties for high order equations [39]), its relative ease of use, its ability to handle high derivatives (e.g., in boundary conditions) directly, its robust adaptivity, and its facility to build scale invariance into the MMPDE by a suitable choice of monitor function. We have demonstrated here that this code can be used to solve a variety of challenging problems with distinct features. We believe that it can be a useful tool for researchers from many different areas.

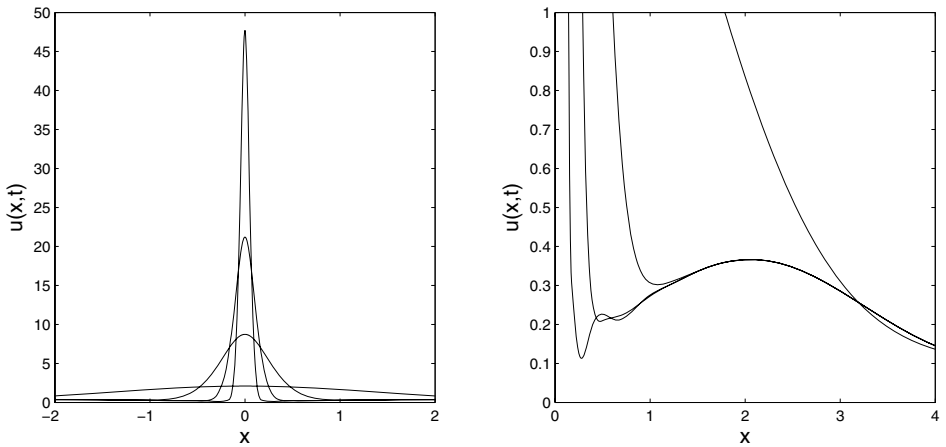


FIG. 7. *Blow-up solutions of an unstable thin-film equation. Left: Sample solutions. Right: Details near the edge of the support of the blow-up solution. As the magnitude of the solution increases at the origin, there is also a second singularity similar to the pinch-off singularity of the previous example. This point requires careful resolution.*

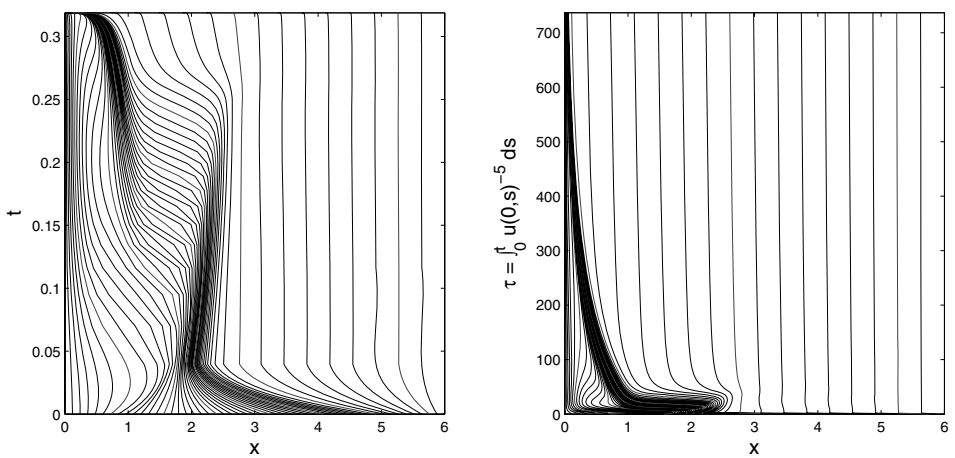


FIG. 8. *Left: Grid in the physical coordinates. Initially the points cluster where the second derivative of the initial data is large but eventually get pulled into the blow-up region. Right: Grid in physical coordinates and rescaled time. Notice in both cases the essentially uniform grid in the outer region away from the blow-up point. Also observe the smooth mesh trajectories in the rescaled time coordinate as blow-up is approached.*

There are a number of questions remaining about which formulations of fourth-order problems prove the most important and useful computationally. Overall solution complexity can increase when going from second to fourth order, so using the best formulation becomes correspondingly more important. Understanding of this issue will increase as experience is gained solving ever more challenging problems using codes such as this one.

It should be noted that MOVCOL4 and a related code have been used to compute

solutions to many other problems; e.g., see [17, 13, 19, 14, 27, 25]. Many of these demonstrate further flexibility in the codes, particularly for solving blow-up problems. In our limited experience, directly solving fourth-order problems with MOVCOL4 generally works at least as well as, and often substantially better than, a method applied to a converted system of second-order problems. In addition, for each of the problems, with solution features which vary over many different scales, having the adaptive features which require no a priori knowledge about the particular solution can be essential.

In conclusion, numerical methods are indispensable for solving the many challenging fourth-order problems having complex solution behavior, largely because the traditional theoretical tools are inapplicable. We expect that software like MOVCOL4 can be extremely useful both to underpin many of the numerical studies and, in other situations, to provide necessary verification of results for scientists who develop their own analytical and numerical methods with features tailored to their particular applications. Conversely, increased analysis of fourth-order problems will play an indispensable role in designing and refining codes such as MOVCOL4, as it would be naive to view current codes as a panacea for understanding the complex structures of these challenging nonlinear problems.

Acknowledgments. The authors are most grateful to Weizhang Huang for his insights and encouragement, and to Andrea Bertozzi for helpful comments on an early version of the paper.

REFERENCES

[1] U. M. Ascher, J. Christiansen, and R. D. Russell, *A collocation solver for mixed order systems of boundary value problems*, Math. Comp., 33 (1979), pp. 659–679.

[2] U. M. Ascher, R. M. M. Mattheij, and R. D. Russell, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*, Prentice–Hall, Englewood Cliffs, NJ, 1988.

[3] J. W. Barrett, J. F. Blowey, and H. Garke, *Finite element approximation of a fourth-order nonlinear degenerate parabolic equation*, Numer. Math., 80 (1998), pp. 525–556.

[4] G. Beckett, J. A. Mackenzie, A. Ramage, and D. M. Sloan, *On the numerical solution of one-dimensional PDEs using adaptive methods based on equidistribution*, J. Comput. Phys., 167 (2001), pp. 372–392.

[5] G. Beckett and J. A. Mackenzie, *On a uniformly accurate finite difference approximation of a singularly perturbed reaction-diffusion problem using grid equidistribution*, J. Comput. App. Math., 131 (2001), pp. 381–405.

[6] E. Beretta, M. Bertsch, and R. Dal Passo, *Nonnegative solutions of a fourth-order nonlinear degenerate parabolic equation*, Arch. Rational Mech. Anal., 129 (1995), pp. 175–200.

[7] M. Berger and R. Kohn, *A rescaling algorithm for the numerical calculation of blowing-up solutions*, Comm. Pure Appl. Math., 41 (1988), pp. 841–863.

[8] F. Bernis and A. Friedman, *Higher-order nonlinear degenerate parabolic equations*, J. Differential Equations, 83 (1990), pp. 179–206.

[9] A. L. Bertozzi, *Loss and gain of regularity in a lubrication equation for thin viscous films*, in Free Boundary Problems: Theory and Applications, Proceedings of the International Colloquium on Free Boundary Problems, (Toledo, Spain, 1993), J. I. Díaz, M. A. Herrero, A. Liñán, and J. L. Vázquez, eds., Pitman Res. Notes Math. Ser. 323, Longman Sci. Tech., Harlow, UK, 1995, pp. 72–85.

[10] A. L. Bertozzi, *The mathematics of moving contact lines in thin liquid films*, Notices Amer. Math. Soc., 45 (1998), pp. 689–697.

[11] A. L. Bertozzi and M. C. Pugh, *The lubrication approximation for thin viscous films: Regularity and long time behavior of weak solutions*, Comm. Pure. Appl. Math., 49 (1996), pp. 85–123.

[12] A. L. Bertozzi and M. C. Pugh, *Long-wave instabilities and saturation in thin film equations*, Comm. Pure Appl. Math., 51 (1998), pp. 625–661.

[13] C. J. Budd, S. Chen, and R. D. Russell, *New self-similar solutions of the nonlinear Schrödinger equation with moving mesh computations*, J. Comput. Phys., 152 (1999), pp. 756–789.
[14] C. J. Budd, V. A. Galaktionov, and J. F. Williams, *Self-similar blow-up in higher-order semilinear parabolic equations*, SIAM J. Appl. Math., 64 (2004), pp. 1775–1809.
[15] C. J. Budd, W. Huang, and R. D. Russell, *Moving mesh methods for problems with blow-up*, SIAM J. Sci. Comput., 17 (1996), pp. 305–327.
[16] C. J. Budd and M. D. Piggott, *The geometric integration of scale-invariant ordinary and partial differential equations*, J. Comput. Appl. Math., 128 (2001), pp. 399–422.
[17] C. J. Budd, R. Caraterro, and R. D. Russell, *Precise computations of chemotactic collapse using moving mesh methods*, J. Comput. Phys., 202 (2005), pp. 462–487.
[18] C. J. Budd and J. F. Williams, *Uniform Error Estimates for PDEs with Singularities*, in preparation.
[19] C. J. Budd, V. Rottschäfer, and J. F. Williams, *Multi-bump, blow-up self-similar solutions of the complex Ginzburg–Landau equation*, SIAM J. Appl. Dyn. Syst., 4 (2005), pp. 649–678.
[20] K. Clark, J. E. Flaherty, and M. S. Shephard, *Introduction to special issue on adaptive methods for partial differential equations*, Appl. Numer. Math., 14 (1994), pp. 1–3.
[21] W. Cao, W. Huang, and R. D. Russell, *An error indicator monitor function for an r-adaptive finite-element method*, J. Comput. Phys., 170 (2001), pp. 871–892.
[22] A. R. Champneys, P. J. McKenna, and P. A. Zegeling, *Solitary waves in nonlinear beam equations: Stability, fission and fusion*, Nonlinear Dynam., 21 (2000), pp. 31–53.
[23] C. de Boor, *Good approximation by splines with variable knots.* II, in Proceedings of the Conference on the Numerical Solution of Differential Equations, Lecture Note in Math. 363, Springer, Berlin, 1974, pp. 12–20.
[24] C. de Boor, *A Practical Guide to Splines*, Springer-Verlag, Berlin, New York, 1978.
[25] J. D. Evans, V. A. Galaktionov, and J. F. Williams, *Blow-up and global asymptotics of the unstable Cahn–Hilliard equation*, SIAM J. Math. Anal., 38 (2006), pp. 64–102.
[26] C. W. Gear and L. R. Petzold, *ODE methods for the solution of differential/algebraic systems*, SIAM J. Numer. Anal., 21 (1984), pp. 716–728.
[27] V. A. Galaktionov and J. F. Williams, *Blow-up in a fourth-order semilinear parabolic equation from explosion-convection theory*, European J. Appl. Math., 14 (2003), pp. 745–764.
[28] D. F. Hawken, J. J. Gottlieb, and J. S. Hansen, *Review of some adaptive node-movement techniques in finite element and finite difference solutions of PDEs*, J. Comput. Phys., 95 (1991), pp. 254–302.
[29] W. Huang, *Convergence analysis of finite element solution of one-dimensional singularly perturbed differential equations on equidistributing meshes*, Int. J. Numer. Anal. Model., 2 (2005), pp. 57–74.
[30] W. Huang, Y. Ren, and R. D. Russell, *Moving mesh methods based on moving mesh partial differential equations*, J. Comput. Phys., 113 (1994), pp. 279–290.
[31] W. Huang, Y. Ren, and R. D. Russell, *Moving mesh partial differential equations (MMPDEs) based on the equidistribution principle*, SIAM J. Numer. Anal., 31 (1994), pp. 709–730.
[32] W. Huang and R. D. Russell, *A moving collocation method for solving time dependent partial differential equations*, Appl. Numer. Math., 20 (1996), pp. 101–116.
[33] W. Huang and R. D. Russell, *Analysis of moving mesh partial differential equations with spatial smoothing*, SIAM J. Numer. Anal., 34 (1997), pp. 1106–1126.
[34] J. A. Mackenzie and M. L. Robertson, *A moving mesh method for the solution of the one-dimensional phase-field equations*, J. Comput. Phys., 181 (2002), pp. 526–544.
[35] E. V. L. de Mello and O. T. da Silveira Filho, *Numerical study of the Cahn–Hilliard equation in one, two, and three dimensions*, Phys. A, 347 (2005), pp. 429–443.
[36] L. Peletier and W. Troy, *Spatial Patterns: Higher-order Models in Physics and Mechanics*, Birkhäuser, New York, 2001.
[37] L. Petzold, *A Description of DASSL: A Differential/Algebraic System Solver*, Tech. report SAND82-8637, Sandia National Labs, Livermore, CA, 1982.
[38] L. G. Reyna and M. J. Ward, *Metastable internal layer dynamics for the viscous Cahn–Hilliard equation*, Methods Appl. Anal., 2 (1995), pp. 285–306.
[39] P. Saucez, A. Vande Wouwer, and P. A. Zegeling, *Adaptive method of lines solutions for the extended fifth order Korteweg–DeVries equation*, J. Comput. Appl. Math., 183 (2005), pp. 343–357.

[40] J. M. Stockie, J. A. MacKenzie, and R. D. Russell, *A moving mesh method for one-dimensional hyperbolic conservation laws*, SIAM J. Sci. Comput., 22 (2000), pp. 1791–1813.

[41] T. P. Witelski, A. J. Bernoff, and A. L. Bertozzi, *Blowup and dissipation in a critical-case unstable thin film equation*, European J. Appl. Math., 15 (2004), pp. 223–256.

[42] Z. Wu, J. Zhao, J. Yin, and H. Li, *Nonlinear Diffusion Equations*, World Scientific, River Edge, NJ, 2001.

[43] L. Zhornitskaya and A. L. Bertozzi, *Positivity-preserving numerical schemes for lubrication-type equations*, SIAM J. Numer. Anal., 37 (2000), pp. 523–555.

SIAM J. SCI. COMPUT.
Vol. 29, No. 1, pp. 221–244
© 2007 Society for Industrial and Applied Mathematics

KINETIC FLUX VECTOR SPLITTING SCHEMES FOR IDEAL QUANTUM GAS DYNAMICS*

JAW-YEN YANG†, TSE-YANG HSIEH†, AND YU-HSIN SHI†

Abstract. Novel quantum kinetic flux vector splitting schemes are presented for the computation of ideal quantum gas dynamical flows. The quantum Boltzmann equation approach is adopted and both Bose–Einstein and Fermi–Dirac gases are considered. Formulas for one spatial dimension are first derived, and the resulting kinetic flux vector splitting scheme is tested for shock tube flows. A modified flux vector splitting based on the BGK model is also presented to reduce the numerical diffusion. Implementation of the efficient weighted essentially nonoscillatory method to yield a class of high resolution methods is also devised. Both the classical limit and the nearly degenerate limit are computed. The flow structures can all be accurately captured by the present kinetic flux vector splitting schemes. Results for one-dimensional boson gas under external potential are also included.

Key words. ideal quantum gases, quantum gas dynamics, kinetic flux vector splitting

AMS subject classifications. 15A15, 15A09, 15A23

DOI. 10.1137/050646664

1. Introduction.

In statistical mechanics, there are three kinds of equilibrium distributions for a system of N identical particles, namely, the Maxwell–Boltzmann, Bose–Einstein, and Fermi–Dirac distributions; the former one is for classical particles and the latter two are for quantum particles. The statistical properties of fermion and boson systems are profoundly different at a sufficiently low temperature. However, in the classical limit, both quantum distributions reduce to the Maxwell–Boltzmann distribution. The ideal classical gas dynamics can be described by the Maxwell–Boltzmann distribution which corresponds to the lowest order solution of the classical Boltzmann equation when the Chapman–Enskog procedure is employed (Chapman and Cowling, 1970). The conservation laws based on the Maxwell–Boltzmann distribution are the well-known Euler equations of classical gas dynamics. Analogous to the classical Boltzmann equation, a quantum Boltzmann equation for the transport phenomenon can be developed for fermions and bosons; see Kadanoff and Baym (1962). A formal solution procedure which generalizes the Chapman–Enskog method to solve the quantum Boltzmann equation was given by Uehling and Uhlenbeck (1933), where the first and second approximations of the distribution function and expressions for the viscosity and heat conductivity coefficients were given. The corresponding quantum gas dynamics can also be explored based on the quantum equilibrium distributions.

In the past 40 years, extensive theoretical and computational methods have been developed to solve the Euler equations of classical gas dynamics. Among the various methods, such as the exact and approximate Riemann solvers (see Hirsch 1988; Toro 1999), most are constructed based on the hyperbolic conservation laws of the Euler equations. Here, we shall denote numerical methods constructed based on distribution functions as kinetic methods. Some kinetic methods have been devised in the past to solve the Euler equations of classical gas dynamics. The beam scheme of

*Received by the editors December 3, 2005; accepted for publication (in revised form) August 10, 2006; published electronically February 2, 2007. This work was done under the auspices of the National Science Council, Taiwan through grants NSC 92-2210-E002-039 and 93-2210-E002-040.
http://www.siam.org/journals/sisc/29-1/64666.html
†Institute of Applied Mechanics, National Taiwan University, 106 Taipei, Taiwan (yangjy@gauss.iam.ntu.edu.tw, hsiehty@gmail.com, f92543072@ntu.edu.tw).

Sanders and Prendergast (1974), the equilibrium flux method of Pullin (1980), the method of Reitz (1981), and the kinetic flux vector splitting (KFVS) of Patterson (1971), Deshpande (1986), and Chou and Baganoff (1997) are some examples. More recent works on kinetic methods have been given by Xu and Prendergast (1994), Xu (1998), and Ohwada (2002). An interesting explicit scheme, which is called the beam scheme, has been presented by Sanders and Prendergast (1974) for solving the equilibrium limit of the classical Boltzmann equation. The derivation was based on the local thermodynamic Maxwell–Boltzmann equilibrium distribution and resulted in a novel method for solving the transport processes governed by the Euler equations of Newtonian gas dynamics. Later, the beam scheme was successfully extended to the relativistic Boltzmann transport equation based on the Jüttner distribution by Yang et al. (1997). Recently, the concept of beam scheme has been extended and generalized to ideal quantum gas dynamics by Yang and Shi (2006) based on the Bose–Einstein and Fermi–Dirac distributions. The beam scheme turns out to also be a flux vector splitting method and has all the desirable features of modern characteristics–based wave propagating numerical methods for hyperbolic conservation laws of gas dynamics. We note that the beam scheme is based on the equilibrium flow limit of the Boltzmann equation, while KFVS is based on the convectional Boltzmann equation which governs the free molecular flow. The basic KFVS scheme is of first order accuracy and thus numerically diffusive. Also, due to the lack of collisions in the evaluation of the numerical flux, KFVS schemes give a smeared solution, especially at the contact discontinuity. In order to include real particle collisions in the gas evolution to reduce overdiffusivity, the BGK relaxation time model (Bhatnagar, Gross, and Krook, 1954) was employed and a class of BGK-type KFVS schemes has been devised by Xu (1998) which improves the original KFVS scheme considerably. However, this numerical diffusion of first order KFVS (or any upwind) methods can also be significantly reduced by the implementation of higher-order methods.

In this work, we shall adopt and generalize the concept of the KFVS scheme to devise a numerical method for the computation of ideal quantum gas dynamics. We start with the quantum Boltzmann equation, in which the particles obey the quantum statistics and the particles of the system are in the excited states. It is well known that the boson or fermion particles in the excited states are in the gas phase and that in the low-density (or high temperature) limit ($\beta\mu \approx -\infty$) they behave like classical ideal gas governed by the Maxwell–Boltzmann distribution. Here μ is the chemical potential and $\beta = 1/k_B T$, with k_B the Boltzmann constant and T the temperature. At low temperatures, in the neighborhood of degeneracy temperature, T_0, corresponding to $n\lambda_B^3 = 1$ (here n is the number density and λ_B is the thermal de Broglie wavelength), the quantum effects become important. The present method considers both the Bose–Einstein and Fermi–Dirac gas behaviors in the regime between the nearly degenerate limit and the classical limit. In Bose–Einstein statistics, the condensed phase of boson gas is not considered. First, the KFVS scheme for ideal quantum gas dynamics is derived based on the solution of the initial value problem of the collisionless Boltzmann equation. Second, we follow Xu (1998) and devise a BGK-type KFVS scheme based on a collisional model Boltzmann equation. Both the basic KFVS and BGK-type KFVS schemes are of first order accuracy and thus numerically too diffusive for practical applications. Nevertheless, they are entropy-satisfying and can exclude the expansion shocks. Furthermore, this numerical diffusion of first order KFVS and BGK–KFVS methods can be significantly reduced by the implementation of higher-order methods. Here we adopt the efficient weighted essentially nonoscillatory (WENO) method of Jiang and Shu (1996) for our basic first order quantum KFVS scheme to yield a

class of high resolution entropy-satisfying methods for computing ideal quantum gas dynamical flows. Implementation of a higher-order BGK–KFVS scheme can be done following Xu (1998) and will not be presented here. Considering the existing extensive works on experimental and computational classical gas dynamics, the present quantum KFVS schemes can be potentially useful for revealing various dynamical aspects of ideal quantum gas through mathematical and physical analogies.

The paper is organized as follows. We first briefly describe the elements of the quantum Boltzmann transport equation in section 2. In section 3, the basic KFVS scheme in one spatial dimension is derived based on the convectional Boltzmann equation for bosons and fermions. In section 4, to reduce the diffusivity of the basic KFVS scheme, a modified KFVS scheme is considered based on the BGK model. Implementations of high-order methods using the WENO scheme are outlined in section 5. In section 6, numerical experiments with quantum shock tube flows in both classical and nearly degenerate limits and one-dimensional boson gas under harmonic potential are given to illustrate the method. Lastly, discussion and concluding remarks are given in section 7.

2. Elements of the quantum Boltzmann equation. In this section, we briefly describe the elements of the quantum Boltzmann transport equation relevant to the development of the present work. Following Kadanoff and Baym (1962), we consider the Boltzmann equation

$$\left(\frac{\partial}{\partial t} + \vec{u}\cdot\nabla_{\vec{x}} - \nabla V(\vec{x},t)\cdot\nabla_{\vec{u}}\right) f(\vec{x},\vec{u},t) = \left(\frac{\delta f}{\delta t}\right)_{col.}, \tag{2.1}$$

where $\vec{u}$ is the particle velocity, $V(\vec{x},t)$ is the externally applied field, and $f(\vec{x},\vec{u},t)$ is the distribution function which represents the average density of particles with velocity $\vec{u}$ at the space-time point $\vec{x}$, t. The $(\delta f/\delta t)_{col.}$ denotes the nonlinear integral collision term. The left-hand side of (2.1) represents the convection of particles in phase space under molecular motions and external forces and, with $(\delta f/\delta t)_{col.} = 0$, it will be denoted as the convectional Boltzmann equation in the present work. A formal solution procedure generalizing the Chapman–Enskog method to solve (2.1) was given by Uehling and Uhlenbeck (1933). In recent years, the development of numerical methods for solving the quantum Boltzmann equation has become an active research subject; see Garcia and Wagner (2003). A recent review on the numerical methods for the dynamics and transport of atomic quantum gases has been given (see Minguzzi et al. 2004 and the references therein). The major mathematical difficulty of solving (2.1) is due to the nonlinear integral collision term. Similar to the simple collision model given by a BGK approach for the classical gas dynamics, the relaxation time collision model has been extended to treat the scattering processes of electrons, phonons, and other quantum particles; see Chen (2005). A corresponding Wigner–BGK equation for the quantum moment equations has also been given by Degond and Ringhofer (2003). Here, we shall consider only the cases where the collision term can be neglected and demand that

$$\left(\frac{\delta f}{\delta t}\right)_{col.} = 0. \tag{2.2}$$

That means we consider the convectional Boltzmann equation for ideal quantum gases. This corresponds to two cases. The first case is the free molecular flow, where the mean free path of the gas system is very large or infinite, and the second is the

equilibrium flow, where the mean collision time is infinitely short and the distribution function is always in local equilibrium. The solution to (2.1) for the equilibrium flow is given by

$$f(\vec{x},\vec{u},t)=\left(\frac{m^4}{h^3}\right)\left\{\exp\left[\frac{m(\vec{u}-\vec{U}(\vec{x},t))^2}{2k_BT(\vec{x},t)}-\frac{\mu(\vec{x},t)}{k_BT(\vec{x},t)}\right]+\theta\right\}^{-1}, \tag{2.3}$$

where m is the particle mass, h is the Planck constant, and $\theta=+1$ denotes the Fermi–Dirac statistics and $\theta=-1$ the Bose–Einstein statistics. To complete the equilibrium solution, we have to determine the unknown functions, namely, the gas temperature, $T(\vec{x},t)$, chemical potential, $\mu(\vec{x},t)$, and velocity vector, $\vec{U}(\vec{x},t)$, which appear in (2.3). These flow parameters can be determined by making use of the conservation laws for number of particles, momentum, and energy. These conservation laws can be obtained by multiplying (2.1) by 1, $\vec{u}$, and $\frac{\vec{u}^2}{2}$ and then integrating the resulting equations over all $\vec{u}$. The integrals of the collision terms in all three cases vanish automatically, and we have the conservation laws for the conserved macroscopic quantities, i.e., the density ρ, momentum $\rho\vec{U}$, and energy density $\rho\epsilon$ as follows:

$$\frac{\partial\rho}{\partial t}+\nabla_{\vec{x}}\cdot\rho\vec{U}=0, \tag{2.4}$$

$$\frac{\partial\rho\vec{U}}{\partial t}+\nabla_{\vec{x}}\cdot(P+\rho\vec{U}\vec{U})=-\rho\nabla_{\vec{x}}V, \tag{2.5}$$

$$\frac{\partial\rho\epsilon}{\partial t}+\nabla_{\vec{x}}\cdot\left(\vec{U}\left(P+\frac{1}{2}\rho\vec{U}^2+e\right)\right)=-\rho\vec{U}\cdot\nabla_{\vec{x}}V, \tag{2.6}$$

where

$$\rho(\vec{x},t)=\int f(\vec{x},\vec{u},t)d\vec{u}, \tag{2.7}$$

$$\rho(\vec{x},t)\vec{U}(\vec{x},t)=\int\vec{u}f(\vec{x},\vec{u},t)d\vec{u}, \tag{2.8}$$

$$\rho\vec{U}\vec{U}=\int\vec{u}\vec{u}f(\vec{x},\vec{u},t)d\vec{u}, \tag{2.9}$$

$$\rho(\vec{x},t)\epsilon(\vec{x},t)=\int\frac{\vec{u}^2}{2}f(\vec{x},\vec{u},t)d\vec{u}. \tag{2.10}$$

Other higher-order moments can also be defined such as stress tensor and the heat flux vector. For the local equilibrium solution, one can obtain these macroscopic quantities in closed form in terms of the Bose or Fermi function (Huang 1987; Pathria 1996). To illustrate the KFVS scheme, we consider the following local quantum equilibrium distribution in one spatial dimension:

$$f(x,u,t)=\left(\frac{m^2}{h}\right)\left\{\frac{1}{z(x,t)}\exp\left\{\frac{m[u-U(x,t)]^2}{2k_BT(x,t)}\right\}+\theta\right\}^{-1}, \tag{2.11}$$

where $z(x,t)=exp\{\mu(x,t)/k_BT(x,t)\}$ is the fugacity. Then the density $\rho(x,t)$ is given by

$$\rho(x,t)=\int_{-\infty}^{\infty}f(x,u,t)du=\frac{m\mathcal{Q}_{\frac{1}{2}}(z)}{\lambda}, \tag{2.12}$$

the momentum $\rho(x,t)U(x,t)$ by

$$\rho(x,t)U(x,t) = \int_{-\infty}^{\infty} uf(x,u,t)du = \frac{m\mathcal{Q}_{\frac{1}{2}}(z)U(x,t)}{\lambda}, \tag{2.13}$$

and the energy density $\rho(x,t)\epsilon(x,t)$ by

$$\rho(x,t)\epsilon(x,t) = \int_{-\infty}^{\infty} \frac{u^2}{2} f(x,u,t)du = \frac{\mathcal{Q}_{\frac{3}{2}}(z)}{2\beta\lambda} + \frac{m\mathcal{Q}_{\frac{1}{2}}(z)U(x,t)^2}{2\lambda}, \tag{2.14}$$

where $\lambda = \sqrt{\frac{\beta h^2}{2\pi m}}$ is the thermal wavelength.

In the above, $\mathcal{Q}_\nu(z)$ denotes either the Bose–Einstein or the Fermi–Dirac function of order ν, which is defined by

$$\mathcal{Q}_\nu(z) = \begin{cases} \mathcal{G}_\nu(z) & \text{for Bose gas,} \\ \mathcal{F}_\nu(z) & \text{for Fermi gas,} \end{cases} \tag{2.15}$$

where $\mathcal{G}_\nu(z)$ denotes the Bose–Einstein integral and $\mathcal{F}_\nu(z)$ denotes the Fermi–Dirac integral of order ν,

$$\mathcal{G}_\nu(z) \equiv \frac{1}{\Gamma(\nu)} \int_0^\infty \frac{x^{\nu-1}}{z^{-1}e^x - 1} dx = \sum_{l=1}^{\infty} \frac{z^l}{l^\nu}, \tag{2.16}$$

$$\mathcal{F}_\nu(z) \equiv \frac{1}{\Gamma(\nu)} \int_0^\infty \frac{x^{\nu-1}}{z^{-1}e^x + 1} dx = \sum_{l=1}^{\infty} \frac{-(-z)^l}{l^\nu}, \tag{2.17}$$

and $\Gamma(\nu)$ is the Gamma function.

We note that the Bose–Einstein integral with $\nu \leq 1$ diverges as $z \to 1$ and that $g_{\frac{3}{2}}(z)$ remains finite at $z = 1$, but with infinite slope, while $g_\nu(z)$ with $\nu > \frac{3}{2}$ have finite values and slopes at $z = 1$. The Fermi-Dirac integral is $\mathcal{F}_\nu(z) \to z$ for small z. Since the chemical potential for a dilute system is large and negative, the classical limit corresponds to small z. Conversely, at low temperature the chemical potential approaches the Fermi energy so that the fugacity for a Fermi system is large. An asymptotic expansion of the Fermi function for large fugacity z due to Sommerfeld can be used.

It is also noted that thermodynamic functions, the Bose–Einstein condensation and the Fermi–Dirac degeneracy of an ideal (noninteracting) gas of N quantum particles, are peculiarly related to the space dimensionality (de Groot, Hooyman, and Ten Seldam 1950; Aguilera-Navarro, de Llano, and Solis 1999; Grether, de Llano, and Solis 2003). Although these results look quite similar, these two functions behave quite differently in the nearly degenerate limit. For example, in boson gas, the fugacity z could not be larger than 1 because of non-negative density, while in Fermi gas such a restriction does not exist.

3. Quantum KFVS scheme. In this section, the basic first order KFVS scheme is derived based on the convectional Boltzmann equation at the gas evolution stage. We consider the problems in one spatial dimension in which the macroscopic quantities are assumed to be functions of x and t. Divide the space into a number of cells. Each cell occupies a small space $x \in [x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}]$, where $j + \frac{1}{2}$ denotes the cell

interface between cells j and $j+1$, and the cell center is located at x_j. With the initial mass, momentum, and energy densities given inside each cell j,

$$\mathbf{W}_j = (\rho_j, \rho_j U_j, \rho_j \epsilon_j)^T, \tag{3.1}$$

the fugacity z_j is obtained from (2.12) and (2.14) through the following relationship:

$$2(\rho_j \epsilon_j) - \left(\frac{h^2}{2\pi m^2}\right)\left(\frac{\rho_j}{\mathcal{Q}_{\frac{1}{2}}(z_j)}\right)^3 \mathcal{Q}_{\frac{3}{2}}(z_j) - \frac{(\rho_j U_j)^2}{\rho_j} = 0. \tag{3.2}$$

With the temperature T_j obtained from (2.12),

$$T_j = \frac{2\pi k_B}{m}\left(\frac{\rho_j}{\mathcal{Q}_{\frac{1}{2}}(z_j)}\right)^2, \tag{3.3}$$

we can then define a local equilibrium state g_j within the jth cell, which is,

$$g_j = \left(\frac{m^2}{h}\right)\left\{z_j^{-1} \exp\left(\frac{m[u-U_j]^2}{2k_B T_j}\right) + \theta\right\}^{-1}. \tag{3.4}$$

Consider the one-dimensional convectional Boltzmann equation

$$\frac{\partial f}{\partial t} + u\frac{\partial f}{\partial x} = 0. \tag{3.5}$$

We get the following initial condition around the cell interface at $x = x_{j+\frac{1}{2}}$:

$$f(x, u, t=0) = f_1(x, u) = (1 - H(x - x_{j+\frac{1}{2}}))g_j + H(x - x_{j+\frac{1}{2}})g_{j+1}, \tag{3.6}$$

where $H(x)$ is the Heaviside function.

It is noted that the initial distribution functions on both sides of each cell interface are assumed to be in local equilibrium.

The solution of the above initial value problem, $f(x, u, t)$ at $x_{j+\frac{1}{2}}$ and time t, becomes

$$f(x_{j+\frac{1}{2}}, u, t) = f_1(x_{j+\frac{1}{2}} - ut, u) = H(u)g_j + (1 - H(u))g_{j+1}. \tag{3.7}$$

From the above distribution function, the numerical fluxes for the mass, momentum, and energy across the cell interface can be constructed, which are

$$\begin{aligned} F^1_{j+\frac{1}{2}} = \begin{pmatrix} F^1_\rho \\ F^1_{\rho U} \\ F^1_{\rho\epsilon} \end{pmatrix}_{j+\frac{1}{2}} &= \int u\psi_\alpha f(x_{j+\frac{1}{2}}, u, t)du \\ &= \int_{u\geq 0} u\psi_\alpha g_j du + \int_{u<0} u\psi_\alpha g_{j+1} du, \end{aligned} \tag{3.8}$$

where ψ_α stands for the moments $\psi_\alpha = (1, u, \frac{u^2}{2})^T$.

The fluxes can be further expressed as

$$F^1_{j+\frac{1}{2}} = F^+_j + F^-_{j+1}, \tag{3.9}$$

where

$$F_j^+ \equiv \int_{u\geq 0} u\psi_\alpha g_j du, \tag{3.10}$$
$$F_{j+1}^- \equiv \int_{u<0} u\psi_\alpha g_{j+1} du.$$

The details of the integrations in (3.10) can be directly obtained, and they are

$$F_j^\pm = \begin{pmatrix} \frac{\sqrt{\pi}}{2}\mathbf{K}_j^{\frac{1}{2}} U_j \mathbf{A}_j^\pm + \frac{1}{2}\mathbf{K}_j \mathbf{B}_j^\pm \\ \frac{\sqrt{\pi}}{2}\mathbf{K}_j^{\frac{1}{2}} U_j^2 \mathbf{A}_j^\pm + \mathbf{K}_j U_j \mathbf{B}_j^\pm + \frac{\sqrt{\pi}}{4}\mathbf{K}_j^{\frac{3}{2}} \mathbf{C}_j^\pm \\ \frac{\sqrt{\pi}}{4}\mathbf{K}_j^{\frac{1}{2}} U_j^3 \mathbf{A}_j^\pm + \frac{3}{4}\mathbf{K}_j U_j^2 \mathbf{B}_j^\pm + \frac{3\sqrt{\pi}}{8}\mathbf{K}_j^{\frac{3}{2}} U_j \mathbf{C}_j^\pm + \frac{1}{4}\mathbf{K}_j^2 \mathbf{D}_j^\pm \end{pmatrix}, \tag{3.11}$$

$$\mathbf{K}_j = \frac{2k_B T_j}{m}, \mathbf{A}_j^\pm = \frac{m^2}{h}(\mathcal{Q}_{\frac{1}{2}}(z_j) \pm sgn(U_j)\mathcal{Q}_{\frac{1}{2}}(h_j, z_j)),$$
$$\mathbf{B}_j^\pm = \pm\frac{m^2}{h}\tilde{\mathcal{Q}}_1(h_j, z_j), \mathbf{C}_j^\pm = \frac{m^2}{h}(\mathcal{Q}_{\frac{3}{2}}(z_j) \pm sgn(U_j)\mathcal{Q}_{\frac{3}{2}}(h_j, z_j)), \tag{3.12}$$
$$\mathbf{D}_j^\pm = \pm\frac{m^2}{h}\tilde{\mathcal{Q}}_2(h_j, z_j), h_j = \frac{mU_j^2}{2k_B T_j}.$$

Here, the sign function is defined as

$$sgn(U) = \begin{cases} +1, U \geq 0, \\ -1, U < 0, \end{cases} \tag{3.13}$$

and the definitions of $\mathcal{Q}_\nu(h,z)$ and $\tilde{\mathcal{Q}}_\nu(h,z)$ are given by

$$\mathcal{Q}_\nu(h,z) = \begin{cases} \mathcal{G}_\nu(h,z) \text{ for Bose gas,} \\ \mathcal{F}_\nu(h,z) \text{ for Fermi gas,} \end{cases}$$
$$\tilde{\mathcal{Q}}_\nu(h,z) = \begin{cases} \tilde{\mathcal{G}}_\nu(h,z) \text{ for Bose gas,} \\ \tilde{\mathcal{F}}_\nu(h,z) \text{ for Fermi gas,} \end{cases} \tag{3.14}$$

where the incomplete Bose–Einstein integrals $\mathcal{G}_\nu(h,z)$ and $\tilde{\mathcal{G}}_\nu(h,z)$ are defined as

$$\mathcal{G}_\nu(h,z) \equiv \frac{1}{\Gamma(\nu)}\int_0^h \frac{x^{\nu-1}}{z^{-1}e^x - 1}dx = \frac{1}{\Gamma(\nu)}\sum_{l=1}^\infty \Gamma(lh,\nu)\frac{z^l}{l^\nu}, \tag{3.15}$$
$$\tilde{\mathcal{G}}_\nu(h,z) \equiv \frac{1}{\Gamma(\nu)}\int_h^\infty \frac{x^{\nu-1}}{z^{-1}e^x - 1}dx = \mathcal{G}_\nu(z) - \mathcal{G}_\nu(h,z),$$

and $\Gamma(lh,\nu)$ is the incomplete Gamma function given in the IMSL (International Mathematical and Statistical Library).

The incomplete Fermi–Dirac integrals $\mathcal{F}_\nu(h,z)$ and $\tilde{\mathcal{F}}_\nu(h,z)$ are defined as

$$\mathcal{F}_\nu(h,z) \equiv \frac{1}{\Gamma(\nu)}\int_0^h \frac{x^{\nu-1}}{z^{-1}e^x + 1}dx = \frac{1}{\Gamma(\nu)}\sum_{l=1}^\infty \Gamma(lh,\nu)\frac{-(-z)^l}{l^\nu}, \tag{3.16}$$
$$\tilde{\mathcal{F}}_\nu(h,z) \equiv \frac{1}{\Gamma(\nu)}\int_h^\infty \frac{x^{\nu-1}}{z^{-1}e^x + 1}dx = \mathcal{F}_\nu(z) - \mathcal{F}_\nu(h,z).$$

Using the above numerical fluxes, the flow variables ρ_j, $\rho_j U_j$, $\rho_j \epsilon_j$ inside each cell can be updated as

$$\mathbf{W}_j^{n+1} = \mathbf{W}_j^n - \sigma\left(F^1_{j+\frac{1}{2}} - F^1_{j-\frac{1}{2}}\right) - \Delta t S_j, \tag{3.17}$$

where n is the time step number, and $\sigma = \Delta t/\Delta x$, where Δt is the time step size, and Δx is the mesh size and $S = (0, -\rho\frac{\partial V}{\partial x}, -\rho\epsilon\frac{\partial V}{\partial x})^T$.

The scheme defined by (3.17) can be considered as a first order upwind scheme (based on KFVS) for the following conservation laws:

$$\frac{\partial \mathbf{W}}{\partial t} + \frac{\partial F}{\partial x} = S, \tag{3.18}$$

where

$$\mathbf{W} = \begin{pmatrix} \rho \\ \rho U \\ \rho\epsilon \end{pmatrix},\ F = \begin{pmatrix} \rho U \\ \rho U^2 + P \\ U(\rho\epsilon + P) \end{pmatrix},\ S = \begin{pmatrix} o \\ -\rho\frac{\partial V}{\partial x} \\ -\rho U\frac{\partial V}{\partial x} \end{pmatrix}. \tag{3.19}$$

The Jacobians of the present kinetic flux vector split fluxes $F^{\pm}$ do not commute, and the Von Neumann stability analysis becomes nonobvious in this case. Here, we follow the approach given by Mandal and Deshpande (1994). The stability for the exact solution, (3.7), can be readily given by

$$\left| u\frac{\Delta t}{\Delta x} \right| \leq 1. \tag{3.20}$$

This condition imposes a rather severe restriction on particle velocity u for finite values of Δx and Δt. The moments of the distribution function in the moment method strategy in the KFVS scheme involve a particle velocity range from $-\infty$ to $+\infty$. Thus, the moments do not receive contributions due to particle velocities outside the cut-off bounds required by (3.20). A reasonable choice is to enlarge the range from $U - \chi c$ to $U + \chi c$ to guarantee the accurate evaluations of moments. The value of χ will be determined by numerical experience. Here we choose $\chi = 3$, and now the time step size can be chosen by the stability condition,

$$\Delta t \leq \frac{\Delta x}{\max(|U_j| + 3c_j)}, \tag{3.21}$$

where c_j is the local speed of sound,

$$c_j = \sqrt{\frac{3}{m\beta}\frac{\mathcal{Q}_{\frac{3}{2}}(z_j)}{\mathcal{Q}_{\frac{1}{2}}(z_j)}}. \tag{3.22}$$

The time step size can be decided by multiplying the right-hand side of (3.21) with a CFL number. With this stability condition, (3.21), for the present KFVS scheme, we can use CFL = 1 in the calculations given below.

At this stage, we have completed the description of the first order KFVS scheme for the ideal quantum gas.

In the computation, we employ dimensionless quantities, and they are defined by the following:

$$\hat{u} = \frac{u}{\xi}, \hat{c} = \frac{c}{\xi} = \sqrt{3\hat{T}\frac{\mathcal{Q}_{3/2}(z)}{\mathcal{Q}_{1/2}(z)}}, \tag{3.23}$$

$$\hat{\rho} = \frac{\rho}{(\Omega/h)}, \hat{\rho}\hat{U} = \frac{\rho U}{(\Omega\xi/h)}, \hat{\rho}\hat{\epsilon} = \frac{\rho\epsilon}{(\Omega\xi^2/h)},$$

where $\Omega = m\sqrt{2\pi m k_B T_{ref}}$, $\xi = \sqrt{(k_B T_{ref})/m}$, and $\hat{T} = T/T_{ref}$.

After each time step, we can get new macroscopic properties $(\rho_j,\ \rho_j U_j,\ \rho_j \epsilon_j)$. In each time step, we need to find the values of fugacity z using (3.2) and find the temperature T (or β) using (3.3) in each cell to determine the numerical fluxes. The dimensionless form of (3.2) is

$$2(\hat{\rho}\hat{\epsilon}) - \left(\frac{\hat{\rho}}{\mathcal{Q}_{\frac{1}{2}}(z)}\right)^3 \mathcal{Q}_{\frac{3}{2}}(z) - \frac{(\hat{\rho}\hat{U})^2}{\hat{\rho}} = 0. \tag{3.24}$$

The above equation can be solved for fugacity z by an iterative method.

After obtaining fugacity z, we can get temperature T, and thus we can construct the local equilibrium distribution function for each cell at the new time step, and finally advance the solution for the next time step by (3.7).

It is appropriate to note at this stage that the physical requirement for the transition from the quantum Boltzmann equation to the quantum Euler equations is based on the assumption of a local equilibrium state. In deriving the KFVS, although at the beginning of each time step the gas distribution in the KFVS scheme is a quantum equilibrium distribution function inside each cell, the real gas distribution function which is used to evaluate the numerical fluxes across the cell interfaces is not a full equilibrium distribution at all—it is composed of two half Bose–Einstein (or Fermi–Dirac) distributions in $u \geq 0$ and $u < 0$ regions separately, as can be seen from (3.7) and (3.8). This nonequilibrium distribution does not correspond to the quantum Euler solutions at all. In order to achieve the Euler solution, physically, the gas molecules encounter many collisions (with an infinitely short collision time) during a CFL time step Δt. The corresponding classical case has been analyzed by Xu (1998, 2001). In the next section, following Xu we shall include particle collisions in the gas evolution stage using a relaxation time model.

4. Kinetic flux vector splitting based on the BGK scheme. The KFVS scheme is based on the convectional Boltzmann equation in the gas evolution stage. Due to free transport dynamics in this stage, the KFVS scheme cannot properly capture the contact discontinuity wave and slip lines. The artificial collisions with collision time equal to the time step $\triangle t$ help the KFVS scheme to capture these waves. In order to include real particle collisions in the gas evolution model to reduce overdiffusivity in the KFVS scheme, following Xu (1998) we choose to use a relaxation model to approximate particle collisions. The classical BGK model (1954) will be used in the gas evolution stage to construct the numerical fluxes across the cell interfaces in each cell. We shall denote this BGK-type KFVS scheme as the BGK–KFVS scheme to distinguish it from the usual KFVS scheme. For one-dimensional problems, the first order BGK–KFVS scheme considers the BGK equation in the gas evolution stage,

$$\frac{\partial f(x,u,t)}{\partial t} + u\frac{\partial f(x,u,t)}{\partial x} = \frac{f_0 - f(x,u,t)}{\tau}, \tag{4.1}$$

where f_0 is the equilibrium distribution function and τ is the collision time. In a classical gas τ is related to gas properties by $\tau = \nu/P$, where ν is the dynamical viscosity and P is the gas pressure. In this work, the relaxation time τ is used purely for numerical purposes and does not have much specific physical meaning. A more rigorous theoretical derivation of the physical relaxation time for the quantum Boltzmann equation is needed, such as the Wigner–BGK equation by Degond and Ringhofer (2003). Other problems can be treated by solving the quantum Boltzmann

equation under the relaxation time approach including the electron and phonon transport in heterojunction structures and ballistic-diffusive phonon transport; see Chen (2005). We shall pursue this relaxation time approach for nanoscale energy transport in separate works.

Under the same initial condition, i.e., (3.6), around the cell interface $x = x_{j+\frac{1}{2}}$, the solution $f(x, u, t)$ based on the BGK equation at $x = x_{j+\frac{1}{2}}$ and time t now becomes

$$\begin{aligned} f(x,u,t) &= (1-e^{-t/\tau})f_0 + e^{-t/\tau} f_1(x_{j+\frac{1}{2}} - ut, u) \\ &= (1-e^{-t/\tau})f_0 + e^{-t/\tau}(g_j H(u) + g_{j+1}(1-H(u))), \end{aligned} \tag{4.2}$$

where $H(u)$ is the Heaviside function and f_1 is the same as that given in (3.7). The equilibrium state f_0 is constructed by applying the compatibility condition along the line ($x = x_{j+1/2}$, t)

$$\int_{-\infty}^{+\infty} \psi_\alpha f_0 du = \int_{u\geq 0} \psi_\alpha g_j du + \int_{u<0} \psi_\alpha g_{j+1} du. \tag{4.3}$$

The underlying physical assumption in the above equation is that the left and right moving particles collapse at the cell interface to form an equilibrium distribution function f_0.

Solution (4.2) is different from the solution that is based on the convectional Boltzmann equation without considering any effects due to collision. In the limit of $\tau \to 0$, (4.2) goes to $f = f_0$, which is an exact equilibrium distribution function at the cell interface. For $\tau \to \infty$, f is equal to $f(x, u, t = 0) = f_1$, which recovers the distribution function in the KFVS scheme (i.e., (3.7)). Thus, (4.2) is the bridge between these two kinds of distribution functions in the cell interface. In the first order BGK–KFVS scheme, $e^{-t/\tau}$ can be assumed to be a constant.

The distribution function at a cell interface for the first order BGK–KFVS scheme is based on the combination of two functions, which are the nonequilibrium initial gas distribution function $f(x, u, t = 0) = f_1$ and the equilibrium distribution function f_0. With the definition $\eta \equiv e^{-t/\tau}$, the final distribution function at $x = x_{j+\frac{1}{2}}$ is

$$f(x_{j+\frac{1}{2}}, u, t) = (1-\eta)f_0 + \eta f_1(x_{j+\frac{1}{2}} - ut, u). \tag{4.4}$$

We note that in this work we assume $0 < \eta < 1$ and that it is taken as a constant $\eta = 0.5$ in the following numerical calculations. The main role of the relaxation time τ is to provide a bridge between the KFVS scheme (based on f_1) and an equilibrium scheme (based on f_0). The scheme with both f_0 and f_1 is called a partially thermalized transport. For further discussion, see Xu (1998).

The procedures for the first order BGK–KFVS scheme are summarized as follows:

1. From the given macroscopic quantities $\mathbf{W}_j = (\rho_j, \rho_j U_j, \rho_j \epsilon_j)^T$ in cell j, we obtain z_j and T_j from (3.2) and (3.3), respectively. We then construct the quantum equilibrium distribution function g_j and set the nonequilibrium initial gas distribution function $f(x, u, t = 0) = f_1$ as given by (3.6).

2. Compute the numerical fluxes at each cell interface $F^1_{j+\frac{1}{2}}$ from the nonequilibrium distribution function. The fluxes are the same as in the previous KFVS scheme defined by (3.8).

3. Obtain the macroscopic quantities $\bar{\mathbf{W}} = (\bar{\rho}, \bar{\rho}\bar{U}, \bar{\rho}\bar{\epsilon})^T$ at the cell interface $x = x_{j+\frac{1}{2}}$ from the compatibility condition (4.3):

$$\bar{\rho}_{j+\frac{1}{2}} = \frac{\sqrt{\pi}}{2}\{\mathbf{K}_j^{\frac{1}{2}} \mathbf{A}_j^+ + \mathbf{K}_{j+1}^{\frac{1}{2}} \mathbf{A}_{j+1}^-\}, \tag{4.5}$$

$$\bar{\rho}_{j+\frac{1}{2}}\bar{U}_{j+\frac{1}{2}} = \frac{\sqrt{\pi}}{2}\{\mathbf{K}_j^{\frac{1}{2}}\mathbf{A}_j^+ + \mathbf{K}_{j+1}^{\frac{1}{2}}U_{j+1}\mathbf{A}_{j+1}^-\} + \frac{1}{2}\{\mathbf{K}_j\mathbf{B}_j^+ + \mathbf{K}_{j+1}\mathbf{B}_{j+1}^-\},$$

$$\bar{\rho}_{j+\frac{1}{2}}\bar{\epsilon}_{j+\frac{1}{2}} = \frac{\sqrt{\pi}}{4}\{\mathbf{K}_j^{\frac{1}{2}}U_j^2\mathbf{A}_j^+ + \mathbf{K}_{j+1}^{\frac{1}{2}}U_{j+1}^2\mathbf{A}_{j+1}^-\} + \frac{1}{2}\{\mathbf{K}_jU_j\mathbf{B}_j^+ + \mathbf{K}_{j+1}U_{j+1}\mathbf{B}_{j+1}^-\} + \frac{\sqrt{\pi}}{8}\{\mathbf{K}_j^{\frac{3}{2}}\mathbf{C}_j^+ + \mathbf{K}_{j+1}^{\frac{3}{2}}\mathbf{C}_{j+1}^-\},$$

where all the parameters $\mathbf{K}$, $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are defined in (3.12).

4. From the macroscopic quantities at the cell interface, $\bar{\mathbf{W}}_{j+\frac{1}{2}} = (\bar{\rho}, \bar{\rho}\bar{U}, \bar{\rho}\bar{\epsilon})^T$, we can get the value of the fugacity $\bar{z}_{j+\frac{1}{2}}$ and temperature $\bar{T}_{j+\frac{1}{2}}$. Then we can construct the equilibrium distribution function $f_{0,j+1/2}$ from (3.4),

$$f_{0,j+1/2} = \left(\frac{m^2}{h}\right)\left\{\bar{z}_{j+\frac{1}{2}}^{-1}\exp\left(\frac{m[u-\bar{U}_{j+\frac{1}{2}}]^2}{2k_B\bar{T}_{j+\frac{1}{2}}}\right) + \theta\right\}^{-1}. \tag{4.6}$$

5. Compute the numerical fluxes $F_{j+1/2}^0$ from the equilibrium distribution function $f_{0,j+1/2}$,

$$F_{j+\frac{1}{2}}^0 = \begin{pmatrix} \sqrt{\pi}\bar{\mathbf{K}}_{j+\frac{1}{2}}^{\frac{1}{2}}\bar{U}_{j+\frac{1}{2}}g_{\frac{1}{2}}(\bar{z}_{j+\frac{1}{2}}) \\ \sqrt{\pi}\bar{\mathbf{K}}_{j+\frac{1}{2}}\bar{U}_{j+\frac{1}{2}}^2g_{\frac{1}{2}}(\bar{z}_{j+\frac{1}{2}}) + \frac{\sqrt{\pi}}{2}\bar{\mathbf{K}}_{j+\frac{1}{2}}^{\frac{3}{2}}g_{\frac{3}{2}}(\bar{z}_{j+\frac{1}{2}}) \\ \frac{\sqrt{\pi}}{2}\bar{\mathbf{K}}_{j+\frac{1}{2}}^{\frac{1}{2}}\bar{U}_{j+\frac{1}{2}}^3g_{\frac{1}{2}}(\bar{z}_{j+\frac{1}{2}}) + \frac{3\sqrt{\pi}}{4}\bar{\mathbf{K}}_{j+\frac{1}{2}}^{\frac{3}{2}}\bar{U}_{j+\frac{1}{2}}g_{\frac{3}{2}}(\bar{z}_{j+\frac{1}{2}}) \end{pmatrix}, \tag{4.7}$$

where $\bar{\mathbf{K}}_{j+\frac{1}{2}} = 2k_B\bar{T}_{j+\frac{1}{2}}/m$.

6. The final flux vector across the cell interface is

$$F_{j+\frac{1}{2}}^2 = (1-\eta)F_{j+\frac{1}{2}}^0 + \eta F_{j+\frac{1}{2}}^1, \tag{4.8}$$

where η is a local constant with $\eta \in [0,1]$ and we use $\eta = 0.5$ in the first order BGK–KFVS scheme.

7. We finally get the first order BGK–KFVS scheme

$$\mathbf{W}_j^{n+1} = \mathbf{W}_j^n - \sigma(F_{j+\frac{1}{2}}^2 - F_{j-\frac{1}{2}}^2). \tag{4.9}$$

Now, we have completed the first order BGK–KFVS scheme for one-dimensional ideal quantum gas dynamics.

5. Implementation of WENO schemes. For the purpose of implementing high-order methods, we can further express the basic quantum KFVS scheme (without external potential) defined by (3.17) in terms of numerical flux as follows:

$$\mathbf{W}_j^{n+1} = \mathbf{W}_j^n - \sigma(F_{j+1/2}^1 - F_{j-1/2}^1), \tag{5.1}$$

where the numerical flux $F_{j+1/2}^1$ as is given in (3.8).

The above scheme is of first order accuracy and thus is very diffusive. In practical applications we need high-order methods. In this section, we apply the WENO interpolation method (Liu, Osher, and Chan 1994; Jiang and Shu 1996) to the basic first order quantum KFVS scheme to result in a class of high resolution methods for the

computation of quantum ideal gas dynamical flows. The numerical flux is expressed as

$$F^3_{j+1/2} = \widehat{F}^+_{j+1/2} + \widehat{F}^-_{j+1/2}. \tag{5.2}$$

Here we consider a fifth-order accurate ($r = 3$) WENO scheme for the spatial difference of numerical fluxes. The WENO scheme for $r = 3$, denoted as WENO3, can be expressed as

$$\widehat{F}^+_{j+1/2} = \omega^+_0\left(\frac{2}{6}F^+_{j-2} - \frac{7}{6}F^+_{j-1} + \frac{11}{6}F^+_j\right) + \omega^+_1\left(-\frac{1}{6}F^+_{j-1} + \frac{5}{6}F^+_j + \frac{2}{6}F^+_{j+1}\right) + \omega^+_2\left(\frac{2}{6}F^+_j + \frac{5}{6}F^+_{j+1} - \frac{1}{6}F^+_{j+2}\right), \tag{5.3}$$

$$\omega^+_k = \frac{\alpha^+_k}{\alpha^+_0 + \alpha^+_1 + \alpha^+_2}, \qquad k = 0, 1, 2, \tag{5.4}$$

$$\alpha^+_0 = \frac{1}{10(\epsilon + IS^+_0)^2}, \alpha^+_1 = \frac{6}{10(\epsilon + IS^+_1)^2}, \alpha^+_2 = \frac{3}{10(\epsilon + IS^+_2)^2}, \tag{5.5}$$

$$\begin{aligned} IS^+_0 &= \frac{13}{12}(F^+_{j-2} - 2F^+_{j-1} + F^+_j)^2 + \frac{1}{4}(F^+_{j-2} - 4F^+_{j-1} + 3F^+_j)^2, \\ IS^+_1 &= \frac{13}{12}(F^+_{j-1} - 2F^+_j + F^+_{j+1})^2 + \frac{1}{4}(F^+_{j-1} - F^+_{j+1})^2, \\ IS^+_2 &= \frac{13}{12}(F^+_j - 2F^+_{j+1} + F^+_j)^2 + \frac{1}{4}(3F^+_j - 4F^+_{j+1} + F^+_{j+2})^2. \end{aligned} \tag{5.6}$$

The numerical flux for the negative part is given by

$$\widehat{F}^-_{j+1/2} = \omega^-_0\left(-\frac{1}{6}F^-_{j-1} + \frac{5}{6}F^-_j + \frac{2}{6}F^-_{j+1}\right) + \omega^-_1\left(\frac{2}{6}F^-_j + \frac{5}{6}F^-_{j+1} - \frac{1}{6}F^-_{j+2}\right) + \omega^-_2\left(\frac{11}{6}F^-_{j+1} - \frac{7}{6}F^-_{j+2} + \frac{2}{6}F^-_{j+3}\right), \tag{5.7}$$

$$\omega^-_k = \frac{\alpha^-_k}{\alpha^-_0 + \alpha^-_1 + \alpha^-_2}, \qquad k = 0, 1, 2, \tag{5.8}$$

$$\alpha^-_0 = \frac{3}{10(\epsilon + IS^-_0)^2}, \alpha^-_1 = \frac{6}{10(\epsilon + IS^-_1)^2}, \alpha^-_2 = \frac{1}{10(\epsilon + IS^-_2)^2}, \tag{5.9}$$

$$\begin{aligned} IS^-_0 &= \frac{13}{12}(F^-_{j-1} - 2F^-_j + F^-_{j+1})^2 + \frac{1}{4}(F^-_{j-1} - 4F^-_j + 3F^-_{j+1})^2, \\ IS^-_1 &= \frac{13}{12}(F^-_j - 2F^-_{j+1} + F^-_{j+2})^2 + \frac{1}{4}(F^-_j - F^-_{j+2})^2, \\ IS^-_2 &= \frac{13}{12}(F^-_{j+1} - 2F^-_{j+2} + F^-_{j+3})^2 + \frac{1}{4}(3F^-_{j+1} - 4F^-_{j+2} + F^-_{j+3})^2. \end{aligned} \tag{5.10}$$

The details of the $F^\pm_j$ are listed in (3.11).

In the following, we shall denote by KFVS–WENO3 the KFVS scheme with WENO3 implementation. The implementation of the high-order method for the BGK–KFVS scheme will be different from that for KFVS and will be investigated separately; see Xu (1998).

6. Numerical experiments. In this section, we report some numerical experiments to illustrate the present quantum KFVS and BGK–KFVS schemes. We consider ideal quantum gas flows in a shock tube. Both Bose–Einstein and Fermi–Dirac gases are considered. In this problem a diaphragm is located at $x = 0.5$ and separates two regions, each of which remains in a constant equilibrium state at initial time $t = 0$. The macroscopic properties on both sides of the diaphragm are different. Then the diaphragm is removed and flow structures of moving shock wave, contact line, and expansion fan are generated, which are similar to those in the classical gas dynamics. We consider two cases for each quantum gas. The first and second cases are set up for boson gas in such a way that one corresponds to the classical limit and the other is set up with the condition that z is very close to unity, which corresponds to the nearly degenerate limit. The third and fourth cases are set up for fermion gas. One corresponds to the classical limit and the other is set up for large z, which is close to the nearly degenerate limit. Finally, we consider a one-dimensional ideal boson gas in a harmonic potential. The harmonic potential is important because of its relevance to the recent experiments on Bose–Einstein condensation and mathematical simplicity (see Ketterle and van Druten 1996 and Pethick and Smith 2002).

' . ,' 1 (boson gas in the near classical limit). The initial condition is specified with $\mathbf{W}_L = (\rho_L, \rho_L U_L, \rho_L \epsilon_L) = (2.337556, 0, 0.629285)$ for $0 < x \leq 0.5$ and $\mathbf{W}_R = (\rho_R, \rho_R U_R, \rho_R \epsilon_R) = (1.249672, 0, 0.248194)$ for $0.5 < x < 1$. This corresponds to $z_L = 0.8$ and $z_R = 0.7$, and we assume that $\mu_L = \mu_R$. The solution behaviors are similar to those in the classical limit. The results are output at time $t = 0.3$ (dimensionless time), and the CFL number used is 1. The computed density, temperature, fugacity, and chemical potential profiles are depicted in Figure 1 for several grid systems. The symbol $\square$ denotes the solution using 200 cells, the symbol $\triangle$ denotes the solution using 400 cells, and the symbol $\circ$ denotes the solution using 800 cells. The flow structures of moving shock, contact line, and expansion fan can be identified and captured by the present first order KFVS scheme. It can be seen from the figures that when the grid becomes finer, the profiles become crisper, which indicates the grid convergence of the solution. In Figure 2, we compare the results using first order KFVS, BGK–KFVS, and KFVS–WENO3 schemes with 200 cells. The symbol $\square$ denotes the solution using the KFVS scheme (CFL number is 1), the symbol $\triangle$ denotes the solution using the BGK–KFVS scheme (CFL number is 1), the symbol $\circ$ denotes the solution using the KFVS–WENO3 scheme (CFL number is 0.2), and the solid line denotes the Riemann solution. As can be seen, the BGK–KFVS solution shows slightly better resolution than that of KFVS, and the effect of the high-order method is clearly depicted. A plot of Bose–Einstein distribution functions at several stations at time $t = 0.3$ is also shown in Figure 2(d). Here, 20 lines are plotted for $0 \leq x \leq 1$ with equal spacing $\Delta x = 0.05$, and the range of u is $-2 \leq u \leq 2$. The profiles at most stations are close to Gaussian distribution in this near classical limit.

' . ,' 2 (boson gas in the nearly degenerate limit). The initial condition in this case is set with $\mathbf{W}_L = (\rho_L, \rho_L U_L, \rho_L \epsilon_L) = (16.221831, 0, 1.135830)$ for $0 < x \leq 0.5$ and $\mathbf{W}_R = (\rho_R, \rho_R U_R, \rho_R \epsilon_R) = (7.768344, 0, 0.375085)$ for $0.5 < x < 1$. We assume $\mu_L = \mu_R$ and set the values of fugacity as $z_L = 0.99$ and $z_R = 0.98$, which are quite close to unity. Although in one spatial dimension Bose–Einstein condensation does not occur because the excited states can accommodate an essentially infinite particle number, due to that, boson functions may diverge at $z = 1$. This example was selected to simulate the situation close to the degenerate limit.

The results are output at time $t = 0.3$ (dimensionless time), and the CFL number used is 1. The computed density, temperature, fugacity, and chemical potential

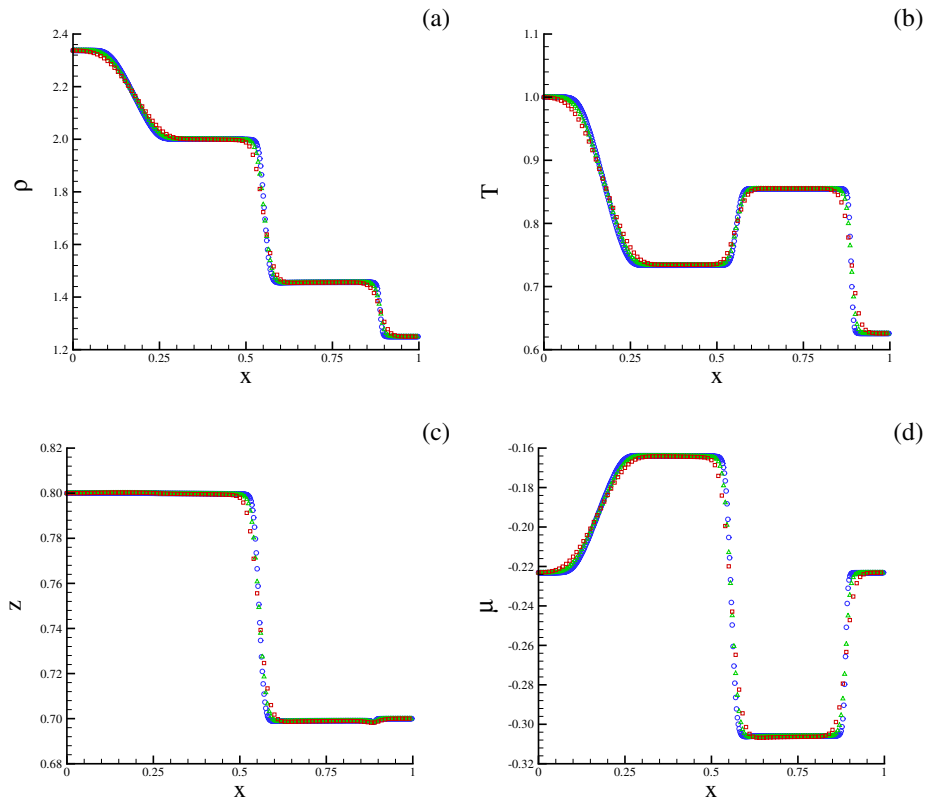


FIG. 1. *Quantum shock tube solution (Example 1).* (a) *density;* (b) *temperature;* (c) *fugacity;* (d) *chemical potential.*

profiles are depicted in Figure 3 for several grid systems. The symbol □ denotes the solution using 200 cells, the symbol △ denotes the solution using 400 cells, and the symbol ○ denotes the solution using 800 cells. The flow structures of moving shock, contact line, and expansion fan can be identified and captured by the present first order KFVS scheme. It can be seen from the figures that when the grid becomes finer, the profiles become crisper, which indicates the grid convergence of the solution. In Figure 4, we compare the results that use first order KFVS, BGK–KFVS, and KFVS–WENO3 schemes with 200 cells. The symbol □ denotes the solution using the first order KFVS scheme (CFL number is 1), the symbol △ denotes the solution using the first order BGK–KFVS scheme (CFL number is 1), the symbol ○ denotes the solution using the KFVS–WENO3 scheme (CFL number is 0.2), and the solid line denotes the Riemann solution. Again, the BGK–KFVS solution indicates slightly better resolution than that of KFVS, and the effect of the high-order method can be clearly observed. In Figure 4(d), a plot of Bose–Einstein distribution functions at several stations is shown. Here, 20 lines are plotted for $0 \le x \le 1$ with equal spacing $\Delta x = 0.05$, and the range of u is $-1 \le u \le 1$. It is noted that the peak values of this

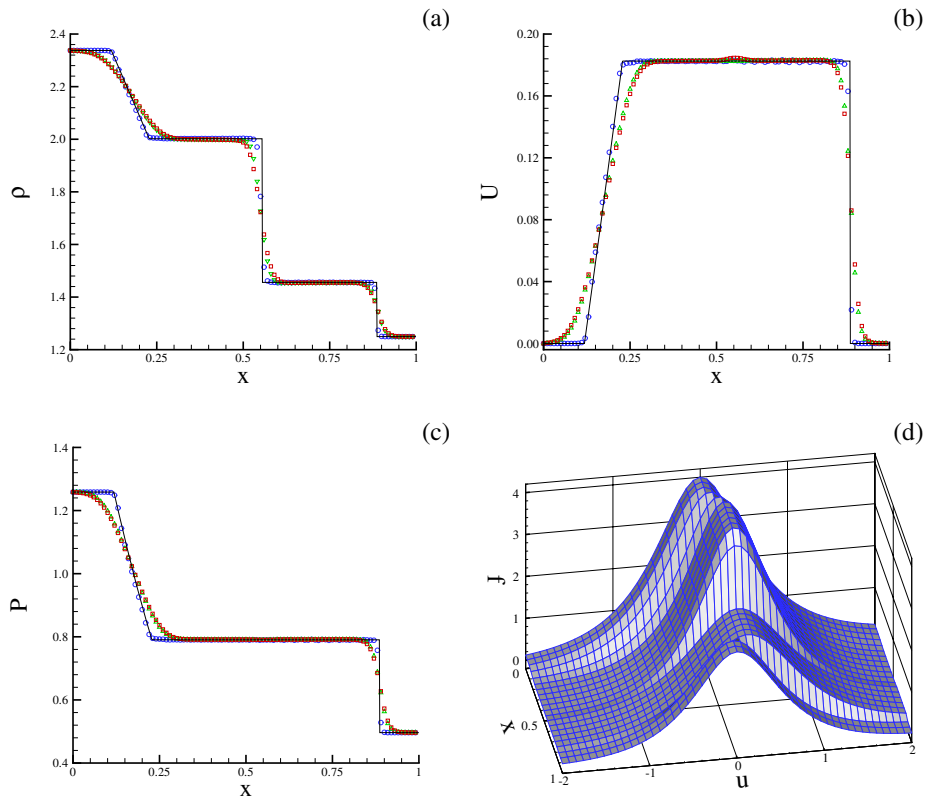


FIG. 2. *Quantum shock tube solution (Example 1).* (a) *density;* (b) *velocity;* (c) *pressure;* (d) *distribution function.*

case are much higher than that of Example 1, as shown in Figure 2(d), and the shape of the distribution function becomes more narrow.

3 (fermion gas in the near classical limit). The initial condition is specified with $\mathbf{W}_L = (\rho_L, \rho_L U_L, \rho_L \epsilon_L) = (0.523188, 0, 0.319701)$ for $0 < x \leq 0.5$ and $\mathbf{W}_R = (\rho_R, \rho_R U_R, \rho_R \epsilon_R) = (0.377659, 0, 0.141679)$ for $0.5 < x < 1$. This corresponds to $z_L = 0.8$ and $z_R = 0.7$, and we assume that $\mu_L = \mu_R$ and $m = 1$. The solution behaviors are similar to those in the classical limit. The results are output at time $t = 0.15$ (dimensionless time), and the CFL number used is 1. The computed density, temperature, fugacity, and chemical potential profiles are depicted in Figure 5 for several grid systems. The symbol □ denotes the solution using 200 cells, the symbol △ denotes the solution using 400 cells, and the symbol ∘ denotes the solution using 800 cells. The flow structures of moving shock, contact line, and expansion fan can be identified and captured by the present first order KFVS scheme. It can be seen from the figures that, again, the grid convergence of the solution is ensured. In Figure 6, we compare the results using the first order KFVS, BGK–KFVS, and KFVS–WENO3

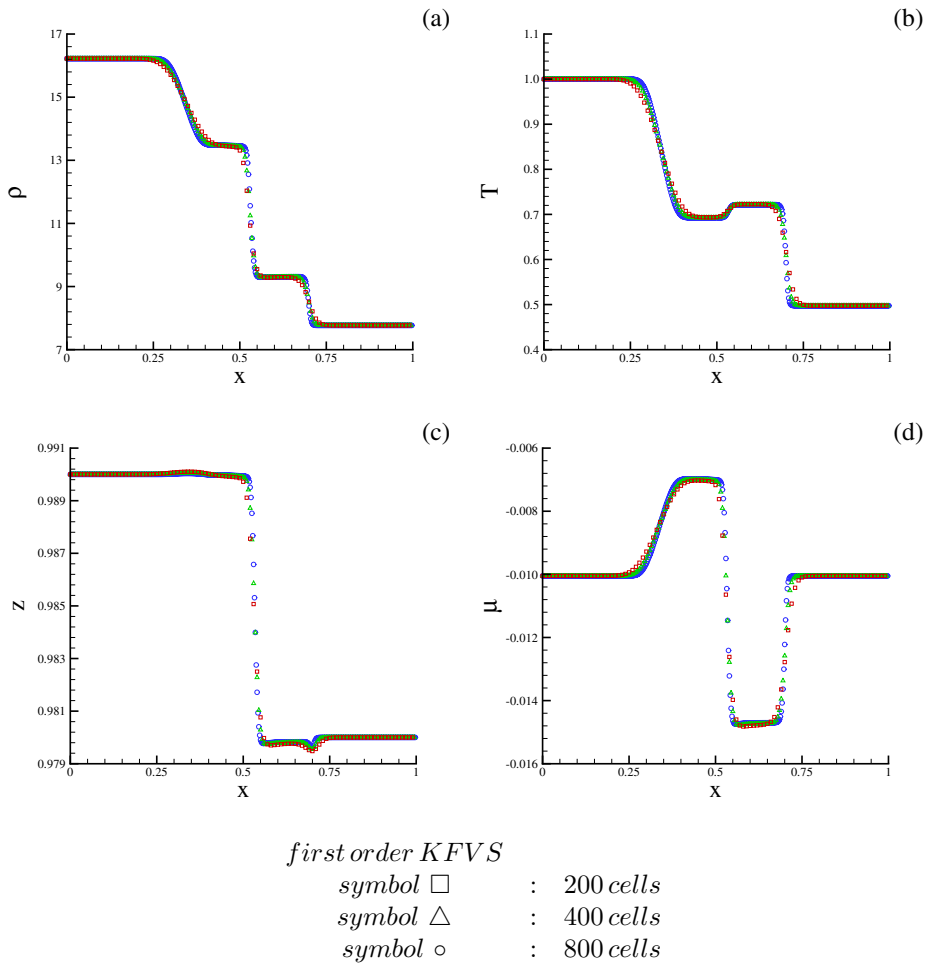


FIG. 3. *Quantum shock tube solution (Example 2).* (a) *density;* (b) *temperature;* (c) *fugacity;* (d) *chemical potential.*

schemes with 200 cells. The symbol □ denotes the solution using the first order KFVS scheme (CFL number is 1), the symbol △ denotes the solution using the first order BGK–KFVS scheme (CFL number is 1), the symbol ○ denotes the solution using the KFVS–WENO3 scheme (CFL number is 0.2), and the solid line denotes the Riemann solution. Again, the BGK–KFVS solution indicates moderately better resolution than that of the KFVS solution, and the effect of the high-order method can be clearly observed. In Figure 6(d), a plot of Fermi–Dirac distribution functions at several stations is shown. Here, 20 lines are plotted for $0 \le x \le 1$ with equal spacing $\Delta x = 0.05$, and the range of u is $-5 \le u \le 5$. The profiles at most stations are close to Gaussian distribution in this near classical limit.

' . ,' 4 (fermion gas in the nearly degenerate limit). The initial condition in this case is set with $\mathbf{W}_L = (\rho_L, \rho_L U_L, \rho_L \epsilon_L) = (3.086643, 0, 8.053134)$ for $0 < x \le 0.5$ and $\mathbf{W}_R = (\rho_R, \rho_R U_R, \rho_R \epsilon_R) = (3.084468, 0, 8.067159)$ for $0.5 < x < 1$. We assume $\mu_L = \mu_R$ and set the values of fugacity as $z_L = 2000$ and $z_R = 1500$, using Sommerfeld expansion to get Fermi function.

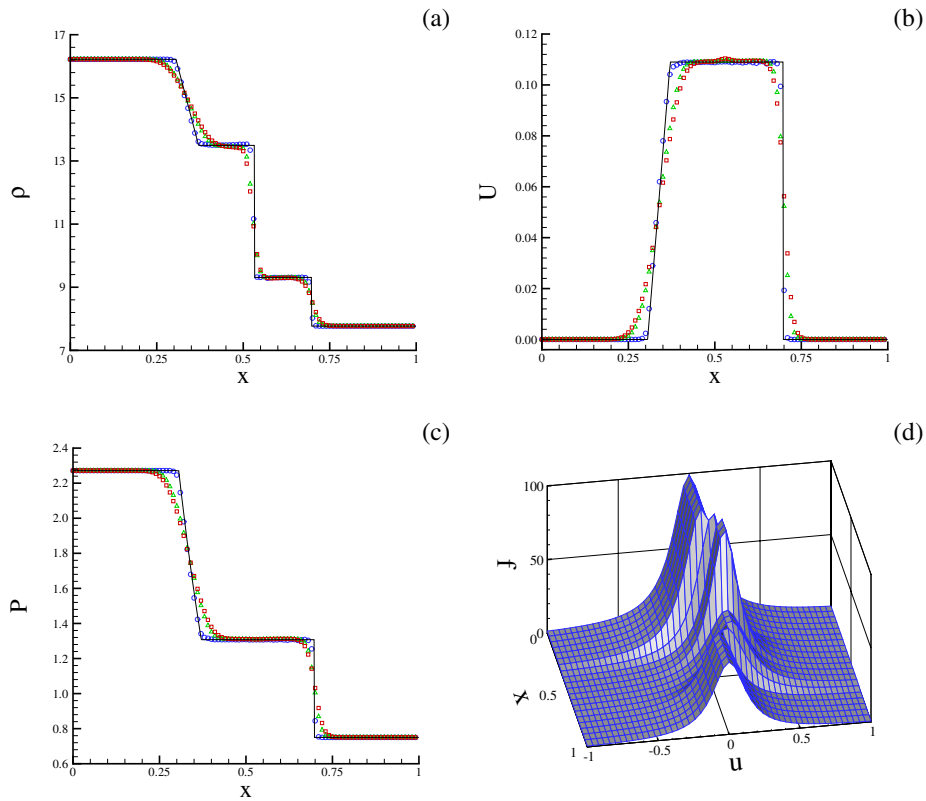


FIG. 4. *Quantum shock tube solution (Example 2).* (a) *density;* (b) *velocity;* (c) *pressure;* (d) *distribution function.*

The results are output at time $t = 0.1$ (dimensionless time), and the CFL number used is 1. The computed density, temperature, fugacity, and chemical potential profiles are depicted in Figure 7 for several grid systems. The symbol □ denotes the solution using 200 cells, the symbol △ denotes the solution using 400 cells, and the symbol ∘ denotes the solution using 800 cells. The flow structures of moving shock, contact line, and expansion fan can be identified and captured by the present first order KFVS scheme. It can be seen from the figures that, again, the grid convergence of the solutions is ensured. In Figure 8, we compare the results using the first order KFVS, BGK–KFVS, and KFVS–WENO3 schemes with 200 cells. The symbol □ denotes the solution using the KFVS scheme (CFL number is 1), the symbol △ denotes the solution using the BGK–KFVS scheme (CFL number is 1), the symbol ∘ denotes the solution using the KFVS–WENO3 scheme (CFL number is 0.2), and the solid line denotes the Riemann solution. In Figure 8(d), a plot of Fermi–Dirac distribution functions at several stations is shown. Here, 20 lines are plotted for $0 \le x \le 1$ with equal spacing $\Delta x = 0.05$, and the range of u is $-8 \le u \le 8$. The approach to the

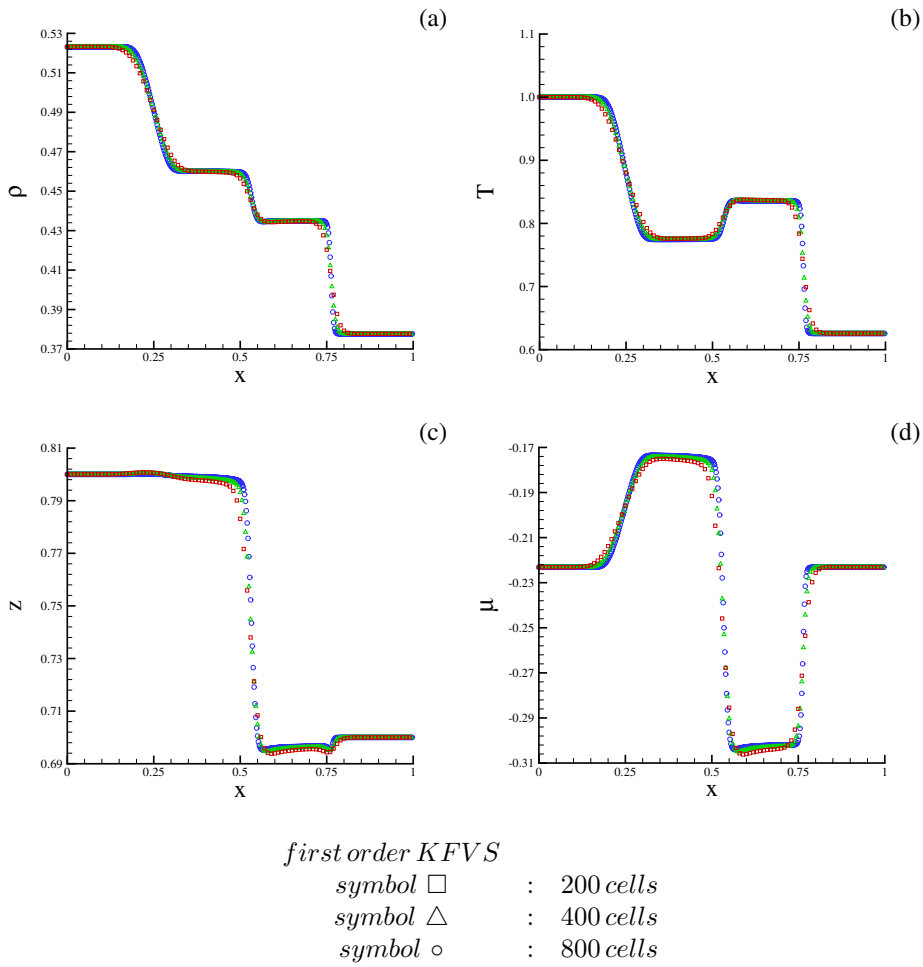


FIG. 5. *Quantum shock tube solution (Example 3).* (a) *density;* (b) *temperature;* (c) *fugacity;* (d) *chemical potential.*

typical double step function shape is well presented in this large fugacity limit.

' . ,' 5 (ideal Bose gas under an external potential). The initial condition for a Bose gas is set as $\mathbf{W} = (\rho, \rho U, \rho\epsilon) = (1.579938, 0, 0.501561)$ within $0 \le x \le L$. The corresponding fugacity and temperature in this case are $z = 0.7$ and $T = 1$. The gas is under the influence of an external potential $V(x)$. In center region $(L/4 \le x \le 3L/4)$, $V(x) = \frac{1}{2}\omega^2(x - L/2)^2$. In boundary regions $(0 \le x < L/4)$ and $(3L/4 < x \le L)$, $V(x) = \frac{1}{2}\omega^2(L/4)^2$. We use $\omega = 1$ and $L = 1$ in this example. At the domain boundaries $(x = 0)$ and $(x = L)$, we apply two different boundary conditions: one is a wall boundary condition and the other is a periodic boundary condition. The wall boundary condition assigns the ghost cells with the same density and pressure but with the velocity direction inverted at the boundary cells. The periodic boundary condition assigns the ghost cells with the same density, momentum, and energy density as the boundary cells that are one period away from the ghost cells. We use first order KFVS with 257 cells within the domain, and the CFL number is 1. The results with these two kinds of boundary conditions are plotted in Figure 9 and Figure 10, respectively.

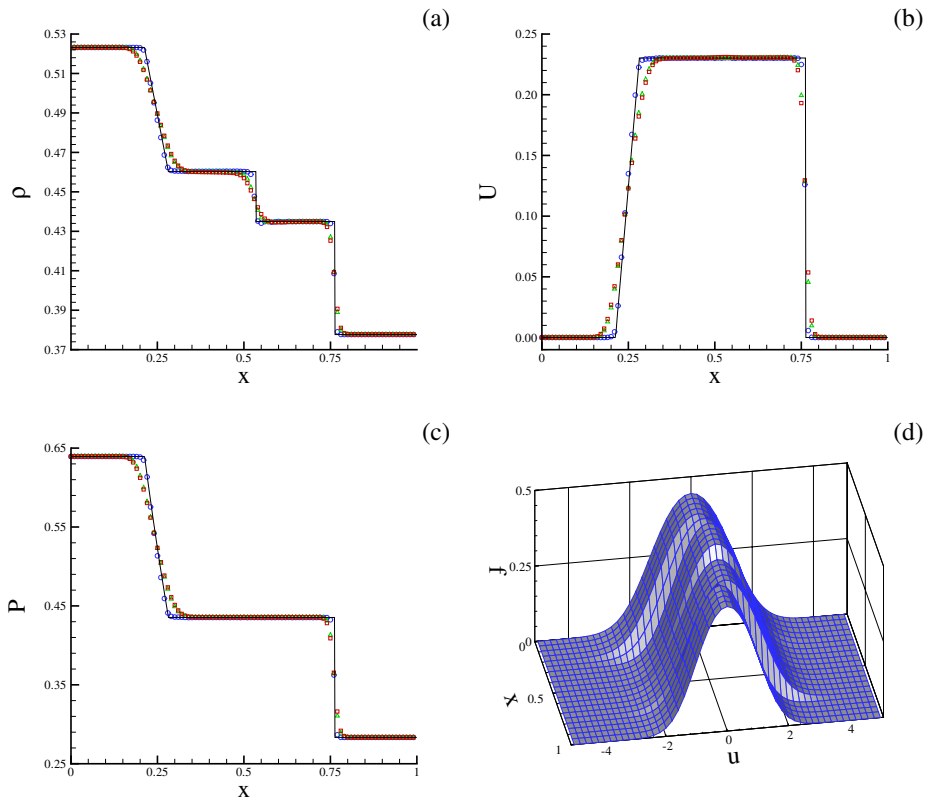


FIG. 6. *Quantum shock tube solution (Example 3).* (a) *density;* (b) *velocity;* (c) *pressure;* (d) *distribution function.*

In Figure 9, the results for the case with the wall boundary condition are exhibited for density, velocity, fugacity, and temperature at $t = 0, 400, 800, 1200$. We can assume that the Bose gas is confined in a box and influenced by a trap potential. We can observe that the density profiles are concentrated in the center region and decrease in the boundary regions (e.g., compare $t = 0$ and $t = 400$). Then, the density profiles decrease at the center region and increase at the boundary regions (t $= 400, 800, 1200$). The velocity profiles oscillate with larger amplitudes (not shown) initially due to the influence of the suddenly applied potential. Then, the velocity profiles are damped to an incline line within the center region. The fugacity rises in the center region and lowers at boundary regions; then, the profiles shift down with time. The temperature profiles rise in the center region and lower at boundary regions; then, the temperature profiles shift up with time, which implies the system is under constant energy input from the trap potential. Enlarged views are also plotted to indicate the final stages of calculations.

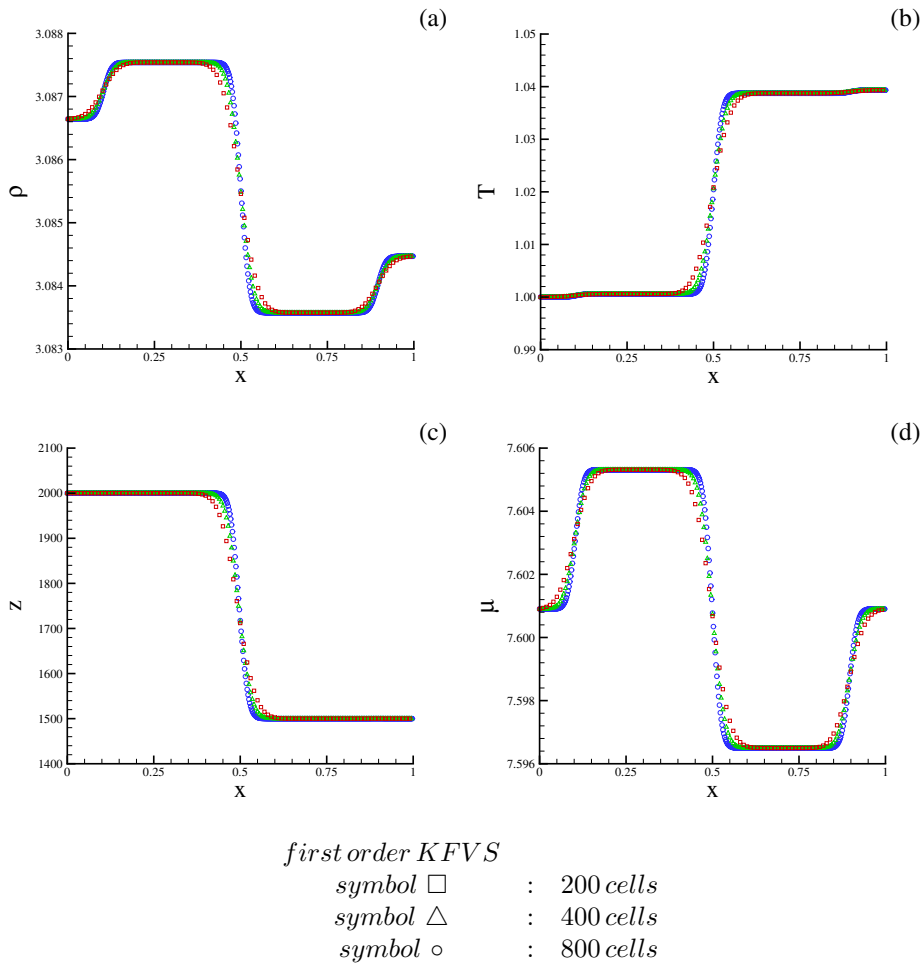
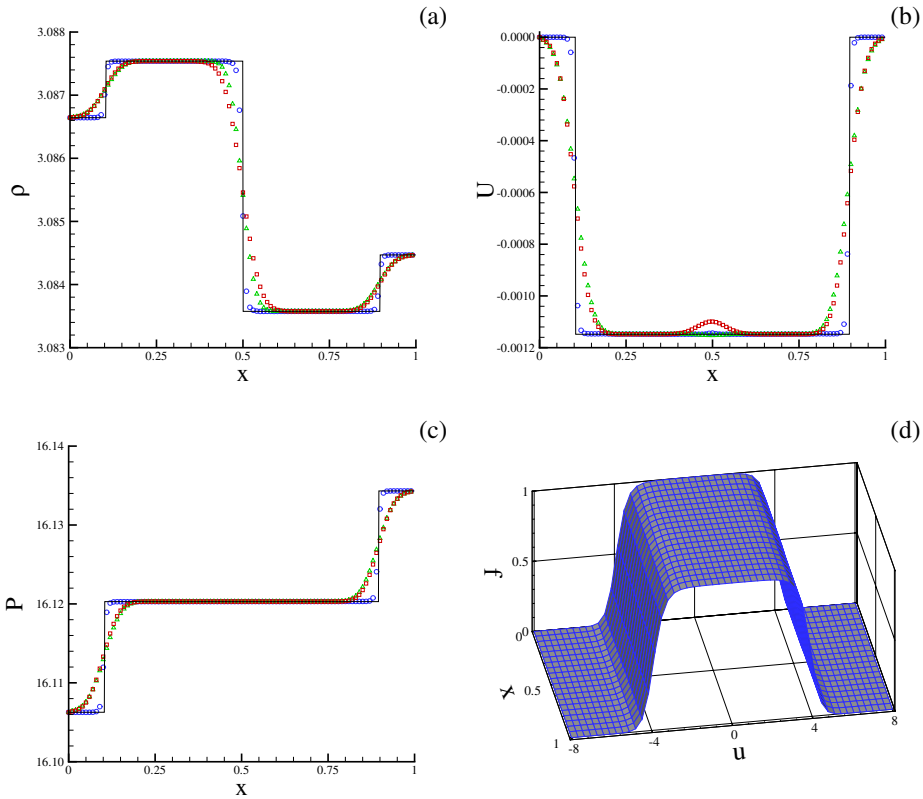


FIG. 7. *Quantum shock tube solution (Example 4).* (a) *density;* (b) *temperature;* (c) *fugacity;* (d) *chemical potential.*

In Figure 10, the results with the periodic boundary condition are exhibited. The density, velocity, fugacity, and temperature at $t = 0, 400, 800, 1200$ are plotted. We observe that the results with the periodic boundary condition are similar to those with wall boundary conditions. These imply that the velocity is always antisymmetric to the center point of the domain and other quantities are symmetric to the center point.

Although only one-dimensional schemes and results are presented here, the extension of the present numerical schemes to multiple dimensions can be done directly using Strang-type dimensional splitting (see Strang, 1968).

7. Concluding remarks. In this work, the basic first order kinetic flux vector splitting (KFVS) method for Newtonian gas dynamics developed by Patterson and by Deshpande have been successfully generalized to the ideal quantum gas dynamics based on the convectional Boltzmann equation for Bose–Einstein statistics and Fermi–Dirac statistics. To reduce the overdiffusivity of the KFVS scheme due to the inherent free transport mechanism, a modified KFVS scheme based on the BGK relaxation



symbol □ : *first order KFVS*
symbol △ : *first order BGK − KFVS*
symbol ○ : *KFVS − WENO3*
solid line : *Riemann solution*

FIG. 8. *Quantum shock tube solution (Example 4).* (a) *density;* (b) *velocity;* (c) *pressure;* (d) *distribution function.*

time collision model is also considered. The KFVS and BGK–KFVS schemes are first derived for one spatial dimension and then tested and illustrated by computing shock tube flows in ideal quantum gases. The regimes in the classical limit and nearly degenerate limit are tested. Numerical experiments indicate that flow structures such as shock wave, contact line, and expansion fan can be properly captured using the present quantum KFVS schemes. To include the effect of external potential, boson gases under a harmonic trap in the one-dimensional case are also illustrated. Higher-order methods using the WENO scheme are implemented to yield a class of accurate and efficient KFVS schemes for the computation of ideal quantum gas dynamical flows. The present method applies for Bose–Einstein gases and Fermi–Dirac gases at temperatures above the critical temperature, provided that the hydrodynamic limit is applicable. Extension of the present method to multiple space dimensions is of no conceptual difficulty and is currently under investigation. The direct numerical solution of the quantum Boltzmann equation using the relaxation time approach for nanoscale energy transport will be considered elsewhere.

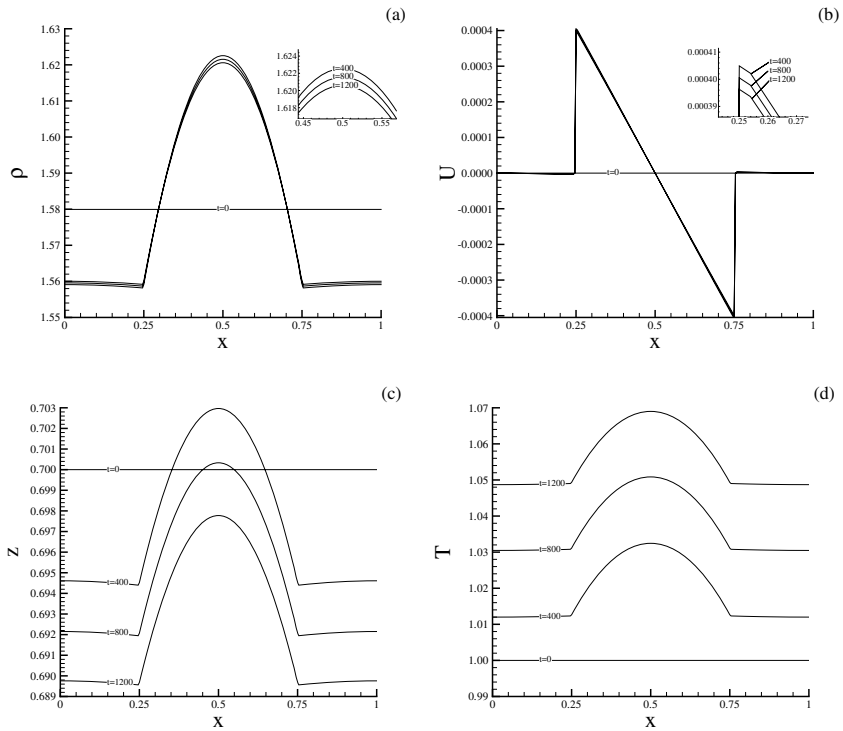


FIG. 9. *Bose gas under harmonic trap with solid wall boundary conditions (Example* 5*).* (a) *density;* (b) *velocity;* (c) *fugacity;* (d) *temperature.*

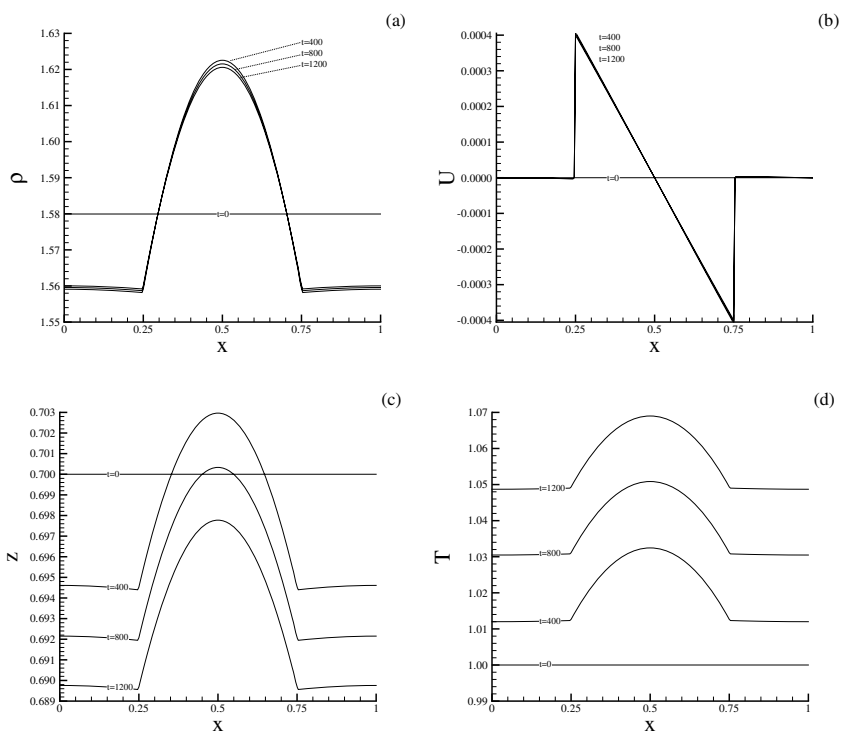


FIG. 10. *Bose gas under harmonic trap with periodic boundary conditions (Example* 5*).* (a) *density;* (b) *velocity;* (c) *fugacity;* (d) *temperature.*

Acknowledgment. The senior author wishes to thank Prof. Kun Xu of Hong Kong University of Science and Technology for many fruitful discussions.

REFERENCES

V. C. Aguilera-Navarro, M. de Llano, and M. A. Solis (1999), *Bose-Einstein condensation for general dispersion relations*, European J. Phys., 20, p. 177–182.

R. Benzi, S. Succi, and M. Vergassola (1992), *The lattice Boltzmann equation: Theory and applications*, Phys. Rep., 222, p. 145–197.

P. L. Bhatnagar, E. P. Gross, and M. Krook (1954), *A model for collision process in gases* I: *Small amplitude processes in charged and neutral one component systems*, Phys. Rev., 94, pp. 511–525.

S. Chapman and T. G. Cowling (1970), *The Mathematical Theory of Non-Uniform Gases*, Cambridge University Press, Cambridge, UK.

G. Chen (2005), *Nanoscale Energy Transport and Conversion*, Oxford University Press, Oxford, UK.

S. Y. Chou and D. Baganoff (1997), *Kinetic flux-vector splitting for the Navier-Stokes equations*, J. Comput. Phys., 130, pp. 217–230.

P. Degond and C. Ringhofer (2003), *Quantum moment hydrodynamics and the entropy principle*, J. Stat. Phys., 112, pp. 587–628.

S. R. de Groot, G. J. Hooyman, and C. A. Ten Seldam (1950), *On the Bose–Einstein condensation,* Proy. Roy. Soc. London Ser. A, 203, pp. 266–286.

S. M. Deshpande (1986), *A Second Order Accurate, Kinetic-Theory Based, Method for Inviscid Compressible Flows*, Technical Paper No. 2613, NASA Langley Research Center, Hampton, VA.

A. L. Garcia and W. Wagner (2003), *Direct simulation Monte Carlo method for the Uehling-Uhlenbeck-Boltzmann equation*, Phys. Rev. E, 68, 056703.

M. Grether, M. de Llano, and M. A. Solis (2003), *Anomalous behavior of ideal Fermi gas below two dimensions*, Eur. Phys. J. D. At. Mol. Opt. Phys., 25, pp. 287–291.

C. Hirsch (1988), *Numerical Computation of Internal and External Flows*, Wiley, New York.

K. Huang (1987), *Statistical Mechanics*, Wiley, New York.

G.-S. Jiang and C.-W. Shu (1996), *Efficient implementation of weighted ENO schemes*, J. Comput. Phys., 126, pp. 202–228.

L. P. Kadanoff and G. Baym (1962), *Quantum Statistical Mechanics*, Benjamin, New York.

W. Ketterle and N. J. van Druten (1996), *Bose-Einstein condensation of a finite number of particles trapped in one or three dimensions*, Phys. Rev. A, 54, pp. 656–660.

X.-D. Liu, S. Osher, and T. Chan (1994), *Weighted essentially non-oscillatory schemes*, J. Comput. Phys., 115, pp. 200–212.

J. C. Mandal and S. M. Deshpande (1994), *Kinetic flux vector splitting for Euler equations*, Comput. & Fluids, 23, p. 447–478.

A. Minguzzi, S. Succi, F. Toschi, M. P. Tosi, and P. Vignolo (2004), *Numerical methods for atomic quantum gases with applications to Bose-Einstein condensates and to ultracold fermions*, Phys. Rep., 395, pp. 223–355.

T. Ohwada (2002), *On the construction of kinetic schemes*, J. Comput. Phys., 177, pp. 156–175.

R. K. Pathria (1996), *Statistical Mechanics*, 2nd ed., Butterworth-Heinemann, Oxford, UK.

G. N. Patterson (1971), *Introduction to Kinetic Theory of Gas Flow*, University of Toronto Press, Toronto.

C. J. Pethick and H. Smith (2002), *Bose-Einstein Condensation in Dilute Gases*, Cambridge University Press, Cambridge, UK.

P. L. Pullin (1980), *Direct simulation methods for compressible inviscid ideal-gas flow*, J. Comput. Phys., 34, pp. 231–244.

R. D. Reitz (1981), *One-dimensional compressible gas dynamics calculations using the Boltzmann equation*, J. Comput. Phys., 42, pp. 108–123.

R. H. Sanders and K. H. Prendergast (1974), *The possible relation of the* 3*-kiloparsec arm to explosions in the galactic nucleus*, Astrophys. J., 188, pp. 489–500.

G. Strang (1968), *On the construction and comparison of difference schemes*, SIAM J. Numer. Anal., 5, pp. 506–517.

E. F. Toro (1999), *Riemann Solvers and Numerical Methods for Fluid Dynamics*. A Practical Introduction, 2nd ed., Springer-Verlag, Berlin.

E. A. Uehling and G. E. Uhlenbeck (1933), *Transport phenomena in Einstein-Bose and Fermi-Dirac gases.* I, Phys. Rev., 43, pp. 552–561.

K. Xu (1998), *Gas-kinetic schemes for unsteady compressible flow simulations*, in Proceedings of the 29th Computational Fluid Dynamics Lecture Series, von Karman Institute for Fluid Dynamics, Rhode-Saint-Geneses, Belgium.

K. Xu (2001), *A gas-kinetic BGK scheme for the Navier-Stokes equations and its connection with artificial dissipation and Godunov method*, J. Comput. Phys., 171, pp. 289–335.

K. Xu and W. Z. Li (2001), *Dissipative mechanism in Godunov-type schemes*, Internat. J. Numer. Methods Fluids, 37, pp. 1–22.

K. Xu and K. H. Prendergast (1994), *Numerical Navier-Stokes solutions from gas kinetic theory*, J. Comput. Phys., 114, pp. 9–17.

J. Y. Yang, M. H. Chen, I. N. Tsai, and J. W. Chang (1997), *A kinetic beam scheme for relativistic gas dynamics*, J. Comput. Phys., 136, pp. 19–40.

J.-Y. Yang and Y.-H. Shi (2006), *A kinetic beam scheme for ideal quantum gas dynamics*, R. Soc. Lond. Proc. Ser. A. Math. Phys. Eng. Sci., 462, pp. 1553–1572.

SIAM J. SCI. COMPUT.
Vol. 29, No. 1, pp. 245–264
© 2007 Society for Industrial and Applied Mathematics

NUMERICAL METHODS FOR SECOND-ORDER STOCHASTIC DIFFERENTIAL EQUATIONS*

KEVIN BURRAGE†, IAN LENANE†, AND GRANT LYTHE‡

Abstract. We seek numerical methods for second-order stochastic differential equations that reproduce the stationary density accurately for all values of damping. A complete analysis is possible for scalar linear second-order equations (damped harmonic oscillators with additive noise), where the statistics are Gaussian and can be calculated exactly in the continuous-time and discrete-time cases. A matrix equation is given for the stationary variances and correlation for methods using one Gaussian random variable per timestep. The only Runge–Kutta method with a nonsingular tableau matrix that gives the exact steady state density for all values of damping is the implicit midpoint rule. Numerical experiments, comparing the implicit midpoint rule with Heun and leapfrog methods on nonlinear equations with additive or multiplicative noise, produce behavior similar to the linear case.

Key words. damped harmonic oscillators with noise, stationary distribution, stochastic Runge–Kutta methods, implicit midpoint rule, multiplicative noise

AMS subject classifications. 60-08, 65C30

DOI. 10.1137/050646032

1. Introduction. Newton's second law of motion relates force to acceleration. Consequently, second-order differential equations are common in scientific applications, in the guise of "Langevin," "Monte Carlo," "molecular," or "dissipative particle" dynamics [1, 2, 3], and the study of methods for second-order ordinary differential equations is one of the most mature branches of numerical analysis [4]. The most exciting advances in recent decades have been the development of symplectic methods, capable of exactly preserving an energy-like quantity over very long times [5], and their extension to stochastic systems [6]. In the stochastic setting, the long-time dynamics of a typical physical system is governed by fluctuation-dissipation, so that the amount of time spent in different regions of phase space can be calculated from the stationary density [7]. This density can have a relatively simple explicit expression even when the dynamics is highly nonlinear [8]. Numerical methods replace continuous-time with discrete-time dynamics, generating values at times $t_0, t_1, \ldots$. Usually $t_{n+1} - t_n$ is a fixed number Δt. The criterion for a good numerical method that will be examined in this work is that its discrete-time dynamics has a stationary density as close as possible to that of the continuous-time system.

The differential equations describing second-order systems contain a parameter known as damping. The stationary density is independent of damping, but dynamical quantities, and the usefulness of numerical algorithms, are strongly dependent on it. In the infinite-damping limit, the system becomes first order. The limit of zero damping, on the other hand, corresponds to Hamiltonian systems, where symplectic methods can be applied. The aim in this paper is to devise methods capable of accurately reproducing the stationary density

*Received by the editors November 24, 2005; accepted for publication (in revised form) August 10, 2006; published electronically February 2, 2007.
http://www.siam.org/journals/sisc/29-1/64603.html
†Advanced Computational Modelling Centre, Department of Mathematics, University of Queensland, Queensland 4072, Australia (kb@maths.uq.edu.au, ijl@maths.uq.edu.au).
‡Department of Applied Mathematics, University of Leeds, Leeds LS2 9JT, England (grant@maths.leeds.ac.uk).

We shall consider equations of the following form:

$$\ddot{x} = f(x) - \eta s^2(x)\dot{x} + \epsilon s(x)\xi(t), \tag{1.1}$$

where $\langle \xi(t)\xi(t') \rangle = \delta(t - t')$ and the damping parameter is denoted η. Angled brackets denote mean over realizations. The second-order stochastic differential equation (SDE) (1.1) describes the position of a particle subject to deterministic forcing $f(x)$ and random forcing $\xi(t)$. The deterministic forcing is related to the potential function $V(x)$ via $f(x) = -V'(x)$. The amplitude of the random forcing, ϵ, is related to the temperature T and damping coefficient η by the fluctuation-dissipation relation [8] $\epsilon^2 = 2\eta KT$.

We can write (1.1) as a pair of first-order equations for $\mathbf{X}_t$ and $\mathbf{V}_t$, the position and velocity variables:

$$\begin{aligned} \mathrm{d}\mathbf{X}_t &= \mathbf{V}_t \mathrm{d}t, \\ \mathrm{d}\mathbf{V}_t &= -\eta s^2(\mathbf{X}_t)\mathbf{V}_t \mathrm{d}t + f(\mathbf{X}_t)\mathrm{d}t + \epsilon s(\mathbf{X}_t)\mathrm{d}\mathbf{W}_t, \end{aligned} \tag{1.2}$$

where $\mathbf{W}_t$ is a Wiener process satisfying $\langle \mathbf{W}_t \mathbf{W}_s \rangle = \min(t, s)$. If $s(x)$ is not a constant, the noise amplitude is a function of position and the equation is commonly said to have "multiplicative noise." However, because the coefficient of $\mathrm{d}\mathbf{W}_t$ in the SDE for $\mathbf{V}_t$ is a function of $\mathbf{X}_t$ only, there is no difference between the Ito and Stratonovich (see Gardiner [8]) forms of (1.2). The probability density at time t is $P(x, v; t)$, where

$$P(x, v; t) = \frac{\mathrm{d}}{\mathrm{d}x}\frac{\mathrm{d}}{\mathrm{d}v}\mathrm{Prob}\left(\mathbf{X}_t < x, \mathbf{V}_t < v\right). \tag{1.3}$$

The stationary density, $P_\infty(x, v)$, defined as

$$P_\infty(x, v) = \lim_{t\to\infty} P(x, v; t), \tag{1.4}$$

has the following analytical form, independent of η and $s(x)$ as long as $\eta > 0$ and suitable conditions on $V(x)$ are satisfied [8]:

$$P_\infty(x, v) = N \exp\left(-v^2/2KT - V(x)/KT\right). \tag{1.5}$$

Thus the late-time statistics of the velocity are Gaussian and uncorrelated with the position. It is notable that the stationary density has a closed tractable form for many nonlinear functions $f(x)$ when analytical study of the full evolution is not possible.

In this paper we examine how faithfully the stationary density is reproduced by standard timestepping methods for SDEs. These methods produce approximate values for position X_n and velocity V_n at discrete times $t_0, t_1, \ldots$, where $t_{n+1} - t_n = \Delta t$. We consider the evolution of X_n and V_n, and their statistical properties as $t_n \to \infty$, and compare it with the exact form (1.4). In particular, we consider $P^*(y, u; t_n)$, the discrete-time analogue of (1.3), and compare the limit

$$P^*_\infty(y, u) = \lim_{t_n\to\infty} P^*(y, u; t_n),$$

where it exists, with $P_\infty(y, u)$. In this work we base our analysis on linear second-order equations, where the statistics are Gaussian and completely characterized by

three quantities: the mean squares of the position and velocity variables, and the correlation between the position and velocity.

The Euler and Heun methods simply treat (1.2) as a pair of SDEs, without attempting to take advantage of their special structure due to their origin in a single second-order differential equation. We shall show that they perform reasonably well at intermediate values of damping but fail at high damping, when the equations are stiff, and low damping, when it is important to conserve energy-like quantities over long times.

The Verlet algorithm [1, 9, 10] produces numerical solutions of second-order differential equations by updating the position variable without reference to the velocity variable. Extensions of this idea to Langevin equations [10, 11, 12, 13] have proved successful, and convergence properties may be analyzed using second-order difference equations [12, 14, 15]. The implicit midpoint rule cannot be written as a difference equation but can be written as a matrix equation [15]. Schurz [16, 17] studied multidimensional Ornstein–Uhlenbeck processes and showed that the implicit midpoint rule possesses the correct stationary distribution. Time-dependent properties of numerical algorithms for the class of linear equations considered in this paper can be considered by means of a modified frequency [15] or by the method of modified equations [18].

Recent analysis by Mannella [19, 20] of the stationary distribution resulting from numerical timestepping methods, based on an expansion of the exponent in (1.5), has led to a proposed modification of the leapfrog method. We shall take this method as one of our examples and find that it provides a notable improvement on the standard leapfrog method.

A slightly different system has been studied by Strømmen Melbø and Higham [21]. They considered a second-order system, but without a damping term. Rather than approaching a stationary density, the sum of the mean squared velocity and position grows proportional to time. Partitioned methods are superior to the Euler method for this system.

In section 2, we consider the case $f(x) = -gx$, where $g > 0$ is a constant, and $s(x) = 1$. That is, we consider linear second-order equations describing harmonic oscillators with additive white noise and damping. The statistics for linear systems are Gaussian, described by a covariance matrix, and can be calculated exactly in the continuous-time and discrete-time cases. A matrix equation is derived for the stationary variances and correlations resulting from a large class of numerical methods. We then calculate the stationary variances produced by some well-known methods. All methods we shall consider use one realization per timestep of a Gaussian distribution with mean zero and variance Δt. This framework includes many methods with multiple intermediate steps and implicit methods (which are in fact explicit for linear equations).

In section 3 we seek "measure-exact" methods for linear second-order SDEs, that is, methods that give the correct late-time mean square of the position and velocity and absence of correlation between them. In section 4, we restrict our consideration to Runge–Kutta methods for systems of additive noise SDEs and show that the unique measure-exact method is the implicit midpoint rule. In section 5 we report on numerical experiments carried out with $f(x) = x - x^3$ and $s(x) = 1$; in section 6 we report on numerical experiments carried out with $f(x) = x - x^3$ and $s(x) = x$. In both cases, of additive and of multiplicative noise in the double-well system, the implicit midpoint rule is the most satisfactory, with no error in two of the three quantities of

interest over the full range of values of damping. However, leapfrog methods have the benefit of being fully explicit.

2. Linear equation and matrix notation. If $f(x) = -gx$, then the stationary density (1.4) is

$$P_\infty(x,v) = N \exp\left(-g\,x^2/2KT - v^2/2KT\right).$$

The distributions of the position and velocity variables are Gaussian with

$$\lim_{t\to\infty} \left\langle \mathbf{X}_t{}^2 \right\rangle = \frac{1}{g} KT\ , \qquad \lim_{t\to\infty} \left\langle \mathbf{V}_t^2 \right\rangle = KT, \qquad \text{and} \qquad \lim_{t\to\infty} \left\langle \mathbf{X}_t \mathbf{V}_t \right\rangle = 0. \tag{2.1}$$

If $f(x) = -gx$ and $s(x) = 1$, then the linear second-order SDE can be written in matrix notation as

$$\mathrm{d}\begin{pmatrix} \mathbf{X}_t \\ \mathbf{V}_t \end{pmatrix} = Q \begin{pmatrix} \mathbf{X}_t \\ \mathbf{V}_t \end{pmatrix} \mathrm{d}t + \epsilon \begin{pmatrix} 0 \\ 1 \end{pmatrix} \mathrm{d}\mathbf{W}_t, \tag{2.2}$$

where

$$Q = \begin{pmatrix} 0 & 1 \\ -g & -\eta \end{pmatrix}. \tag{2.3}$$

We consider numerical updates for the linear system (2.2) that can be summarized as

$$\begin{pmatrix} X_{n+1} \\ V_{n+1} \end{pmatrix} = R \begin{pmatrix} X_n \\ V_n \end{pmatrix} + \epsilon r \Delta W_n, \tag{2.4}$$

where

$$R = \begin{pmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{pmatrix}, \qquad r = \begin{pmatrix} r_1 \\ r_2 \end{pmatrix}$$

and ΔW_n is sampled from a Gaussian distribution with mean zero and variance Δt, independently of ΔW_m for $n \neq m$. Since the numerical update (2.4) is a linear transformation, $P^*_\infty(x,v)$ is Gaussian. Let the correlation matrix be

$$\Sigma = \begin{pmatrix} \sigma_x^2 & \mu \\ \mu & \sigma_v^2 \end{pmatrix}, \tag{2.5}$$

where

$$\sigma_x^2 = \lim_{t_n\to\infty} \left\langle X_n^2 \right\rangle\ , \qquad \sigma_v^2 = \lim_{t_n\to\infty} \left\langle V_n^2 \right\rangle, \qquad \text{and} \qquad \mu = \lim_{t_n\to\infty} \left\langle X_n V_n \right\rangle.$$

Then

$$P^*_\infty(x,v) = \frac{1}{2\pi} |\Sigma^{-1}|^{\frac{1}{2}} \exp\left(-\frac{1}{2}(x,v)\Sigma^{-1} \begin{pmatrix} x \\ v \end{pmatrix} \right).$$

The stationary density $P^*_\infty(x,v)$ has the property of invariance under the transformation (2.4). Now, if $\binom{X_n}{V_n}$ is Gaussian with mean zero and correlation matrix Σ, then $R\binom{X_n}{V_n}$ is Gaussian with mean zero and correlation matrix $R\Sigma R^{\mathrm{T}}$. Thus the correlation

matrix that results from a method of the form (2.4) satisfies $\Sigma = R\Sigma R^{\mathrm{T}} + \epsilon^2 r r^{\mathrm{T}} \Delta t$, or

$$R\Sigma R^{\mathrm{T}} = \Sigma - \epsilon^2 r r^{\mathrm{T}} \Delta t. \tag{2.6}$$

In the remainder of this section, we shall use (2.6) to calculate the σ_x^2, μ, and σ_v^2 as a function of η and Δt under six numerical methods for the case $s(x) = 1$. We can rewrite (2.6) in a form suitable for inversion:

$$\begin{pmatrix} r_{11}^2 - 1 & 2r_{11}r_{12} & r_{12}^2 \\ r_{11}r_{21} & r_{11}r_{22} + r_{12}r_{21} - 1 & r_{12}r_{22} \\ r_{21}^2 & 2r_{21}r_{22} & r_{22}^2 - 1 \end{pmatrix} \begin{pmatrix} \sigma_x^2 \\ \mu \\ \sigma_v^2 \end{pmatrix} = -\epsilon^2 \Delta t \begin{pmatrix} r_1^2 \\ r_1 r_2 \\ r_2^2 \end{pmatrix}.$$

The stability boundaries of the methods are functions of the parameters $\eta\Delta t$ and $g\Delta t^2$. Of the methods discussed below, the implicit midpoint rule and the leapfrog methods have the virtue of reducing to a symplectic method if $\eta = 0$ [5]. Further explicit methods with this property have been devised [6].

2.1. Forward Euler method. Under the Euler method, the position and velocity variables are updated as follows:

$$X_{n+1} = X_n + V_n \, \Delta t,$$

$$V_{n+1} = V_n - \eta V_n \Delta t + f(X_n) \, \Delta t + \epsilon \Delta W_n.$$

With the notation of (2.4), $R = R_{\mathrm{E}}$ and $r = r_{\mathrm{E}}$, where

$$R_{\mathrm{E}} = 1 + \Delta t Q = \begin{pmatrix} 1 & \Delta t \\ -g\Delta t & 1 - \eta \Delta t \end{pmatrix}, \tag{2.7}$$

$r_{\mathrm{E}} = \binom{0}{1}$ and the solution of (2.6) yields

$$\Sigma_{\mathrm{E}} = \frac{KT}{1 - \frac{g}{\eta}\Delta t} \left(2 - \eta \Delta t + \frac{1}{2} g \Delta t^2 \right)^{-1} \begin{pmatrix} g^{-1}(2 - \eta\Delta t + g\Delta t^2) & -\Delta t \\ -\Delta t & 2 \end{pmatrix}. \tag{2.8}$$

In Figure 1 we display the differences between the stationary variances, obtained by applying the Euler method to the linear equation, and the exact values (2.1). The forward Euler method works best at intermediate values of η. It is unstable if $\eta\Delta t < g\Delta t^2$ or $\eta\Delta t > 2 + \frac{1}{2} g \Delta t^2$. In Figure 2, we plot the correlation between velocity and position, which is zero in the exact solution but is proportional to Δt under the Euler method.

2.2. The Heun method. Under the Heun method, intermediate values are obtained via the Euler method:

$$\hat{X} = X_n + V_n \, \Delta t,$$

$$\hat{V} = V_n - \eta V_n \Delta t + f(X_n) \, \Delta t + \epsilon \, \Delta W_n.$$

The update is

$$X_{n+1} = X_n + \frac{1}{2}(V_n + \hat{V}) \, \Delta t,$$

$$V_{n+1} = V_n - \frac{1}{2}\eta(V_n + \hat{V})\Delta t + \frac{1}{2}(f(X_n) + f(\hat{X})) \, \Delta t + \epsilon \, \Delta W_n.$$

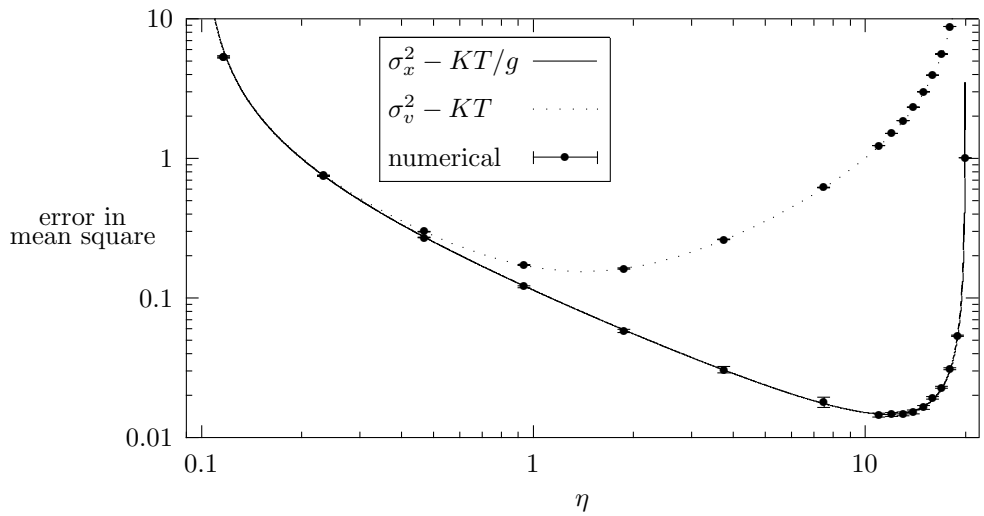


FIG. 1. *Mean squares versus damping for the forward Euler method. The differences between the late-time mean squares and the exact values are plotted against* η *for fixed* $\Delta t = 0.1$, $g = 1$, *and* $KT = 1.0$. *The lines use* (2.8). *Dots with error bars are obtained from the numerical solution of the linear SDE.*

With the notation of (2.4), $R = R_{\mathrm{H}}$ and $r = r_{\mathrm{H}}$, where

$$R_{\mathrm{H}} = I + \Delta t Q + \frac{1}{2}\Delta t^2 Q^2 = \begin{pmatrix} 1 - \frac{1}{2}g\Delta t^2 & \Delta t - \frac{1}{2}\eta\Delta t^2 \\ -g\Delta t + \frac{1}{2}\eta g\Delta t^2 & 1 - \eta\Delta t + \frac{1}{2}(-g+\eta^2)\Delta t^2 \end{pmatrix}$$

and

$$r_{\mathrm{H}} = \begin{pmatrix} \frac{1}{2}\Delta t \\ 1 - \frac{1}{2}\eta\Delta t \end{pmatrix}.$$

Let $k_1 = \eta\Delta t$ and $k_2 = g\Delta t^2$. Then, for the stochastic Heun method,

(2.9)

$$\sigma_x^2 = \frac{KT}{g}\frac{1 - \frac{1}{4}k_1 + \frac{1}{8}k_1^2 - \frac{1}{32}(k_1k_2 + k_1^3) + \frac{3}{64}k_1^2k_2 - \frac{1}{32}k_1k_2^2 + \frac{1}{128}k_2^3}{\left(1 - \frac{1}{2}k_1 + \frac{1}{4}(k_1^2 - k_2) - \frac{1}{8}k_1k_2^2 + \frac{1}{16}k_2^2\right)\left(1 - \frac{1}{2}k_1 + \frac{1}{2}k_2 - \frac{1}{4}k_2^2/k_1\right)},$$

$$\sigma_v^2 = KT\frac{(1 - \frac{1}{2}k_2)^2}{\left(1 - \frac{1}{2}k_1 + \frac{1}{4}(k_1^2 - k_2) - \frac{1}{8}k_1k_2^2 + \frac{1}{16}k_2^2\right)\left(1 - \frac{1}{2}k_1 + \frac{1}{2}k_2 - \frac{1}{4}k_2^2/k_1\right)},$$

$$\mu = KT\Delta t\left(k_1 - \frac{1}{2}k_2\right)\frac{1 - \frac{1}{2}k_1}{\left(1 - \frac{1}{2}k_1 + \frac{1}{4}(k_1^2 - k_2) - \frac{1}{8}k_1k_2^2 + \frac{1}{16}k_2^2\right)\left(1 - \frac{1}{2}k_1 + \frac{1}{2}k_2 - \frac{1}{4}k_2^2/k_1\right)}.$$

Compared with the Euler method, the upper limit on η for stability is little changed. The lower limit is, however, proportional to Δt^3. As $\Delta t \to 0$, the Heun method is stable if $(\frac{1}{2}g\Delta t^2)^2 < \eta\Delta t < 2$, and the correlation between position and velocity is proportional to Δt^2. See Figure 3.

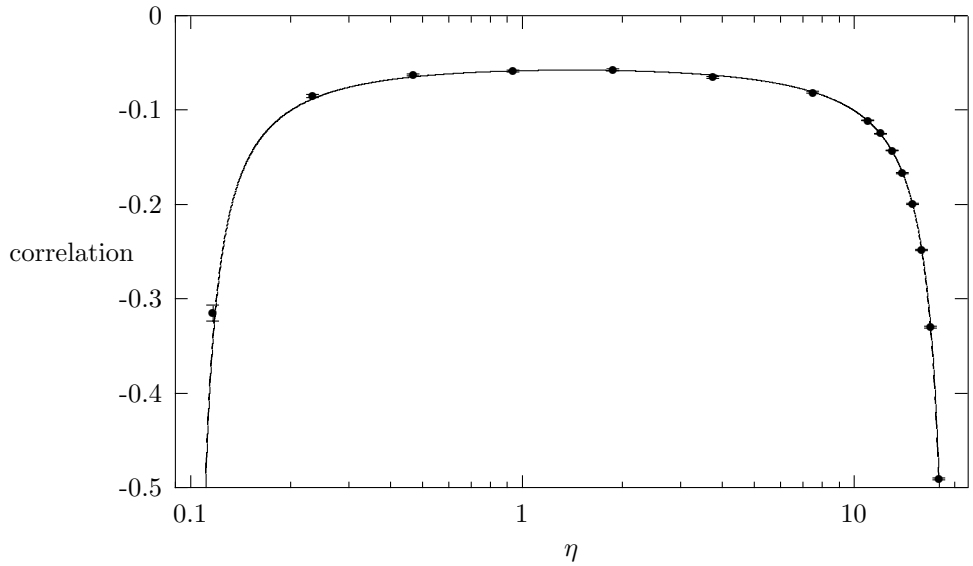


FIG. 2. *Position-velocity correlation versus damping for the forward Euler method. The late-time correlation, μ, obtained using the forward Euler method, is plotted against η for fixed $\Delta t = 0.1$, $g = 1$, and $KT = 1.0$. The line uses* (2.8). *Dots with error bars are obtained from the numerical solution of the linear SDE.*

2.3. The leapfrog method. Under leapfrog methods, velocity and position are updated successively rather than together. The simplest possible method is

$$\hat{X} = X_n + \frac{1}{2} V_n \Delta t,$$

$$V_{n+1} = V_n - \eta V_n \Delta t + f(\hat{X})\,\Delta t + \epsilon\,\Delta W_n,$$

$$X_{n+1} = \hat{X} + \frac{1}{2} V_{n+1} \Delta t. \tag{2.10}$$

Under this method, with the notation of (2.4), $R = R_{\mathrm{le}}$ and $r = r_{\mathrm{le}}$, where

$$R_{\mathrm{le}} = \begin{pmatrix} 1 - \frac{1}{2} g \Delta t^2 & \Delta t - \frac{1}{2}\eta \Delta t^2 - \frac{1}{4} g \Delta t^3 \\ -g\Delta t & 1 - \eta \Delta t - \frac{1}{2} g \Delta t^2 \end{pmatrix}$$

and

$$r_{\mathrm{le}} = \begin{pmatrix} \frac{1}{2}\Delta t \\ 1 \end{pmatrix}.$$

The leapfrog method maintains the independence of position and velocity and produces the exact stationary variance of the position variable. However, the error in σ_v^2 is an increasing function of η:

$$\Sigma_{\mathrm{le}} = KT \begin{pmatrix} g^{-1} & 0 \\ 0 & (1 - \frac{1}{2}\eta \Delta t - \frac{1}{4} g \Delta t^2)^{-1} \end{pmatrix}. \tag{2.11}$$

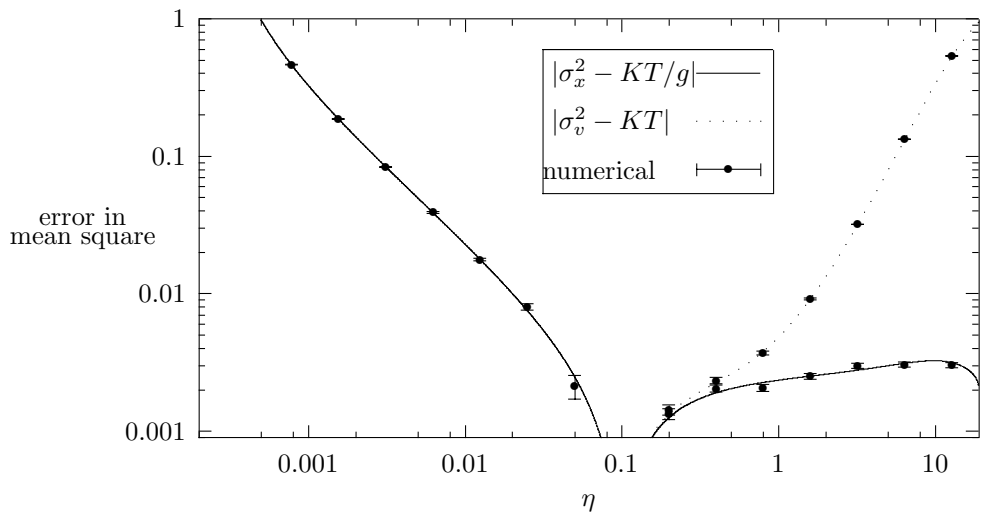


FIG. 3. *Mean squares versus damping for the Heun method. The differences between the late-time mean squares and the exact values are plotted against* η *for fixed* $\Delta t = 0.1$, $g = 1$, *and* $KT = 1.0$. *The lines use* (2.9). *Dots with error bars are obtained from the numerical solution of the linear SDE.*

Mannella [19, 20] has proposed the following modification of the leapfrog method:

$$\hat{X} = X_n + \frac{1}{2}V_n\Delta t,$$

$$V_{n+1} = c_2\left(c_1V_n + f(\hat{X})\,\Delta t + \epsilon\,\Delta W_n\right),$$

$$X_{n+1} = \hat{X} + \frac{1}{2}V_{n+1}\Delta t,$$

where $c_1 = 1 - \frac{1}{2}\eta\Delta t$ and $c_2 = (1 + \frac{1}{2}\eta\Delta t)^{-1}$. The corresponding quantities are

$$R_{\mathrm{M}} = \begin{pmatrix} 1 - c_2\frac{1}{2}g\Delta t^2 & \frac{1}{2}\Delta t\left(1 + c_1c_2 - \frac{1}{2}c_2\Delta t^2\right) \\ -c_2g\Delta t & c_1c_2 - \frac{1}{2}c_2g\Delta t^2 \end{pmatrix},$$

$$r_{\mathrm{M}} = c_2\begin{pmatrix} \frac{1}{2}\Delta t \\ 1 \end{pmatrix},$$

and

$$\Sigma_{\mathrm{M}} = KT\begin{pmatrix} g^{-1} & 0 \\ 0 & (1 - \frac{1}{4}g\Delta t^2)^{-1} \end{pmatrix}. \tag{2.12}$$

The only error, in σ_v^2, is independent of η.

2.4. The BBK method. Under the Brünger–Brooks–Karplus (BBK) method [11], the position variable is updated without reference to the velocity variable. The

two most recent values of the position variable need to be retained, and the update is

$$X_{n+1} = X_n + \frac{1-\frac{1}{2}\eta\Delta t}{1+\frac{1}{2}\eta\Delta t}(X_n - X_{n-1}) + \frac{\Delta t}{1+\frac{1}{2}\eta\Delta t}\left(f(X_n) + \epsilon\,\Delta W_n\right).$$

With the definition of velocity $V_n\Delta t = X_n - X_{n-1}$, proposed in [12], the BBK method can be written in the form (2.4) with [22]

$$R_{\mathrm{B}} = \begin{pmatrix} 1-\frac{g\Delta t^2}{1+\frac{1}{2}\eta\Delta t} & \frac{1-\frac{1}{2}\eta\Delta t}{1+\frac{1}{2}\eta\Delta t}\Delta t \\ -\frac{g\Delta t}{1+\frac{1}{2}\eta\Delta t} & \frac{1-\frac{1}{2}\eta\Delta t}{1+\frac{1}{2}\eta\Delta t} \end{pmatrix}$$

and

$$r_{\mathrm{B}} = \frac{1}{1+\frac{1}{2}\eta\Delta t}\begin{pmatrix}\Delta t \\ 1\end{pmatrix}.$$

Whatever the definition of velocity, we find [12, 15, 18]

$$\sigma_x^2 = \frac{KT}{g}\frac{1}{1-\frac{1}{4}g\Delta t^2}. \tag{2.13}$$

However, σ_v^2 and μ do depend on the definition of velocity. Note that, with the velocity definition $2V_n\Delta t = X_{n+1} - X_{n-1}$, the BBK update cannot be put into the form (2.4).

2.5. The implicit midpoint method. Under the implicit midpoint method, intermediate values are obtained via the implicit procedure

$$\hat{X} = X_n + \frac{1}{2}\hat{V}\,\Delta t,$$

$$\hat{V} = V_n - \frac{1}{2}\eta\hat{V}\Delta t + \frac{1}{2}f(\hat{X})\,\Delta t + \frac{1}{2}\epsilon\,\Delta W_n.$$

The update is

$$X_{n+1} = X_n + \hat{V}\,\Delta t,$$

$$V_{n+1} = V_n - \eta\hat{V}\Delta t + f(\hat{X})\,\Delta t + \epsilon\,\Delta W_n.$$

With the notation of (2.4), $R = R_{\mathrm{im}}$ and $r = r_{\mathrm{I}}$, where

$$R_{\mathrm{im}} = \left(I - \frac{1}{2}\Delta t Q\right)^{-1}\left(I + \frac{1}{2}\Delta t Q\right) = \begin{pmatrix} 1-\frac{1}{2}\kappa^{-1}g\Delta t^2 & \kappa^{-1}\Delta t \\ -\kappa^{-1}g\Delta t & 2\kappa^{-1}-1 \end{pmatrix},$$

and

$$r_{\mathrm{im}} = \kappa^{-1}\begin{pmatrix}\frac{1}{2}\Delta t \\ 1\end{pmatrix},$$

where $\kappa = 1 + \frac{1}{2}\eta\Delta t + \frac{1}{4}g\Delta t^2$. Thus

$$\Sigma_{\mathrm{im}} = KT\begin{pmatrix} g^{-1} & 0 \\ 0 & 1\end{pmatrix}. \tag{2.14}$$

Since R_{im} and r_{im} satisfy (2.6), the implicit midpoint method gives the exact stationary variances for all values of η. Only an algebraic error in Appendix A of [15] prevented this observation from being made by Mishra and Schlick. The behavior of the implicit midpoint method is explored further in section 4.

3. Measure-exact algorithms. We shall call a method "measure-exact" if its correlation matrix is the exact one for (2.2):

$$\Sigma_{\mathrm{e}} = KT \begin{pmatrix} g^{-1} & 0 \\ 0 & 1 \end{pmatrix}. \tag{3.1}$$

The R and r matrices for a measure-exact method satisfy (2.6) with $\Sigma = \Sigma_e$, that is,

$$R \begin{pmatrix} g^{-1} & 0 \\ 0 & 1 \end{pmatrix} R^{\mathrm{T}} = \begin{pmatrix} g^{-1} & 0 \\ 0 & 1 \end{pmatrix} - 2\eta r r^{\mathrm{T}} \Delta t. \tag{3.2}$$

We shall assume Σ is positive definite and let Δt be chosen small enough to ensure $G = \Sigma - \epsilon^2 r r^{\mathrm{T}} \Delta t$ is also positive definite. Matrices $\Sigma^{\frac{1}{2}}$ and $G^{\frac{1}{2}}$ can be constructed to satisfy $G = G^{\frac{1}{2}}(G^{\frac{1}{2}})^{\mathrm{T}}$ and $\Sigma = \Sigma^{\frac{1}{2}}(\Sigma^{\frac{1}{2}})^{\mathrm{T}}$. Let $\Sigma^{-\frac{1}{2}}$ and $G^{-\frac{1}{2}}$ be the inverses of $\Sigma^{\frac{1}{2}}$ and $G^{\frac{1}{2}}$. Condition (2.6) can then be rewritten as $PP^{\mathrm{T}} = I$, where $P = G^{-\frac{1}{2}} R \Sigma^{\frac{1}{2}}$ and I is the identity matrix. Any 2×2 orthogonal matrix P generates a solution of (3.2) as

$$R = G^{\frac{1}{2}} P \Sigma^{-\frac{1}{2}}. \tag{3.3}$$

A real 2×2 orthogonal matrix can be written as

$$P = \begin{pmatrix} (1-a^2)^{1/2} & a \\ -a & (1-a^2)^{1/2} \end{pmatrix},$$

for some $|a| \leq 1$. This form is convenient because we can take $a \propto \Delta t$, so that P reduces to the identity matrix when $a = 0$.

With Σ given by (3.1), we can choose

$$\Sigma^{\frac{1}{2}} = (KT)^{1/2} \begin{pmatrix} g^{-1/2} & 0 \\ 0 & 1 \end{pmatrix}.$$

So

$$G = \Sigma - r r^{\mathrm{T}} \Delta t = \begin{pmatrix} KT/g - r_1^2 \Delta t & -r_1 r_2 \Delta t \\ -r_1 r_2 \Delta t & KT - r_2^2 \Delta t \end{pmatrix},$$

and

$$G^{\frac{1}{2}} = \begin{pmatrix} \left(KT/g - r_1^2 \Delta t\right)^{\frac{1}{2}} & 0 \\ \frac{-r_1 r_2 \Delta t}{\left(KT/g - r_1^2 \Delta t\right)^{\frac{1}{2}}} & \left(KT - r_2^2 \Delta t + \frac{(r_1 r_2 \Delta t)^2}{KT/g - r_1^2 \Delta t}\right)^{\frac{1}{2}} \end{pmatrix}.$$

Any measure-exact method, using one Gaussian random variable per timestep, can be obtained from (3.3) with some choice of a, r_1, and r_2. For example, we may obtain a measure-exact Euler-like method by choosing $r = \binom{0}{1}$. That is,

$$G^{1/2} = (KT)^{1/2} \begin{pmatrix} g^{-1/2} & 0 \\ 0 & (1 - 2\eta \Delta t)^{1/2} \end{pmatrix}$$

and

$$R = G^{1/2} P \Sigma^{-1/2} = \begin{pmatrix} (1-a^2)^{1/2} & a g^{-1/2} \\ -a g^{1/2} (1 - 2\eta \Delta t)^{1/2} & (1-a^2)^{1/2} (1 - 2\eta \Delta t)^{1/2} \end{pmatrix}.$$

The Euler method is obtained as $\Delta t \to 0$ if $a = g^{1/2}\Delta t$. Thus

$$R = \begin{pmatrix} (1 - g\Delta t^2)^{1/2} & \Delta t \\ -g\Delta t(1 - 2\eta\Delta t)^{1/2} & (1 - g\Delta t^2)^{1/2}(1 - 2\eta\Delta t)^{1/2} \end{pmatrix}. \tag{3.4}$$

Notice that the difference between (3.4) and (2.7) is proportional to Δt^2. This modified Euler method reproduces the [illegible] as long as $2\eta\Delta t < 1$. As a next step, we may impose agreement of R with $\exp(\Delta t Q)$ to order Δt^2:

$$R = I + \Delta t Q + \frac{1}{2}\Delta t^2 Q^2 + \cdots .$$

Then $a = g^{\frac{1}{2}}\Delta t - \frac{1}{2}g^{\frac{1}{2}}\eta\Delta t^2$, $r_1 = \frac{1}{2}\Delta t + \cdots$, and $r_2 = 1 - \eta\frac{1}{2}\Delta t + \cdots$. Methods constructed in this way will have a stationary density differing from the exact density by some power of Δt.

Having established that measure-exact numerical methods do exist, in the next section we return to the question of solving condition (3.2) exactly, this time in the context of Runge–Kutta methods.

4. Runge–Kutta methods for additive noise. From the point of view of general Runge–Kutta methods, the system of two SDEs (1.2) with $s(x) = 1$ is a special case of the m-dimensional additive noise SDE that can be written [23]

$$\mathrm{d}\mathbf{Y}_t = f(\mathbf{Y}_t)\mathrm{d}t + \epsilon H \mathrm{d}\mathbf{B}_t,$$

where $\mathbf{Y}_t$ and $\mathbf{B}_t$ are $m \times 1$ column vectors, the entries of $\mathbf{B}_t$ are independent Wiener processes, and H is an $m \times m$ matrix with constant entries. Let the numerically generated approximations be denoted by column vectors y_n. Under an s-stage Runge–Kutta method, y_{n+1} is obtained from y_n as a weighted sum of s evaluations of the function f at intermediate values Y_i:

$$y_{n+1} = y_n + \sum_{j=1}^{s} b_j f(Y_j)\Delta t + \epsilon H \Delta B_n,$$

where $\sum_j b_j = 1$. The m entries of ΔB_n are drawn independently from a Gaussian distribution with mean zero and variance Δt. The column vectors of intermediate values satisfy

$$Y_i = y_n + \sum_{j=1}^{s} a_{ij} f(Y_j)\Delta t + \epsilon c_i H \Delta B_n. \tag{4.1}$$

Runge–Kutta methods for the case $s(x) \neq 1$ also exist [23].

We apply this general formalism to the linear equation (2.2) with $m = 2$. We have

$$y_n = \begin{pmatrix} X_n \\ V_v \end{pmatrix}, \ f(y) = Qy, \text{ and } H = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}.$$

Let $Y = (Y_1, Y_2, \ldots, Y_s)^{\mathrm{T}}$, $c = (c_1, c_2, \ldots, c_s)^{\mathrm{T}}$, $e = (1, 1, \ldots, 1)^{\mathrm{T}}$ and A be the $s \times s$ matrix whose entries are the a_{ij} in (4.1). Then (4.1) can be written

$$Y_i = e \otimes y_n + A \otimes Q Y_i \Delta t + \epsilon c \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} \Delta W_n.$$

If $b = (b_1, b_2, \ldots, b_s)^{\mathrm{T}}$, we can write

$$\begin{aligned} y_{n+1} &= y_n + b^{\mathrm{T}} Y \otimes Q\Delta t + \epsilon \begin{pmatrix} 0 \\ 1 \end{pmatrix} \Delta W_n \\ &= y_n + b^{\mathrm{T}} \otimes Q\Delta t \left(I_s \otimes I_m - A \otimes Q\Delta t\right)^{-1} \left(e \otimes y_n + \epsilon c \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} \Delta W_n \right) \\ &\qquad + \epsilon \begin{pmatrix} 0 \\ 1 \end{pmatrix} \Delta W_n \\ &= \left(I_m + b^{\mathrm{T}} \otimes Q\Delta t \left(I_s \otimes I_m - A \otimes Q\Delta t\right)^{-1} e \right) y_n \\ &\qquad + \epsilon \left(I_m + b^{\mathrm{T}} \otimes Q\Delta t \left(I_s \otimes I_m - A \otimes Q\Delta t\right)^{-1} c \right) \begin{pmatrix} 0 \\ 1 \end{pmatrix} \Delta W_n. \end{aligned}$$

Thus

$$y_{n+1} = R(\Delta t Q) y_n + R_1(\Delta t Q) \begin{pmatrix} 0 \\ 1 \end{pmatrix} \epsilon \Delta W_n,$$

where, for scalar z,

$$R(z) = 1 + b^{\mathrm{T}} z (I_s - Az)^{-1} e$$

and

$$R_1(z) = 1 + b^{\mathrm{T}} z (I_s - Az)^{-1} c.$$

In the notation of (2.4), $r = R_1 \binom{0}{1}$. For example, the Euler method has $R = \Delta t Q$ and $r^{\mathrm{T}} = \begin{pmatrix} 0 & 1 \end{pmatrix}$.

Let us now examine the condition for an exact stationary measure. We shall take $g = 1$, which can always be achieved by rescaling time, so

$$\Sigma_{\mathrm{e}} = KT \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

The equation to be satisfied is now

$$RR^{\mathrm{T}} - I + 2\eta \Delta t R_1 \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} R_1^{\mathrm{T}} = 0. \tag{4.2}$$

Let

$$R(\Delta t Q) = I + \sum_{i=1}^{\infty} \alpha_i (\Delta t Q)^i$$

and

$$R_1(\Delta t Q) = I + \sum_{i=1}^{\infty} \beta_i (\Delta t Q)^i.$$

Then (4.2) can be expanded in powers of Δt as

$$\begin{aligned}
&\Delta t\left(\alpha_1(Q+Q^{\mathrm{T}})+2\eta\begin{pmatrix}0&0\\0&1\end{pmatrix}\right)\\
&+\Delta t^2\left(\alpha_1^2QQ^{\mathrm{T}}+\alpha_2(Q^2+(Q^{\mathrm{T}})^2)+2\eta\beta_1\left(Q\begin{pmatrix}0&0\\0&1\end{pmatrix}+\begin{pmatrix}0&0\\0&1\end{pmatrix}Q^{\mathrm{T}}\right)\right)\\
&+\Delta t^3\Bigg(\alpha_3(Q^3+(Q^{\mathrm{T}})^3)+\alpha_1\alpha_2(Q^2Q^{\mathrm{T}}+Q(Q^2)^{\mathrm{T}})\\
&\qquad+2\eta\left(\beta_2Q^2\begin{pmatrix}0&0\\0&1\end{pmatrix}+\beta_2\begin{pmatrix}0&0\\0&1\end{pmatrix}(Q^{\mathrm{T}})^2+\beta_1^2Q\begin{pmatrix}0&0\\0&1\end{pmatrix}Q^{\mathrm{T}}\right)\Bigg)+\cdots=0.
\end{aligned}$$

If this equation is to be satisfied for all Δt, then each coefficient in the expansion must be zero. It is easily shown that this can hold if and only if $\alpha_i=(\frac{1}{2})^{i-1}$ and $\beta_i=(\frac{1}{2})^i$. Thus the family of Runge–Kutta methods that gives the exact stationary density is characterized by

$$R(\Delta tQ)=\left(I-\frac{1}{2}\Delta tQ\right)^{-1}\left(I+\frac{1}{2}\Delta tQ\right) \tag{4.3}$$

and

$$R_1(\Delta tQ)=(\Delta tQ)^{-1}(R(\Delta tQ)-I).$$

Explicitly, with Q given by (2.3),

$$r=R_1\begin{pmatrix}0\\1\end{pmatrix}=\Delta t^{-1}\begin{pmatrix}-r_{22}-\eta r_{12}+1\\ r_{12}\end{pmatrix}.$$

While Runge–Kutta methods with more than one stage can be constructed that have the stability function given by (4.3), they all have a singular tableau matrix.

The method also has the virtue of being symplectic in the limit $\eta\to 0$ [5]. For linear equations, the method is explicit.

In the case of (2.2), the implicit midpoint rule is implemented as follows. We first generate $\hat{X}$ and $\hat{V}$:

$$\hat{V}=\kappa^{-1}\left(V_n-gX_n\frac{1}{2}\Delta t+\frac{1}{2}\epsilon\,\Delta W_n\right) \tag{4.4}$$

$$\hat{X}=X_n+\hat{V}\,\frac{1}{2}\Delta t, \tag{4.5}$$

where $\kappa=1+\frac{1}{2}\eta\Delta t+\frac{1}{4}g\Delta t^2$. Then

$$X_{n+1}=X_n+\hat{V}\,\Delta t, \tag{4.6}$$

$$V_{n+1}=V_n-\eta\hat{V}\Delta t-g\hat{X}\,\Delta t+\epsilon\Delta W_n. \tag{4.7}$$

Steps (4.4)–(4.7) are carried out in the order given.

5. Double-well system: Additive noise. The good properties of the implicit midpoint and leapfrog methods applied to linear second-order SDEs prompt us to investigate their accuracy for nonlinear equations. In this section we take as our example system (1.2), with the double-well potential

$$V(x) = -\frac{1}{2}x^2 + \frac{1}{4}x^4 \tag{5.1}$$

and with $s(x) = 1$. The stationary density is given explicitly by (1.5). The statistics of the position variable are non-Gaussian, but the mean is zero and σ_x^2 can be evaluated to arbitrary accuracy by a numerical evaluation of an integral.

In general, using an implicit method on a nonlinear equation requires an iterative procedure at each timestep. However, the structure of second-order systems makes a very simple iteration possible. First, the intermediate value $\hat{X}$ is generated by fixed-point iteration with the starting value $\hat{X} = X_n$. This amounts to repeated evaluation of

$$\hat{X} = X_n + \left(1 + \frac{1}{2}\eta\Delta t\right)^{-1} \frac{1}{2}\Delta t \left(V_n + \frac{1}{2}\Delta t f(\hat{X}) + \frac{1}{2}\epsilon\Delta W_n\right). \tag{5.2}$$

The rest of the algorithm is explicit:

$$\hat{V} = \left(V_n + \frac{1}{2}f(\hat{X})\,\Delta t + \frac{1}{2}\epsilon\,\Delta W_n\right) \Big/ \left(1 + \frac{1}{2}\eta\Delta t\right),$$

$$X_{n+1} = X_n + \hat{V}\,\Delta t,$$

$$V_{n+1} = V_n - \eta\hat{V}\Delta t + f(\hat{X})\Delta t + \epsilon\Delta W_n.$$

In our numerical experiments with the double-well system we used six iterations of (5.2), which was not a large overhead, comparable to an explicit predictor-corrector approach. However, in larger systems, where evaluation of $f(x)$ is computationally expensive, the number of iterations needs to be considered along with Δt.

In Figure 4 we display results obtained at $KT = 0.1$, using the Heun, leapfrog, and implicit midpoint methods, as a function of η for $\Delta t = 0.1$. The Heun method is seen to be poor both at large and small values of damping, while the leapfrog method is accurate for small damping but inaccurate in σ_v^2 at large values of damping. The implicit midpoint method produces the exact values of σ_v^2 and μ at all values of damping but has a nonzero error in σ_x^2. Results for Mannella's modification of the leapfrog method are not shown in Figure 4 but are summarized as follows: the errors show little or no dependence on η, the error in σ_x^2 is comparable to that of the implicit midpoint method, while the error in σ_v^2 is similar to that of the standard leapfrog method as $\eta \to 0$.

In Figure 5 we display results obtained using the Heun and implicit midpoint methods, the standard leapfrog method, and Mannella's modified leapfrog method as a function of Δt, with $\eta = 1$ and $KT = 0.1$. Both leapfrog methods and the implicit midpoint method maintain the independence of position and velocity for all values of Δt. The implicit midpoint method also gives the exact value of the late-time mean square of the velocity variable. Mannella's modification of the leapfrog method leads to a reduced error in σ_v^2.

It is appropriate to compare numerical methods in a way that reflects their demand for computer resources. Figure 6 plots the errors in σ_x^2 and σ_v^2 as a function of

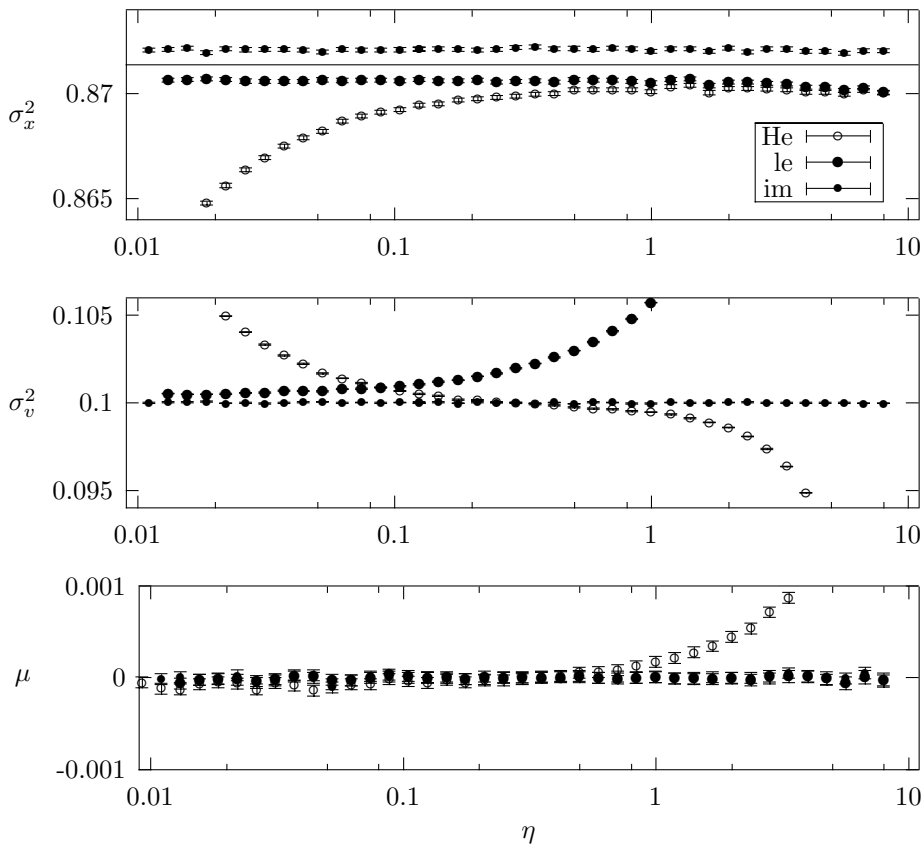


FIG. 4. *Mean squares and correlation versus damping: double-well system. Results obtained at* $KT = 0.1$ *using the Heun (circles), leapfrog (larger filled circles), and implicit midpoint (small filled circles) methods with* $\Delta t = 0.1$. *The top graph shows the late-time mean square of the position variable; the exact value is shown as a solid line. The middle graph shows the late-time mean square of the velocity variable; the exact value is* 0.1. *The lower graph shows the correlation between the position and velocity variables; the exact value is* 0.

the number of evaluations of $f(x)$ per time unit. In the cases of the leapfrog methods, the horizontal axis is simply Δt^{-1}. The Heun method requires two evaluations of $f(x)$ per timestep; the horizontal axis in this case is $2\Delta t^{-1}$. The implicit midpoint rule requires a total of eight such evaluations, six of them in the iterative step; the horizontal axis is thus $8\Delta t^{-1}$. All four methods have errors in σ_x^2 proportional to Δt^2; the extra computer time required for the iteration makes the implicit midpoint rule more expensive. On the other hand, the implicit midpoint rule reproduces the exact value of σ_v^2. Mannella's modification reduces the error in σ_v^2 associated with the leapfrog method.

6. Double-well system: Multiplicative noise. It is of interest to see how the methods analyzed and tested in this paper perform when extended to multiplicative noise systems, i.e., $s(x) \neq 1$. We first describe in the following steps how we have implemented them in our numerical experiments.

(i) Under the Heun method, intermediate values are obtained via the Euler

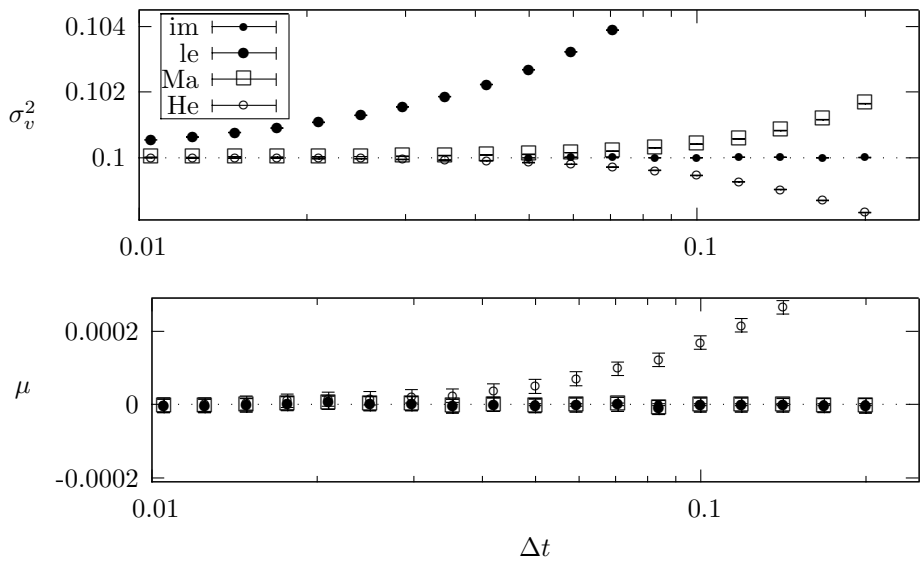


FIG. 5. *Mean-squared velocity and correlation versus Δt: double-well system. Results obtained at $KT = 0.1$ using the Heun method (circles), leapfrog method (larger filled circles), Mannella's modification (squares), and the implicit midpoint method (small filled circles) with $\eta = 1$. The top graph shows the late-time mean square of the velocity variable; the exact value is* 0.1. *The lower graph shows the correlation between the position and velocity variables.*

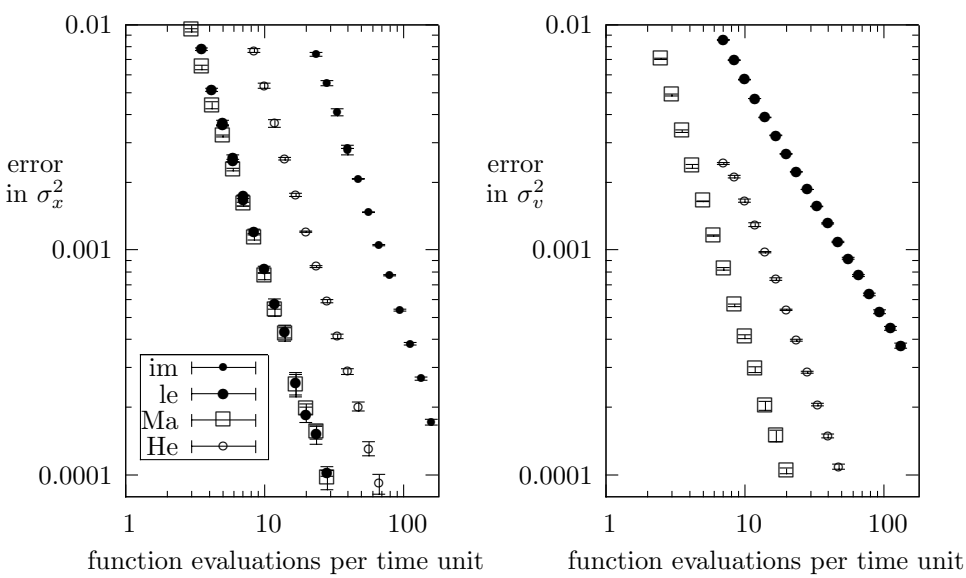


FIG. 6. *Error in mean squares: double-well system. Results obtained at $KT = 0.1$ and $\eta = 1$ using the Heun method (circles), leapfrog method (larger filled circles), Mannella's modification (squares), and the implicit midpoint method (small filled circles) with $\eta = 1$. The horizontal coordinate is Δt^{-1} multiplied by the number of evaluations of $f(x)$ per timestep.*

method:

$$\hat{X} = X_n + V_n\,\Delta t,$$

$$\hat{V} = V_n - \eta s^2(X_n)V_n\Delta t + f(X_n)\,\Delta t + s(X_n)\epsilon\,\Delta W_n.$$

The update is

$$X_{n+1} = X_n + \frac{1}{2}(V_n + \hat{V})\,\Delta t,$$

$$V_{n+1} = V_n - \frac{1}{2}\eta(s^2(X_n)V_n + s^2(\hat{X})\hat{V})\Delta t + \frac{1}{2}(f(X_n) + f(\hat{X}))\,\Delta t + \frac{1}{2}\epsilon(s(X_n) + s(\hat{X}))\Delta W_n.$$

(ii) The standard leapfrog method is now

$$\hat{X} = X_n + \frac{1}{2}V_n\Delta t,$$

$$V_{n+1} = V_n - \eta s^2(\hat{X})V_n\Delta t + f(\hat{X})\,\Delta t + \epsilon s(\hat{X})\Delta W_n,$$

$$X_{n+1} = \hat{X} + \frac{1}{2}V_{n+1}\Delta t.$$

(iii) Mannella's leapfrog method is now

$$\hat{X} = X_n + \frac{1}{2}V_n\Delta t,$$

$$V_{n+1} = \left(1 + \frac{1}{2}s^2(\hat{X})\eta\Delta t\right)^{-1}\left(\left(1 - \frac{1}{2}s^2(\hat{X})\eta\Delta t\right)V_n + f(\hat{X})\,\Delta t + \epsilon s(\hat{X})\Delta W_n\right),$$

$$X_{n+1} = \hat{X} + \frac{1}{2}V_{n+1}\Delta t.$$

(iv) When using the implicit midpoint method, intermediate values $\hat{X}$ are generated by repeated evaluation of

$$\hat{X} = X_n + \left(1 + \frac{1}{2}s(\hat{X})^2\eta\Delta t\right)^{-1}\frac{1}{2}\Delta t\left(V_n + \frac{1}{2}\Delta t f(\hat{X}) + \frac{1}{2}\epsilon s(\hat{X})\Delta W_n\right),$$

with the starting value $\hat{X} = X_n$. The rest of the algorithm is explicit:

$$\begin{aligned}\hat{V} &= V_n + \frac{1}{2}\Delta t\hat{X},\\ X_{n+1} &= X_n + \hat{V}\,\Delta t,\\ V_{n+1} &= V_n - \eta s^2(X_{n+1})\hat{V}\Delta t + f(\hat{X})\Delta t + s(X_{n+1})\epsilon\Delta W_n.\end{aligned}$$

We have performed numerical experiments with $s(x) = x$. In Figure 7 we display results obtained at $KT = 0.1$, using the Heun method, Mannella's modification of the leapfrog method, and the implicit midpoint method, as a function of η for $\Delta t = 0.1$.

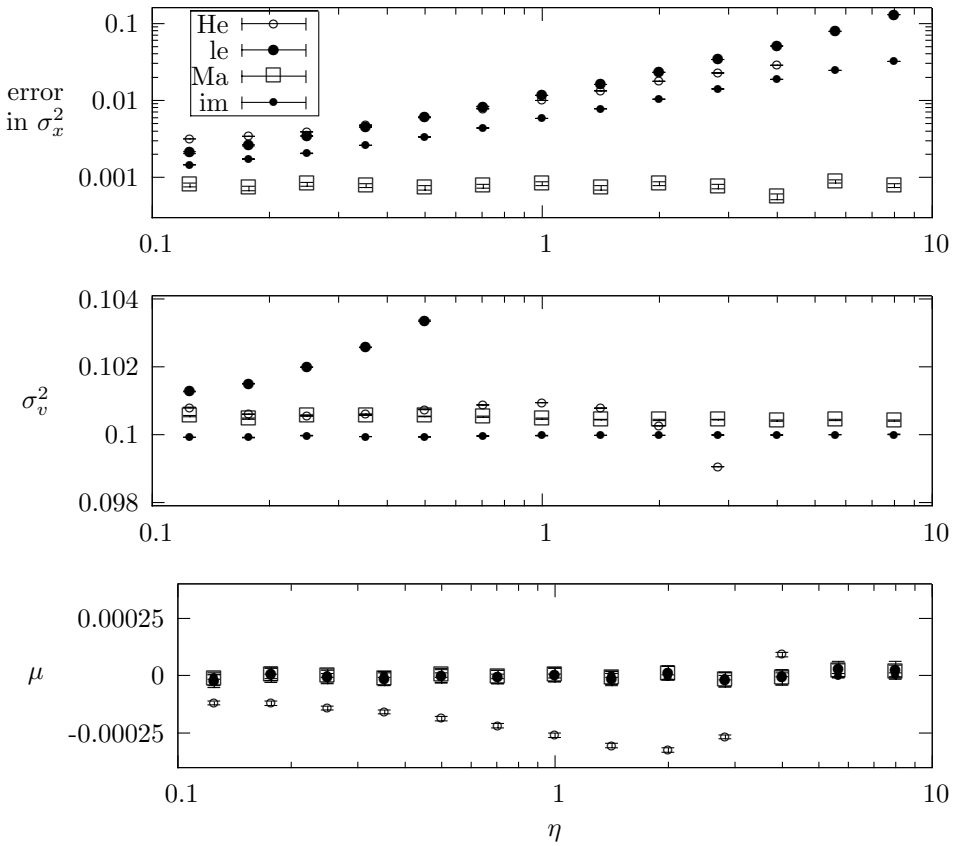


FIG. 7. *Mean squares and correlation versus damping: double-well system with multiplicative noise. Results obtained at* $KT = 0.1$ *and* $\Delta t = 0.1$ *using the Heun method (circles), the leapfrog method (larger filled circles), Mannella's modification of the leapfrog method (squares), and the implicit midpoint method (small filled circles). The top graph shows the error in the late-time mean square of the position variable (log scale). The middle graph shows the late-time mean square of the velocity variable; the exact value is* 0.1. *The lower graph shows the correlation between the position and velocity variables.*

In terms of the error in σ_x^2, Mannella's method performs best. The implicit midpoint method, however, is the only one that appears to give the exact value of σ_v^2 at all values of η. In Figure 8 we display numerical results with $KT = 0.1$ and $\eta = 1$ as a function of Δt. Mannella's method is the only one that has second-order convergence in σ_x^2, but the implicit midpoint method is more accurate in σ_v^2.

Note that, for systems of SDEs where there is a difference between Ito and Stratonovich forms, the implicit midpoint rule will converge to the Stratonovich form. However, it will only have strong order 0.5 for noncommutative SDEs with more than one noise term, in which case other approaches are needed to obtain a strong order of 1 [23, 24].

7. Conclusion. Exact calculations for linear equations are of interest in their own right and because the qualitative form of the error as a function of Δt and η carries over to the nonlinear equations used as examples here. The implicit midpoint rule is the only Runge–Kutta method with a nonsingular tableau matrix that reproduces

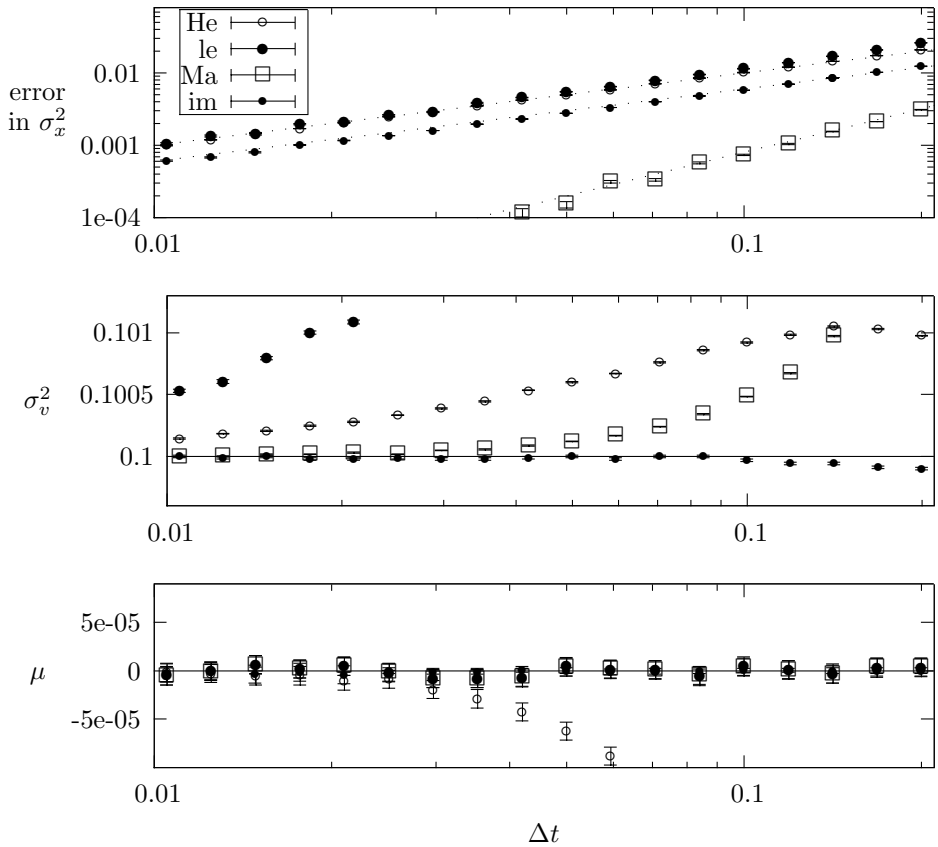


FIG. 8. *Mean squares and correlation versus Δt: double-well system with multiplicative noise. Results obtained at $KT = 0.1$ and $\eta = 1$ using the Heun method (circles), the leapfrog method (larger filled circles), Mannella's modification of the leapfrog method (squares), and the implicit midpoint method (small filled circles). The top graph shows the error in the late-time mean square of the position variable (log scale); the dotted lines are $0.1\Delta t$, $0.06\Delta t$, and $0.08\Delta t^2$. The middle graph shows the late-time mean square of the velocity variable; the exact value is 0.1. The lower graph shows the correlation between the position and velocity variables.*

the exact stationary distribution for all values of damping. It also performs well when applied to second-order nonlinear and multiplicative noise equations, with no apparent error in the stationary mean square of the velocity variable even for the nonlinear double-well system with multiplicative noise. However, leapfrog-type methods have the advantage of being fully explicit even for nonlinear equations. In particular, they require only one evaluation of the deterministic force per timestep.

Applied to linear equations, the simplest leapfrog method gives the exact stationary variance of the position variable and maintains the independence of position and velocity. Mannella's modification of the leapfrog method maintains this virtue while reducing the magnitude of the error in the variance of the velocity variable. We found similar qualitative behavior in our numerical experiments on the nonlinear double-well system: the implicit midpoint method and Mannella's method perform best, the former being superior in the error in σ_v^2 and the latter in σ_x^2.

In the deterministic case when solving separable Hamiltonian problems, explicit partitioned Runge–Kutta methods can be constructed that are symplectic. A chal-

lenge is to extend the idea in a stochastic setting to construct explicit partitioned methods that approximate the stationary correlation matrix with high-order accuracy.

Acknowledgments. GL thanks the Ethel Raybould Foundation and the Advanced Computational Modelling Centre of the University of Queensland for funding a visit in 2002. KB would like to thank the Australian Research Council for its funding of a Federation Fellowship.

REFERENCES

[1] M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids*, Oxford University Press, Oxford, UK, 1997.

[2] C. A. Marsh and J. M. Yeomans, *Dissipative particle dynamics: The equilibrium for finite time steps*, Europhys. Lett., 37 (1997), pp. 511–516.

[3] T. Shardlow, *Splitting for dissipative particle dynamics*, SIAM J. Sci. Comput., 24 (2003), pp. 1267–1282.

[4] E. Hairer, S. P. Norsett, and G. Wanner, *Solving Ordinary Differential Equations* I: *Non-stiff Problems*, 2nd ed., Springer-Verlag, Berlin, 1993.

[5] E. Hairer, C. Lubich, and G. Wanner, *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*, Springer-Verlag, Berlin, 2002.

[6] G. N. Milstein and M. V. Tretyakov, *Quasi-symplectic methods for Langevin-type equations*, IMA J. Numer. Anal., 23 (2003), pp. 593–626.

[7] J. C. Mattingley, A. M. Stuart, and D. J. Higham, *Ergodicity for SDEs and approximations: Locally Lipschitz vector fields and degenerate noise*, Stochastic Process. Appl., 101 (2002), pp. 185–232.

[8] C. W. Gardiner, *Handbook of Stochastic Methods for Physics, Chemistry, and the Natural Sciences*, 3rd ed., Springer-Verlag, Berlin, 2004.

[9] L. Verlet, *Computer experiments on classical fluids.* I. *Thermodynamical properties of Lennard–Jones molecules*, Phys. Rev., 159 (1967), pp. 98–103.

[10] W. F. van Gunsteren and H. J. C. Berendsen, *Algorithms for Brownian dynamics*, Mol. Phys., 45 (1982), pp. 637–647.

[11] A. Brünger, C. L. Brooks III, and M. Karplus, *Stochastic boundary conditions for molecular dynamics simulations of st2 water*, Chem. Phys. Lett., 105 (1984), pp. 495–500.

[12] R. W. Pastor, B. P. Brooks, and A. Szabo, *An analysis of the accuracy of Langevin and molecular dynamics algorithms*, Mol. Phys., 65 (1988), pp. 1409–1419.

[13] J. A. Izaguirre and R. D. Skeel, *An impulse integrator for Langevin dynamics*, Mol. Phys., 100 (2002), pp. 3885–3891.

[14] G. Zhang and T. Schlick, *Implicit discretization schemes for Langevin dynamics*, Mol. Phys., 84 (1995), pp. 1077–1098.

[15] B. Mishra and T. Schlick, *The notion of error in Langevin dynamics.* I. *Linear analysis*, J. Chem. Phys., 105 (1996), pp. 299–318.

[16] H. Schurz, *The invariance of asymptotic laws of linear stochastic systems under discretization*, Z. Angew Math. Mech., 6 (1999), pp. 375–382.

[17] H. Schurz, *Preservation of probabilistic laws through Euler methods for Ornstein–Uhlenbeck processes*, Stochastic Anal. Appl., 17 (1999), pp. 463–486.

[18] W. Wang and R. D. Skeel, *Analysis of a few numerical integration methods for the Langevin equation*, Mol. Phys., 101 (2003), pp. 2149–2156.

[19] R. Mannella, *Quasisymplectic integrators for stochastic differential equations*, Phys. Rev. E, 69 (2004), 041107.

[20] R. Mannella, *Numerical stochastic integration for quasi-symplectic flows*, SIAM J. Sci. Comput., 27 (2006), pp. 2121–2139.

[21] A. H. Strømmen Melbø and D. J. Higham, *Numerical simulation of a linear oscillator with additive noise*, Appl. Numer. Math., 51 (2004), pp. 89–99.

[22] W. Wang, *Analysis of a Few Numerical Integration Methods for the Langevin Equation*, Master's thesis, University of Illinois at Urbana-Champaign, Urbana, IL, 2001.

[23] K. Burrage and P. M. Burrage, *Order conditions of stochastic Runge–Kutta methods by B-series*, SIAM J. Numer. Anal., 38 (2000), pp. 1626–1646.

[24] G. Lord, S. J. A. Malham, and A. Wiese, *Efficient strong integrators for stochastic linear systems*, submitted.

SIAM J. SCI. COMPUT.
Vol. 29, No. 1, pp. 265–289
© 2007 Society for Industrial and Applied Mathematics

STABILITY THROUGH SYNCHRONIZATION IN NONLINEAR MULTISCALE TRANSFORMATIONS*

FRANCESC ARÀNDIGA† AND ROSA DONAT†

Abstract. Nonlinear analogues of classical wavelet transformations have been designed within spatially oriented multiresolution frameworks such as Harten's framework [*J. Appl. Numer. Math.*, 12 (1993), pp. 153–193] and the lifting framework [R. L. Claypoole et al., *IEEE Trans. Image Process.*, 12 (2003), pp. 1449–1459]. In designing and using nonlinear multiscale transformations for data compression purposes, the main issue is that of ensuring stability. In [*J. Appl. Numer. Math.*, 12 (1993), pp. 153–193], Harten ensures stability by designing a special error-control mechanism that provides estimates for the compression error. In [R. L. Claypoole et al., *IEEE Trans. Image Process.*, 12 (2003), pp. 1449–1459] no theoretical results are provided, but the authors argue that stability is a consequence of synchronization between the nonlinear decisions made by the prediction operators in the direct and inverse transformations. By interpreting the nonlinear multiscale transformation of R. L. Claypoole et al. within Harten's cell-average framework, we are able to provide stability bounds for the scheme of R. L. Claypoole et al. and establish a theoretical basis for comparison between both nonlinear multiresolution schemes.

Key words. multiscale decomposition, nonlinear prediction, stability, image compression, lifting scheme

AMS subject classifications. 65D05, 65T50, 94A08

DOI. 10.1137/050630817

1. Introduction.

Multiscale transformations have recently been used in the first step of transform coding algorithms for image compression. A multiscale transformation allows for an [illegible] representation of the image data since, typically, a significant number of the transform coefficients are small and can be quantized or discarded with little loss of real information. A lossy multiresolution-based compression scheme acting on a sequence of data sampled at a given resolution, v^L, can be schematically described as follows:

$$v^L \longrightarrow v_d = Mv^L = (v^0, d^1, \ldots, d^L) \longrightarrow \hat{v}_d = Q_{\bar{\varepsilon}} Mv^L = (\hat{v}^0, \hat{d}^1, \ldots, \hat{d}^L). \tag{1.1}$$

Here Mv^L is the multiresolution representation of the signal v^L and $Q_{\bar{\varepsilon}}$ is the [illegible], which proceeds by quantizing or truncating the elements of Mv^L with respect to a set of user-dependent parameters $\bar{\varepsilon} = (\varepsilon_0, \ldots, \varepsilon_L)$ and then coding the resulting representation.

Even though compression is achieved through $Q_{\bar{\varepsilon}}$, and not during the transformation step, the properties of the multiscale transformation, M, are most important in the overall performance of the transform coding algorithm. Mv^L is composed by v^0, which corresponds to a sampling of the data at a much coarser resolution, and a sequence of details $(d^1, \ldots, d^L)$. Each d^k represents the intermediate information which is necessary to recover the data sampled at level k, v^k from the data sampled at level $k-1$, v^{k-1}. Multiresolution transforms are designed in such a way that most

*Received by the editors May 6, 2005; accepted for publication (in revised form) August 10, 2006; published electronically February 2, 2007. This research was supported in part by the Spanish government through grant MTM 2005-07214 and by EU project HPRN-CT-2002-00286.
http://www.siam.org/journals/sisc/29-1/63081.html
†Departament de Matemàtica Aplicada, Universitat de València, Spain (arandiga@uv.es, donat@uv.es).

of these coefficients are small and can be quantized or discarded with little loss of global information content. Wavelet bases typically employed for image compression (such as the Daubechies $(7, 9)$ system) use smooth scaling and wavelet functions and are very efficient for smooth signals; however, they produce many large transform coefficients around the edges in an image which reduces the efficiency of the transform coding algorithm.

To avoid this loss in efficiency, one would like to adapt the multiscale transformation so that the local properties of the image are taken into account, which leads up to considering data-dependent (nonlinear) multiscale transforms. Adaptivity is a hard issue when the basic design of a multiscale transform is done through the standard wavelet technology. However, data-dependent multiscale transformations have been successfully designed within Harten's framework for multiresolution (see [2, 1, 4, 14, 15]) and within the (modified) lifting framework [7].

Under conditions of uniformity, these two frameworks can be seen as alternate implementations of classical wavelet transforms, however, and in contrast with the traditional frequency-domain construction, their design is intrinsically [illegible]. The detail coefficients are obtained as [illegible] that take into account local correlations present in the signal. The spatial construction of the prediction operator is the key advantage in building multiscale transformations specifically adapted to complex geometries and irregular sampling, as well as data-dependent multiscale transformations. It is shown in [2, 1, 4, 14, 15, 7] that a more [illegible] performance around edges in an image can be obtained when using nonlinear multiresolution transformations obtained from data-dependent prediction schemes.

The practical use of a lossy compression scheme, such as (1.1), relies on its stability. It is mandatory that the error in the reconstructed signal obtained from the decoder, i.e., $\hat{v}^L = M^{-1}Q_{\bar{\varepsilon}}Mv^L$, and the original signal, v^L, be somehow compatible with the error made in the uninvertible processing step.

The compression operator $Q_{\bar{\varepsilon}}$ is designed so that v_d and $\hat{v}_d$ in (1.1) are [illegible], in the sense that for some prescribed discrete norm $||\cdot||$,

$$||\hat{v}^0 - v^0|| \le \varepsilon_0, \quad ||\hat{d}^k - d^k|| \le \varepsilon_k, \quad 1 \le k \le L,$$

as is the case in, for example, [illegible] or simple [illegible]. Stability follows if it is possible to establish a relation such as

$$||\hat{v}^L - v^L|| \le \sigma(\varepsilon_0, \varepsilon_1, \ldots, \varepsilon_L), \quad \text{with} \quad \lim_{\varepsilon_l \to 0, 0 \le l \le L} \sigma(\varepsilon_0, \varepsilon_1, \ldots, \varepsilon_L) = 0. \tag{1.2}$$

For linear multiresolution transformations, stability is analyzed in terms of the underlying basis functions of the wavelet system. However, such techniques are no longer applicable to nonlinear transformations, for which linear superposition does not hold. Ensuring stability becomes hence the major issue in practical applications involving nonlinear multiscale transformations.

In Harten's framework, data-dependent prediction operators have been constructed following the design principles of the essentially nonoscillatory (ENO) interpolation technique described in [16]. The solution proposed by Harten to ensure the stability of the nonlinear multiresolution transformations that result from using ENO-type prediction operators involves a modification of the direct transform, with the ultimate goal of controlling the accumulation of errors due to the compression mechanism. Because of this feature, Harten named the modified algorithm an [illegible] (EC) multiresolution transform.

Throughout this paper we shall denote Harten's modified transformation as $\widetilde{M}_{\bar{\varepsilon}}$, because it merges the direct transformation with the processing procedure (quantization or truncation). In [14], Harten shows that (1.2) holds for $\hat{v}^L = M^{-1}\widetilde{M}_{\bar{\varepsilon}} v^L$, which guarantees stability. Harten's EC strategies in various multiresolution frameworks have been analyzed and the error bounds in [14] improved in [4, 1].

In the lifting framework, data-dependent predictors are constructed in [7] by adaptively choosing a linear filter, from a finite pool, at each spatial location where a detail coefficient must be computed. Close to an edge in an image, the length of the filter is reduced so that no data across the edge is used in the prediction step, in an attempt to avoid the large prediction errors associated to the use of linear filters across edges.

It is well known, however, that the use of locally adapted filters can cause a series of incorrect filter choices in the decoder, leading to serious reconstruction errors (see, e.g., [7] and the references therein). In [7], Claypoole et al. argue that the stability of the transform coding algorithm must follow from encoder/decoder [illegible], i.e., from ensuring that the same choice of filter is made at a given location and resolution level in both the encoder and the decoder. For efficiency reasons, this encoder/decoder synchronization must be achieved [illegible], i.e., no side information should be attached to the transform coefficients.

The analysis in [7] shows that it is impossible to guarantee synchronization within the classical split-predict-update lifting routine when using data-dependent prediction schemes. As a solution, Claypoole et al. propose switching instead to an [illegible] form of lifting which is based on Donoho's average-interpolation setting [10], which fits precisely within Harten's cell-average framework.

Once the connection has been established, it is fairly simple to realize that the nonlinear multiscale transformation introduced in [7] is similar in nature to the nonlinear transformations within Harten's framework obtained from ENO prediction operators. In both cases, the nonlinearity is a consequence of the use of a (nonlinear) selection function to adaptively choose a linear filter among a finite pool.

The central subject of this paper is the comparison of the synchronization strategy proposed in [7] with Harten's EC strategy for the cell-average framework [14, 1]. Indeed, although independently developed, the basic structure of the synchronized nonlinear transformation described in [7] is precisely the cell-average EC strategy described by Harten in [14], the main difference lying in the computation of the scale coefficients.

The analysis in this paper shows that the essential ingredient for stability is, in fact, the [illegible] between the prediction choices carried out in the direct transformation and in the inverse transformation at each particular location. In addition, we analyze the different possibilities for the computation of the scale coefficients from a theoretical as well as computational point of view and conclude that Harten's original choice for the scale coefficients provides the best balance between theoretical and practical effectiveness.

The paper is organized as follows: In section 2 we briefly review Harten's cell-average framework for multiresolution and establish its relation with the lifting schemes in [7]. In section 3 we recall the class of prediction operators that are obtained by considering piecewise polynomial reconstruction techniques, linear as well as nonlinear. In section 4 we study the relation between stability and encoder/decoder synchronization for nonlinear prediction schemes based on the use of an adaptive, position-dependent filter selection. In section 5 we give specific stability bounds for several [illegible]

nonlinear multiresolution transformations. The corresponding tensor-product extensions to two-dimensional (2D) signals, used for image compression, are briefly described in section 6. Numerical experiments are presented in section 7. Finally some conclusions are drawn in section 8.

2. The cell-average setting as an update-first lifting scheme. Harten's general framework for multiresolution [15] relies on two operators, decimation and prediction, that define the basic interscale relations. These operators act on linear vector spaces, V^j, that represent the different resolution levels (j increasing implies more resolution)

$$\text{(2.1)} \qquad \text{(a)} \quad D^j : V^j \to V^{j-1}, \qquad \text{(b)} \quad P_j : V^{j-1} \to V^j,$$

and must satisfy two requirements of algebraic nature: (a) D^j must be a [illegible] operator and (b) $D^j P_j = I_{V^{j-1}}$ [illegible], i.e., the identity operator on the lower resolution level represented by V^{j-1}. For all practical purposes in this paper, V^j shall be considered a space of finite-dimensional sequences.

Using these two operators, a vector (i.e., a discrete sequence) $v^j \in V^j$ can be decomposed and reassembled as follows:

$$\text{(2.2)} \qquad \text{(a)} \quad \begin{array}{rcl} v^j & \to & v^{j-1} = D^j v^j \\ & \searrow & e^j = v^j - P_j v^{j-1} \end{array}, \qquad \text{(b)} \quad v^j = P_j v^{j-1} + e^j.$$

The consistency requirement implies that the prediction errors e^j satisfy $D^j e^j = 0$. The detail coefficients are computed as the nonredundant information contained in the prediction error.

To be more precise, and to establish the correspondence with the modified lifting scheme of [7], we concentrate on the one-dimensional (1D) cell-average setting, in which the levels of resolution are specified by a set of uniform nested dyadic grids on the interval $[0,1]$, $\{X^j\}$, $j \geq 0$, of size $h_j = 2^{-j} h_0$: $X^j = \{x_i^j = i h_j\}_{i=0}^{N_j}$, $N_j \cdot h_j = 1$. The cell-average setting is characterized by the interpretation of the discrete data at level j as cell-averages of a function on the grid X^j. Hence, the decimation operator is given as (see, e.g., [14, 4] for full details)

$$\text{(2.3)} \qquad (D^j v^j)_i = \frac{1}{2}(v^j_{2i-1} + v^j_{2i}), \qquad 1 \leq i \leq N_j.$$

The consistency requirement for P_j becomes

$$\text{(2.4)} \qquad v_i^{j-1} = (D^j P_j v^{j-1})_i = \frac{1}{2}\left[(P_j v^{j-1})_{2i-1} + (P_j v^{j-1})_{2i}\right].$$

Hence, if $v^{j-1} = D^j v^j$, then (2.3) and (2.4) imply that

$$e^j_{2i-1} = v^j_{2i-1} - (P_j v^{j-1})_{2i-1} = (P_j v^{j-1})_{2i} - v^j_{2i} = -e^j_{2i},$$

which shows the redundancy inherent in the prediction error. By considering only the prediction errors at (for example) the odd points of the grid X^j, one immediately obtains a one-to-one correspondence between the sets $\{v^j\}_i \leftrightarrow (\{v^{j-1}\}_i, \{d^j\}_i)$, where $d_i^j = e^j_{2i-1}$ and $v^{j-1} = D^j v^j$.

$$\text{(2.5)} \qquad \left\{ \begin{array}{l} v_i^{j-1} = (v^j_{2i} + v^j_{2i-1})/2 \\ d_i^j = v^j_{2i-1} - (P_j v^{j-1})_{2i-1} \end{array} \right\} \iff \left\{ \begin{array}{l} v^j_{2i-1} = (P_j v^{j-1})_{2i-1} + d_i^j \\ v^j_{2i} = 2v_i^{j-1} - v^j_{2i-1} \end{array} \right\}.$$

By repeating the two-step transformation for all levels, we immediately get a multiresolution transformation, M, and its corresponding inverse, M^{-1}.

When comparing with the classical split-predict-update (SPU) form of the lifting schemes, we observe that the design strategy is conceptually different. In the SPU-lifting routine the signal v^j is split into even and odd polyphase coefficients and the prediction operator acts only on even coefficients v^j_{2i}; then the scale coefficients d^j_i represent the prediction residuals at odd locations. The update step is necessary in order to recover a low-pass downsampled version of the original signal.

If the order of the predict and update steps in the lifting scheme is reversed, a low-pass downsampled version of the original signal (i.e., $v^{j-1} = D^j v^j$) is produced first and is subsequently used to predict the odd samples: This is the precise structure of Harten's two-step invertible transformation (2.5), and hence the update-first lifting scheme proposed in [7] has the exact same structure of Harten's framework.

It should be noted, anyhow, that the multiscale transformations obtained from the repeated application of (2.5) share common features with the lifting framework: First, the inverse transformation is computed by undoing the steps made in the direct transform. Second, the computations can be done "in place," i.e., no auxiliary memory is needed. In addition, in both frameworks, the prediction operator can be adjusted to account for boundaries (see [15, 4, 18] for details).

3. Prediction via piecewise-polynomial average interpolation. A well-known construction of prediction operators in the cell-average setting is based on the so-called [illegible] interpolation [14, 10]: For each $1 \le i \le N_{j-1}$ on level $j-1$, construct a polynomial $p_i(x)$ of fixed degree, say R, by specifying its averages on a [illegible] of cells $\mathcal{S}_i = \{i-A, \ldots, i+B\}$.

$$\frac{1}{h_{j-1}} \int_{x^{j-1}_{l-1}}^{x^{j-1}_l} p_i(x)dx = v^{j-1}_l \quad \text{for } l = i-A, \ldots, i+B; \qquad \begin{cases} R = A+B, \\ A \ge 0, B \ge 0. \end{cases} \tag{3.1}$$

The $A+B+1$ relations in (3.1) lead always to a solvable linear system for the coefficients of $p_i(x)$.

The prediction operator is defined by considering the averages of the polynomial piece $p_i(x)$ on the next finer grid as follows:

$$(P_j v^{j-1})_{2i} := \frac{1}{h_j} \int_{x^j_{2i-1}}^{x^j_{2i}} p_i(x)dx, \qquad (P_j v^{j-1})_{2i-1} := \frac{1}{h_j} \int_{x^j_{2i-2}}^{x^j_{2i-1}} p_i(x)dx. \tag{3.2}$$

Since $i \in \mathcal{S}_i$, definition (3.2) clearly implies that P_j satisfies the consistency relation (2.4). Hence, $(P_j v^{j-1})_{2i}$ can easily be computed as $(P_j v^{j-1})_{2i} = 2v^{j-1}_i - (P_j v^{j-1})_{2i-1}$.

For the construction above, a straightforward calculation leads to (see, e.g., [15])

(3.3)

$$\text{(a) } (P_j v^{j-1})_{2i-1} = \sum_{l=-A}^{B} \beta_l v^{j-1}_{i+l}, \quad \text{(b) } \beta_l = \begin{cases} 2\sum_{k=l}^{B} L^{(A,B)}_k\left(-\frac{1}{2}\right), & 0 \le l \le B, \\ -2\sum_{k=-(A+1)}^{l-1} L^{(A,B)}_k\left(-\frac{1}{2}\right), & -A \le l \le -1, \end{cases}$$

where $L^{(A,B)}_m(y)$, $m = -(A+1), \ldots, B$, is the standard Lagrange interpolatory basis for the stencil of points $\{-(A+1), \ldots, B\}$. Then $(P_j v^{j-1})_{2i-1}$ is obtained as the result of applying a particular filter to the lower resolution data v^{j-1}. The filter coefficients

depend solely on $\mathcal{S}_i$, the stencil assigned to the polynomial piece $p_i(x)$. We shall write $p_i(x) = p_i(x, v^{j-1}, \mathcal{S}_i)$ when necessary.

3.1. Let $v_i^j = \frac{1}{h_j} \int_{x_{i-1}^j}^{x_i^j} f(x)dx$, $f(x)$ being a piecewise smooth function. Then

$$f(x) = p_i(x, v^j, \mathcal{S}_i) + O(h_j^{R+1}), \qquad x \in [x_{i-1}^j, x_i^j],$$

if and only if the convex hull of the stencil $\mathcal{S}_i$ is entirely contained in a region of smoothness of $f(x)$ [4]. The corresponding prediction scheme is then $(R+1)$st order accurate on smooth functions.

3.1. Data-independent stencils: Linear prediction operators. A classical prediction follows from considering $\mathcal{S}_i = \{i-s, \ldots, i+s\}$ for all i, where s is a fixed integer. Since $A = B = s$ for all i, the coefficients in (3.3) are independent of the spatial location, i.e., the same filter is used at all odd locations. The resulting prediction scheme is then a linear operator, intimately related to the interpolatory subdivision schemes of Deslauries and Dubuc [9, 10],

It was noticed by Harten [15] and proven later (see, e.g., [12]) that the multiscale transformations in the cell-average setting associated to this class of prediction operators turn out to be the $(1, M = 2s+1)$ branch of the Cohen–Daubechies–Feauveau biorthogonal wavelet (CDF-$(1, M)$) family [8].

3.2. Data-dependent stencils: Nonlinear ENO prediction operators. On the contrary, if the stencil is chosen in a data-dependent way, a nonlinear prediction operator results. The corresponding multiscale transformation then becomes a nonlinear analogue of the CDF$(1, M)$ family [4].

ENO interpolatory techniques [16] have often been used in the construction of $p_i(x)$ in (3.1) [14, 4, 2, 1]. Once R, the accuracy parameter, is chosen, the practical implementation of an ENO-type prediction operator is done in two steps. The first step is the stencil selection mechanism: Given v^{j-1}, we denote by $\mathcal{S}_i(v^{j-1}) =: \mathcal{S}_i^{v^{j-1}}$ the (data-dependent) stencil corresponding to the polynomial piece $p_i(x)$. The stencil $\mathcal{S}_i^{v^{j-1}}$ determines the particular filter to be used at location $2i-1$ through (3.3)(b). The second step is the actual computation of $(P_j v^{j-1})_{2i-1}$ as in (3.3)(a).

There are several algorithms for an ENO selection of stencil, but all involve a nonlinear decision process based on the use of the divided differences of the data (see, e.g., [4] and the references therein for details on various ENO stencil selection mechanisms). For ENO prediction operators of accuracy $R+1$ there is a finite pool of $R+1$ filters that can be chosen at location $2i-1$ (see the examples below).

It is important to remark that the nonlinearity of ENO prediction schemes lies exclusively in the use of a nonlinear selection function to adaptively choose between a finite pool of linear predictors at each spatial location. Indeed, the computation of $(P_j v^{j-1})_{2i-1}$ is carried out by applying the chosen filter on the data v^{j-1}.

A distinctive feature of ENO predictors is that all filters in the pool have and, hence, the same spatial accuracy.

3.2. The data used for the filter selection can be different from the data used in the interpolation process: Let $\hat{v}^{j-1}$ be the data set to be used in the stencil-selection step, and v^{j-1} the data to be interpolated (in the cell-average sense). Then the polynomial pieces $p_i(x) = p_i(x, v^{j-1}, \mathcal{S}_i^{\hat{v}^{j-1}})$ can be uniquely determined for any pair of data sets v^{j-1}, $\hat{v}^{j-1}$. To denote the nonlinear dependence of the prediction scheme from the data set $\hat{v}^{j-1}$, we shall use the notation $P_j = P_{\hat{v}^{j-1}}$ when appropriate.

As mentioned before, $P_{\hat{v}^{j-1}}$ acts on the data in V^{j-1}, i.e.,

$$P_{\hat{v}^{j-1}}u^{j-1} + P_{\hat{v}^{j-1}}w^{j-1} = P_{\hat{v}^{j-1}}(u^{j-1} + w^{j-1}). \tag{3.4}$$

For $R = 2$, there are three possible stencil selections for each i. Each one leads to a different filter to be applied to the interpolation data:

$$(P_j v^{j-1})_{2i-1} = \begin{cases} -\frac{1}{8}v_{i-2}^{j-1} + \frac{1}{2}v_{i-1}^{j-1} + \frac{5}{8}v_i^{j-1} & \text{if } \mathcal{S}_i^{\hat{v}^{j-1}} = \{i-2, i-1, i\}, \\ \frac{1}{8}v_{i-1}^{j-1} + v_i^{j-1} - \frac{1}{8}v_{i+1}^{j-1} & \text{if } \mathcal{S}_i^{\hat{v}^{j-1}} = \{i-1, i, i+1\}, \\ \frac{11}{8}v_i^{j-1} - \frac{1}{2}v_{i+1}^{j-1} + \frac{1}{8}v_{i+2}^{j-1} & \text{if } \mathcal{S}_i^{\hat{v}^{j-1}} = \{i, i+1, i+2\}. \end{cases} \tag{3.5}$$

To obtain $(P_j v^{j-1})_{2i}$ we simply use (2.4).

For $R = 4$, there are five different possibilities:

(3.6)

$$(P_j v^{j-1})_{2i-1} \begin{cases} -\frac{7}{128}v_{i-4}^{j-1} + \frac{19}{64}v_{i-3}^{j-1} + \frac{11}{16}v_{i-2}^{j-1} - \frac{61}{64}v_{i-1}^{j-1} + \frac{63}{128}v_i^{j-1}, & z_i = i-4, \\ \frac{3}{128}v_{i-3}^{j-1} - \frac{9}{64}v_{i-2}^{j-1} + \frac{13}{32}v_{i-1}^{j-1} + \frac{49}{64}v_i^{j-1} - \frac{7}{128}v_{i+1}^{j-1}, & z_i = i-3, \\ -\frac{3}{128}v_{i-2}^{j-1} + \frac{11}{64}v_{i-1}^{j-1} + v_i^{j-1} - \frac{11}{64}v_{i+1}^{j-1} + \frac{3}{128}v_{i+2}^{j-1}, & z_i = i-2, \\ \frac{7}{128}v_{i-1}^{j-1} + \frac{79}{64}v_i^{j-1} - \frac{13}{32}v_{i+1}^{j-1} + \frac{9}{64}v_{i+2}^{j-1} - \frac{3}{128}v_{i+3}^{j-1}, & z_i = i-1, \\ \frac{193}{128}v_i^{j-1} - \frac{61}{64}v_{i+1}^{j-1} + \frac{11}{16}v_{i+2}^{j-1} - \frac{19}{64}v_{i+3}^{j-1} + \frac{7}{128}v_{i+4}^{j-1}, & z_i = i, \end{cases}$$

where $(\mathcal{S}_i^{\hat{v}^{j-1}}) = \{z_i, \ldots, z_i + R\}$.

3.3. Other adaptive prediction schemes. In [7], Claypoole et al. propose reducing the length of the prediction filters near edges in order to avoid using data across image edges in the prediction step. The main goal in their adaptive scheme is to maintain high accuracy away from edges while avoiding large errors in their vicinity. The nonlinearity thus introduced is similar to that considered in ENO-type prediction operators; i.e., at each location and scale a nonlinear selection function is used to choose a linear predictor from a finite set. In this case, the filters in the pool have different lengths and, thus, different orders of accuracy.

A more involved nonlinear prediction scheme is obtained by considering the ENO-SR (ENO with subcell resolution) technique. This technique, introduced in [13], elaborates on the ENO interpolation in order to account for discontinuities within a cell. The nonlinearity in an ENO-SR prediction operator also derives from the use of a nonlinear procedure (slightly more involved than the one in ENO predictors) that selects a particular filter from a finite pool of filters of the same length (and order). When the polynomial pieces are of degree R, the pool is composed of $R + 3$ different filters. We refer the reader to [4, 2, 1] for more information on the ENO-SR technique and its application in multiscale transformations.

Remark 3.2 applies also to all these adaptive prediction schemes.

4. Synchronization and stability in nonlinear multiscale transformations. In lossy multiresolution-based compression algorithms, the original signal v^L is transformed and processed in such a way that the data available in the decoder, $\hat{v}_d = (\hat{v}^0, \hat{d}^1, \ldots, \hat{d}^L)$, is no longer equivalent to v^L. Stability ensures that the difference $v^L - M^{-1}\hat{v}_d$ can be adequately controlled through the compression parameters.

It is well known (see [7] and the references therein) that the cascading of quantization errors, due to incorrect filter choices in nonlinear prediction schemes, prevents

one from obtaining stable data-dependent multiscale transformations. In order to avoid this phenomenon, Claypoole et al. argue that the nonlinear decisions made by the prediction scheme at a particular location and scale must be [illegible] both in the encoder and in the decoder; i.e., [illegible] is a necessary requirement for stability.

Only $\hat{v}_d = (\hat{v}^0, \hat{d}^1, \ldots, \hat{d}^L)$ is available in the decoder; therefore the nonlinear decisions made there are based on [illegible] $\hat{v}^j = M^{-1}(\hat{v}^0, \hat{d}^1, \ldots, \hat{d}^j)$. Synchronization will follow only if (a) the [illegible] data $\hat{v}^j = M^{-1}(\hat{v}^0, \hat{d}^1, \ldots, \hat{d}^j)$ are also made available to the encoder; and (b) the nonlinear decisions involved in the application of the prediction operator in the direct transformation are based on $\hat{v}^j$ instead of on the original low-pass data $v^j = D^j D^{j+1} \ldots D^L v^L$.

As explained in [7], the usual SPU-lifting routine does not allow automatic synchronization. The authors switch then to the update-first strategy, i.e., to Harten's framework, and start the prediction [illegible], i.e., from the lowest resolution level. Working from coarse to fine scales makes it possible to have the processed data $\hat{v}^j$ already available in the encoder and ensure synchronization. Claypoole et al. relate their synchronized nonlinear multiscale transformation with the Laplacian pyramid of Burt and Adelson [6] and the work of Gerek and Cetin [11], but they apparently did not realize the intrinsic similarity between their approach and the EC algorithms developed by Harten in [14].

In order to establish the specific similarities and differences between Harten's EC algorithm and the synchronized transformation in [7], we now give a schematic description of the EC-modified transformation for the cell-average framework (see [1]),

(4.1)

$$v^L \longrightarrow \hat{v}_d = \widetilde{M}_{\bar{\varepsilon}} v^L = \{\hat{v}^0, \hat{d}^1, \ldots, \hat{d}^L\} \qquad \left\{ \begin{array}{l} \text{for} \quad j = L, \ldots, 1 \\ \quad v^{j-1} = D^j v^j \\ \text{end} \\ \hat{v}^0 = \mathbf{pr}(v^0, \varepsilon_0) \\ \text{for} \quad j = 1, \ldots, L \\ \quad [\hat{v}^j, \hat{d}^j] = \mathbf{ECj}(\hat{v}^{j-1}, v^{j-1}, v^j; \varepsilon_j) \\ \text{end.} \end{array} \right.$$

The first step consists in iterating the linear procedure (the decimation, or, equivalently, the update, operator) to obtain v^0, which is subsequently processed to $\hat{v}^0$ and sent. Then the processed coefficients $\hat{d}^j$ are computed in reverse order [illegible], together with [illegible] j-level values $\hat{v}^j$, in such a way that $\hat{v}^j = M^{-1}(\hat{v}^0, \hat{d}^1, \ldots, \hat{d}^j)$.

Again, to establish similarities and differences between the cell-average [illegible] routine (**ECj** henceforth) in [14, 1] and in [7], we provide next a schematic description of **ECj**:

(4.2)

$$[\hat{v}^j, \hat{d}^j] = \mathbf{ECj}(\hat{v}^{j-1}, v^{j-1}, v^j; \varepsilon_j) \left\{ \begin{array}{l} \tilde{d}^j = \mathbf{pe}(v^j, v^{j-1}, \hat{v}^{j-1}) \\ \hat{d}^j = \mathbf{proc}(\tilde{d}^j, \varepsilon_j) \\ \text{for } i = 1, \ldots, N_{j-1} \\ \quad \hat{v}^j_{2i-1} = (P_{\hat{v}^{j-1}} \hat{v}^{j-1})_{2i-1} + \hat{d}^j_i; \\ \quad \hat{v}^j_{2i} = (P_{\hat{v}^{j-1}} \hat{v}^{j-1})_{2i} - \hat{d}^j_i \quad (\equiv 2\hat{v}^{j-1}_i - \hat{v}^j_{2i-1}); \\ \text{end.} \end{array} \right.$$

First, it proceeds by computing a set of $\tilde{d}^j = \mathbf{pe}(v^j, v^{j-1}, \hat{v}^{j-1})$. This set has the same cardinality as d^j in the direct transformation M and should represent the error in predicting v^j from $\hat{v}^{j-1}$ at level $j-1$. The set $\tilde{d}^j$ is subsequently **proc**essed to obtain $\hat{d}^j$.

The modified j-level values $\hat{v}^j$ are computed in such a way that $\hat{v}^j = M^{-1}(\hat{v}^0, \hat{d}^1, \ldots, \hat{d}^j)$. This is easily checked by adding up the two equalities in the for-loop of (4.2) and taking into account the consistency relation (2.4),

$$\hat{v}^j_{2i-1} + \hat{v}^j_{2i} = (P_{\hat{v}^{j-1}}\hat{v}^{j-1})_{2i-1} + (P_{\hat{v}^{j-1}}\hat{v}^{j-1})_{2i} = 2\hat{v}^{j-1}_i.$$

This implies that $D^j\hat{v}^j = \hat{v}^{j-1}$, $j = 1, \ldots, L$. Moreover,

$$\hat{d}^j_i = \hat{v}^j_{2i-1} - (P_{\hat{v}^{j-1}}\hat{v}^{j-1})_{2i-1} = -(\hat{v}^j_{2i} - (P_{\hat{v}^{j-1}}\hat{v}^{j-1})_{2i});$$

hence, the detail coefficients, $\{\hat{d}^j\}_{j=1}^L$, are the prediction errors for the data, $\{\hat{v}^j\}_{j=0}^L$. As a consequence, we have

$$M\hat{v}^j = (\hat{v}^0, \hat{d}^1, \ldots, \hat{d}^j) \equiv M^{-1}(\hat{v}^0, \hat{d}^1, \ldots \hat{d}^j) = \hat{v}^j, \qquad j = 1, \ldots, L. \tag{4.3}$$

By embedding the quantizer into the transformation, as specified in **ECj**, synchronization follows: are used in both the encoder and the decoder in the decision step of the nonlinear prediction operator at each resolution level. This guarantees that exactly the same filter will be used at each location and resolution level both in the encoder and in the decoder.

We note the fact that synchronization in $\widetilde{M}_{\bar{\varepsilon}}$ is achieved of the definition of $\tilde{d}^j$, and that it will be attained independently of the source of the perturbation, as long as the perturbation can be correctly modeled in the encoder by the **proc** routine. For example $\hat{d}^j$ might represent the scale coefficients available in the decoder after compression effects and transmission errors have been taken into account. Clearly, random transmission errors will inevitably lead to a loss of synchronization.

4.1. Harten versus lifting: The definition of the prediction residuals. Written in the form (4.1) and (4.2), it is apparent that the difference between Harten's EC algorithms in [14] and the nonlinear lifting scheme in [7] lies precisely in the definition of $\tilde{d}^j$.

In [14], $\tilde{d}^j_i$ is defined as the average of odd and even prediction errors (**hc** stands for Harten's choice):

$$\tilde{d}^j = \mathbf{hc}(v^j, \hat{v}^{j-1}) \quad \begin{cases} \text{for } i = 1, \ldots, N_{j-1} \\ \quad \tilde{d}^j_i = \dfrac{1}{2}[(v^j_{2i-1} - (P_{\hat{v}^{j-1}}\hat{v}^{j-1})_{2i-1}) - (v^j_{2i} - (P_{\hat{v}^{j-1}}\hat{v}^{j-1})_{2i})] \\ \text{end,} \end{cases} \tag{4.4}$$

while in [7] two different definitions are considered:

$$\tilde{d}^j = \mathbf{pe1}(v^j, \hat{v}^{j-1}) \quad \begin{cases} \text{for } i = 1, \ldots, N_{j-1} \\ \quad \tilde{d}^j_i = v^j_{2i-1} - (P_{\hat{v}^{j-1}}\hat{v}^{j-1})_{2i-1} \\ \text{end,} \end{cases} \tag{4.5}$$

$$\tilde{d}^j = \mathbf{pe2}(v^j, v^{j-1}, \hat{v}^{j-1}) \quad \begin{cases} \text{for } i = 1, \ldots, N_{j-1} \\ \quad \tilde{d}^j_i = v^j_{2i-1} - (P_{\hat{v}^{j-1}}v^{j-1})_{2i-1} \\ \text{end.} \end{cases} \tag{4.6}$$

Notice that **pe1** and **hc** do not use the raw low-pass data v^{j-1}. Indeed the choice **pe1** is discarded in [7] and substituted by **pe2** because numerical experiments revealed that, even for moderate quantization levels, the prediction errors for **pe1** tend to be large, regardless of the prediction filter, thus reducing the compression potential of the adaptive algorithm.

Even though encoder/decoder synchronization is achieved independently of the definition of $\tilde{d}^j$, the following basic consistency requirement must be imposed: when no processing takes place, i.e., $\varepsilon_j = 0$ for all j, the modified transformation produces exactly the same outcome as the direct transformation, i.e., $\widetilde{M}_{\bar{0}} = M = MQ_{\bar{0}}$. If no compression is needed, the modified transformation becomes an alternative way to compute the multiscale representation Mv^L: it is simply a reordering in the computation of the detail coefficients.

PROPOSITION 4.1. [illegible] $\tilde{d}^j$ [illegible] (4.4), (4.5) [illegible] (4.6) [illegible] $\widetilde{M}_{\bar{\epsilon}}$ [illegible] $\widetilde{M}_{\bar{0}} = M$ [illegible]. For **pe1**, we easily deduce that

$$v^j_{2i-1} - \hat{v}^j_{2i-1} = \tilde{d}^j_i - \hat{d}^j_i, \qquad v^j_{2i} - \hat{v}^j_{2i} = 2(v^{j-1}_i - \hat{v}^{j-1}_i) - (\tilde{d}^j_i - \hat{d}^j_i). \tag{4.7}$$

For **hc** we have instead

$$\begin{aligned}(v^j - \hat{v}^j)_{2i-1} - (v^j - \hat{v}^j)_{2i} &= 2(\tilde{d}^j - \hat{d}^j)_i,\\ (v^j - \hat{v}^j)_{2i-1} + (v^j - \hat{v}^j)_{2i} &= 2(v^{j-1} - \hat{v}^{j-1})_i,\end{aligned}$$

and hence

$$v^j_{2i-1} - \hat{v}^j_{2i-1} = (v^{j-1}_i - \hat{v}^{j-1}_i) + (\tilde{d}^j_i - \hat{d}^j_i), \qquad v^j_{2i} - \hat{v}^j_{2i} = (v^{j-1}_i - \hat{v}^{j-1}_i) - (\tilde{d}^j_i - \hat{d}^j_i). \tag{4.8}$$

For **pe2**, we easily deduce from (4.6)

$$\begin{aligned}v^j_{2i-1} - \hat{v}^j_{2i-1} &= (P_{\hat{v}^{j-1}} v^{j-1})_{2i-1} - (P_{\hat{v}^{j-1}} \hat{v}^{j-1})_{2i-1} + \tilde{d}^j_i - \hat{d}^j_i,\\ v^j_{2i} - \hat{v}^j_{2i} &= (P_{\hat{v}^{j-1}} v^{j-1})_{2i} - (P_{\hat{v}^{j-1}} \hat{v}^{j-1})_{2i} - (\tilde{d}^j_i - \hat{d}^j_i).\end{aligned} \tag{4.9}$$

If no processing takes place, i.e., $\hat{v}^0 = v^0$ and $\hat{d}^j = \tilde{d}^j$, then either (4.7), (4.8), or (4.9) directly imply that $v^j_{2i-1} = \hat{v}^j_{2i-1}$ and $v^j_{2i} = \hat{v}^j_{2i}$. Therefore $\hat{v}^j = v^j$ for all j, i.e., $\widetilde{M}_{\bar{0}} \equiv M$ in all three cases. □

Relations (4.7), (4.8), (4.9) hold the key to the behavior of the resulting adaptive multiscale transformation. It is important to recall that the detail coefficients $\tilde{d}^j$ should correctly represent the error in predicting "raw" data at level j, from "known" data at level $j-1$. For **pe1** and **hc** the prediction error is $e^j = v^j - P_{\hat{v}^{j-1}} \hat{v}^{j-1}$. Consistency of the prediction operator implies $D^j e^j = v^{j-1} - \hat{v}^{j-1}$, and hence the even and odd components no longer satisfy $e^j_{2i} = -e^j_{2i-1}$. Thus, the **pe1** choice, which keeps only the odd prediction residuals, does not correctly represent the full prediction error; this, most likely, is the cause of the bad behavior observed in the numerical experiments reported in [7] when this choice of scale coefficients was implemented. On the other hand, in Harten's choice, **hc**, $\tilde{d}^j$ is computed as the average between the even and odd components of e^j, hence yielding a better representation of the the full prediction error.

For the choice in (4.6), the prediction error is $e^j = v^j - P_{\hat{v}^{j-1}} v^{j-1}$; hence as a consequence of the consistence of the prediction operator,

$$(P_{\hat{v}^{j-1}} v^{j-1})_{2i-1} + (P_{\hat{v}^{j-1}} v^{j-1})_{2i} = 2v^{j-1}_i = v^j_{2i-1} + v^j_{2i}$$

so that

$$v^j_{2i-1} - (P_{\hat{v}^{j-1}} v^{j-1})_{2i-1} = -(v^j_{2i} - (P_{\hat{v}^{j-1}} v^{j-1})_{2i}) \tag{4.10}$$

and we recover $e^j_{2i} = -e^j_{2i-1}$.

4.1. Notice that, in light of the observation above on the prediction error $v^j - P_{\hat{v}^{j-1}} v^{j-1}$, a fourth obvious alternative becomes equivalent to **pe2**

$$\tilde{d}^j_i = \frac{1}{2}[(v^j_{2i-1} - (P_{\hat{v}^{j-1}} v^{j-1})_{2i-1}) - (v^j_{2i} - (P_{\hat{v}^{j-1}} v^{j-1})_{2i})] = v^j_{2i-1} - (P_{\hat{v}^{j-1}} v^{j-1})_{2i-1}.$$

In [7], the authors provide no specific error bounds on the compression error. We shall exploit the connection between the synchronized nonlinear multiscale transformation in [7] and Harten's cell-average EC algorithms to demonstrate that the claim that stability follows from synchronization is essentially correct, at least for the type of nonlinear prediction schemes considered in section 3.

5. Error estimates for compression schemes based on the modified multiscale transformation. The goal in this section is to obtain precise estimates of the error between the original signal v^L and the signal obtained from decoding its compressed multiscale representation, i.e., $\hat{v}^L = M^{-1}\widetilde{M}_{\bar{\varepsilon}} v^L$, with $\widetilde{M}_{\bar{\epsilon}}$ as specified in (4.1)–(4.2).

For ease of reference we fix the notation as follows:

- UFL1 refers to $\widetilde{M}_{\bar{\epsilon}}$ with **pe** = **pe1** as specified in (4.5). This is the first (discarded) choice for scale coefficients in [7], and hence the acronym stands for update-first lifting, choice 1.
- HC refers to $\widetilde{M}_{\bar{\epsilon}}$ with **pe** = **hc** as specified in (4.4). This corresponds to Harten's original choice of scale coefficients in his EC algorithms.
- UFL2 refers to $\widetilde{M}_{\bar{\epsilon}}$ with **pe** = **pe2** as specified in (4.6). This corresponds to the preferred choice of scale coefficients in [7], and the acronym stands for update-first lifting, choice 2.

For the remainder of the paper we use the same notation as in [14, 15, 1]: Given a $v^j \in V^j$, a finite-dimensional signal with N_j samples, we define

$$\|v^j\|_\infty = \sup_i |v^j_i|; \quad \|v^j\|_1 = \frac{1}{N_j}\left(\sum_i^{N_j} |v^j_i|\right); \quad \|v^j\|_2^2 = \frac{1}{N_j}\left(\sum_i^{N_j} |v^j_i|^2\right) \tag{5.1}$$

and give two types of error estimates. We obtain first [illegible] error estimates, i.e., error estimates obtained in terms of the differences $\|\hat{d}^i - \tilde{d}^i\|$, $\|\hat{v}^0 - v^0\|$, which can be computed as the algorithm proceeds. They provide estimates on the accumulated compression error $||v^j - \hat{v}^j||$ at each resolution level and, hence, a bound on the global compression error already at the encoder. From these, we can easily deduce error bounds in terms of the threshold parameters $\bar{\varepsilon} = (\varepsilon_0, \ldots, \varepsilon_L)$ which could be used to ensure a certain accuracy in the decoded representation a priori and, hence, are referred to as [illegible] error bounds.

The compression error bounds obtained next reveal that Harten's choice seems to be optimal: distributing evenly the quantization error between odd and even locations, as done in **hc**, allows us to obtain more accurate global bounds on the compression error. It is a remarkable fact, which Harten did not notice, that the [illegible] of the 2-norm of the compression error can be computed in the encoder (see Remark 5.1 and [1]).

5.1. A posteriori error bounds. The structure of the modified transformation allows us to prove that estimates of the form

$$||v^L - \hat{v}^L|| \le C_0||v^0 - \hat{v}^0|| + \sum_{j=1}^{L} C_j||\tilde{d}^j - \hat{d}^j||$$

hold for the UFL1, HC, and UFL2 algorithms. For the latter, the estimates are obtained provided the prediction operators satisfy certain uniform bounds. The constants depend on the choice made for $\tilde{d}^j$ and the norm considered.

PROPOSITION 5.1. [illegible] v^L [illegible] $\hat{v}_d = \widetilde{M}_{\bar{\epsilon}} v^L = (\hat{v}^0, \hat{d}^1, \dots, \hat{d}^L)$ [illegible] $\hat{v}^L = M^{-1}\hat{v}_d$ [illegible] M^{-1} [illegible] (2.5) [illegible] $\widetilde{M}_{\bar{\epsilon}}$ [illegible] (4.1)–(4.2) [illegible] (4.5), (4.4), [illegible] (4.6) [illegible]

$$\|v^L - \hat{v}^L\|_p \le C_{0,p}\|v^0 - \hat{v}^0\|_p + \sum_{j=1}^{L} C_{j,p}\|\tilde{d}^j - \hat{d}^j\|_p, \quad p = \infty, 1, \tag{5.2}$$

$$\|v^L - \hat{v}^L\|_2^2 \le C_{0,2}\|v^0 - \hat{v}^0\|_2^2 + \sum_{j=1}^{L} C_{j,2}\|\tilde{d}^j - \hat{d}^j\|_2^2, \tag{5.3}$$

[illegible]

$C_{0,p}$	[illegible] 1	[illegible]	[illegible] 2
$p = \infty$	2^L	1	$\mathcal{B}_\infty^L$
$p = 1$	1	1	$\mathcal{B}_1^L$
$p = 2$	3^L	1	$2^L\mathcal{B}_2^{2L}$

$C_{j,p}$	[illegible] 1	[illegible]	[illegible] 2
$p = \infty$	2^{L-j}	1	$\mathcal{B}_\infty^{L-j}$
$p = 1$	1	1	$\mathcal{B}_1^{L-j}$
$p = 2$	$2 \cdot 3^{L-j}$	1	$2(2\mathcal{B}_2^2)^{L-j}$

[illegible] $\mathcal{B}_p$ [illegible] j [illegible]

$$||P_{\hat{v}^j}(w^j)||_p \le \mathcal{B}_p||w^j||_p, \quad \forall \hat{v}^j \in V^j. \tag{5.4}$$

[illegible] $P_{\hat{v}^j}$ We shall establish two-level estimates in all three cases. The global estimates are easily found by setting $j = L$ and applying recursively the two-level estimates.

Let us consider first the UFL1 choice. At each level, we use (4.7) to obtain

$$\begin{aligned}
&\max\{|v_{2i-1}^j - \hat{v}_{2i-1}^j|, |v_{2i}^j - \hat{v}_{2i}^j|\} \le 2|v_i^{j-1} - \hat{v}_i^{j-1}| + |\tilde{d}_i^j - \hat{d}_i^j|,\\
&(|v_{2i-1}^j - \hat{v}_{2i-1}^j| + |v_{2i}^j - \hat{v}_{2i}^j|) \le (2|\tilde{d}_i^j - \hat{d}_i^j| + 2|v_i^{j-1} - \hat{v}_i^{j-1}|),\\
&(v_{2i-1}^j - \hat{v}_{2i-1}^j)^2 + (v_{2i}^j - \hat{v}_{2i}^j)^2 = 4(v_i^{j-1} - \hat{v}_i^{j-1})^2 + 2(\tilde{d}_i^j - \hat{d}_i^j)^2 - 4(v_i^{j-1} - \hat{v}_i^{j-1})(\tilde{d}_i^j - \hat{d}_i^j)\\
&\qquad\qquad \le 6(v_i^{j-1} - \hat{v}_i^{j-1})^2 + 4(\tilde{d}_i^j - \hat{d}_i^j)^2.
\end{aligned}$$

Hence

$$\|v^j - \hat{v}^j\|_\infty \le 2\|v^{j-1} - \hat{v}^{j-1}\|_\infty + \|\tilde{d}^j - \hat{d}^j\|_\infty,$$

$$\|v^j - \hat{v}^j\|_1 = \frac{1}{N_j}\sum_{i=1}^{N_{j-1}}(|v^j_{2i-1} - \hat{v}^j_{2i-1}| + |v^j_{2i} - \hat{v}^j_{2i}|)$$
$$\leq \frac{1}{2N_{j-1}}\sum_{i=1}^{N_{j-1}}(2|\tilde{d}^j_i - \hat{d}^j_i| + 2|v^{j-1}_i - \hat{v}^{j-1}_i|) = \|\tilde{d}^j - \hat{d}^j\|_1 + \|v^{j-1} - \hat{v}^{j-1}\|_1,$$
$$\|v^j - \hat{v}^j\|_2^2 = \frac{1}{N_j}\sum_{i=1}^{N_{j-1}}(|v^j_{2i-1} - \hat{v}^j_{2i-1}|^2 + |v^j_{2i} - \hat{v}^j_{2i}|^2)$$
$$\leq 3\|v^{j-1} - \hat{v}^{j-1}\|_2^2 + 2\|\tilde{d}^j - \hat{d}^j\|_2^2.$$

For Harten's original choice, HC, we use the equality $\max(|a|,|b|) = (|a+b|+|a-b|)/2$ together with relation (4.8) to obtain

$$\max\{|v^j_{2i-1} - \hat{v}^j_{2i-1}|, |v^j_{2i} - \hat{v}^j_{2i}|\} = |\tilde{d}^j_i - \hat{d}^j_i| + |v^{j-1}_i - \hat{v}^{j-1}_i|,$$

$$|v^j_{2i-1} - \hat{v}^j_{2i-1}| + |v^j_{2i} - \hat{v}^j_{2i}| = \max(|\tilde{d}^j_i - \hat{d}^j_i|, |v^{j-1}_i - \hat{v}^{j-1}_i|),$$

$$|v^j_{2i-1} - \hat{v}^j_{2i-1}|^2 + |v^j_{2i} - \hat{v}^j_{2i}|^2 = (2|\tilde{d}^j_i - \hat{d}^j_i|^2 + 2|v^{j-1}_i - \hat{v}^{j-1}_i|^2).$$

Hence

$$\|v^j - \hat{v}^j\|_\infty \;\leq\; \|v^{j-1} - \hat{v}^{j-1}\|_\infty + \|\tilde{d}^j - \hat{d}^j\|_\infty,$$

$$\|v^j - \hat{v}^j\|_1 = \frac{1}{N_{j-1}}\sum_{i=1}^{N_{j-1}}\max(|\tilde{d}^j_i - \hat{d}^j_i|, |v^{j-1}_i - \hat{v}^{j-1}_i|)$$
$$\leq \frac{1}{N_{j-1}}\sum_{i=1}^{N_{j-1}}|\tilde{d}^j_i - \hat{d}^j_i| + |v^{j-1}_i - \hat{v}^{j-1}_i| = \|\tilde{d}^j - \hat{d}^j\|_1 + \|v^{j-1} - \hat{v}^{j-1}\|_1,$$

$$\|v^j - \hat{v}^j\|_2^2 = \frac{1}{2N_{j-1}}\sum_{i=1}^{N_{j-1}}(2|\tilde{d}^j_i - \hat{d}^j_i|^2 + 2|v^{j-1}_i - \hat{v}^{j-1}_i|^2) = \|\tilde{d}^j - \hat{d}^j\|_2^2 + \|v^{j-1} - \hat{v}^{j-1}\|_2^2,$$

from which (5.5) follows.

We consider lastly the UFL2 choice. Notice that $P_{\hat{v}^j}$ acts linearly on the data, and hence we have

$$(P_{\hat{v}^{j-1}}v^{j-1})_k - (P_{\hat{v}^{j-1}}\hat{v}^{j-1})_k = (P_{\hat{v}^{j-1}}(v^{j-1} - \hat{v}^{j-1}))_k$$

so that under the hypothesis of the proposition

$$||P_{\hat{v}^{j-1}}v^{j-1} - P_{\hat{v}^{j-1}}\hat{v}^{j-1}||_p \leq \mathcal{B}_p||v^{j-1} - \hat{v}^{j-1}||_p.$$

From (4.9) we get

$$\max_i|v^j_{2i-1} - \hat{v}^j_{2i-1}| \leq \max_i|\tilde{d}^j_i - \hat{d}^j_i| + \max_i|(P_{\hat{v}^{j-1}}v^{j-1})_{2i-1} - (P_{\hat{v}^{j-1}}\hat{v}^{j-1})_{2i-1}|,$$

$$\max_i|v^j_{2i} - \hat{v}^j_{2i}| \leq \max_i|\tilde{d}^j_i - \hat{d}^j_i| + \max_i|(P_{\hat{v}^{j-1}}v^{j-1})_{2i} - (P_{\hat{v}^{j-1}}\hat{v}^{j-1})_{2i}|.$$

Hence

$$||v^j - \hat{v}^j||_\infty \leq ||\tilde{d}^j - \hat{d}^j||_\infty + \mathcal{B}_\infty ||v^{j-1} - \hat{v}^{j-1}||_\infty,$$

$$\begin{aligned}\|v^j - \hat{v}^j\|_1 &\leq \frac{1}{2N_{j-1}} \sum_{i=1}^{N_{j-1}} 2|\tilde{d}_i^j - \hat{d}_i^j| + \frac{1}{N_j} \sum_{i=1}^{N_j} |(P_{\hat{v}^{j-1}} v^{j-1})_i - (P_{\hat{v}^{j-1}} \hat{v}^{j-1})_i| \\ &= \|\tilde{d}^j - \hat{d}^j\|_1 + \|P_{\hat{v}^{j-1}}(v^{j-1} - \hat{v}^{j-1})\|_1 \leq \|\tilde{d}^j - \hat{d}^j\|_1 + \mathcal{B}_1 \|v^{j-1} - \hat{v}^{j-1}\|_1,\end{aligned}$$

$$\begin{aligned}\|v^j - \hat{v}^j\|_2^2 &\leq \frac{1}{N_j} \sum_{i=1}^{N_{j-1}} 4(\tilde{d}_i^j - \hat{d}_i^j)^2 + 2\frac{1}{N_j} \sum_{i=1}^{N_j} ((P_{\hat{v}^{j-1}} v^{j-1})_i - (P_{\hat{v}^{j-1}} \hat{v}^{j-1})_i)^2 \\ &= 2\|\tilde{d}^j - \hat{d}^j\|_2^2 + 2\|P_{\hat{v}^{j-1}}(v^{j-1} - \hat{v}^{j-1})\|_2^2 \\ &\leq 2\|\tilde{d}^j - \hat{d}^j\|_2^2 + 2(\mathcal{B}_2)^2 \|v^{j-1} - \hat{v}^{j-1}\|_2^2. \qquad \square\end{aligned}$$

[illegible] 5.1. Notice that for HC (and [illegible] for HC) we obtain

$$\|v^L - \hat{v}^L\|_2^2 = \|v^0 - \hat{v}^0\|_2^2 + \sum_{j=1}^{L} \|\tilde{d}^j - \hat{d}^j\|_2^2. \tag{5.5}$$

Therefore, the 2-compression error $||v^L - M^{-1}\widetilde{M}_\varepsilon v^L||_2$ can be exactly known at the encoder.

Uniform bounds as described in (5.4) can be computed for all the nonlinear prediction operators of the type considered in section 3, as well as those in [7], as follows: Recall that for a fixed v^j, the operator $P_{v^j} : V^j \to V^{j+1}$ is linear. Since V^j is a linear vector space of dimension N_j, P_{v^j} can be reinterpreted as a linear mapping $P_{v^j} : \mathbb{R}^{N_j} \longrightarrow \mathbb{R}^{N_{j+1}}$, and hence it can be represented as an $N_{j+1} \times N_j$ matrix. To compute the uniform bounds, we simply use standard linear algebra results for matrix norms. We take into account that the relation between the p-norms in $\mathbb{R}^{N_j}$ and the p-norms defined in (5.1) is as follows:

(5.6)
$$||w^j||_{l^\infty(\mathbb{R}^{N_j})} = ||w^j||_\infty, \qquad ||w^j||_{l^1(\mathbb{R}^{N_j})} = N_j ||w^j||_1, \qquad ||w^j||^2_{l^2(\mathbb{R}^{N_j})} = N_j ||w^j||_2^2.$$

Proposition 5.2. [illegible] 5.1 [illegible]

$$\mathcal{B}_\infty = \sup\{||P_{v^j}||_{l^\infty(\mathbb{R}^{N_j}, \mathbb{R}^{N_{j+1}})}, \quad v^j \in \mathbb{R}^{N_j}\}, \tag{5.7}$$

$$\mathcal{B}_1 = \frac{1}{2} \sup\{||P_{v^j}||_{l^1(\mathbb{R}^{N_j}, \mathbb{R}^{N_{j+1}})}, \ v^j \in \mathbb{R}^{N_j}\}, \tag{5.8}$$

$$\mathcal{B}_2 = \frac{1}{\sqrt{2}} \sup\{||P_{v^j}||_{l^2(\mathbb{R}^{N_j}, \mathbb{R}^{N_{j+1}})}, \ v^j \in \mathbb{R}^{N_j}\}. \tag{5.9}$$

[illegible] Since $P_{v^j} : \mathbb{R}^{N_j} \longrightarrow \mathbb{R}^{N_{j+1}}$ is a linear mapping, we have

$$||P_{v^j} w^j||_{l^p(\mathbb{R}^{N_{j+1}})} \leq ||P_{v^j}||_{l^p(\mathbb{R}^{N_j}, \mathbb{R}^{N_{j+1}})} ||w^j||_{l^p(\mathbb{R}^{N_j})}.$$

Using the relation between the p-norms in $\mathbb{R}^{N_j}$ and the p-norms in V^j, we easily obtain the result. $\square$

TABLE 5.1
Coefficients $\mathcal{B}_p$, $p = \infty, 1, 2$, for various prediction operators.

Prediction	$\mathcal{B}_\infty$	$\mathcal{B}_1$	$\mathcal{B}_2$
Linear-2	1.25	1.25	1.0308
ENO-2	2	2.25	1.5416
ENO-SR-2	7	11	6.0725
Linear-4	1.3907	1.3907	1.0614
ENO-4	3.5	4.8750	3.009
ENO-SR-4	19.25	25.3438	18.4715

For data-dependent prediction schemes of the type considered in [7] and in this paper, P_{v^j} can be represented as a banded matrix whose rows are composed of a finite number of nonzero entries. The rows depend on the data v^j but can only be chosen from a finite pool; e.g., for ENO predictors of degree R, the pool is composed of $R+1$ filters; for ENO-SR predictors the pool has $R+3$ filters (see section 3).

A simple MATLAB program can grind up the $\mathcal{B}_p$ uniform bounds. Some of these bounds are compiled in Table 5.1. We have also included the data-independent biorthogonal wavelet prediction operator (linear-R). In this case $P_{\hat{v}^{j-1}} = P_j$ is a specific $N_{j+1} \times N_j$ banded matrix and $\mathcal{B}_p = \|P_j\|_p$.

5.2. "A priori" error bounds. In applications, the processing step for the detail coefficients is carried out in such a way that $||\tilde{d}^j - \hat{d}^j|| \le \varepsilon_j$, $||v^0 - \hat{v}^0|| \le \varepsilon_0$ is ensured for a predetermined norm. Then Proposition 5.1 leads to simple error bounds that depend only on the processing parameters ε_j and that can be used to guarantee, a priori, a certain accuracy in the decoded signal.

COROLLARY 5.3. [illegible] 5.1 [illegible]

$$||\tilde{d}^j - \hat{d}^j|| \le \varepsilon_j, \quad ||v^0 - \hat{v}^0|| \le \varepsilon_0,$$

[illegible]
[illegible] 1 [illegible]

$$||v^L - \hat{v}^L||_\infty \le \sum_{j=0}^{L} 2^{L-j}\varepsilon_j, \quad ||v^L - \hat{v}^L||_1 \le \sum_{j=0}^{L} \varepsilon_j, \quad ||v^L - \hat{v}^L||_2^2 \le 3^L\varepsilon_0^2 + 2\sum_{j=1}^{L} 3^{L-j}\varepsilon_j^2.$$

[illegible]

$$||v^L - \hat{v}^L||_p \le \sum_{j=0}^{L} \varepsilon_j, \quad p = \infty, 1, \qquad ||v^L - \hat{v}^L||_2^2 \le \sum_{j=0}^{L} \varepsilon_j^2.$$

[illegible] 2 [illegible]

$$||v^L - \hat{v}^L||_p \le \sum_{j=0}^{L} (\mathcal{B}_p)^{L-j}\varepsilon_j, \quad p = \infty, 1, \qquad ||v^L - \hat{v}^L||_2^2 \le \sum_{j=0}^{L} (2(\mathcal{B}_2)^2)^{L-j}\varepsilon_j^2.$$

This corollary suggests appropriate processing strategies to ensure a target accuracy in the decompressed signal. In particular, choosing the processing parameters of the form

$$\varepsilon_j = \varepsilon/q^{L-j},$$

with q a given number, leads to a priori estimates of the form

$$||v^L - \hat{v}^L|| \leq \varepsilon \sum_{j=0}^{L} \alpha^{L-j},$$

where, according to Corollary 5.3, α depends on the specific strategy being used. In order to have an a priori error estimate independent of L we need $\alpha < 1$. It is worth noting that $q = 2$ is "appropriate" for the HC strategy, while the "appropriate" q for the UFL2 strategy would depend on the quantities $\mathcal{B}_p$.

6. Tensor product algorithms in two dimensions. In the 2D tensor-product extension of the algorithms described in section 4, synchronization follows, exactly as in the 1D case, from the fact that $M^{-1}(\hat{v}^0, \hat{d}^1, \ldots, \hat{d}^j) = \hat{v}^j$, $j = 1, \ldots, L$. This is due to the structure of the **ECj** routine, independently of the definition for $\tilde{d}^j$. It should be remarked that, due to the intertwined structure of the error control algorithm, the details, $\tilde{d}^j$, will be different (but of the same order of magnitude) if one acts first on the rows or first on the columns. This, however, does not affect the synchrony between the prediction in the encoder and decoder, or the error bounds obtained.

A 2D tensor-product version of the modified transformation $\widetilde{M}_{\bar{\varepsilon}}$ together with the **hc** strategy has been described in [1], where the authors carry out a computational study on the performance of this particular algorithm for various linear and nonlinear prediction operators. The tensor product strategy allows us to obtain 2D error bounds in much the same way as in the 1D case. A posteriori bounds relate the global accumulated error $||v^L - M^{-1}\widetilde{M}_{\bar{\varepsilon}} v^L||$ to the differences $||v^0 - \hat{v}^0||$ and $||\tilde{d}^j - \hat{d}^j||$ for a given norm. A priori error bounds relate the global accumulated error to the processing parameters ε_j.

In the following proposition the modified transformation $\widetilde{M}_{\bar{\varepsilon}}$ is the 2D tensor product extension of Algorithm 1, as described in [1]. The acronyms are the same as in last section and the norm definitions for 2D signals are [15, 1]

(6.1)

$$\|v^j\|_\infty = \sup_{k,l} |v^j_{k,l}|; \quad \|v^j\|_1 = \frac{1}{N_j^2} \left(\sum_{k,l}^{N_j} |v^j_{k,l}| \right); \quad \|v^j\|_2^2 = \frac{1}{N_j^2} \left(\sum_{k,l}^{N_j} |v^j_{k,l}|^2 \right).$$

As in [15, 1], we assume that the processing strategy for the scale coefficients is such that

$$\|\tilde{d}^j(m) - \hat{d}^j(m)\|_p \leq \varepsilon_j, \qquad m = 1, 2, 3. \tag{6.2}$$

PROPOSITION 6.1. [illegible] 2 [illegible] v^L [illegible] $\hat{v}_d = \widetilde{M}_{\bar{\varepsilon}} v^L$ [illegible] $\hat{v}^L = M^{-1}\hat{v}_d$ [illegible]

$$\begin{aligned} \|v^L - \hat{v}^L\|_\infty \leq C_{0,\infty} \|v^0 - \hat{v}^0\|_\infty + \sum_{j=1}^{L} C^2_{j,\infty} \|\tilde{d}^j(2) - \hat{d}^j(2)\|_\infty \\ + \sum_{j=1}^{L} C^*_{j,\infty} \max \left\{ \|\tilde{d}^j(1) - \hat{d}^j(1)\|_\infty, \|\tilde{d}^j(3) - \hat{d}^j(3)\|_\infty \right\}, \end{aligned} \tag{6.3}$$

TABLE 6.1

$p=\infty$	UFL1	HC	UFL2
$C_{0,\infty}$	4^L	1	$\mathcal{B}_\infty^{2L}$
$C_{j,\infty}^2$	$2\cdot 4^{L-j}$	1	$\mathcal{B}_{j,\infty}\mathcal{B}_\infty^{2(L-j)}$
$C_{j,\infty}^*$	4^{L-j}	1	$\mathcal{B}_\infty^{2(L-j)}$

	$C_{0,1}$	$C_{j,1}^2$	$C_{j,1}^*$	$C_{0,2}$	$C_{j,2}^2$	$C_{j,2}^*$
UFL1	1	1	1/2	9^L	$6\cdot 9^{(L-j)}$	$9^{(L-j)}$
HC	1	1	1/2	1	1	1/2
UFL2	$\mathcal{B}_1^{2L}$	$\mathcal{B}_1\mathcal{B}_1^{2(L-j)}$	$\frac{1}{2}\mathcal{B}_1^{2(L-j)}$	$(2\mathcal{B}_2{}^2)^{2L}$	$4\mathcal{B}_2^2(2\mathcal{B}_2{}^2)^{2(L-j)}$	$(2\mathcal{B}_2{}^2)^{2(L-j)}$

$$\|v^L-\hat v^L\|_p^p \le C_{0,p}\|v^0-\hat v^0\|_p^p+\sum_{j=1}^{L} C_{j,p}^2\|\tilde d^j(2)-\hat d^j(2)\|_p^p + \sum_{j=1}^{L} C_{j,p}^*\left\{\|\tilde d^j(1)-\hat d^j(1)\|_p^p+\|\tilde d^j(3)-\hat d^j(3)\|_p^p\right\}, \qquad p=1,2. \tag{6.4}$$

6.1 $\mathcal{B}_p$ 1 (5.4)

Proofs for the HC strategy can be found in [1]; the proofs corresponding to the other two cases are found exactly in the same manner, using the corresponding estimates for the univariate case.

For the HC-strategy, it is worth mentioning that, as in the univariate case, the 2-norm of the compression error can be computed (the a posteriori error estimate in the proposition above is indeed an equality; see [1] for details).

From the a posteriori estimates, it is easy to obtain a priori bounds. As in the univariate case, these lead to estimates of the form

$$\|v^L-\hat v^L\|_p \le C_p\varepsilon$$

for appropriate choices of the parameter vector $\bar\epsilon = (\varepsilon_0,\dots,\varepsilon_L)$. These bounds, however, seem to be useless for the practical compression of 2D signals, except for the case of the HC strategy.

7. Numerical experiments. The theoretical study carried out in the previous sections allows us to conclude that the modified transformation $\widetilde{M}_{\bar\varepsilon}$ ensures an explicit control on the compression error, independently of the particular form used for the computation of the prediction residuals. In this section, we carry out a numerical comparison of the different techniques considered in this paper for the computation of the scale coefficients in $\widetilde{M}_{\bar\varepsilon}$.

The results in Proposition 5.1 point out that the UFL1 and HC versions of $\widetilde{M}_{\bar\varepsilon}$ lead to error bounds that are independent of the prediction operators, while a strong dependence on the specific form of the prediction scheme is to be expected for the UFL2 version. We shall numerically demonstrate this dependence by performing a series of numerical tests on the image depicted in the leftmost corner of Figure 7.5. The image has 512×512 pixel values and we shall consider $L=4$ in the multiscale transformations.

In our numerical tests, we use prediction operators derived from piecewise polynomial average interpolation, as described in section 3, with $R = 2s$. Each prediction scheme is identified by an acronym which refers to the type of reconstruction that serves to define it, as follows:

- FcS-R: Fixed (centered) stencil polynomial pieces of degree R, as specified in section 3.1.
- ENO-R: ENO polynomial pieces of degree R as described in section 3.2.
- ENO-SR-R: ENO-R polynomial pieces, together with the subcell resolution technique (see section 3.3).

The order of the corresponding prediction schemes is $R + 1$ in all cases.

Once the prediction scheme is specified, the compressed multiscale representation of the image, i.e., $\hat{v}_d$, can be obtained in two different ways:

1. the [illegible] way (acronym ST henceforth), i.e., $\hat{v}_d^{ST} = Q_{\bar{\epsilon}} M v^L$, M being the direct multiscale transformation specified in (2.5);
2. using the synchronized [illegible] transformation described in (4.1), i.e., $\hat{v}_d^M = \widetilde{M}_{\bar{\epsilon}} v^L$.

In the latter case, the choice made for the computation of $\tilde{d}^j$ identifies the multiresolution scheme, as in section 5. We recall the acronyms that shall be used throughout this section:

- UFL1 refers to $\widetilde{M}_{\bar{\epsilon}}$ with $\mathbf{pe} = \mathbf{pe1}$ as specified in (4.5).
- HC (Harten's choice) refers to $\widetilde{M}_{\bar{\epsilon}}$ with $\mathbf{pe} = \mathbf{hc}$ as specified in (4.4).
- UFL2 (update-first-lifting choice) refers to $\widetilde{M}_{\bar{\epsilon}}$ with $\mathbf{pe} = \mathbf{pe2}$ as specified in (4.6).

The notation used in the caption of each of the figures should be self-explanatory, e.g., UFL1-ENO-2 refers to computing $\hat{v}_d = \widetilde{M}_{\bar{\epsilon}} v^L$ with (4.5) for the definition of $\tilde{d}^j$ and an ENO nonlinear prediction operator with polynomial pieces of degree $R = 2$. On the other hand ST-ENO-2 refers to computing $\hat{v}^d = Q_{\bar{\epsilon}} M v^L$ with an ENO-2 prediction operator in the direct transformation M.

The reconstructed image is always obtained from the compressed representation by applying the inverse multiscale transformation, i.e., $\hat{v}^L = M^{-1} \hat{v}_d$.

[illegible] 7.1. We recall that the multiscale transformation M obtained from a prediction operator based on FcS-R with $R = 2s$ corresponds to the biorthogonal wavelet $(1, R + 1 = 2s + 1)$ transformation. Hence ST-FcS-R will always be referred to as BOW$(1, R + 1)$.

For the sake of simplicity we shall consider only the simplest processing technique: thresholding according to a level-dependent parameter vector, $\bar{\varepsilon} = (\varepsilon_0, \varepsilon_1, \ldots, \varepsilon_L)$,

$$\hat{d}^j = \mathbf{proc}(\tilde{d}^j, \varepsilon_j) = \begin{cases} 0 & \text{if } |\tilde{d}^j| \le \varepsilon_j, \\ \tilde{d}^j & \text{otherwise.} \end{cases} \tag{7.1}$$

In all cases, we choose the parameter vector as $\varepsilon_j = \varepsilon / 2^{L-j+1}$, or, equivalently, $\varepsilon_L = \varepsilon/2$, $\varepsilon_j = \varepsilon_{j+1}/2$; in addition, and unless specifically stated, we use $v^0 = \hat{v}^0$. In this manner, ε is the only free parameter in the compression algorithm.

7.1. Stability bounds. In Figure 7.1 we plot the 2-norm of the compression error, $||v^L - \hat{v}^L||_2$, versus the compression parameter ε. In the left plot we show the results when using a linear (data-independent) prediction scheme, FcS-2 in this case. We readily observe that computing the compressed representation as $\hat{v}_d^{ST} = Q_{\bar{\epsilon}} M v^L$ or as $\hat{v}_d^M = \widetilde{M}_{\bar{\epsilon}} v^L$ does not alter the stability properties of the transform coding algorithm. In particular, the stability-behavior of the transform coding algorithm

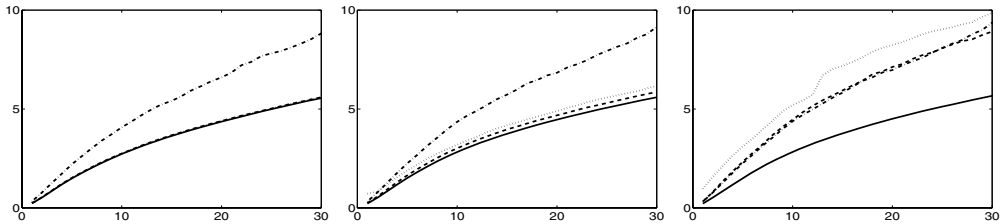


FIG. 7.1. *Plots of the compression error,* $||v^L - \hat{v}^L||_2$*, versus* ε *for the peppers image. Line convention: the* dotted line *corresponds to* $\hat{v}^L = M^{-1}Q_{\tilde{\epsilon}}Mv^L$ *(ST). Other cases,* $\hat{v}^L = M^{-1}\widetilde{M}_{\tilde{\epsilon}}v^L$ *with UFL1 (dash-dot), UFL2 (dashed), HC (solid). Left plot: FcS-2 prediction. Middle plot: ENO-2 prediction. Right plot: ENO-SR-2 prediction.*

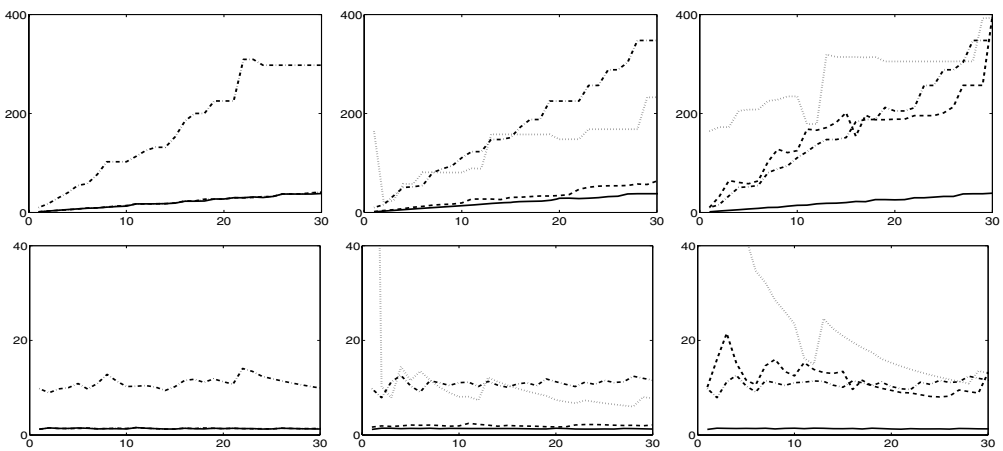


FIG. 7.2. *Top:* $||v^L - \hat{v}^L||_\infty$ *versus* $\varepsilon(q = 2)$*. Bottom:* $||v^L - \hat{v}^L||_\infty/\varepsilon$ *versus* $\varepsilon(q = 2)$*. Peppers image. Same line convention as in Figure* 7.1*. Left column: FcS-2 prediction scheme. Middle column: ENO-2 prediction scheme. Right column: ENO-SR-2 prediction scheme.*

is absolutely equivalent for either ST or the HC and UFL2 versions of the modified transform. We shall see in further tests that this is always the case for linear prediction operators (see also [1] for further comparisons between HC and ST).

The middle and right plots in Figure 7.1 correspond to nonlinear prediction operators of the same order, in this case ENO-2 and ENO-SR-2, respectively. The behavior of the dotted line, which corresponds to ST, confirms a relatively well-known fact: the compression error $||v^L - M^{-1}\hat{v}_d^{ST}||$ does not tend to zero with ϵ when using ENO and ENO-SR nonlinear prediction schemes in the direct multiscale transformation M. The same behavior is observed for the ∞-norm, in the top row of Figure 7.2, and also for the 1-norm (not shown).

In Figures 7.2 (bottom row) and 7.3, we plot $||v^L - \hat{v}^L||_p/\epsilon$ $(p = \infty, 2)$. The plots on the middle and right of Figure 7.3 further confirm that a bound of type $||v^L - \hat{v}^L||_2 \leq C\epsilon$ does not hold when using a nonlinear prediction operator to compute $\hat{v}_d = Q_{\tilde{\epsilon}}Mv^L$.

This should be compared with the stable behavior, in the sense of (1.2), displayed by all the versions of the modified transformation $\widetilde{M}_{\tilde{\epsilon}}$ described in section 5. In Figures 7.1, 7.2, and 7.3, it can be readily appreciated that the stability bounds for HC are not influenced by the choice of prediction scheme: exactly the same behavior

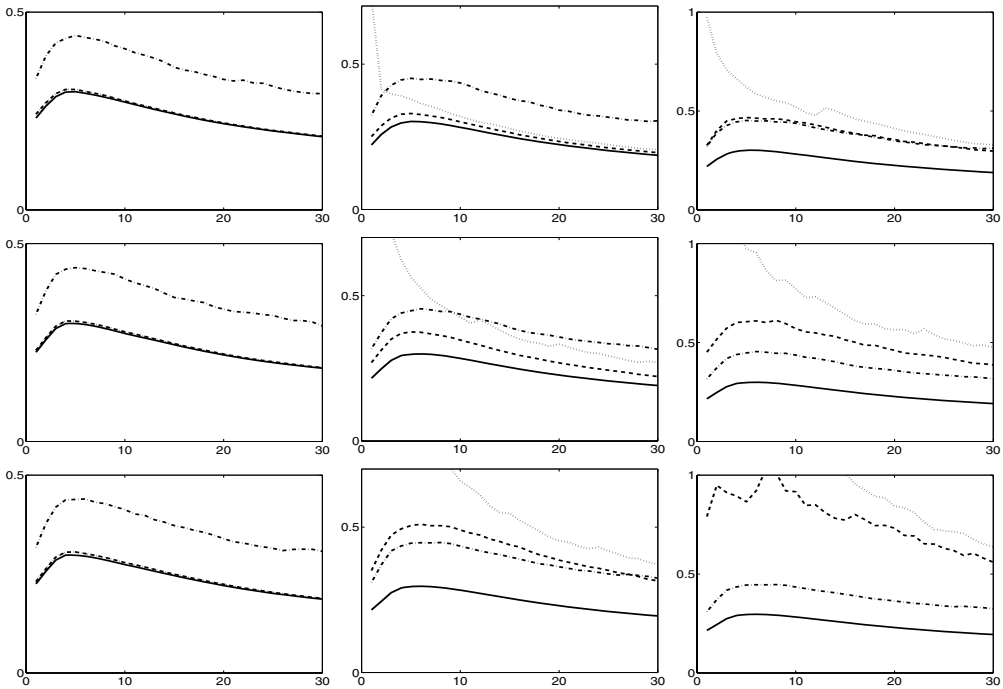


FIG. 7.3. *Plot of* $||v^L - \hat{v}^L||_2/\varepsilon$ *versus* ε *for the peppers image. Same line convention as in Figure* 7.1. *Left column: FcS-R prediction scheme. Middle column: ENO-R prediction scheme. Right column: ENO-SR-R prediction scheme. Top row:* $R = 2$. *Middle row:* $R = 4$. *Bottom row:* $R = 6$.

is observed for the linear or nonlinear prediction operators, independently of their order. The same can be said of UFL1, although the stability bounds are larger for this choice (in agreement with the theoretical observations). It is here where we find the most important difference between HC and UFL2: the UFL2 choice displays a heavy dependence on the specific form of the prediction scheme. As mentioned before, it is interesting to note that the behavior of HC and UFL2 is absolutely similar for [illegible] (FcS-R). It is only when using nonlinear prediction schemes that the two choices behave differently.

Our numerical tests (in particular results in Figures 7.3 and 7.4) seem to indicate that nonlinear predictors such as those used in [7], which reduce the length of the filter when approaching a previously detected edge, should be more appropriate than ENO predictors, which maintain the length of the filter and the order of the reconstruction close to the edges, when using the UFL2 version of the modified transformation, as proposed by Claypoole et al. A full comparison between the prediction proposed in [7] and ENO predictors is, however, beyond the scope of this paper.

A comment on the reliability and sharpness of the error bounds is appropriate here: The results on the peppers image show that we have $||v^L - \hat{v}^L||_2 \leq 0.4\epsilon$ for the HC strategy. This is consistent with the numerical experiments presented in [2, 1], and the same behavior has been observed in all other images for which the algorithm has been applied. This "experimental" bound is smaller than the theoretical error bound of section 6, but of the same order of magnitude. Hence, the theoretical error bounds can still be used for the HC strategy, while they are useless overestimations for the UFL1 and UFL2 cases.

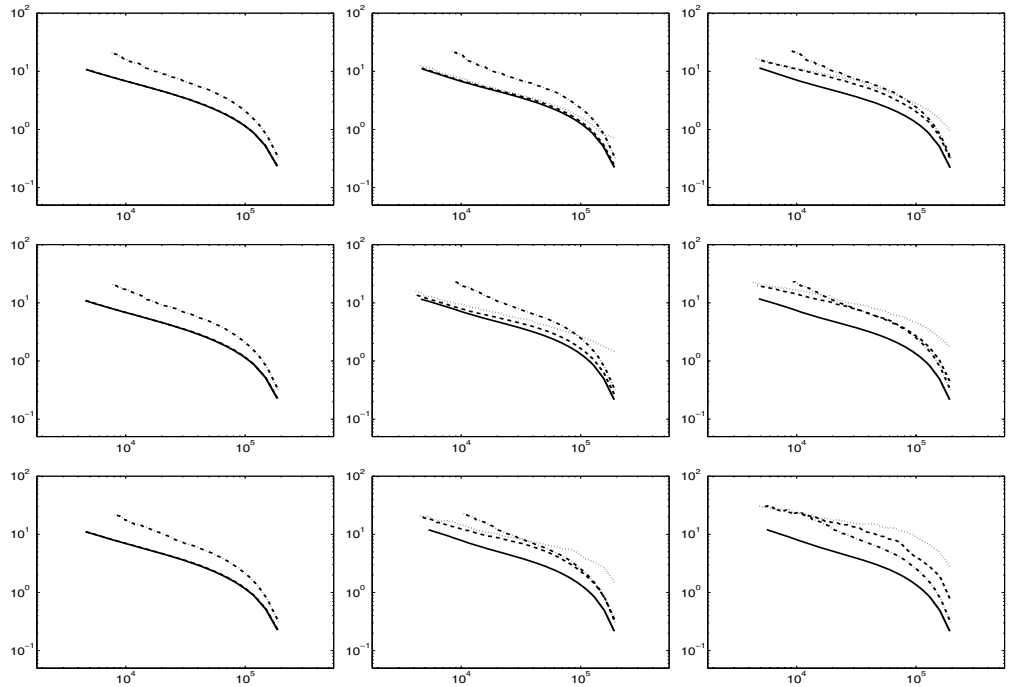


FIG. 7.4. $\log ||v^L - \hat{v}^L||_2$ *versus the number of nonzero details in* $\hat{v}^L$ *for the peppers image. Same line convention as in Figure* 7.1. *Left column: FcS-R prediction scheme. Middle column: ENO-R prediction scheme. Right column: ENO-SR-R prediction scheme. Top row:* $R = 2$. *Middle row:* $R = 4$. *Bottom row:* $R = 6$.

7.2. Compression performance versus quality. To compare the compression performance of the different synchronization strategies we focus on the error curves with respect to the number of preserved coefficients.

Some results are displayed in Figure 7.4 for $R = 2, 4$, and 6. Here, we also observe that UFL2, HC, and ST are essentially equivalent for linear prediction schemes. For ENO predictors, HC is more efficient than UFL2 (smaller error for the same compression rate), and the difference between both alternatives becomes more important for higher order predictors. For the ENO-SR case, the HC alternative is always clearly superior to the others.

We focus next on the [illegible] of the reconstructed images. In Figure 7.5 we show the original peppers image and, for the sake of comparison, the reconstructed figure $\hat{v}^L = M^{-1}Q_{\bar{\epsilon}}Mv^L$ with M a linear BOW$(1, 3)$-transform. The latter has been obtained from approximately 4000 nonzero details (see Table 7.1), and we readily observe the typical [illegible] effect at the image edges, characteristic of compression schemes based on linear multiscale transformations.

Since ENO-SR predictors seem to accentuate the differences between the various compression strategies, we display in Figure 7.6 the reconstructed images obtained (from approximately 4000 nonzero details; see Table 7.1) when using ENO-SR-2 prediction schemes in $\widetilde{M}_{\bar{\epsilon}}$ for each one of the options under consideration. The actual compression errors in various norms are also shown in Table 7.1 (right).

From the results compiled in the table, we observe that [illegible] the HC algorithm maintains the best accuracy not only on the L_2-norm, but also on the L_∞-norm.

FIG. 7.5. *Left: Original peppers image. Right: Reconstruction with* ≈ 4000 *coefficients and a ST-FcS-*2 *transform (i.e., BOW*$(1,3)$ *transform).*

FIG. 7.6. *Reconstruction with* ≈ 4000 *coefficients. Top left: HC-ENO-SR-*2*. Top right: UFL*2*-ENO-SR-*2*. Bottom left: UFL*1*-ENO-SR-*2*. Bottom right: ST-ENO-SR-*2*.*

TABLE 7.1
Compression errors and number of nonzero coefficients. Left: Geotex. Right: Peppers.

	Step	$\|\|\cdot\|\|_\infty$	$\|\|\cdot\|\|_1$	$\|\|\cdot\|\|_2$	nnz	$\|\|\cdot\|\|_\infty$	$\|\|\cdot\|\|_1$	$\|\|\cdot\|\|_2$	nnz
FcS-2	UFL1	1876	6.31	16.55	2007	2173	11.1	33.2	4004
	HC	94.1	4.93	7.45	2005	110	7.86	11.8	4003
	UFL2	98.9	4.91	7.46	2007	110	7.85	11.8	4000
	ST	98.9	4.91	7.46	2007	110	7.85	11.8	4000
ENO-2	UFL1	1752	5.48	18.99	2004	2525	11.2	39.7	4004
	HC	145	4.09	6.55	2006	147	7.57	12.1	4002
	UFL2	150	4.02	6.61	2003	185	7.56	12.6	4007
	ST	150	4.03	6.64	2001	656	7.85	13.8	4003
ENO-SR-2	UFL1	1557	4.26	12.47	1996	2716	12.1	42.1	3999
	HC	67.7	3.37	4.66	1996	152	7.92	12.9	4005
	UFL2	302	4.19	8.23	1994	400	9.23	17.2	3999
	ST	224	3.95	7.27	1997	660	9.05	17.3	4005

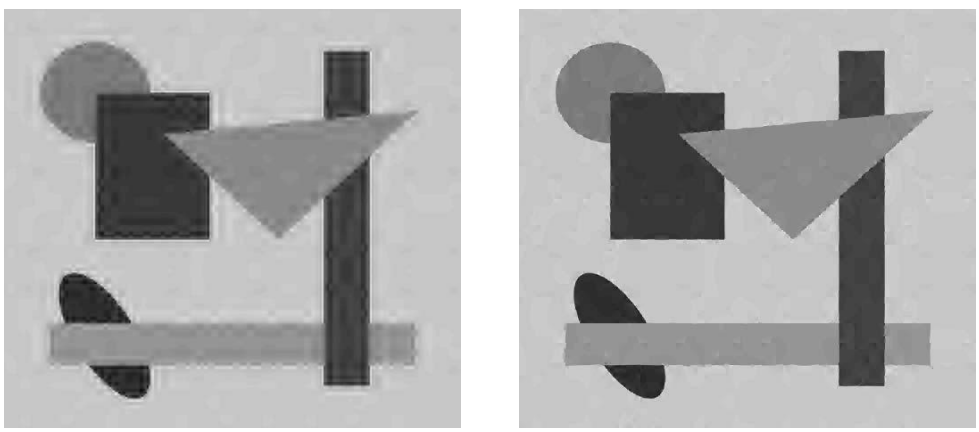


FIG. 7.7. *Reconstruction with* ≈ 2000 *coefficients for the geotex image. Left: HC-FcS-2. Right: HC-ENO-SR-2.*

Notice, in addition, the sharp reconstruction of image edges aligned with the tensor-product underlying grid obtained by the HC-ENO-SR-2. In the larger pepper, however, we can observe an example of the bad handling of texture-dominated areas by ENO-SR predictors. It is interesting to compare this image with the one obtained from the HC-FcS-2 algorithm (essentially the $(1, 3)$ biorthogonal wavelet algorithm). Here, the texture regions look "better," but on the other hand, all edges are blurred. For consistency, all our predictors have the same accuracy on smooth signals, but we emphasize that our goal in this section is merely to illustrate the theoretical results in sections 4 and 5, and that a full comparison between prediction schemes is beyond the scope of the present paper. It is worth mentioning, however, that the "rugged" behavior in texture areas obtained by the ENO-SR prediction scheme, which is caused by the detection of "false edges" ("false alarms"), has recently been the subject of active research [3, 17]. These works have lead to a "robust" ENO-SR-based prediction scheme that is able to eliminate many of these false alarms.

Table 7.1 shows compression errors, $||v^L - \hat{v}^L||_p$ for $p = \infty, 1$, and 2, for different prediction schemes of the same order $R + 1 = 3$. We carry out the computation for

the peppers image and also for a geometric image obtained by superimposing (white noise) texture on shapes of different magnitudes and orientations. This synthetic image (with white noise) is referred to as (for metric image with ture).

For geotex, the ENO-SR prediction combined with the HC algorithm gives the minimum number of coefficients for a given accuracy. This property of the ENO-SR prediction operator in the case of geometric images was already observed in [4, 2, 1], and we observe here that the addition of a moderate amount of noise does not significantly alter the good reconstruction properties of the ENO-SR technique. In Figure 7.7 we display the reconstructed image using approximately 2000 coefficients (the original image is 512×512) with HC-FcS-2 and HC-ENO-SR-2. In the latter case, the reconstructed image is free from edge-blurring and other visually detectable artifacts.

8. Conclusions and further perspectives. We analyze the synchronization mechanism proposed in [7] and observe that it essentially coincides with Harten's EC multiresolution algorithms (see [14, 4, 1]). We theoretically demonstrate that the EC multiscale transformation always leads to encoder/decoder synchronization and provide explicit stability bounds on the resulting cumulative compression errors in the corresponding transform coding algorithm.

The error bounds obtained depend on the specific strategy chosen to compute the detail coefficients. The numerical experiments confirm this dependence. For the preferred strategy proposed in [7], the characteristics of the synchronized compression algorithm depend heavily on the type of prediction scheme, and this is confirmed by a series of numerical tests. On the other hand, Harten's original choice seems to provide the best balance between theoretical and practical effectiveness.

The EC transformations become a useful tool when there is a need to exert a specific control on the cumulative compression error. A potential application lies in compression algorithms, as explored in [5] within the interpolatory multiresolution framework. It could also be potentially used for "optimal quantization," i.e., tuning the quantization mechanism in a dynamical manner so that the total effect on future coefficients is minimized.

Acknowledgments. We would like to thank Richard Baraniuk and Albert Cohen for reading various drafts of this paper. Their comments, as well as the comments of the anonymous referees, have been very valuable to us.

REFERENCES

[1] S. Amat, F. Aràndiga, A. Cohen, and R. Donat, *Tensor product multiresolution analysis with error control for compact image representation*, Signal Process., 4 (2002), pp. 587–608.

[2] S. Amat, F. Aràndiga, A. Cohen, R. Donat, G. Garcia, and M. von Oehsen, *Data compression with ENO schemes: A case study*, Appl. Comput. Harmon. Anal., 11 (2001), pp. 273–288.

[3] F. Aràndiga, A. Cohen, M. Doblas, and B. Matei, *Edge adapted nonlinear multiscale transform for compact image representation*, in Proceedings of the 2003 IEEE International Conference on Image Processing, Vol. 1, 2003, pp. 701–704.

[4] F. Aràndiga and R. Donat, *Nonlinear multiscale decompositions: The approach of A. Harten*, Numer. Algorithms, 23 (2000), pp. 175–216.

[5] F. Aràndiga and R. Donat, *Data compression algorithms with maximum error control*, in Proceedings of the 2003 IEEE International Conference on Image Processing, Vol. 2, 2003, pp. 243–246.

[6] P. J. Burt and E. H. Adelson, *The Laplacian pyramid as a compact image code*, IEEE Trans. Commun., 31 (1983), pp. 532–540.

[7] R. L. Claypoole, G. Davis, W. Sweldens, and R. Baraniuk, *Nonlinear wavelet transforms for image coding via lifting scheme*, IEEE Trans. Image Process., 12 (2003), pp. 1449–1459.
[8] A. Cohen, I. Daubechies, and J. C. Feauveau, *Biorthogonal bases of compactly supported wavelets*, Comm. Pure Appl. Math., 45 (1992), pp. 485–560.
[9] D. L. Donoho, *Interpolating Wavelet Transforms*, Technical report, Department of Statistics, Stanford University, Stanford, CA, 1992.
[10] D. L. Donoho, *Smooth wavelet decompositions with blocky coefficient kernels*, in Recent Advances in Wavelet Analysis, Wavelet Anal. Appl. 3, Academic Press, Boston, MA, 1994, pp. 259–308.
[11] O. Gerek and E. Cetin, *Adaptive polyphase subband decomposition structures for image compression*, IEEE Trans. Image Process., 9 (2000), pp. 1649–1660.
[12] M. Guichaoua, *Analyses Multirésolution Biorthogonales associées à la Résolution d'Equations aux Dérivées Partielles*, Ph.D. thesis, Ecole Superieure de Mecanique de Marseille, Universite de la Mediterranee Aix-Marseille II, 1999.
[13] A. Harten, *ENO schemes with subcell resolution*, J. Comput. Phys., 83 (1989), pp. 148–184.
[14] A. Harten, *Discrete multiresolution analysis and generalized wavelets*, J. Appl. Numer. Math., 12 (1993), pp. 153–193.
[15] A. Harten, *Multiresolution representation of data: A general framework*, SIAM J. Numer. Anal., 33 (1996), pp. 1205–1256.
[16] A. Harten, B. Engquist, S. Osher, and S. R. Chakravarthy, *Uniformly high-order accurate essentially nonoscillatory schemes.* III, J. Comput. Phys., 71 (1987), pp. 231–303.
[17] B. Matei, *Métodes multirésolution nonlinéaires—applications au traitement d'image*, Ph.D. thesis, Université Pierre et Marie Curie, Paris, 2003.
[18] W. Sweldens and P. Schröder, *Building your own wavelets at home*, in Wavelets in Computer Graphics, ACM SIGGRAPH Course Notes, ACM, New York, 1996, pp. 15–87.

SIAM J. SCI. COMPUT.
Vol. 29, No. 1, pp. 290–314
© 2007 Society for Industrial and Applied Mathematics

INEXACT DATA-SPARSE BOUNDARY ELEMENT TEARING AND INTERCONNECTING METHODS*

U. LANGER†§, G. OF‡, O. STEINBACH‡, AND W. ZULEHNER§

Abstract. The boundary element tearing and interconnecting (BETI) methods have recently been introduced as boundary element counterparts of the well-established finite element tearing and interconnecting (FETI) methods. In this paper we present inexact data-sparse versions of the BETI methods which avoid the elimination of the primal unknowns and dense matrices. However, instead of symmetric and positive definite systems, we finally have to solve twofold saddle point problems. The proposed iterative solvers and preconditioners result in almost optimal solvers whose complexity is proportional to the number of unknowns on the skeleton up to some polylogarithmical factor. Moreover, the solvers are robust with respect to large coefficient jumps.

Key words. potential equations, heterogeneous coefficients, boundary element methods, domain decomposition, iterative solvers, preconditioners

AMS subject classifications. 65F10, 65N38, 65N55

DOI. 10.1137/050636243

1. Introduction.

The finite element tearing and interconnecting (FETI) methods were introduced by Farhat and Roux [18] in 1991 as a dual version of the classical iterative substructuring methods (see also the survey paper [19] by the same authors). Meanwhile the classical FETI methods and, in particular, the more recently developed dual-primal FETI (FETI-DP) and balanced domain decomposition by constraints (BDDC) methods are well established as powerful and robust parallel solvers for large-scale finite element equations in different fields of applications; see, e.g., [15, 16, 17, 57]. The practical success of the FETI, FETI-DP, and BDDC methods in real-life applications is supported by a rigorous theoretical analysis [6, 13, 38, 39, 46, 47]. In particular, for second-order self-adjoint elliptic problems such as the potential equation or the linear elasticity problem, the iteration numbers of the conjugate gradient solver preconditioned by a properly scaled FETI preconditioner grow only like $\mathcal{O}(1+\log(H/h))$, where H and h denote the usual scalings of the subdomains and the finite elements, respectively. Furthermore, the iteration process is insensitive to large jumps in the coefficients of the partial differential equations (PDEs). The latter property is very important for many practical applications, e.g., in electromagnetics, where we frequently have to solve interface problems with large jumps in the material coefficients. We refer the reader to the recently published monograph [64] by Toselli and Widlund for more informations about the analysis of FETI methods.

*Received by the editors July 18, 2005; accepted for publication (in revised form) August 14, 2006; published electronically February 2, 2007. This work was supported by the Radon Institute for Computational and Applied Mathematics (RICAM) of the Austrian Academy of Sciences, the Austrian Science Fund "Fonds zur Förderung der wissenschaftlichen Forschung (FWF)" under grant P14953, and the German Research Foundation "Deutsche Forschungsgemeinschaft (DFG)" under grant SFB 404 "Multifield Problems in Continuum Mechanics."

http://www.siam.org/journals/sisc/29-1/63624.html

†Radon Institute for Computational and Applied Mathematics, Austrian Academy of Sciences, Altenberger Strasse 69, A-4040 Linz, Austria (ulrich.langer@assoc.oeaw.ac.at).

‡Institute for Computational Mathematics, Graz University of Technology, Steyrergasse 30, A-8010 Graz, Austria (of@tugraz.at, o.steinbach@tugraz.at).

§Institute for Computational Mathematics, Johannes Kepler University of Linz, Altenberger Strasse 69, A-4040 Linz, Austria (ulanger@numa.uni-linz.ac.at, zulehner@numa.uni-linz.ac.at).

Recently, the boundary element tearing and interconnecting (BETI) methods [43] and the coupled BETI/FETI methods [44] have been introduced. The dual-primal version of the BETI method (BETI-DP) can be introduced along the same lines as the FETI-DP method. It is widely recognized that boundary element methods (BEM) have some advantages over the finite element methods (FEM) in the treatment of unbounded regions. But we can also benefit from the BEM in some other situations such as in the case of large air subdomains without sources in electromagnetics or in the case of moving parts or interfaces in magnetomechanics [34]. The coupling of BEM and FEM within a domain decomposition framework seems to be very attractive (see, e.g., [24] and [60]). We also refer to some early papers on FEM-BEM coupling by Brezzi and Johnson [7] and Johnson and Nédélec [33]. The symmetric FEM-BEM coupling technique was originally proposed by Costabel [10]. The symmetric coupling technique was then used by Hsiao and Wendland to construct a (primal) boundary element substructuring method [32]; see also [31]. As in the finite element substructuring method, they eliminate the interior subdomain unknowns, which are the fluxes (tractions) in the BEM, and arrive at the symmetric and positive definite boundary element Schur complement system that can be solved by a direct or an iterative method. For large-scale problems, a preconditioned conjugate gradient method should be used for efficiency reasons. However, in every iteration step, the interior boundary element subdomain unknowns must be eliminated; i.e., a discrete single layer potential must be inverted to compute the solution of a Dirichlet boundary value problem in every subdomain. To avoid the expensive elimination of these interior boundary element unknowns, an inexact boundary element substructuring method was introduced in [40] which requires the solution of a saddle point problem. The saddle point problem was solved by the preconditioned conjugate gradient method proposed by Bramble and Pasciak for solving systems of algebraic equations with symmetric but indefinite system matrices [4]. The availability of good preconditioners is very essential for the efficiency of the solver. In the inexact boundary element substructuring case, preconditioners are needed for the local (subdomain) discrete single layer potential operators and for the global boundary element Schur complement. The latter usually provides the global information exchange. In [9] such preconditioners were constructed and analyzed. Inexact finite element substructuring solvers were proposed and investigated in [3, 25, 26, 27]. The first inexact FETI solver was introduced by Klawonn in [35] and analyzed by Klawonn and Widlund in [37]. Recently, inexact FETI-DP methods and inexact BDDC methods have been considered in [36] and [14, 45], respectively. Let us mention that inexact versions are usually more efficient than Schur complement techniques, especially in the case of sufficiently large local problems. This is typical for really large-scale problems. Midsize problems on PC clusters can also benefit from inexact solution techniques.

In this paper we introduce inexact BETI methods which lead to threefold saddle point problems in the first instance. However, applying an appropriate projection, we can reduce the threefold saddle point problem to a twofold saddle point problem. Following the approach given in [67] we present preconditioned Krylov subspace solution methods for twofold saddle point problems and give sharp convergence rate estimates. Similar questions arise when considering the Hu–Washizu principle for linear elasticity problems [21]. The standard boundary element discretization of boundary integral operators with nonlocal kernel functions would lead to fully populated matrices. This is totally unacceptable for three-dimensional boundary value problems. Indeed, already the matrix-vector multiplication costs $\mathcal{O}(N_h^2)$ arithmetical operations in the case of dense matrices. Here N_h denotes the number of boundary unknowns which is

of the order $\mathcal{O}(h^{-(d-1)})$, where h denotes the usual mesh size parameter and d is the spatial dimension ($d = 2$, or $d = 3$). The same complexity is required for the storage. Data-sparse approximations of the system matrix, such as multipole techniques [8, 58], panel clustering methods [29], $\mathcal{H}$-matrix approaches [28], and adaptive cross approximation (ACA) methods [1, 2], can reduce the complexity to almost $\mathcal{O}(N_h)$, except for a polylogarithmic factor, for both the arithmetic and the memory costs. Here we use the fast multipole method, developed in [53], to approximate the single layer potential, the double layer potential, and the hypersingular boundary integral operators which appear in the symmetric boundary integral domain decomposition formulation. Our preconditioned Krylov subspace solvers for twofold saddle point problems require appropriately scaled preconditioners for the local discrete single layer potential operators, for the local boundary element Schur complements, and for the BETI matrix in their data-sparse (fast multipole) representations. We propose data-sparse preconditioners which result in an almost optimal solver requiring $\mathcal{O}((H/h)^{(d-1)}(1+\log(H/h))^s \log \varepsilon^{-1})$ arithmetic operations in a parallel regime and $\mathcal{O}((H/h)^{(d-1)}(1+\log(H/h))^2)$ memory per processor, where $s \in \{3,4,5\}$ depending on the quality of the preconditioners for the local problems, and $\varepsilon \in (0,1)$ is the relative accuracy of the iteration error in a suitable norm. Moreover, our solvers are insensitive to large jumps in the coefficients of the potential equation which serves as our model problem.

The rest of the paper is organized as follows: In section 2, we introduce the data-sparse symmetric boundary element domain decomposition method. Section 3 is devoted to the inexact BETI approach that results in a twofold saddle-point problem after a suitable subspace projection. In section 4, we present and analyze iterative methods for solving twofold saddle point problems. Optimal data-sparse boundary element preconditioners required by the solvers presented in section 4 are given in section 5. This completes the solution procedure. Finally, we discuss some numerical results in section 6 and draw some conclusions in section 7.

2. Data-sparse boundary element domain decomposition methods. As a model problem we consider the Dirichlet boundary value problem for the potential equation

$$-\mathrm{div}[\alpha(x)\nabla\hat{u}(x)] = 0 \quad \text{for } x \in \Omega \subset \mathbb{R}^3, \quad \hat{u}(x) = g(x) \quad \text{for } x \in \Gamma = \partial\Omega, \tag{2.1}$$

where Ω is a bounded Lipschitz domain that is assumed to be decomposed into p nonoverlapping subdomains Ω_i with Lipschitz boundaries $\Gamma_i = \partial\Omega_i$. We further assume that the coefficient function $\alpha(\cdot)$ in the potential equation (2.1) is piecewise constant with $\alpha(x) = \alpha_i > 0$ for $x \in \Omega_i$, $i = 1, \ldots, p$. The variational formulation of the boundary value problem (2.1) reads as follows: find $\hat{u} \in H^1(\Omega)$ with $\hat{u}_{|\Gamma} = g$ such that the variational equation

$$\sum_{i=1}^{p} \alpha_i \int_{\Omega_i} \nabla\hat{u}(x)\,\nabla v(x)dx \;=\; 0 \tag{2.2}$$

is satisfied for all test functions $v \in H^1_0(\Omega)$, where g is a given function from $H^{1/2}(\Gamma)$. Applying Green's first formula with respect to the local subdomains Ω_i gives

$$\sum_{i=1}^{p} \alpha_i \int_{\Gamma_i} \frac{\partial}{\partial n_i}\hat{u}(x)v(x)ds_x \;=\; 0 \quad \text{for all } v \in H^1_0(\Omega) \tag{2.3}$$

due to $-\Delta\hat{u}(x) = 0$ for $x \in \Omega_i$, where $n_i = n_{i,x}$ denotes the exterior normal vector to Γ_i at $x \in \Gamma_i$. For $x \in \Omega_i$ the solution of the local Laplace equation is given by the local representation formula

$$\hat{u}(x) = \int_{\Gamma_i} U^*(x,y) \frac{\partial}{\partial n_{i,y}} \hat{u}(y) ds_y - \int_{\Gamma_i} \frac{\partial}{\partial n_{i,y}} U^*(x,y) \hat{u}(y) ds_y, \tag{2.4}$$

where

$$U^*(x,y) = \frac{1}{4\pi} \frac{1}{|x-y|}$$

is the fundamental solution of the Laplace operator; see, e.g., [61]. Taking the trace and the normal derivative of the local representation formula (2.4) we obtain the Calderon system

$$\begin{pmatrix} u_i \\ t_i \end{pmatrix} = \begin{pmatrix} \frac{1}{2} I - K_i & V_i \\ D_i & \frac{1}{2} I + K_i' \end{pmatrix} \begin{pmatrix} u_i \\ t_i \end{pmatrix} \tag{2.5}$$

of local boundary integral equations on Γ_i for the Cauchy data $u_i = \hat{u}_{|\Gamma_i} \in H^{1/2}(\Gamma_i)$ and $t_i = n_i \cdot \nabla \hat{u}_{|\Gamma_i} \in H^{-1/2}(\Gamma_i)$, where the local single layer potential operator $V_i : H^{-1/2}(\Gamma_i) \to H^{1/2}(\Gamma_i)$, the local double layer potential operator $K_i : H^{1/2}(\Gamma_i) \to H^{1/2}(\Gamma_i)$, its adjoint $K_i' : H^{-1/2}(\Gamma_i) \to H^{-1/2}(\Gamma_i)$, and the local hypersingular boundary integral operator $D_i : H^{1/2}(\Gamma_i) \to H^{-1/2}(\Gamma_i)$ are defined by

$$\begin{aligned}
(V_i t_i)(x) &= \int_{\Gamma_i} U^*(x,y) t_i(y) ds_y, \\
(K_i u_i)(x) &= \int_{\Gamma_i} \frac{\partial}{\partial n_{i,y}} U^*(x,y) u_i(y) ds_y, \\
(K_i' t_i)(x) &= \int_{\Gamma_i} \frac{\partial}{\partial n_{i,x}} U^*(x,y) t_i(y) ds_y, \\
(D_i u_i)(x) &= -\frac{\partial}{\partial n_{i,x}} \int_{\Gamma_i} \frac{\partial}{\partial n_{i,y}} U^*(x,y) u_i(y) ds_y,
\end{aligned}$$

respectively. The mapping properties of these boundary integral operators are now well known; see, e.g., [11, 48, 61]. In particular, the local single layer potential operators V_i are self-adjoint and $H^{-1/2}(\Gamma_i)$-elliptic. The local hypersingular operators D_i are self-adjoint as well, but only positive semidefinite. Indeed, the kernel $\ker D_i$ of the hypersingular operator D_i is spanned by all constant functions; i.e., $\ker D_i = \operatorname{span}\{1\}$.

Let us denote the skeleton of the domain decomposition by $\Gamma_S = \cup_{i=1}^p \Gamma_i$ and the skeleton trace space by $H^{1/2}(\Gamma_S) = \{v_{|\Gamma_S} : v \in H^1(\Omega)\}$. Inserting the second boundary integral equation of (2.5) into (2.3), we have to find $\hat{u} \in H^{1/2}(\Gamma_S)$ with $\hat{u}_{|\Gamma} = g$ such that

$$\sum_{i=1}^p \alpha_i \left[\langle D_i \hat{u}_{|\Gamma_i}, v_{|\Gamma_i} \rangle_{\Gamma_i} + \left\langle \left(\frac{1}{2} I + K_i' \right) t_i, v_{|\Gamma_i} \right\rangle_{\Gamma_i} \right] = 0 \tag{2.6}$$

for all test functions $v \in H_0^{1/2}(\Gamma_S) = \{v \in H^{1/2}(\Gamma_S) : v_{|\Gamma} = 0\}$, where $t_i \in H^{-1/2}(\Gamma_i)$ are the unique solutions of the local variational problems

$$\alpha_i \left[\langle V_i t_i, w_i \rangle_{\Gamma_i} - \left\langle \left(\frac{1}{2} I + K_i \right) \hat{u}_{|\Gamma_i}, w_i \right\rangle_{\Gamma_i} \right] = 0 \tag{2.7}$$

for all test functions $w_i \in H^{-1/2}(\Gamma_i)$, where $\langle \cdot, \cdot \rangle_{\Gamma_i}$ denotes the duality pairing in $H^{-1/2}(\Gamma_i) \times H^{1/2}(\Gamma_i)$ that is the extension of the scalar product in $L_2(\Gamma_i)$ for $i = 1, 2, \ldots, p$. After homogenization of the Dirichlet boundary condition via the ansatz $\hat{u} = \hat{g} + u$ with some extension $\hat{g}_{|\Gamma} = g$ and $u_{|\Gamma} = 0$, we can rewrite (2.6) and (2.7) as a mixed variational problem to find $t = (t_1, t_2, \ldots, t_p) \in T = T_1 \times T_2 \times \cdots \times T_p = H^{-1/2}(\Gamma_1) \times H^{-1/2}(\Gamma_2) \times \cdots \times H^{-1/2}(\Gamma_p)$ and $u \in U = H_0^{1/2}(\Gamma_S)$ such that

$$\alpha_i \left[\langle w_i, V_i t_i \rangle_{\Gamma_i} - \left\langle w_i, \left(\frac{1}{2} I + K_i \right) u_{|\Gamma_i} \right\rangle_{\Gamma_i} \right] = \alpha_i \left\langle w_i, \left(\frac{1}{2} I + K_i \right) \hat{g}_{|\Gamma_i} \right\rangle_{\Gamma_i} \tag{2.8}$$

for all $w_i \in T_i$, $i = 1, 2, \ldots, p$, and

$$\sum_{i=1}^{p} \alpha_i \left[- \left\langle \left(\frac{1}{2} I + K_i' \right) t_i, v_{|\Gamma_i} \right\rangle_{\Gamma_i} - \langle D_i u_{|\Gamma_i}, v_{|\Gamma_i} \rangle_{\Gamma_i} \right] = \sum_{i=1}^{p} \alpha_i \langle D_i \hat{g}_{|\Gamma_i}, v_{|\Gamma_i} \rangle_{\Gamma_i} \tag{2.9}$$

for all $v \in U$. It is well known that the mixed variational problem (2.8) and (2.9) is uniquely solvable [32]. Therefore, problem (2.6)–(2.7), which is equivalent to problem (2.8)–(2.9), is also uniquely solvable.

Let us now introduce the boundary element trial spaces

$$U_h = S_h^1(\Gamma_S) = \operatorname{span}\{\varphi_m\}_{m=1}^{M} \subset U = H_0^{1/2}(\Gamma_S) \tag{2.10}$$

and

$$T_{i,h} = S_h^0(\Gamma_i) = \operatorname{span}\{\psi_k^i\}_{k=1}^{N_i} \subset T_i = H^{-1/2}(\Gamma_i) \tag{2.11}$$

spanned by continuous piecewise linear basis functions φ_m and by piecewise constant basis functions ψ_k^i with respect to a regular globally quasi-uniform boundary element mesh $\Gamma_i = \cup_{\ell=1}^{N_i} \tau_\ell^i$ with the average mesh size h on Γ_S and Γ_i, respectively. The restriction of the global trial space U_h onto the local subdomain boundaries Γ_i is denoted by

$$U_{i,h} = S_h^1(\Gamma_i) = S_h^1(\Gamma_S)_{|\Gamma_i} = \operatorname{span}\{\varphi_m^i\}_{m=1}^{M_i} \subset H_0^{1/2}(\Gamma_i), \tag{2.12}$$

where $H_0^{1/2}(\Gamma_i) = \{v_{|\Gamma_i} : v \in H_0^{1/2}(\Gamma_S)\}$. The Galerkin discretization of the symmetric mixed variational formulation (2.8) and (2.9) now gives immediately the mixed boundary element equations to find $t_h = (t_{1,h}, t_{2,h}, \ldots, t_{p,h}) \in T_h = T_{1,h} \times T_{2,h} \times \cdots \times T_{p,h}$ and $u_h \in U_h$ such that

$$\alpha_i \left[\langle w_{i,h}, V_i t_{i,h} \rangle_{\Gamma_i} - \left\langle w_{i,h}, \left(\frac{1}{2} I + K_i \right) u_{h|\Gamma_i} \right\rangle_{\Gamma_i} \right] = \alpha_i \left\langle w_{i,h}, \left(\frac{1}{2} I + K_i \right) \hat{g}_{|\Gamma_i} \right\rangle \tag{2.13}$$

for all $w_{i,h} \in T_{i,h}$, $i = 1, 2, \ldots, p$, and

$$\sum_{i=1}^{p} \alpha_i \left[- \left\langle \left(\frac{1}{2} I + K_i' \right) t_{i,h}, v_{h|\Gamma_i} \right\rangle_{\Gamma_i} - \langle D_i u_{h|\Gamma_i}, v_{h|\Gamma_i} \rangle_{\Gamma_i} \right] = \sum_{i=1}^{p} \alpha_i \langle D_i \hat{g}_{|\Gamma_i}, v_{h|\Gamma_i} \rangle_{\Gamma_i} \tag{2.14}$$

for all $v_h \in U_h$. Applying standard arguments, it follows that the mixed boundary element equations (2.13) and (2.14) admit a unique solution. Moreover, the discretization error is bounded by the approximation error,

$$\|u-u_h\|_U^2 + \|t-t_h\|_T^2 \le c \left\{ \inf_{v_h \in U_h} \|u-v_h\|_U^2 + \inf_{w_h \in T_h} \|t-w_h\|_T^2 \right\}.$$

From the approximation properties of the boundary element trial spaces U_h and T_h we then obtain the a priori error estimate

$$\|u-u_h\|_U^2 + \|t-t_h\|_T^2 \;\le\; c\, h^{2s+1} \left[\|u\|_{H^{s+1}(\Gamma_S)}^2 + \sum_{i=1}^{p} \|t_i\|_{H^s_{\mathrm{pw}}(\Gamma_i)}^2 \right] \tag{2.15}$$

for a sufficiently smooth solution (u,t) with $u \in H^{s+1}(\Gamma_S)$ and $t_i \in H^s_{\mathrm{pw}}(\Gamma_i)$, $i = 1,2,\ldots,p$, for some $s \in (0,1]$. Here the subscript pw indicates the piecewise definition of the corresponding smoothness; see [61, p. 43] for the precise definition of these spaces. Estimate (2.15) gives an optimal convergence rate of $\mathcal{O}(h^{3/2})$; see, e.g., [61]. Here and in the following, c denotes a generic constant that is always independent of the discretization parameters.

Once the basis functions are chosen, the boundary element equations (2.13) and (2.14) are equivalent to the following system of algebraic equations: Find the coefficient vectors $\underline{t}_i = [t_{i,k}]_{k=1}^{N_i} \in \mathbb{R}^{N_i}$ for $i = 1,2,\ldots,p$ and $\underline{u} = [u_m]_{m=1}^{M} \in \mathbb{R}^M$ of the boundary element functions $t_{i,h} = \sum_{k=1}^{N_i} t_{i,k}\psi_k^i$ and $u_h = \sum_{m=1}^{M} u_m \varphi_m$ as the unique solution of the Galerkin system

$$\begin{pmatrix} \alpha_1 V_{1,h} & & & -\alpha_1 \bar{K}_{1,h} R_{1,h} \\ & \ddots & & \vdots \\ & & \alpha_p V_{p,h} & -\alpha_p \bar{K}_{p,h} R_{p,h} \\ -\alpha_1 R_{1,h}^\top \bar{K}_{1,h}^\top & \ldots & -\alpha_p R_{p,h}^\top \bar{K}_{p,h}^\top & -D_h \end{pmatrix} \begin{pmatrix} \underline{t}_1 \\ \vdots \\ \underline{t}_p \\ \underline{u} \end{pmatrix} = \begin{pmatrix} \alpha_1 \underline{g}_1 \\ \vdots \\ \alpha_p \underline{g}_p \\ \underline{f} \end{pmatrix}, \tag{2.16}$$

where

$$D_h = \sum_{i=1}^{p} \alpha_i R_{i,h}^\top D_{i,h} R_{i,h} \quad \text{and} \quad \underline{f} = \sum_{i=1}^{p} \alpha_i R_{i,h}^\top \underline{f}_i. \tag{2.17}$$

The entries of the matrices $V_{i,h}$, $\bar{K}_{i,h}$, $D_{i,h}$ and the coefficients of the right-hand sides $\underline{g}_i$ and $\underline{f}$ are defined by the relations

$$\begin{aligned} V_{i,h}[\ell,k] &= \langle \psi_\ell^i, V_i \psi_k^i \rangle_{\Gamma_i}, \\ \bar{K}_{i,h}[\ell,m] &= \left\langle \psi_\ell^i, \left(\frac{1}{2} I + K_i \right) \varphi_m^i \right\rangle_{\Gamma_i}, \\ D_{i,h}[n,m] &= \langle \varphi_n^i, D_i \varphi_m^i \rangle_{\Gamma_i} \end{aligned}$$

and $g_{i,\ell} = \langle \psi_\ell^i, (\frac{1}{2} I + K_i)\hat{g}_{|\Gamma_i} \rangle_{\Gamma_i}$, $f_{i,n} = \langle D_i \hat{g}_{|\Gamma_i}, \varphi_n^i \rangle_{\Gamma_i}$ for all $k,\ell = 1,\ldots,N_i$ and $m,n = 1,\ldots,M_i$, respectively. The restriction operator $R_{i,h}$ maps any global coefficient vector $\underline{v} \in \mathbb{R}^M$ to the local vector $\underline{v}_i \in \mathbb{R}^{M_i}$ containing those components of $\underline{v}$ which correspond to Γ_i only, $i = 1,2,\ldots,p$. The matrices $R_{i,h}$ are Boolean matrices which are sometimes also called subdomain connectivity matrices.

The system matrix of (2.16) is symmetric but indefinite. The matrices $V_{i,h}$ are symmetric and positive definite, while the blocks $D_{i,h}$ are symmetric but in general only positive semidefinite, where the kernel is spanned by all vectors with constant entries. Unfortunately, all matrix blocks involved in (2.16) are fully populated due to the nonlocal fundamental solution. Especially for three-dimensional boundary value problems, this fact would require unacceptable large resources of memory and already a high complexity of a single matrix-vector multiplication. Since the latter operation is the basic operation in every iteration method, we need some data-sparse approximation of these matrices that reduces the quadratic complexity with respect to the number of unknowns to an almost linear one, but without affecting the accuracy given by the quasi-optimal error estimate (2.15).

As already pointed out in the introduction, there exist several possibilities of fast BEM to reduce the computational costs, for example, panel clustering [29], algebraic approximation techniques such as the ACA method [2], and hierarchical matrices [28] or wavelets [12]. Following [53], we describe here the application of the fast multipole method to the discrete boundary integral operators introduced above. We will mainly use the original version of the fast multipole method as described in [22, 23], but the main difference in our version of the method will be the use of reformulated spherical harmonics [55, 65, 66].

As already mentioned above, when using an iterative solver it is necessary to efficiently perform the matrix-by-vector multiplications of the local stiffness matrices with given vectors. In particular, if the vectors $\underline{r}_1^i \in \mathbb{R}^{N_i}$ and $\underline{r}_2^i \in \mathbb{R}^{M_i}$ are given, we need a fast computation of the matrix-vector products $\underline{w}_1^i = V_{i,h}\underline{r}_1^i \in \mathbb{R}^{N_i}$, $\underline{w}_2^i = K_{i,h}\underline{r}_2^i \in \mathbb{R}^{N_i}$, $\underline{v}_1^i = D_{i,h}\underline{r}_2^i \in \mathbb{R}^{M_i}$, and $\underline{v}_2^i = K_{i,h}^\top \underline{r}_1^i \in \mathbb{R}^{M_i}$.

We begin with the local matrix-vector product $\underline{w}_1^i = V_{i,h}\underline{r}_1^i$ defined componentwise as follows:

$$w_{1,\ell}^i \;=\; \sum_{k=1}^{N_i} V_{i,h}[\ell,k] r_{1,k}^i \;=\; \sum_{k=1}^{N_i} \frac{r_{1,k}^i}{4\pi} \int_{\tau_\ell^i} \int_{\tau_k^i} \frac{1}{|x-y|} ds_y ds_x \quad \text{for } \ell = 1,\dots,N_i.$$

As in most of the fast BEM, one first substitutes the kernel function by an appropriate series expansion separating the integration variables x and y from each other. The kernel of the single layer potential can be written in terms of Legendre polynomials:

$$k(x,y) \;=\; \frac{1}{|x-y|} \;=\; \sum_{n=0}^{\infty} \frac{|x|^n}{|y|^{n+1}} P_n(\hat{x}\cdot\hat{y}), \quad \hat{x} = \frac{x}{|x|}, \quad \hat{y} = \frac{y}{|y|},$$

where

$$P_n(u) \;=\; \frac{1}{2^n n!} \frac{d^n}{du^n}(u^2-1)^n \quad \text{for } |u| \le 1.$$

An approximation of the kernel function $k(x,y)$ is then defined by truncating the infinite sum at certain expansion degree ϱ:

$$k_\varrho(x,y) \;=\; \sum_{n=0}^{\varrho} \frac{|x|^n}{|y|^{n+1}} P_n(\hat{x}\cdot\hat{y}). \tag{2.18}$$

Note that for $|y| > |x|$ the following error estimate holds:

$$|k(x,y) - k_\varrho(x,y)| \;\le\; \frac{1}{|y|-|x|} \left(\frac{|x|}{|y|}\right)^{\varrho+1}. \tag{2.19}$$

Since the product of the normalized integration variables $\hat{x}$ and $\hat{y}$ still appears as an argument of the Legendre polynomials in (2.18), one may instead use spherical harmonics [30] to obtain the kernel approximation

$$k_\varrho(x,y) \;=\; \sum_{n=0}^{\varrho}\sum_{m=-n}^{n} \frac{|x|^n}{|y|^{n+1}} Y_n^{-m}(\hat{x}) Y_n^m(\hat{y}) \;=\; \sum_{n=0}^{\varrho}\sum_{m=-n}^{n} \overline{S_n^m}(y) R_n^m(x) \tag{2.20}$$

when assuming $|y| > |x|$. The truncated series expansion (2.20) is the first ingredient of this fast multipole method. The second is the use of an artificial geometric hierarchy which is built upon the boundary elements. All boundary elements $\tau_k^i \subset \Gamma_i$ belong to a cluster $\omega_{i,1}^0$ which is a box containing the whole subdomain Ω_i. The clusters of the level $\lambda+1$ are constructed by the refinement of all boxes $\omega_{i,j}^\lambda$ into eight similar subboxes $\omega_{i,j'}^{\lambda+1}$. The clusters $\omega_{i,j'}^{\lambda+1}$ are called the sons of the father cluster $\omega_{i,j}^\lambda$. This recursive process is repeated until a maximum level L is attained. Clusters containing no boundary elements are neglected. All boundary elements $\{\tau_k^i\}_{k=1}^{N_i}$ are assigned to the cluster $\omega_{i,j}^L$ with respect to their midpoints.

Since the expansion (2.20) is valid only for $|x| < |y|$, one has to distinguish the farfield (FF), where this expansion is admissible, from the nearfield (NF), where the standard kernel has to be used. A cluster ω_j^λ on the same level λ is said to be in the NF of the cluster ω_i^λ if the condition

$$\operatorname{dist}\{C_i^\lambda, C_j^\lambda\} \;\le\; (d+1)\max\{r_i^\lambda, r_j^\lambda\} \tag{2.21}$$

is satisfied with $d > 1$ a suitably chosen parameter. C_i^λ denotes the center and r_i^λ is the radius of the cluster ω_i^λ which is determined by the cluster center and the extension of the boundary elements τ_k contained in ω_i^λ.

Hence we obtain the approximate local matrix-vector product

$$\begin{aligned}\widetilde{w}_{1,\ell}^i &= \sum_{k\in \mathrm{NF}(\ell)} V_{i,h}[\ell,k] r_{1,k}^i + \sum_{k\in \mathrm{FF}(\ell)} \frac{r_{1,k}^i}{4\pi} \int_{\tau_\ell}\int_{\tau_k} \sum_{n=0}^{\varrho}\sum_{m=-n}^{n} \overline{S_n^m}(y) R_n^m(x)\, ds_x\, ds_y \\ &= \sum_{k\in \mathrm{NF}(\ell)} V_{i,h}[\ell,k] r_{1,k}^i + \sum_{n=0}^{\varrho}\sum_{m=-n}^{n}\sum_{k\in \mathrm{FF}(\ell)} \frac{r_{1,k}^i}{4\pi} \int_{\tau_k} R_n^m(x)\, ds_x \int_{\tau_\ell} \overline{S_n^m}(y)\, ds_y\end{aligned}$$

for all $\ell = 1,\dots,N_i$. As the FFs differ from each other for most indices ℓ, the question arises of how to compute the coefficients of the series expansion for different ℓ efficiently. This can be done by using the cluster hierarchy constructed above. Starting on the finest level all coefficients for the corresponding clusters are computed and later translated to their common fathers on the coarser levels up to level zero. Next, these multipole expansions are converted into local expansions in terms of their coefficients on the highest level, in which the kernel expansion is admissible. This results in an algorithm of $\mathcal{O}(N_i \log^2 N_i)$ complexity. For a more detailed description of the efficient matrix-vector multiplication, see [53] and the references given there.

In what follows we will focus on the stability and error analysis of the approximate matrix-vector product $\underline{\widetilde{w}}_1^i = \widetilde{V}_{i,h}\underline{r}_1^i$ in the case of a fixed NF parameter $d > 0$.

LEMMA 2.1 (see [53, Lemma 4.1]). [illegible] $\varrho = \mathcal{O}(\log N_i)$ [illegible] $\widetilde{V}_i$ [illegible]

$$\|(V_i - \widetilde{V}_i) t_{i,h}\|_{L_2(\Gamma_i)} \;\le\; c\, h^2 \|t_{i,h}\|_{L_2(\Gamma_i)} \quad \text{[illegible]} \quad t_{i,h} \in S_h^0(\Gamma_i) \subset L_2(\Gamma_i).$$

From the previous lemma and an inverse inequality we see that it is possible to choose the expansion degree $\varrho = \mathcal{O}(\log N_i)$ such that the approximate single layer potential $\widetilde{V}_{i,h}$ is $S_h^0(\Gamma_i)$-elliptic,

$$\langle \widetilde{V}_{i,h} w_{i,h}, w_{i,h} \rangle_{\Gamma_i} \geq c_1^{\widetilde{V}_i} \|w_{i,h}\|^2_{H^{-1/2}(\Gamma)} \quad \text{for all } w_{i,h} \in S_h^0(\Gamma_i).$$

The multipole expansion of the double layer potential operator $\widetilde{K}_i$ can be computed by applying the normal derivative to the expansion (2.20) of the single layer potential V_i. In the case of the adjoint double layer potential K_i', the normal derivative has to be applied to the coefficients in the evaluation of the expansions. Therefore the transposition property between the Galerkin matrices of the double layer potential and its adjoint operator will be maintained if the same expansion degree p is chosen for both. As for the approximate single layer potential $\widetilde{V}_{i,h}$, corresponding error estimates follow; see [53, Lemmas 4.3 and 4.4].

The hypersingular boundary integral operator D_i is treated via integration by parts [50]. The bilinear form of the hypersingular integral operator D_i can be written as a sum of bilinear forms of the single layer potential V_i,

$$\langle D_i u_i, v_i \rangle_{\Gamma_i} = \sum_{k=1}^{3} \langle V_i \underline{\mathrm{curl}}_{|\Gamma_i} u_{i|k}, \underline{\mathrm{curl}}_{|\Gamma_i} v_{i|k} \rangle_{\Gamma_i},$$

with the surface curl $\underline{\mathrm{curl}}_{|\Gamma_i} u_i(x) = n_x \times \nabla u^*_{i,\Gamma_i}(x)$ defined for a constant extension $u^*_{i|\Gamma_i}$ of $u(x)$ along the normal n_x into a small neighborhood of Γ_i. In the case of piecewise linear, continuous basis functions $\varphi_m^i(x)$ and plane triangles as boundary elements, the vector $\underline{\mathrm{curl}}_{\Gamma_i} \varphi_m^i(x)$ is piecewise constant and constant on each boundary element. Therefore the Galerkin matrix of the single layer potential V_i for piecewise constant basis functions ψ_k^i can be reused here, leading to the approximate bilinear form

$$\langle \widetilde{D}_{i,h} u_{i,h}, v_{i,h} \rangle_{\Gamma_i} = \sum_{k=1}^{3} \langle \widetilde{V}_{i,h} \underline{\mathrm{curl}}_{|\Gamma_i} u_{i,h|k}, \underline{\mathrm{curl}}_{|\Gamma_i} v_{i,h|k} \rangle_{\Gamma_i}. \tag{2.22}$$

The stability and error analysis for the approximation $\widetilde{D}_{i,h}$ then follows as for the approximate single layer potential.

Using the fast multipole approximations for the application of the discrete boundary integral operators, instead of (2.16) we have to solve the perturbed linear system

$$\begin{pmatrix} \alpha_1 \widetilde{V}_{1,h} & & & -\alpha_1 \widetilde{K}_{1,h} R_{1,h} \\ & \ddots & & \vdots \\ & & \alpha_p \widetilde{V}_{p,h} & -\alpha_p \widetilde{K}_{p,h} R_{p,h} \\ -\alpha_1 R_{1,h}^\top \widetilde{K}_{1,h}^\top & \dots & -\alpha_p R_{p,h}^\top \widetilde{K}_{p,h}^\top & -\widetilde{D}_h \end{pmatrix} \begin{pmatrix} \widetilde{\underline{t}}_1 \\ \vdots \\ \widetilde{\underline{t}}_p \\ \widetilde{\underline{u}} \end{pmatrix} = \begin{pmatrix} \alpha_1 \widetilde{\underline{g}}_1 \\ \vdots \\ \alpha_p \widetilde{\underline{g}}_p \\ \widetilde{\underline{f}} \end{pmatrix}. \tag{2.23}$$

As in the case of the symmetric boundary integral formulation considered in [53, Theorem 4.1], we obtain the unique solvability of (2.23) as well as the quasi-optimal error estimate

$$\|u - \widetilde{u}_h\|_U^2 + \|t - \widetilde{t}_h\|_T^2 \leq c\, h^{2s+1} \left[\|u\|^2_{H^{s+1}(\Gamma_S)} + \sum_{i=1}^{p} \|t_i\|^2_{H^s_{\mathrm{pw}}(\Gamma_i)} \right] \tag{2.24}$$

for the associated approximate solutions $\widetilde{t}_h \in T_h$ and $\widetilde{u}_h \in U_h$ when assuming $\varrho = \mathcal{O}(\log N_i)$ and smooth solutions as in (2.15), i.e., $s \in (0,1]$. Hence we can ensure the same asymptotic behavior as for the solution of the standard Galerkin boundary element method.

A standard approach to solving the linear system (2.23) is to consider its Schur complement system

$$\sum_{i=1}^{p} \alpha_i R_{i,h}^{\top} \left[\widetilde{D}_{i,h} + \widetilde{K}_{i,h}^{\top} \widetilde{V}_{i,h}^{-1} \widetilde{K}_{i,h} \right] R_{i,h} \underline{u} = -\sum_{i=1}^{p} \alpha_i R_{i,h}^{\top} \left[\underline{f}_i + \widetilde{K}_{i,h}^{\top} \widetilde{V}_{i,h}^{-1} \underline{g}_i \right] \tag{2.25}$$

with the symmetric and positive definite system matrix

$$\widetilde{S}_h = \sum_{i=1}^{p} \alpha_i R_{i,h}^{\top} \widetilde{S}_{i,h} R_{i,h}, \quad \widetilde{S}_{i,h} = \widetilde{D}_{i,h} + \widetilde{K}_{i,h}^{\top} \widetilde{V}_{i,h}^{-1} \widetilde{K}_{i,h}.$$

The linear system (2.25) can be solved by the preconditioned conjugate gradient method where a possible preconditioner is given by

$$C_S^{-1} = \sum_{i=1}^{p} \frac{1}{\alpha_i} R_{i,h}^{\top} \bar{V}_{i,h} R_{i,h}. \tag{2.26}$$

Here $\bar{V}_{i,h}$ is the single layer potential discretized by piecewise linear continuous basis functions and approximated by means of a fast multipole method. In the case of many subdomains a coarse grid solver must be added in order to ensure a global information exchange. Note that each application of the discrete Steklov–Poincaré operators $\widetilde{S}_{i,h}$ within one step of the conjugate gradient scheme requires one inversion of the discrete single layer potential $\widetilde{V}_{i,h}$ which can be realized by an interior conjugate gradient scheme.

3. Inexact BETI methods. In order to avoid assembled matrices and vectors, we tear the global potential vector $\underline{u}$ on the subdomain boundaries Γ_i by introducing the individual local unknowns

$$\underline{u}_i = R_{i,h} \underline{u}. \tag{3.1}$$

Conversely, the global continuity of the potentials is enforced by the constraints

$$\sum_{i=1}^{p} B_i \underline{u}_i = \underline{0} \tag{3.2}$$

interconnecting the local potential vectors across the subdomain boundaries. Each row of the matrix $B = (B_1, \ldots, B_p)$ is connected with a pair of matching nodes across the subdomain boundaries. The entries of such a row are 1 and -1 for the indices corresponding to the matching nodes on the interface (coupling boundaries) $\Gamma_C = \Gamma_S \setminus \Gamma$ and are 0 otherwise. Therefore, (3.2) implies that the corresponding boundary element functions $u_{i,h}$ are continuous across the interface Γ_C; i.e., $u_{i,h} = u_{j,h}$ on $\Gamma_i \cap \Gamma_j \neq \emptyset$. We assume here that the number of constraints at any such matching node is equal to the number of matching subdomains minus one. This method with a minimal number of constraints (respectively, multipliers) is called nonredundant; see, e.g., [38] for the use of redundant constraints.

By introducing Lagrange multipliers $\underline{\lambda} \in \mathbb{R}^L$, the linear system (2.23) is equivalent to

$$\begin{pmatrix} \alpha_1\widetilde{V}_{1,h} & & & -\alpha_1\widetilde{K}_{1,h} & & & \\ & \ddots & & & \ddots & & \\ & & \alpha_p\widetilde{V}_{p,h} & & & -\alpha_p\widetilde{K}_{p,h} & \\ -\alpha_1\widetilde{K}_{1,h}^\top & & & -\alpha_1\widetilde{D}_{1,h} & & & B_1^\top \\ & \ddots & & & \ddots & & \vdots \\ & & -\alpha_p\widetilde{K}_{p,h}^\top & & & -\alpha_p\widetilde{D}_{p,h} & B_p^\top \\ & & & B_1 & \dots & B_p & \end{pmatrix} \begin{pmatrix} \underline{\widetilde{t}}_1 \\ \vdots \\ \underline{\widetilde{t}}_p \\ \underline{\widetilde{u}}_1 \\ \vdots \\ \underline{\widetilde{u}}_p \\ \underline{\lambda} \end{pmatrix} = \begin{pmatrix} \alpha_1\underline{g}_1 \\ \vdots \\ \alpha_p\underline{g}_p \\ \alpha_1\underline{f}_1 \\ \vdots \\ \alpha_p\underline{f}_p \\ \underline{0} \end{pmatrix}.$$

Setting $V = \operatorname{diag}(\alpha_i\widetilde{V}_{i,h})$, $K = \operatorname{diag}(\alpha_i\widetilde{K}_{i,h})$, $D = \operatorname{diag}(\alpha_i\widetilde{D}_{i,h})$ as well as $B = (B_1, \dots, B_p)$, we have to solve the twofold saddle point problem

$$\begin{pmatrix} V & -K & 0 \\ -K^\top & -D & B^\top \\ 0 & B & 0 \end{pmatrix} \begin{pmatrix} \underline{t} \\ \underline{u} \\ \underline{\lambda} \end{pmatrix} = \begin{pmatrix} \underline{g} \\ \underline{f} \\ \underline{0} \end{pmatrix} \tag{3.3}$$

after ordering the vectors in an appropriate way. The matrix $\widetilde{D}_{i,h}$ is symmetric and positive definite if and only if the corresponding subdomain Ω_i is nonfloating. A subdomain Ω_i is called floating if the subdomain boundary Γ_i does not touch the Dirichlet boundary part Γ_D that equals Γ for our model problem (2.1). Otherwise the subdomain is called nonfloating. For a floating subdomain Ω_i the kernel of the matrix $\widetilde{D}_{i,h}$ is spanned by the vector $\underline{e}_i = (1, \dots, 1)^\top \in \mathbb{R}^{M_i}$; i.e., $\ker \widetilde{D}_{i,h} = \operatorname{span}\{\underline{e}_i\}$. Let us assume for simplicity that the first q ($q < p$) subdomains are floating and the remaining subdomains are nonfloating. The case $q = 0$ (no floating subdomains) is trivial and does not need any discussion. In order to avoid singular block entries $\widetilde{D}_{i,h}$ and, therefore, singular Schur complements $\widetilde{S}_{i,h} = \alpha_i(\widetilde{D}_{i,h} + \widetilde{K}_{i,h}^\top \widetilde{V}_{i,h}^{-1} \widetilde{K}_{i,h})$ for the floating subdomains, we replace the singular blocks $\widetilde{D}_{i,h}$ by the regularized matrices $\widetilde{D}_{i,h} + \beta_i \underline{e}_i \underline{e}_i^\top$ with appropriately chosen positive constants β_i for $i = 1, \dots, q$. We note that

$$\widetilde{K}_{i,h}\underline{e}_i = \underline{0} \quad \text{for } i = 1, \dots, q. \tag{3.4}$$

Let us denote the unique solutions of the regularized equations

$$\alpha_i(\widetilde{D}_{i,h} + \beta_i \underline{e}_i \underline{e}_i^\top)\underline{v}_i \;=\; -\alpha_i \underline{f}_i + B_i\underline{\lambda} - \alpha_i \widetilde{K}_{i,h}^\top \underline{t}_i \tag{3.5}$$

by $\underline{v}_i$, $i = 1, \dots, q$. If the right-hand side of (3.5) fulfills the solvability conditions for the original equations without the regularization term,

$$\underline{e}_i^\top(-\alpha_i \underline{f}_i + B_i\underline{\lambda} - \alpha_i \widetilde{K}_{i,h}^\top \underline{t}_i) \;=\; \underline{e}_i^\top(-\alpha_i \underline{f}_i + B_i\underline{\lambda}) \;=\; 0, \tag{3.6}$$

then the solution $\underline{v}_i$ of (3.5) is orthogonal to $\ker \widetilde{D}_{i,h}$; i.e.,

$$\underline{e}_i^\top \underline{v}_i = 0. \tag{3.7}$$

Now the solution $\underline{u} = (\underline{u}_1, \dots, \underline{u}_p)^\top$ of the original system (3.3) can be recovered by the formulae

$$\underline{u}_i = \underline{v}_i + \gamma_i \underline{e}_i, \tag{3.8}$$

with $\gamma_i = 0$ for $i = q+1, \ldots, p$ and appropriately chosen γ_i for $i = 1, \ldots, q$. Taking (3.4)–(3.8) into account, we easily observe that the unique solution $\underline{t} \in \mathbb{R}^M$, $\underline{v} \in \mathbb{R}^N$, $\underline{\lambda} \in \mathbb{R}^L$, and $\underline{\gamma} \in \mathbb{R}^q$ of the threefold saddle point problem

$$\begin{pmatrix} V & -K & 0 & 0 \\ -K^\top & -D & B^\top & 0 \\ 0 & B & 0 & G \\ 0 & 0 & G^\top & 0 \end{pmatrix} \begin{pmatrix} \underline{t} \\ \underline{v} \\ \underline{\lambda} \\ \underline{\gamma} \end{pmatrix} = \begin{pmatrix} \underline{g} \\ \underline{f} \\ \underline{0} \\ \underline{e} \end{pmatrix} \tag{3.9}$$

immediately yields the solution $(\underline{t}, \underline{u}, \underline{\lambda})$ of the twofold saddle point problem (3.3), where now D is symmetric and positive definite since we have replaced the singular blocks $\widetilde{D}_{j,h}$ by the regularized ones. The last equation in (3.9) is nothing but the solvability condition (3.6). Note that the $L \times q$ matrix G and the vector $\underline{e} \in \mathbb{R}^q$ are defined by the relations

$$G = (B_1 \underline{e}_1, \ldots, B_q \underline{e}_1) \quad \text{and} \quad \underline{e} = (\alpha_1 \underline{e}_1^\top \underline{f}_1, \ldots, \alpha_q \underline{e}_q^\top \underline{f}_q)^\top. \tag{3.10}$$

Now we use a subspace projection in order to separate the determination of $\underline{\gamma}$ from the determination of the rest of the unknowns in (3.9). Thus, we introduce the orthogonal projection

$$P = I - G(G^\top G)^{-1} G^\top \tag{3.11}$$

from the space $\Lambda := \mathbb{R}^M$ onto the subspace $\Lambda_0 = \ker G^\top = (\operatorname{range} G)^\perp$ with respect to the scalar product $(\cdot, \cdot) = (\cdot, \cdot)_\Lambda = (\cdot, \cdot)_{\mathbb{R}^M}$. We mention that in case of large jumps in the coefficients of the PDE the scalar product in Λ has to be changed according to the approach proposed in [38]; see also [6] and [64]. Of course, the change of the scalar product changes the orthoprojection P too.

Since $PG = 0$, the application of P to the third equation of (3.9) gives $PG\underline{\gamma} = 0$ that excludes $\underline{\gamma}$ from the first three equations of (3.9). Let us represent $\underline{\lambda}$ in the form

$$\underline{\lambda} = T_0 \underline{\lambda}_0 + \underline{\lambda}_e \tag{3.12}$$

with known $\underline{\lambda}_e = G(G^\top G)^{-1} \underline{e} \in (\ker G^\top)^\perp = \operatorname{range} G$, fulfilling the constraints $G^\top \underline{\lambda}_e = \underline{e}$, and an unknown $T_0 \underline{\lambda}_0 \in \ker G^\top$, i.e., $G^\top T_0 \underline{\lambda}_0 = 0$, where $\underline{\lambda}_0 \in \mathbb{R}^{L_0}$ and $L_0 = \dim \Lambda_0$. The columns of T_0 span a basis of Λ_0. We refer to Remark 3.1 for the practical implementation of the determination of $\underline{\lambda}_0$ by a subspace iteration. We can now define $\underline{t}$, $\underline{v}$, and $\underline{\lambda}_0$ from the twofold saddle point problem

$$\begin{pmatrix} V & -K & 0 \\ -K^\top & -D & B^\top P^\top T_0 \\ 0 & T_0^\top P B & 0 \end{pmatrix} \begin{pmatrix} \underline{t} \\ \underline{v} \\ \underline{\lambda}_0 \end{pmatrix} = \begin{pmatrix} \underline{g} \\ \underline{d} \\ \underline{0} \end{pmatrix}, \tag{3.13}$$

where here $\underline{d} = \underline{f} - B^\top \underline{\lambda}_e$. Once the vectors $\underline{t}$, $\underline{v}$, and $\underline{\lambda}_0$ are defined from (3.13), we get $\underline{\lambda}$ from (3.12), $\underline{\gamma}$ from the third equation of (3.9), i.e.,

$$\underline{\gamma} = -(G^\top G)^{-1} G^\top B \underline{v}, \tag{3.14}$$

and, finally, $\underline{u}$ from (3.8).

Remark 3.1. We mention that the iteration updates for $\underline{\lambda}$ completely live in the subspace Λ_0 if the initial guess is chosen as $\underline{\lambda}_e$. Therefore, a basis of Λ_0, forming the

columns of T_0, is not explicitly needed in the computation (see also Algorithm 1 in section 5).

Since $V = \mathrm{diag}(\alpha_i \widetilde{V}_{i,h})$ is block diagonal and therefore easily invertible we also may eliminate the vector $\underline{t}$ in (3.13) to obtain

$$\begin{pmatrix} S & -B^\top P^\top T_0 \\ -T_0^\top P B & 0 \end{pmatrix} \begin{pmatrix} \underline{v} \\ \underline{\lambda}_0 \end{pmatrix} = \begin{pmatrix} -\underline{d} - K^\top V^{-1} \underline{g} \\ \underline{0} \end{pmatrix} \tag{3.15}$$

with

$$S = \mathrm{diag}(\alpha_i \widetilde{S}_{i,h}), \quad \widetilde{S}_{i,h} = \widetilde{D}_{i,h} + \widetilde{K}_{i,h}^\top \widetilde{V}_{i,h}^{-1} \widetilde{K}_{i,h}.$$

Eliminating $\underline{v}$ we have to solve the Schur complement system of (3.15),

$$\hat{F} \underline{\lambda}_0 = T_0^\top P B S^{-1} B^\top P^\top T_0 \underline{\lambda}_0 = T_0^\top P B S^{-1} [\underline{d} + K^\top V^{-1} \underline{g}]. \tag{3.16}$$

Appropriate strategies for solving (3.16) have been considered in [43]; another choice is the application of a Bramble–Pasciak conjugate gradient scheme to the onefold saddle point problem (3.15). Note that appropriate preconditioners for both methods are given by preconditioners already needed for solving (3.13); see section 5.

4. Solution of twofold saddle point problems. In this section we construct and analyze preconditioned CG-like iterative methods for solving twofold saddle point problems possessing exactly the structure and the properties of the twofold saddle point problem (3.13), namely,

$$\begin{pmatrix} A_1 & B_1^\top & 0 \\ B_1 & -A_2 & B_2^\top \\ 0 & B_2 & 0 \end{pmatrix} \begin{pmatrix} \underline{x}_1 \\ \underline{x}_2 \\ \underline{x}_3 \end{pmatrix} = \begin{pmatrix} \underline{b}_1 \\ \underline{b}_2 \\ \underline{b}_3 \end{pmatrix}, \tag{4.1}$$

where $S_1 = A_1$, A_2, $S_2 = A_2 + B_1 A_1^{-1} B_1^\top$, and $S_3 = B_2 S_2^{-1} B_2^\top$ are symmetric and positive definite. It is easy to see that the system matrix

$$\mathcal{K}_3 = \begin{pmatrix} A_1 & B_1^\top & 0 \\ B_1 & -A_2 & B_2^\top \\ 0 & B_2 & 0 \end{pmatrix}$$

of (4.1) possesses a block LU-decomposition:

$$\mathcal{K}_3 = \mathcal{L}_3 \mathcal{U}_3 \tag{4.2}$$

with

$$\mathcal{L}_3 = \begin{pmatrix} S_1 & & \\ B_1 & -S_2 & \\ & B_2 & -S_3 \end{pmatrix} \quad \text{and} \quad \mathcal{U}_3 = \begin{pmatrix} I & S_1^{-1} B_1^T & \\ & I & -S_2^{-1} B_2^T \\ & & I \end{pmatrix}.$$

The factor $\mathcal{L}_3$ motivates the following construction of a preconditioner $\hat{\mathcal{L}}_3$: Start by replacing S_1, S_2, and S_3 by preconditioners $\hat{S}_1$, $\hat{S}_2$, and $\hat{S}_3$. Additionally, we introduce two scaling factors. First the left upper block is premultiplied by some factor $\tau_1 > 0$ leading to a block 2×2 submatrix

$$\hat{\mathcal{L}}_2 = \begin{pmatrix} \tau_1 \hat{S}_1 & \\ B_1 & -\hat{S}_2 \end{pmatrix}.$$

We end up with the final form of the preconditioner by premultiplying this submatrix by a second scaling factor $\tau_2 > 0$:

$$\hat{\mathcal{L}}_3 = \begin{pmatrix} \tau_2 \hat{\mathcal{L}}_2 & \\ B_2 \mathsf{E}_2 & \hat{S}_3 \end{pmatrix} = \begin{pmatrix} \tau_2\tau_1 \hat{S}_1 & & \\ \tau_2 B_1 & -\tau_2 \hat{S}_2 & \\ & B_2 & \hat{S}_3 \end{pmatrix}$$

with $\mathsf{E}_2 = \begin{pmatrix} 0 & I \end{pmatrix}$. Now we have the following theorem.

THEOREM 4.1. [illegible] τ_1 [illegible]

$$0 < \tau_1 < \lambda_{\min}(\hat{S}_1^{-1} A_1),$$

[illegible] $\hat{\mathcal{L}}_2^{-1}\mathcal{K}_2$ [illegible] $(u, v)_2 = (\mathcal{D}_2 u, v)_{\ell_2}$ [illegible]

$$\mathcal{K}_2 = \begin{pmatrix} A_1 & B_1^T \\ B_1 & -A_2 \end{pmatrix} \qquad \mathcal{D}_2 = \begin{pmatrix} A_1 - \tau_1 \hat{S}_1 & \\ & \hat{S}_2 \end{pmatrix}.$$

[illegible] τ_2 [illegible]

$$0 < \tau_2 < \lambda_{\min}(\hat{\mathcal{L}}_2^{-1}\mathcal{K}_2),$$

[illegible] $\hat{\mathcal{L}}_3^{-1}\mathcal{K}_3$ [illegible] $(u, v)_3 = (\mathcal{D}_3 u, v)_{\ell_2}$ [illegible]

$$\mathcal{D}_3 = \begin{pmatrix} \mathcal{D}_2(\hat{\mathcal{L}}_2^{-1}\mathcal{K}_2 - \tau_2 I) & 0 \\ 0 & \hat{S}_3 \end{pmatrix}.$$

For the proof, see [68].

This theorem shows that we can apply the standard conjugate gradient method in the scalar product $(u, v)_3 = (\mathcal{D}_3 u, v)_{\ell_2}$ to the preconditioned system

$$\hat{\mathcal{L}}_3^{-1}\mathcal{K}_3 \, \underline{x} = \hat{\mathcal{L}}_3^{-1} \underline{b},$$

which is equivalent to (4.1), with $\underline{x} = (\underline{x}_1, \underline{x}_2, \underline{x}_3)^\top$ and $\underline{b} = (\underline{b}_1, \underline{b}_2, \underline{b}_3)^\top$. Then we obtain the following well-known error estimate for the nth iterate $x^{(n)}$ of the conjugate gradient method in the corresponding energy norm:

$$\|x - x^{(n)}\|_{\mathcal{D}_3 \hat{\mathcal{L}}_3^{-1}\mathcal{K}_3} \leq \frac{2q^n}{1 + q^{2n}} \|x - x^{(0)}\|_{\mathcal{D}_3 \hat{\mathcal{L}}_3^{-1}\mathcal{K}_3}$$

with $q = (\sqrt{\kappa(\hat{\mathcal{L}}_3^{-1}\mathcal{K}_3)} - 1)/(\sqrt{\kappa(\hat{\mathcal{L}}_3^{-1}\mathcal{K}_3)} + 1)$, where $\kappa(\hat{\mathcal{L}}_3^{-1}\mathcal{K}_3)$ denotes the relative condition number, $\kappa(\hat{\mathcal{L}}_3^{-1}\mathcal{K}_3) = \lambda_{\max}(\hat{\mathcal{L}}_3^{-1}\mathcal{K}_3)/\lambda_{\min}(\hat{\mathcal{L}}_3^{-1}\mathcal{K}_3)$. In the following we call this preconditioned conjugate gradient method for twofold saddle point problems 2SPCG.

The next theorem provides an estimate for this relative condition number depending on the quality of the preconditioners for the Schur complements.

THEOREM 4.2. [illegible]

$$\underline{\sigma}_1 \hat{S}_1 \leq S_1 \leq \overline{\sigma}_1 \hat{S}_1, \quad \underline{\sigma}_2 \hat{S}_2 \leq S_2 \leq \overline{\sigma}_2 \hat{S}_2, \quad \underline{\sigma}_3 \hat{S}_3 \leq S_3 \leq \overline{\sigma}_3 \hat{S}_3,$$

[illegible] τ_1 [illegible] τ_2 [illegible]

$$\tau_1 < \underline{\lambda}_1, \quad \tau_2 < \underline{\lambda}_2,$$

$\hat{\mathcal{L}}_2^{-1}\mathcal{K}_2$ $\hat{\mathcal{L}}_3^{-1}\mathcal{K}_3$ $(\cdot,\cdot)_2$ $(\cdot,\cdot)_3$

$$\lambda_{\min}(\hat{\mathcal{L}}_3^{-1}\mathcal{K}_3) \geq \underline{\lambda}_3, \quad \lambda_{\max}(\hat{\mathcal{L}}_3^{-1}\mathcal{K}_3) \leq \overline{\lambda}_3, \qquad \kappa(\hat{\mathcal{L}}_3^{-1}\mathcal{K}_3) \leq \frac{\overline{\lambda}_3}{\underline{\lambda}_3},$$

$\underline{\lambda}_1 = \underline{\sigma}_1$

$$\underline{\lambda}_{i+1} = \frac{1}{2\tau_i}\left(\overline{\sigma}_i(\underline{\sigma}_{i+1}+1) - \sqrt{\overline{\sigma}_i^2(\underline{\sigma}_{i+1}+1)^2 - 4\tau_i\overline{\sigma}_i\underline{\sigma}_{i+1}}\right) \quad i = 1, 2,$$

$\overline{\lambda}_3$

$$\overline{\theta}_3(\lambda) = -\frac{\lambda}{\overline{\sigma}_3} + \cfrac{1-\lambda}{-\cfrac{\tau_2\lambda}{\overline{\sigma}_2} + \cfrac{1-\tau_2\lambda}{-\cfrac{\tau_1\tau_2\lambda}{\overline{\sigma}_1}+1}}.$$

For the proof, see [68].

4.1.

1. It is necessary to keep the parameters τ_1 and τ_2 below the limiting values $\underline{\lambda}_1$ and $\underline{\lambda}_2$ in order to guarantee a well-defined scalar product $(u, v)_3$. On the other hand, the closer these parameters are to their limiting values, the better it is, i.e., the smaller is the upper bound $\overline{\lambda}_3/\underline{\lambda}_3$ for the relative condition number.

2. If the parameters τ_1 and τ_2 are chosen close to their limiting values $\underline{\lambda}_1$ and $\underline{\lambda}_2$ in the sense

$$\frac{\underline{\lambda}_i}{\tau_i} = \mathcal{O}(1) \quad \text{for } i = 1, 2$$

and if the spectral bounds are reasonably scaled, i.e.,

$$1 + \underline{\sigma}_i\overline{\sigma}_i \leq \mathcal{O}(\overline{\sigma}_i) \quad \text{for } i = 1, 2, 3,$$

it can be shown (see [68]) that

$$\frac{\overline{\lambda}_3}{\underline{\lambda}_3} = \mathcal{O}\left(\frac{\overline{\sigma}_1}{\underline{\sigma}_1}\frac{\overline{\sigma}_2}{\underline{\sigma}_2}\frac{\overline{\sigma}_3}{\underline{\sigma}_3}\right).$$

5. Data-sparse BEM preconditioners. As seen in the previous section, we need appropriate preconditioners C_{V_i}, C_{S_i}, and C_F for the local discrete single layer potentials $\widetilde{V}_{i,h}$, the local discrete Schur complements $\widetilde{S}_{i,h}$, and the assembled BETI Schur complement $\hat{F}$, respectively.

Since the bilinear form of the local approximate discrete single layer potential $\widetilde{V}_{i,h}$ is spectrally equivalent to the Galerkin matrix $V_{i,h}$ and therefore to the $H^{-1/2}(\Gamma_i)$-norm, we may use any multilevel representation of an equivalent norm in $H^{-1/2}(\Gamma_i)$ [20, 54]. A suitable multilevel [5] operator is defined by

$$C_{V_i}^{-1}w_{i,h} = \sum_{j=0}^{J} h_{i,j}(Q_{i,j} - Q_{i,j-1})w_{i,h}$$

for $w_{i,h} \in T_{i,h} = T_{i,J}$, where $T_{i,0} \subset T_{i,1} \subset \cdots \subset T_{i,J} = T_{i,h} \subset \cdots$ is a nested sequence of boundary element spaces of piecewise constant basis functions with related mesh

sizes $h_{i,j} = \frac{1}{2}h_{i,j-1}$. Moreover, $Q_{i,j} : L_2(\Gamma_i) \to T_{i,j}$ is the L_2 Galerkin projection onto $T_{i,j}$.

LEMMA 5.1 (see [54, Theorem 2]).

$$c_1^{V_i}\, J^{-2}\, \langle C_{V_i} w_{i,h}, w_{i,h}\rangle_{\Gamma_i} \;\le\; (\widetilde{V}_{i,h}\underline{w}_i, \underline{w}_i) \;\le\; c_2^{V_i}\, \langle C_{V_i} w_{i,h}, w_{i,h}\rangle_{\Gamma_i}$$

$w_{i,h} \in T_{i,J} = T_{i,h}$ $J = \mathcal{O}(1 + \log(H/h))$

Note that in the case of piecewise constant basis functions the L_2 projections $Q_{i,j}w_i$ can be computed directly by inverting diagonal matrices.

The crucial issue in the multilevel preconditioning approach is the availability of a nested hierarchy of boundary element spaces $T_{i,j}$. Starting from a coarse mesh with an associated space $T_{i,0}$, one can define all boundary element spaces $T_{i,j}$ and therefore $T_{i,h} = T_{i,J}$ by a recursive refinement procedure. Instead of this more academic situation we now consider the case where the boundary element space $T_{i,h} = T_{i,J}$ is already given and no further refinement is applicable. In this case we have to construct an artificial mesh hierarchy to define an appropriate multilevel operator. For this we will again use ideas from fast BEM; in particular we will introduce an appropriate clustering hierarchy of all boundary elements as is used in the fast multipole method itself. Using a recursive bisection algorithm, all boundary elements τ_k^i belonging to a subdomain boundary Γ_i can be clustered hierarchically. Since the boundary element spaces $T_{i,h}$ are spanned by piecewise constant basis functions, one can define corresponding piecewise constant coarse grid functions with respect to the cluster of underlying boundary elements. This results in an artificial hierarchy of piecewise constant trial spaces which can be used for the definition of the multilevel operator for preconditioning [59].

To construct preconditioning matrices C_{S_i} for the local discrete Schur complement matrices $\widetilde{S}_{i,h} = \widetilde{D}_{i,h} + \widetilde{K}_{i,h}^\top \widetilde{V}_{i,h}^{-1} \widetilde{K}_{i,h}$, we will apply the concept of boundary integral operators of the opposite order [62]. Based on the local trial space $U_{i,h} = S_h^1(\Gamma_i)$ of piecewise linear basis functions φ_m^i as used for the Galerkin discretization of the local hypersingular boundary integral operators D_i, we define the Galerkin matrices

$$\bar{V}_{i,h}[n,m] \;=\; \langle \varphi_n^i, V\varphi_m^i\rangle_{\Gamma_i}, \quad \bar{M}_{i,h}[n,m] \;=\; \langle \varphi_n^i, \varphi_m^i\rangle_{\Gamma_i}$$

for $m, n = 1, \ldots, M_i$, and the application of the resulting preconditioning matrix is given by

$$C_{S_i}^{-1} \;=\; \bar{M}_{i,h}^{-1} \bar{V}_{i,h} \bar{M}_{i,h}^{-1} \quad \text{for } i = 1, \ldots, p. \tag{5.1}$$

When the application of the discrete single layer potential $\bar{V}_{i,h}$ is realized by using a fast multipole method as described in section 2, this results in an approximate preconditioning matrix $\widetilde{C}_{S_i}$.

LEMMA 5.2. Ω_i $\Gamma_i \cap \Gamma \neq \emptyset$

$$c_1^{S_i}\, [1 + \log(H/h)]^{-2}\, (\widetilde{C}_{S_i}\underline{v}_i, \underline{v}_i) \;\le\; (\widetilde{S}_{i,h}\underline{v}_i, \underline{v}_i) \;\le\; c_2^{S_i} (\widetilde{C}_{S_i}\underline{v}_i, \underline{v}_i) \tag{5.2}$$

$\underline{v}_i \in \mathbb{R}^{M_i}$ Ω_i $\Gamma_i \cap \Gamma = \emptyset$

$$c_1^{S_i}\, (\widetilde{C}_{S_i}\underline{v}_i, \underline{v}_i) \;\le\; (\widetilde{S}_{i,h}\underline{v}_i, \underline{v}_i) \;\le\; c_2^{S_i} (\widetilde{C}_{S_i}\underline{v}_i, \underline{v}_i) \tag{5.3}$$

$\underline{v}_i \in \mathbb{R}^{M_i}$ [illegible] $\underline{e}_i^\top \underline{v}_i = 0$ [illegible] $c_1^{S_i}$ [illegible] $c_2^{S_i}$ [illegible] h [illegible] H [illegible]. Since the approximated discrete single layer potential $\widetilde{V}_{i,h}$ is spectrally equivalent to the Galerkin matrix $V_{i,h}$ and therefore positive definite, and using the representation (2.22) of the approximate hypersingular boundary integral operator $\widetilde{D}_i$, we first conclude that

$$\begin{aligned}(\widetilde{S}_{i,h}\underline{v}_i, \underline{v}_i) &\geq (\widetilde{D}_{i,h}\underline{v}_i, \underline{v}_i) + (\widetilde{V}_{i,h}^{-1}\widetilde{K}_{i,h}\underline{v}_i, \widetilde{K}_{i,h}\underline{v}_i) \;\geq\; (\widetilde{D}_{i,h}\underline{v}_i, \underline{v}_i) \\ &= \sum_{k=1}^{3} \langle \widetilde{V}_{i,h}\underline{\mathrm{curl}}_{|\Gamma_i} v_{i,h|k}, \underline{\mathrm{curl}}_{|\Gamma_i} v_{i,h|k}\rangle_{\Gamma_i} \\ &\geq c \sum_{k=1}^{3} \langle V_i \underline{\mathrm{curl}}_{|\Gamma_i} v_{i,h|k}, \underline{\mathrm{curl}}_{|\Gamma_i} v_{i,h|k}\rangle_{\Gamma_i} \;=\; c\,(D_{i,h}\underline{v}_i, \underline{v}_i)\end{aligned}$$

for all $\underline{v}_i \in \mathbb{R}^{M_i}$. The approximate discrete Steklov–Poincaré operator $\widetilde{S}_{i,h}$ is also bounded from above [9, 60]; i.e.,

$$(\widetilde{S}_{i,h}\underline{v}_i, \underline{v}_i) \;\leq\; (S_{i,h}\underline{v}_i, \underline{v}_i) \;=\; \langle S_i v_{i,h}, v_{i,h}\rangle_{\Gamma_i} \;\leq\; c\,\langle D_i v_{i,h}, v_{i,h}\rangle_{\Gamma_i} \;=\; c\,(D_{i,h}\underline{v}_i, \underline{v}_i)$$

holds for all $\underline{v}_i \in \mathbb{R}^{M_i}$. Hence, in order to find a preconditioner for $\widetilde{S}_{i,h}$ it is sufficient to find a preconditioner for the discrete hypersingular integral operator $D_{i,h}$. In the case of a floating subdomain Ω_i, the preconditioning matrix $C_{S_i} = \bar{M}_{i,h}\bar{V}_{i,h}M_{i,h}$ is spectrally equivalent to the discrete hypersingular integral operator $D_{i,h}$ [62], and since the multipole approximation of $\bar{V}_{i,h}$ is spectrally equivalent to $\bar{V}_{i,h}$, the same is true for the approximate preconditioner $\widetilde{C}_{S_i}$; see also [52]. Hence we obtain the spectral equivalence inequalities (5.3).

If the subdomain Ω_i is a nonfloating subdomain, then we have to use the trial space $U_{i,h} \subset \widetilde{H}^{1/2}(\Gamma_i \backslash (\Gamma \cap \Gamma_i))$ of piecewise linear basis functions vanishing on the Dirichlet boundary Γ. Due to the mapping properties of the boundary integral operators

$$\begin{aligned}D_i &: \widetilde{H}^{1/2}(\Gamma_i \backslash (\Gamma \cap \Gamma_i)) \to H^{-1/2}(\Gamma_i \backslash (\Gamma \cap \Gamma_i)), \\ V_i &: \widetilde{H}^{-1/2}(\Gamma_i \backslash (\Gamma \cap \Gamma_i)) \to H^{1/2}(\Gamma_i \backslash (\Gamma \cap \Gamma_i)),\end{aligned}$$

we have to consider the discrete embedding

$$\|v_h\|^2_{H^{1/2}(\Gamma_i \backslash (\Gamma \cap \Gamma_i))} \;\leq\; c\,[1 + \log(H/h)]^2\, \|v_h\|^2_{\widetilde{H}^{1/2}(\Gamma_i \backslash (\Gamma \cap \Gamma_i))}$$

for $v_h \in U_{i,h}$ yielding the spectral equivalence inequalities (5.2); see [49] together with an appropriate scaling argument. □

We finally have to describe a suitable preconditioning matrix for the BETI Schur complement

$$F \;=\; \sum_{i=1}^{q} \frac{1}{\alpha_i} B_i [\widetilde{S}_{i,h} + \beta_i \underline{e}_i \underline{e}_i^\top]^{-1} B_i^\top \;+\; \sum_{i=q+1}^{p} \frac{1}{\alpha_i} B_i \widetilde{S}_{i,h}^{-1} B_i^\top \,.$$

Following [43] we can define the scaled sparse hypersingular BETI preconditioner

$$C_F^{-1} = (BC_\alpha^{-1}B^T)^{-1} BC_\alpha^{-1} \widetilde{D}_h C_\alpha^{-1} B^T (BC_\alpha^{-1}B^\top)^{-1}. \tag{5.4}$$

LEMMA 5.3 (see [43, Theorem 3.2]).

(5.4)

$$c_1^F\,(C_F\underline{\mu},\underline{\mu}) \le (F\underline{\mu},\underline{\mu}) \le c_2^F\,(1+\log(H/h))^2\,(C_F\underline{\mu},\underline{\mu}) \tag{5.5}$$

$\underline{\mu} \in \Lambda_0 = \ker G^\top$ c_1^F c_2^F
h H p α_i

Combining Theorem 4.2 with the results of Lemmata 5.1–5.3 and taking into account the complexity estimate for the fast multipole method, we finally arrive at our main theorem.

THEOREM 5.4. (3.13) 2
C_V C_S
C_F $I(\varepsilon) = \mathcal{O}((1+\log(H/h))^3 \log \varepsilon^{-1})$
$(\varepsilon) = \mathcal{O}((H/h)^2(1+\log(H/h))^5 \log \varepsilon^{-1})$
$\varepsilon \in (0,1)$
$I(\varepsilon)$
$\mathcal{O}((H/h)^2(1+\log(H/h))^2)$

Note that the complexity estimates given in Theorem 5.4 do not take into account the costs for the projection (3.11) that involves the global information exchange.

5.1. If we use optimal preconditioners $\widetilde{C}_{V_i}$ for the local single layer potentials $\widetilde{V}_{i,h}$ and $\widetilde{C}_{S_i}$ for the local boundary element Schur complements $\widetilde{S}_{i,h}$, the number of iterations $I(\varepsilon)$ of our 2SPCG solver would behave like $\mathcal{O}((1+\log(H/h))\log\varepsilon^{-1})$, and the arithmetical complexity would decrease from $\mathcal{O}((H/h)^2(1+\log(H/h))^5\log\varepsilon^{-1})$ to $\mathcal{O}((H/h)^2(1+\log(H/h))^3\log\varepsilon^{-1})$. Such preconditioners are available; see, e.g., [42, 56, 62, 63]. If we convert the nonfloating subdomains having a Dirichlet boundary part to floating subdomains by including the Dirichlet boundary condition into the constraints (all-floating BETI), then the data-sparse opposite order preconditioners $\widetilde{\mathcal{S}}_{i,h}$ given above would be optimal.

We mention that h/H can be replaced by $\max\{h_i/H_i\}$, where H_i and h_i are now the scaling parameters of Ω_i and the mesh on Γ_i. Therefore, the global quasi-uniformity assumption can be replaced by a local one.

Applying the 2SPCG introduced in section 4 to our twofold saddle point problem (3.13) with the preconditioners presented above and taking Remark 3.1 into account, we obtain a subspace iteration version of the 2SPCG that can be rewritten in the algorithmic form given in Algorithm 1.

Note that, for an efficient implementation, the applications of $\hat{\mathcal{L}}_3^{-1}$ and $\mathcal{D}_3\hat{\mathcal{L}}_3^{-1}$ have to be concatenated; see [61, 68]. Here, we have used the notation of section 4 for the matrix in (3.13) and a vector $\underline{x} = (\underline{x}_1, \underline{x}_2, \underline{x}_3)^T$, where $\underline{x}_i$ corresponds to the ith block of the system (3.13).

Algorithm 1 was used in our numerical experiments, the results of which are given in the next section.

6. Numerical results. For the numerical examples we consider the unit cube which is subdivided into eight similar subdomains as shown in Figure 6.1. The given Dirichlet data $g(x)$ is chosen as the trace of a regular solution of the boundary value problem (2.1). This allows us to check the convergence of the computed numerical solution to the exact one.

In Table 6.1, the geometric information about the domain decomposition and the discretization is listed for the refinement levels L used. Starting from the coarsest grid with 192 triangles for the whole domain Ω, the refined meshes are recursively

ALGORITHM 1. Inexact BETI twofold saddle point problem CG.

$\underline{\lambda}_e = QG^\top (G^\top QG)^{-1} \underline{e}$ {forcing the constraint $G^\top \underline{\lambda}_e = \underline{e}$}
$\underline{x}_1^0 = \underline{0}$, $\underline{x}_2^0 = \underline{0}$, $\underline{x}_3^0 = \underline{\lambda}_e$ {choose the initial guess}
$\underline{b}_1 = \underline{g}$, $\underline{b}_2 = \underline{d} = \underline{f} - B^\top \underline{\lambda}_e$, $\underline{b}_3 = \underline{0}$ {compute right-hand side}
$\underline{d} = \mathcal{K}_3 \underline{x}^0 - \underline{b}$ {compute the defect}
$\underline{r}^0 = \mathcal{D}_3 \hat{\mathcal{L}}_3^{-1} \underline{d}$ {apply the transformation}
$\underline{p}^0 = \underline{w}^0 = \mathcal{D}_3^{-1} \underline{r}^0$ {preconditioning}
$\rho_0 = (\underline{w}^0, \underline{r}^0)$
for $n = 0$ **step** 1 **until** $\rho_n \le \varepsilon \rho_0$ **do**
  $\underline{v}^n = \mathcal{D}_3 \hat{\mathcal{L}}_3^{-1} \mathcal{K}_3 \underline{p}^n$ {matrix vector multiplication and transformation}
  $\sigma = (\underline{v}^n, \underline{p}^n)$
  $\alpha = \rho_{n+1} / \sigma$
  $\underline{x}^{n+1} = \underline{x}^n - \alpha \underline{p}^n$ {update the iterate}
  $\underline{r}^{n+1} = \underline{r}^n - \alpha \underline{v}^n$ {update the defect}
  $\underline{w}^{n+1} = \mathcal{D}_3^{-1} \underline{r}^{n+1}$ {preconditioning}
  $\rho_{n+1} = (\underline{w}^{n+1}, \underline{r}^{n+1})$
  $\beta = \rho_{n+1} / \rho_n$
  $\underline{p}^{n+1} = \underline{w}^{n+1} + \beta \underline{p}^n$ {update of the search direction}
end for

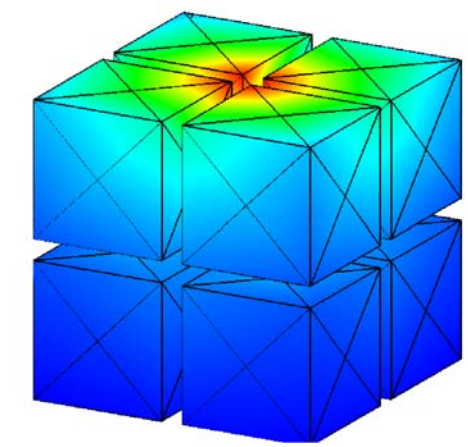


FIG. 6.1. *Domain decomposition with* 8 *subdomains.*

constructed by subdividing each triangle into four smaller similar triangles. N and M are the total numbers of triangles and nodes, respectively. M_c is the total number of coupling nodes. The numbers of local triangles and nodes of a single subdomain Ω_i are given by N_i and M_i, respectively. If the boundary mesh of one subdomain Ω_i with 98304 triangles was uniformly extended to the interior of the subdomain, the corresponding FEM grid would consist of 6291456 tetrahedra. For the whole domain, more than 50 million tetrahedra would be used in a finite element discretization.

In all following tables, a unified notation is used. L again denotes the refinement level. Table 6.1 provides the corresponding information about the grids. The times t_1 and t_2 are the measured times in seconds for setting up the systems of linear equations

TABLE 6.1
Geometric information.

L	N	M	M_c	N_i	M_i
0	192	63	13	24	14
1	768	261	67	96	50
2	3072	1089	319	384	194
3	12288	4473	1399	1536	770
4	49152	18153	5863	6144	3074
5	196608	73161	24007	24576	12290
6	786432	293769	97159	98304	49154

TABLE 6.2
Primal Schur complement system (2.25).

	Unpreconditioned				Preconditioned			
L	t_1	t_2	it	error	t_1	t_2	it	error
0	0	0	4(6)	3.2954e-2	0	0	4(11)	3.2954e-2
1	1	0	12(21)	5.1662e-3	1	0	14(22)	5.1662e-3
2	2	3	17(33)	1.1907e-3	4	3	20(28)	1.1907e-3
3	9	30	25(46)	2.8996e-4	13	19	22(31)	2.8996e-4
4	35	380	36(65)	6.9990e-5	48	150	24(35)	6.9989e-5
5	159	4912	52(87)	1.7508e-5	178	1194	26(38)	1.7506e-5
6	807	40294	74(122)	4.3532e-6	844	6099	29(41)	4.3488e-6

and for their solution. The column shows the number of iterations needed to solve the corresponding system of linear equations with a relative accuracy of $\varepsilon = 10^{-8}$. In the column the error $\|u - u_h\|_{L_2(\Gamma_i)}$ is listed for the different levels considered.

The effects of the preconditioners for inverting the single layer potential in the representation of the Steklov–Poincaré operator and for the Schur complement system (2.25) by the preconditioner (2.26) are shown in Table 6.2. Setting up the preconditioners takes some time as can be seen in the differences of t_1 for the unpreconditioned and preconditioned cases. But the preconditioning pays off as the numbers of iterations are reduced significantly. In parentheses, the average numbers of iterations are given for the inversion of the local single layer potentials. Accordingly, the times t_2 for solving the system of linear equations are reduced, while the errors are about the same. These calculations have been executed on a standard PC cluster with 3.06 MHz Intel processors and 1 GB of RAM.

Next, the Schur complement system (2.25) will be compared to three different BETI formulations. Since the times t_1 to set up the system of linear equations differ only slightly from each other, only the times t_2 for solving and the numbers *it* of iterations will be compared in Table 6.3. Only the numbers for the preconditioned systems are treated. The numbers of iterations differ slightly from those of Table 6.2, as a slightly different preconditioner C_V^{-1} was used for the inversion of the local single layer potentials $\widetilde{V}_{i,h}$. The computational times are a little bit higher than in the example of Table 6.2 as the computations have been executed on a different cluster and different boundary data have been used. All further computations have been executed in parallel with one processor per subdomain on the Mozart cluster of the Chair of Simulation of Large Systems and the Chair of Numerics for Supercomputers at the University of Stuttgart. The cluster consists of 64 nodes with 2 CPU Intel Xeon 3,066 and 4 GB RAM each. The errors are almost the same for the different formulations.

TABLE 6.3
Comparison of standard domain decomposition (DD) and BETI for uniform coefficients.

	(2.25)		(3.16)		(3.15)		(3.3)	
L	t_2	it	t_2	it	t_2	it	t_2	it
0	0	0(0)	1	2(2(4))	1	5(4)	2	36
1	2	14(19)	5	6(6(18))	3	18(18)	2	50
2	4	18(27)	32	11(14(28))	10	28(28)	8	62
3	23	21(32)	248	12(16(33))	72	34(32)	56	78
4	155	22(37)	2113	14(19(36))	546	39(35)	363	89
5	1523	27(40)	20908	15(21(39))	4873	43(38)	2938	101
6	8267	30(43)	123454	16(25(43))	25759	47(42)	15038	116

The first BETI formulation is the Schur complement system (3.16). This system involves two recursive inversions in each iteration step. The iteration numbers given in parentheses are average numbers needed to invert the local Steklov–Poincaré operators and the local single layer potential, respectively, while the other number is the number of iterations for solving the BETI Schur complement system (3.16). For example, on the sixth refinement level about $16 \cdot 25$ inversions of the local single layer potentials are executed, and the average number of iterations needed for such an inversion is 43.

The second BETI formulation is the saddle point problem (3.15). This system needs only the inversions of the local single layer potentials in each iteration step of the system, while the local potentials $\underline{u}_i$ are determined simultaneously in the global system. Due to the larger number of unknowns in the global system, the number of iterations increases but not too strongly. Therefore the total number of applications of the Galerkin matrices of the local boundary integral operators is less than that of the BETI Schur complement system. Correspondingly, the improvement on the times t_2 for solving the system is significant.

A further improvement on the times t_2 can be made by using the third BETI formulation, the twofold saddle point problem given by (3.3). In this formulation, no inner inversion is needed, but the number of unknowns in the system is increased to almost 800000 for the local fluxes $\underline{t}_i$ on the sixth level. Therefore, the number of iterations again increases. Correspondingly, the number of applications of the Galerkin matrices of the local single layer potentials is again decreased. The benefit of solving the system is smaller than in the case before, as the number of applications of the Galerkin matrices of the other boundary integral operator increases. Unfortunately, these applications are more expensive. The twofold saddle point formulation (3.3) results in the most efficient BETI algorithm in comparison with algorithms based on the other BETI formulations. But it still performs worse than the Schur complement system (2.25).

The systems (3.15) and (3.3) have been solved as the transformed systems described in section 4 by using Algorithm 1. The preconditioners used have been scaled optimally according to the estimates of the eigenvalues of the transformed systems given by Theorem 4.2.

The BETI methods perform much better when considering jumping coefficients instead of constant coefficients as in the previous example. In Table 6.4, the times t_2 for solving the linear systems and the corresponding numbers of iterations are itemized for the coefficients $\alpha_1 = 1$ and $\alpha_2 = 10^5$ distributed on the subdomains as on a checkerboard. Again, the three different BETI formulations are compared to the Schur complement system (2.25).

Comparing the iteration numbers in Tables 6.3 and 6.4, we observe that the scaled

TABLE 6.4
Comparison of standard DD and BETI for jumping coefficients.

	(2.25)		(3.16)		(3.15)		(3.3)	
L	t_2	it	t_2	it	t_2	it	t_2	it
0	2	8(12)	2	2(2(4))	1	5(4)	1	30
1	2	16(19)	4	6(6(18))	4	17(18)	2	43
2	5	23(27)	29	10(14(28))	10	25(26)	8	58
3	29	28(32)	245	12(16(32))	70	32(31)	51	71
4	226	33(36)	1929	13(19(36))	497	36(34)	328	80
5	2439	44(39)	17762	13(21(39))	4442	40(37)	2593	89
6	15045	55(44)	106341	14(23(43))	24748	45(41)	13667	106

version of the BETI preconditioner performs even better. Moreover, this example confirms the independence of the iteration numbers from the jumps in the coefficients as predicted by the theory presented in sections 4 and 5. On the other hand, the iteration numbers for solving the Schur complement system of (2.16) increase in the case of jumping coefficients, even though the preconditioner still works.

Comparing the times t_2 for solving the corresponding systems, the numbers are a lot better for the BETI method. The twofold saddle point formulation (3.3) of the BETI method outperforms the Schur complement system (2.25) on the sixth refinement level, and one would expect an even larger difference on the seventh level.

The improvement of the preconditioners C_{V_i} for the single layer potential blocks $\widetilde{V}_{i,h}$ leads to a significant improvement of the preconditioner for the complete twofold BETI saddle point system (see [41], where AMG preconditioners are used for $\widetilde{V}_{i,h}$).

7. Conclusions and final remarks. In this paper, we have presented inexact data-sparse boundary element tearing and interconnecting domain decomposition methods for solving boundary value problems for the potential equation with piecewise constant coefficients. In contrast to the classical approach we have avoided the elimination of local unknowns $\underline{t}_i$ and $\underline{u}_i$. This finally led us to a twofold saddle point problem. We have presented preconditioned CG-like iterative methods for solving such twofold saddle point problems and have given a rigorous analysis of the preconditioning problem resulting in precise CG-like convergence rate estimates. Furthermore, we have presented data-sparse preconditioners which yield almost optimal iterative solvers for the inexact data-sparse BETI equations. Moreover, the convergence rate is not affected by large jumps in the coefficients of the PDE. This is confirmed by all our numerical experiments. The data-sparse preconditioners proposed in this paper do not need special constructions since they are available in the BETI discretization anyway. Thus, they are completely algebraic. This is very important for the practical acceptance of our method. It is clear that inexact data-sparse BETI allows us to solve really large-scale problems with several million boundary unknowns on distributed memory parallel computers very efficiently.

The treatment of the exterior Dirichlet problem and other boundary conditions is straightforward. The generalization of our method to linear elasticity problems with piecewise constant materials is also possible [51]. The all-floating BETI method that has recently been proposed by Of in [51] provides a unified framework for treating mixed boundary value problems. Using the symmetric coupling technique proposed by Costabel [10] and combining the results of this paper with the results obtained by Klawonn and Widlund [37], we can easily construct and analyze inexact FETI-BETI solvers for large-scale coupled finite and boundary element equations. Exact FETI-BETI solvers were already proposed and analyzed in [44].

Acknowledgment. The authors would like to express their thanks to the anonymous referees for many helpful hints and suggestions.

REFERENCES

[1] M. Bebendorf, *Approximation of boundary element matrices*, Numer. Math., 86 (2000), pp. 565–589.
[2] M. Bebendorf and S. Rjasanow, *Adaptive low-rank approximation of collocation matrices*, Computing, 70 (2003), pp. 1–24.
[3] C. Börgers, *The Neumann–Dirichlet domain decomposition method with inexact solvers on the subdomains*, Numer. Math., 55 (1989), pp. 123–136.
[4] J. H. Bramble and J. E. Pasciak, *A preconditioning technique for indefinite systems resulting from mixed approximations of elliptic problems*, Math. Comp., 50 (1988), pp. 1–17.
[5] J. H. Bramble, J. E. Pasciak, and J. Xu, *Parallel multilevel preconditioners*, Math. Comp., 55 (1990), pp. 1–22.
[6] S. C. Brenner, *An additive Schwarz preconditioner for the FETI method*, Numer. Math., 94 (2003), pp. 1–31.
[7] F. Brezzi and C. Johnson, *On the coupling of boundary integral and finite element methods*, Calcolo, 16 (1979), pp. 189–201.
[8] J. Carrier, L. Greengard, and V. Rokhlin, *A fast adaptive multipole algorithm for particle simulations*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 669–686.
[9] C. Carstensen, M. Kuhn, and U. Langer, *Fast parallel solvers for symmetric boundary element domain decomposition equations*, Numer. Math., 79 (1998), pp. 321–347.
[10] M. Costabel, *Symmetric methods for the coupling of finite elements and boundary elements*, in Boundary Elements IX, C. A. Brebbia, G. Kuhn, and W. L. Wendland, eds., Springer, Berlin, 1987, pp. 411–420.
[11] M. Costabel, *Boundary integral operators on Lipschitz domains: Elementary results*, SIAM J. Math. Anal., 19 (1988), pp. 613–626.
[12] W. Dahmen, S. Prössdorf, and R. Schneider, *Wavelet approximation methods for pseudodifferential equations* II: *Matrix compression and fast solution*, Adv. Comput. Math., 1 (1993), pp. 259–335.
[13] C. R. Dohrmann, *A preconditioner for substructuring based on constrained energy minimization*, SIAM J. Sci. Comput., 25 (2003), pp. 246–258.
[14] C. Dohrmann, *An Approximate BDDC Preconditioner*, Technical report SAND 2005-5424, Sandia National Laboratories, Albuquerque, NM, 2005.
[15] C. Farhat, M. Lesoinne, and A. P. Macedo, *A two-level domain decomposition method for the iterative solution of high frequency exterior Helmholtz problems*, Numer. Math., 85 (2000), pp. 283–308.
[16] C. Farhat, M. Lesoinne, P. Le Tallec, K. Pierson, and D. Rixen, *FETI-DP: A dual-primal unified FETI method* I: *A faster alternative to the two-level FETI method*, Internat. J. Numer. Methods Engrg., 50 (2001), pp. 1523–1544.
[17] C. Farhat, M. Lesoinne, and K. Pierson, *A scalable dual-primal domain decomposition method*, Numer. Linear Algebra Appl., 7 (2000), pp. 687–714.
[18] C. Farhat and F.-X. Roux, *A method of finite element tearing and interconnecting and its parallel solution algorithm*, Internat. J. Numer. Methods Engrg., 32 (1991), pp. 1205–1227.
[19] C. Farhat and F.-X. Roux, *Implicit parallel processing in structural mechanics*, Comput. Mech. Adv., 2 (1994), pp. 1–124.
[20] S. A. Funken and E. P. Stephan, *The BPX preconditioner for the single layer potential operator*, Appl. Anal., 67 (1997), pp. 327–340.
[21] G. N. Gatica and N. Heuer, *Conjugate gradient method for dual-dual mixed formulations*, Math. Comp., 71 (2002), pp. 1455–1472.
[22] L. Greengard, *The Rapid Evaluation of Potential Fields in Particle Simulation*, The MIT Press, Cambridge, MA, 1987.
[23] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comput. Phys., 73 (1987), pp. 325–348.
[24] G. Haase, B. Heise, M. Kuhn, and U. Langer, *Adaptive domain decomposition methods for finite and boundary element equations*, in Boundary Element Topics, W. L. Wendland, ed., Springer, Berlin, Heidelberg, 1997, pp. 121–147.
[25] G. Haase, U. Langer, and A. Meyer, *The approximate Dirichlet domain decomposition method*, Computing, 47 (1991), pp. 137–167.
[26] G. Haase, U. Langer, A. Meyer, and S. Nepomnyaschikh, *Hierarchical extension operators*

and local multigrid methods in domain decomposition preconditioners, East-West J. Numer. Math., 2 (1994), pp. 173–193.

[27] G. Haase and S. Nepomnyaschikh, *Explicit extension operators on hierarchical grids*, East-West J. Numer. Math., 5 (1997), pp. 231–248.

[28] W. Hackbusch, *A sparse matrix arithmetic based on $\mathcal{H}$-matrices*, Computing, 62 (1999), pp. 89–108.

[29] W. Hackbusch and Z. P. Nowak, *On the fast matrix multiplication in the boundary element method by panel clustering*, Numer. Math., 54 (1989), pp. 463–491.

[30] E. W. Hobson, *The Theory of Spherical and Ellipsoidal Harmonics*, Chelsea, New York, 1955.

[31] G. C. Hsiao, O. Steinbach, and W. L. Wendland, *Domain decomposition methods via boundary integral equations*, J. Comput. Appl. Math., 125 (2000), pp. 521–537.

[32] G. C. Hsiao and W. L. Wendland, *Domain decomposition in boundary element methods*, in Proceedings of the Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations, R. Glowinski, Y. A. Kuznetsov, G. Meurant, J. Périaux, and O. B. Widlund, eds., SIAM, Philadelphia, 1991, pp. 41–49.

[33] C. Johnson and C. Nédélec, *On coupling of boundary integral and finite element methods*, Math. Comp., 35 (1980), pp. 1063–1079.

[34] M. Kaltenbacher, *Numerical Simulation of Mechatronic Sensors and Actuators*, Springer, Heidelberg, 2004.

[35] A. Klawonn, *An iterative substructuring method with Lagrange multipliers for elasticity problems using approximate Neumann subdomain solvers*, in Multifield Problems: State of the Art, A.-M. Sändig, W. Schiehlen, and W. L. Wendland, eds., Springer, Berlin, 2000, pp. 193–200.

[36] A. Klawonn and O. Rheinbach, *Inexact FETI-DP methods*, Internat. J. Numer. Methods Engrg., 69 (2007), pp. 284–307.

[37] A. Klawonn and O. B. Widlund, *A domain decomposition method with Lagrange multipliers and inexact solvers for linear elasticity*, SIAM J. Sci. Comput., 22 (2000), pp. 1199–1219.

[38] A. Klawonn and O. B. Widlund, *FETI and Neumann–Neumann iterative substructuring methods: Connections and new results*, Comm. Pure Appl. Math., 54 (2001), pp. 57–90.

[39] A. Klawonn, O. B. Widlund, and M. Dryja, *Dual-primal FETI methods for three-dimensional elliptic problems with heterogeneous coefficients*, SIAM J. Numer. Anal., 40 (2002), pp. 159–179.

[40] U. Langer, *Parallel iterative solution of symmetric coupled FE/BE-equations via domain decomposition*, in Domain Decomposition Methods in Science and Engineering, Contemp. Math. 157, AMS, Providence, RI, 1994, pp. 335–344.

[41] U. Langer, G. Of, O. Steinbach, and W. Zulehner, *Inexact fast multipole BETI methods*, in Proceedings of the 16th International Conference on Domain Decomposition, Lect. Notes Comput. Sci. Eng. 55, D. Keyes and O. Widlund, eds., Springer, Heidelberg, 2007, pp. 405–412.

[42] U. Langer, D. Pusch, and S. Reitzinger, *Efficient preconditioners for boundary element matrices based on grey-box algebraic multigrid methods*, Internat. J. Numer. Methods Engrg., 58 (2003), pp. 1937–1953.

[43] U. Langer and O. Steinbach, *Boundary element tearing and interconnecting methods*, Computing, 71 (2003), pp. 205–228.

[44] U. Langer and O. Steinbach, *Coupled boundary and finite element tearing and interconnecting methods*, in Proceedings of the 15th International Conference on Domain Decomposition, Lect. Notes Comput. Sci. Eng. 40, R. Kornhuber, R. Hoppe, J. Periaux, O. Pironneau, O. Widlund, and J. Xu, eds., Springer, Heidelberg, 2004, pp. 83–97.

[45] J. Li and O. B. Widlund, *On the Use of Inexact Subdomain Solvers for BDDC Algorithms*, Technical report TR2005-871, Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, New York, 2005.

[46] J. Mandel and R. Tezaur, *Convergence of a substructuring method with Lagrange multipliers*, Numer. Math., 73 (1996), pp. 473–487.

[47] J. Mandel and R. Tezaur, *On the convergence of a dual-primal substructuring method*, Numer. Math., 88 (2001), pp. 543–558.

[48] W. McLean, *Strongly Elliptic Systems and Boundary Integral Equations*, Cambridge University Press, Cambridge, UK, 2000.

[49] W. McLean and O. Steinbach, *Boundary element preconditioners for a hypersingular integral equation on an interval*, Adv. Comput. Math., 11 (1999), pp. 271–286.

[50] J. Nédélec, *Integral equations with nonintegrable kernels*, Integral Equations Operator Theory, 5 (1982), pp. 562–572.

[51] G. Of, *BETI-Gebietszerlegungsmethoden mit schnellen Randelementverfahren und Anwendun-*

gen, Doctoral Thesis, Universität Stuttgart, Stuttgart, Germany, 2006.

[52] G. Of and O. Steinbach, *A fast multipole boundary element method for a modified hypersingular boundary integral equation*, in Proceedings of the International Conference on Multifield Problems, Lect. Notes Appl. Comput. Mech. 12, M. Efendiev and W. L. Wendland, eds., Springer, Berlin, 2003, pp. 163–169.

[53] G. Of, O. Steinbach, and W. L. Wendland, *The fast multipole method for the symmetric boundary integral formulation*, IMA J. Numer. Anal., 26 (2006), pp. 272–296.

[54] P. Oswald, *Multilevel norms in* $H^{-1/2}(\Gamma)$, Computing, 61 (1998), pp. 235–255.

[55] J. M. Perez-Jorda and W. Yang, *A concise redefinition of the solid spherical harmonics and its use in the fast multipole method*, J. Chem. Phys., 104 (1996), pp. 8003–8006.

[56] T. V. Petersdorff and E. P. Stephan, *Multigrid solvers and preconditioners for first kind integral equations*, Numer. Methods Partial Differential Equations, 8 (1992), pp. 443–450.

[57] K. H. Pierson, P. Raghaven, and G. M. Reese, *Experiences with FETI-DP in a production level finite element application*, in Proceedings of the 14th International Conference on Domain Decomposition Methods, I. Herrera, D. E. Keyes, O. Widlund, and R. Yates, eds., DDM.org, 2003, pp. 233–240.

[58] V. Rokhlin, *Rapid solution of integral equations of classical potential theory*, J. Comput. Phys., 60 (1985), pp. 187–207.

[59] O. Steinbach, *Artificial multilevel boundary element preconditioners*, Proc. Appl. Math. Mech., 3 (2003), pp. 539–542.

[60] O. Steinbach, *Stability Estimates for Hybrid Coupled Domain Decomposition Methods*, Lecture Notes in Math. 1809, Springer, Heidelberg, 2003.

[61] O. Steinbach, *Numerische Näherungsverfahren für elliptische Randwertprobleme. Finite Elemente und Randelemente*, Teubner, Stuttgart, Leipzig, Wiesbaden, 2003.

[62] O. Steinbach and W. L. Wendland, *The construction of some efficient preconditioners in the boundary element method*, Adv. Comput. Math., 9 (1998), pp. 191–216.

[63] E. P. Stephan, *Multilevel methods for the h-, p-, and hp-versions of the boundary element method*, J. Comput. Appl. Math., 125 (2000), pp. 503–519.

[64] A. Toselli and O. Widlund, *Domain Decomposition Methods—Algorithms and Theory*, Springer Ser. Comput. Math. 34, Springer, New York, 2004.

[65] C. A. White and M. Head-Gordon, *Derivation and efficient implementation of the fast multipole method*, J. Chem. Phys., 101 (1994), pp. 6593–6605.

[66] K. Yosida, N. Nishimura, and S. Kobayashi, *A fast multipole boundary integral equation method for crack problems in* $3D$, Eng. Anal. Bound. Elem., 23 (1999), pp. 97–105.

[67] W. Zulehner, *Analysis of iterative methods for saddle point problems: A unified approach*, Math. Comp., 71 (2002), pp. 479–505.

[68] W. Zulehner, *Uzawa-Type Methods for Block-Structured Indefinite Linear Systems*, SFB-Report 2005-05, SFB F013, Johannes Kepler University Linz, Linz, Austria, 2005.

SIAM J. SCI. COMPUT.
Vol. 29, No. 1, pp. 315–330
© 2007 Society for Industrial and Applied Mathematics

A PROJECTION-BASED APPROACH TO GENERAL-FORM TIKHONOV REGULARIZATION*

MISHA E. KILMER†, PER CHRISTIAN HANSEN‡, AND MALENA I. ESPAÑOL†

Abstract. We present a projection-based iterative algorithm for computing general-form Tikhonov regularized solutions to the problem $\min_x\{\|Ax-b\|_2^2+\lambda^2\|Lx\|_2^2\}$, where the regularization matrix L is not the identity. Our algorithm is designed for the common case where λ is not known a priori. It is based on a joint bidiagonalization algorithm and is appropriate for large-scale problems when it is computationally infeasible to transform the regularized problem to standard form. By considering the projected problem, we show how estimates of the corresponding optimal regularization parameter can be efficiently obtained. Numerical results illustrate the promise of our projection-based approach.

Key words. general-form Tikhonov regularization, iterative bidiagonalization methods

AMS subject classification. 65F22

DOI. 10.1137/050645592

1. Introduction.

This paper presents a new iterative algorithm for computing an approximate solution to a Tikhonov regularized, linear discrete ill-posed problem with a general smoothing term. Linear ill-posed problems in the form of first-kind Fredholm integral equations occur frequently in the physical sciences, where they model situations where hidden information is computed from external observations [6]. The term discrete ill-posed problem refers to a discretization of such an integral equation, taking the form

$$Ax \approx b = b_{\text{true}} + e, \qquad b_{\text{true}} = Ax_{\text{true}}, \tag{1.1}$$

where A is a discretization of the integral operator, x_{true} is the true solution, and e denotes noise which we here assume to be white noise (i.e., the covariance matrix for e is a scaled identity matrix). In discrete ill-posed problems the singular values of A always decay gradually towards zero, and often there are many singular values at the machine precision level. This renders the least squares solution to the system (1.1) worthless, in the sense that the least squares solution does not approximate the solution x_{true} to the noise-free problem.

A popular technique for computing approximations to the true solution in the presence of noise is to compute the Tikhonov regularized solution:

$$x_\lambda = \operatorname{argmin}_x \left\{\|Ax-b\|_2^2+\lambda^2\|Lx\|_2^2\right\} = \operatorname{argmin}_x \left\| \begin{bmatrix} A \\ \lambda L \end{bmatrix} x - \begin{bmatrix} b \\ 0 \end{bmatrix} \right\|_2. \tag{1.2}$$

Here A is $m \times n$ with $m \geq n$, L is a $p \times n$ regularization matrix which defines the smoothing term $\|Lx\|_2$, and λ is a positive scalar called the [illegible]

*Received by the editors November 18, 2005; accepted for publication (in revised form) September 1, 2006; published electronically February 2, 2007. This work was supported in part by NSF grants 0208548 and 0139968 and CenSSIS (the Center for Subsurface Sensing and Imaging Systems) under the ERC Program of the NSF (EEC-9986821). Additional partial funding was provided by the Danish Research Agency, research grant 21-03-0574.

http://www.siam.org/journals/sisc/29-1/64559.html

†Department of Mathematics, Tufts University, Medford, MA 02155 (misha.kilmer@tufts.edu, malena.espanol@tufts.edu).

‡Informatics and Mathematical Modelling, Technical University of Denmark, DK-2800 Lyngby, Denmark (pch@imm.dtu.dk).

There are no restrictions on the dimensions of L, and we assume that the null spaces of A and L intersect trivially such that the matrix $[A^T \ L^T]$ has full rank and the solution x_λ in (1.2) is unique. If λ and L are chosen appropriately, the solution to the Tikhonov problem should approximate the desired solution x_{true}.

We emphasize that there are many examples of inverse problems where $L = I$ is not suited, such as two-dimensional (2D) spline interpolation [2], geophysics [4], and heat distribution [13]. The matrix L is problem dependent, and typical choices for L include (scaled) discrete first or second derivative operators in the appropriate dimension. The most common situation is that these L matrices are rectangular.

For large-scale problems, x_λ is usually computed by applying an iterative least squares method, such as CGLS or LSQR, to (1.2)—possibly supplemented with an appropriate preconditioner. However, this approach is computationally very demanding if the regularization parameter is not known a priori, and many different values of λ must be tried. Moreover, as we shall see shortly, the Krylov subspace underlying this approach is not necessarily the right choice.

When $L = I$ (the standard-form case) one can instead use a [illegible], where the problem is first projected onto a lower-dimensional subspace—typically a Krylov subspace—and then Tikhonov regularization combined with a parameter-choice method is applied to the projected problem. Finally the solution is transformed back to the original domain, and the result is an approximation to the Tikhonov solution. The projection is typically implemented via Lanczos bidiagonalization applied to A, with starting vector b, and the projected problem involves a lower bidiagonal matrix; cf. [14] for details.

The common approach to treating the general case with $L \neq I$ is to use a standard-form transformation. If L is invertible, then we can use the substitution $y = Lx$. Otherwise—and this is the common situation—we must use the A-weighted pseudoinverse of L, formally given by $L_A^\dagger = (I - (A(I - L^\dagger L))^\dagger A)L^\dagger$; cf. [3]. The standard-form problem is then

$$y_\lambda = \text{argmin}_y \left\{ \|AL_A^\dagger y - b\|_2^2 + \lambda^2 \|y\|_2^2 \right\}, \qquad x_\lambda = L_A^\dagger y_\lambda + x_0, \tag{1.3}$$

in which x_0 is the component of the solution in the null space of L. The key idea is to apply the bidiagonalization algorithm to the matrix $AL_A^\dagger$ and again use the regularization and parameter-choice algorithms on the projected problem. Multiplication with $L_A^\dagger$ involves solving a least squares problem followed by an oblique projection; see, e.g., [11] or [13] for some implementation details. This approach is useful from a practical standpoint whenever it is possible to implement these operations efficiently.

However, there are many problems for which it is complicated to work with $L_A^\dagger$. This is often the case for higher-dimensional problems, e.g., when the matrix L is a sum of Kronecker products. Therefore we are interested in an alternative projection-based approach that avoids the use of $L_A^\dagger$, which involves only multiplications with L and L^T and is based on a suited Krylov subspace.

This paper describes a new projection method which simultaneously bidiagonalizes both A and L in such a way that we can easily apply Tikhonov regularization and parameter-choice methods to the projected problem involving the two bidiagonal matrices. The algorithm is inspired by a bidiagonalization algorithm by Zha [17] for computing principal generalized singular values and vectors of a matrix pair, but our algorithm and the underlying theory are different.

Our paper is organized as follows. In section 2 we give the basic notation and state several decompositions that we will make use of throughout the paper. In section 3

we introduce the iterative algorithm for approximately solving (1.2) and consider how to select a suitable λ as the iterative method progresses. In section 4 some insight is presented which helps to interpret the outcome of the iterative method. Numerical results are presented in section 5, and section 6 contains our conclusions.

2. Background. This section summarizes the facts and decompositions that are the basis for our algorithm. Throughout the rest of the paper, we assume that L and A are real and have been scaled so that the norms of A and L are of the same order of magnitude. We also assume that the exact right-hand side b_{true} satisfies the discrete Picard condition, as specified below.

2.1. GSVD and regularization. To understand why (1.2) produces a reasonable approximate solution to the noisy problem (1.1), it is convenient to look at the generalized singular value decomposition (GSVD) of the matrix pair $\{A, L\}$. First, consider the special QR factorization of the stacked matrix:

$$\begin{bmatrix} A \\ L \end{bmatrix} = QR = \begin{bmatrix} Q_A \\ Q_L \end{bmatrix} R, \qquad Q_A^T Q_A + Q_L^T Q_L = I, \tag{2.1}$$

where $R \in \mathbb{R}^{n\times n}$ is upper triangular and $Q \in \mathbb{R}^{(m+p)\times n}$ has orthonormal columns. We partition Q such that $Q_A \in \mathbb{R}^{m\times n}$ and $Q_L \in \mathbb{R}^{p\times n}$, and therefore $A = Q_A R$ and $L = Q_L R$. Now let

$$Q_A = P_A C_A W^T, \qquad Q_L = P_L S_L W^T \tag{2.2}$$

be the CS decomposition of $\{Q_A, Q_L\}$ (see, e.g., [1, section 4.2] or [5, section 2.6]), where $P_A \in \mathbb{R}^{m\times m}$, $P_L \in \mathbb{R}^{p\times p}$, and $W \in \mathbb{R}^{n\times n}$ are orthogonal matrices, and $C_A \in \mathbb{R}^{m\times n}$ and $S_L \in \mathbb{R}^{p\times n}$ are diagonal matrices (not necessarily square) satisfying $C_A^T C_A + S_L^T S_L = I$. Then the GSVD of $\{A, L\}$ is defined as

$$A = P_A C_A G^{-1}, \qquad L = P_L S_L G^{-1} \qquad \text{with} \qquad G = R^{-1} W, \tag{2.3}$$

where the invertibility of G follows from the assumption that $[A^T \ L^T]$ has full rank. For convenience, we order the entries of the diagonal matrices C_A and S_L so that

$$1 \geq c_1 \geq \cdots \geq c_{\min(n,p)} \geq 0, \qquad c_{\min(n,p)+1} = \cdots = c_n = 1,$$

$$0 \leq s_1 \leq \cdots \leq s_{\min(n,p)} \leq 1,$$

which differs from the order usually found in the literature. We do this so that the quantities $c_1, \ldots, c_{\min(n,p)}$ appear in nonincreasing order, similar to the standard SVD. Using this decomposition, the Tikhonov solution takes the form

$$\begin{aligned} x_\lambda &= (A^T A + \lambda^2 L^T L)^{-1} A^T b = G\,(C_A^T C_A + \lambda^2 S_L^T S_L)^{-1} C_A^T P_A^T b \\ &= \sum_{i=1}^{\min(n,p)} \frac{c_i^2}{c_i^2 + \lambda^2 s_i^2} \frac{p_{A,i}^T b}{c_i}\, g_i + \sum_{i=\min(n,p)+1}^{n} p_{A,i}^T b\, g_i, \end{aligned} \tag{2.4}$$

where $p_{A,i}$ and g_i are the columns of P_A and G, respectively. The last term, which vanishes when $p \geq n$, consists of components in the null space of L which are not influenced by the regularization. The discrete Picard condition [10] says that the values $|p_{A,i}^T b_{\text{true}}|$ must, on average, decay faster than the c_i. Hence the values $|p_{A,i}^T b|$ also, on average, decay until we reach an index i such that noise components become

TABLE 2.1
The ingredients of the two algorithms BIDIAG-1 *and* BIDIAG-2.

Matrix	Size	Details
B_k	$(k+1) \times k$	$(B_k)_{i,i} = \alpha_i \quad (B_k)_{i+1,i} = \beta_{i+1}$
$\widehat{B}_k$	$k \times k$	$(\widehat{B}_k)_{i,i} = \hat{\alpha}_i \quad (\widehat{B}_k)_{i,i+1} = \hat{\beta}_i$
U_{k+1}	$m \times (k+1)$	$(u_1, \ldots, u_{k+1})$
$\widehat{U}_k$	$p \times k$	$(\hat{u}_1, \ldots, \hat{u}_k)$
V_k	$n \times k$	$(v_1, \ldots, v_k)$
$\widehat{V}_k$	$n \times k$	$(\hat{v}_1, \ldots, \hat{v}_k)$

dominant; beyond that point $|p_{A,i}^T b|$ becomes roughly constant independent of i, due to our assumption about the noise e being white. The recoverable "signal" is contained primarily in the subspace spanned by the g_i's for small i, and a good value of λ should have the effect of filtering (or damping) the terms in the sum for larger values of i. The solution produced by our algorithm must resemble such a filtered GSVD solution.

2.2. Joint bidiagonalization. Consider the reduced QR factorization in (2.1). Zha [17, section 4.1] derived an algorithm for joint reduction of Q_A and Q_L to upper bidiagonal form, given a common right starting vector. Similar to Zha's algorithm, we do not need the QR factorization explicitly, but to make our presentation clearer we start by formulating our algorithm explicitly in terms of Q_A and Q_L.

Our algorithm is intended for solving systems of equations, and it is crucial to use the right-hand side b as a left starting vector for a , , . bidiagonal reduction of Q_A; therefore we must use the BIDIAG-1 algorithm from [15] (which underlies the LSQR algorithm). We combine this with the algorithm BIDIAG-2 from [15] for the joint , , , bidiagonal reduction of Q_L.

BIDIAG-1. Starting vector b; reduction of Q_A to lower bidiagonal form:
$\beta_1 u_1 = b$
$\alpha_1 v_1 = Q_A^T u_1$
for $i = 1, 2, \ldots, n-1$
  $\beta_{i+1} u_{i+1} = Q_A v_i - \alpha_i u_i$
  $\alpha_{i+1} v_{i+1} = Q_A^T u_{i+1} - \beta_{i+1} v_i$.

BIDIAG-2. Starting vector v_1; reduction of Q_L to upper bidiagonal form:
$\hat{v}_1 = v_1$
$\hat{\alpha}_1 \hat{u}_1 = Q_L \hat{v}_1$
for $i = 1, 2, \ldots, \min(p, n) - 1$
  $\hat{\beta}_i \hat{v}_{i+1} = Q_L^T \hat{u}_i - \hat{\alpha}_i \hat{v}_i$
  $\hat{\alpha}_{i+1} \hat{u}_{i+1} = Q_L \hat{v}_{i+1} - \hat{\beta}_i \hat{u}_i$.

In both algorithms the scalars α_i, β_i, $\hat{\alpha}_i$, and $\hat{\beta}_i$ are chosen so that u_i, v_i, $\hat{u}_i$, and $\hat{v}_i$ have unit 2-norm. From the kth step of the two algorithms we obtain

$$Q_A V_k = U_{k+1} B_k, \qquad Q_A^T U_{k+1} = V_k B_k^T + \alpha_{k+1} v_{k+1} e_{k+1}^T, \tag{2.5}$$

$$Q_L \widehat{V}_k = \widehat{U}_k \widehat{B}_k, \qquad Q_L^T \widehat{U}_k = \widehat{V}_k \widehat{B}_k^T + \hat{\beta}_k \hat{v}_{k+1} e_k^T, \tag{2.6}$$

where e_{k+1} and e_k are the last columns of the identity matrices of orders $k+1$ and k, respectively. The remaining ingredients of the two equations are listed in Table 2.1.

The following theorem, the proof of which follows along the lines of a related proof in [17], gives the relationship between the joint bidiagonalization of Q_A and Q_L computed by the two algorithms above.

THEOREM 2.1. [illegible] Q_A [illegible] Q_L [illegible] BIDIAG-1 [illegible] BIDIAG-2 [illegible]

$$\hat{v}_{j+1} = (-1)^j v_{j+1}, \qquad \hat{\alpha}_j\hat{\beta}_j = \alpha_{j+1}\beta_{j+1}, \qquad j = 1, 2, \ldots, \min(p, n) - 1.$$

[illegible]. We prove Theorem 2.1 by induction. Consider the base case $j = 1$ and note that $\hat{v}_1 = v_1$ by construction. We assume that $\hat{\alpha}_1$, $\hat{\beta}_1$, α_2, and β_2 are strictly positive, and we must show that $\hat{v}_2 = -v_2$ and $\hat{\alpha}_1\hat{\beta}_1 = \alpha_2\beta_2$. Since $\hat{\beta}_1\hat{v}_2 = Q_L^T\hat{u}_1 - \hat{\alpha}_1\hat{v}_1$, we get

$$\hat{\alpha}_1\hat{\beta}_1\hat{v}_2 = Q_L^T(Q_L\hat{v}_1) - \hat{\alpha}_1^2\hat{v}_1 = v_1 - Q_A^TQ_Av_1 - \hat{\alpha}_1^2v_1,$$

where we have used that $Q_L^TQ_L = I - Q_A^TQ_A$ and $v_1 = \hat{v}_1$. Similarly, we find

$$\beta_2\alpha_2v_2 = Q_A^T(Q_Av_1 - \alpha_1u_1) - \beta_2^2v_1 = Q_A^TQ_Av_1 - \alpha_1^2v_1 - \beta_2^2v_1.$$

Adding these two equations, we obtain

$$\hat{\alpha}_1\hat{\beta}_1\hat{v}_2 + \alpha_2\beta_2v_2 = (1 - \hat{\alpha}_1^2 - \alpha_1^2 - \beta_2^2)v_1.$$

Both $\hat{v}_2$ and v_2 are orthogonal to v_1 by construction, and each has unit norm. Therefore, multiplying on the left by v_2^T (likewise $\hat{v}_2^T$) and rearranging give

$$v_2^T\hat{v}_2 = -\frac{\alpha_2\beta_2}{\hat{\alpha}_1\hat{\beta}_1}, \qquad \hat{v}_2^Tv_2 = -\frac{\hat{\alpha}_1\hat{\beta}_1}{\alpha_2\beta_2}.$$

Since $\hat{\alpha}_1$, $\hat{\beta}_1$, α_2, and β_2 are positive, this implies the two relations $\hat{\alpha}_1\hat{\beta}_1 = \alpha_2\beta_2$ and $\hat{v}_2 = -v_2$, and the base case is proved.

Next, suppose the hypothesis is true for indices up to j. We need to show that $\hat{v}_{j+1} = (-1)^jv_{j+1}$ and $\hat{\alpha}_j\hat{\beta}_j = \alpha_{j+1}\beta_{j+1}$. Let $P_j = \mathrm{diag}(1, -1, 1, -1, \ldots) \in \mathbb{R}^{j\times j}$ (note that P_j is orthogonal). Then the induction hypothesis gives $\hat{V}_j = V_jP_j$. Using this fact, the relation $Q_A^TQ_A + Q_L^TQ_L = I$, and (2.5) and (2.6), it is straightforward to show that

$$I_j = B_j^TB_j + P_j\widehat{B}_j^T\widehat{B}_jP_j.$$

Multiplying the leftmost equation in (2.5) by Q_A^T and using the rightmost equation, we obtain

$$Q_A^TQ_AV_j = V_jB_j^TB_j + \alpha_{j+1}\beta_{j+1}v_{j+1}e_j^T.$$

Likewise, we get $Q_L^TQ_L\widehat{V}_j = \widehat{V}_j\widehat{B}_j^T\widehat{B}_j + \hat{\alpha}_j\hat{\beta}_j\hat{v}_{j+1}e_j^T$, which yields

$$Q_L^TQ_LV_j = V_jP_j\widehat{B}_j^T\widehat{B}_jP_j + \hat{\alpha}_j\hat{\beta}_j(-1)^{j+1}\hat{v}_{j+1}e_j^T.$$

Adding the above two equations, we get

$$\overbrace{(Q_A^TQ_A + Q_L^TQ_L)}^{I}V_j = V_j\overbrace{(B_j^TB_j + P_j\widehat{B}_j^T\widehat{B}_jP_j)}^{I_j} + \left(\alpha_{j+1}\beta_{j+1}v_{j+1} + \hat{\alpha}_j\hat{\beta}_j\hat{v}_{j+1}(-1)^{j+1}\right)e_j^T,$$

which means the last term in the sum on the right must vanish. Since the $\hat{\alpha}_j$, $\hat{\beta}_j$, α_{j+1}, and β_{j+1} are nonnegative and since v_{j+1} and $\hat{v}_{j+1}$ have unit norm, this implies $\hat{v}_{j+1} = (-1)^j v_{j+1}$ and $\hat{\alpha}_j \hat{\beta}_j = \alpha_{j+1}\beta_{j+1}$ as desired. ∎

From the above theorem and the relation of Q_A and Q_L to A and L, it follows that A and L can be jointly bidiagonalized.

THEOREM 2.2. $A \in \mathbb{R}^{m\times n}$ $L \in \mathbb{R}^{p\times n}$ $m \geq n$ $U \in \mathbb{R}^{m\times m}$ $\widehat{U} \in \mathbb{R}^{p\times p}$ $V \in \mathbb{R}^{n\times n}$ $B \in R^{m\times n}$ $\bar{B} \in R^{p\times n}$ $n \times n$ Z

$$A = UBZ^{-1}, \qquad L = \widehat{U}\bar{B}Z^{-1}, \qquad Z = R^{-1}V, \tag{2.7}$$

R (2.1) $\bar{B} = \widehat{B}P$ $P = \mathrm{diag}(1, -1, 1, -1, \dots)$ $p < n$ $p+1, \dots, n$ $\bar{B}$. The proof follows by simple insertion. ∎

3. An iterative algorithm. In [17], a joint bidiagonalization (different from ours) was used to estimate generalized singular triplets. In this work we utilize our new joint bidiagonalization to obtain a low-storage, inner-outer iterative technique for computing approximate Tikhonov solutions to (1.2) for multiple values of λ simultaneously.

3.1. Derivation of the projection-based algorithm. First, we replace the Tikhonov problem (1.2) by an approximate problem:

$$x_k^\lambda = \mathrm{argmin}_{x\in\mathcal{Z}_k} \left\| \begin{bmatrix} A \\ \lambda L \end{bmatrix} x - \begin{bmatrix} b \\ 0 \end{bmatrix} \right\|_2, \qquad \mathcal{Z}_k = \mathrm{span}\{z_1, \dots, z_k\}; \tag{3.1}$$

i.e., x_k^λ is a linear combination of the first k columns of the matrix Z from (2.7). The superscript is used to emphasize that this vector depends on λ, since it is an approximation to the Tikhonov solution.

Now from Theorem 2.2 it follows that $AZ_k = U_{k+1}B_k$ and $LZ_k = \widehat{U}_k\bar{B}_k$, where $Z_k = (z_1, \dots, z_k)$. Notice that

$$U_{k+1}(\beta_1 e_1) = b, \qquad \beta_1 = \|b\|_2, \tag{3.2}$$

where e_1 is the first canonical unit vector of length $k+1$. Then, with $x_k^\lambda = Z_k y_k^\lambda$,

$$\begin{aligned} \left\| \begin{bmatrix} A \\ \lambda L \end{bmatrix} x_k^\lambda - \begin{bmatrix} b \\ 0 \end{bmatrix} \right\|_2 &= \left\| \begin{bmatrix} AZ_k \\ \lambda L Z_k \end{bmatrix} y_k^\lambda - \begin{bmatrix} b \\ 0 \end{bmatrix} \right\|_2 \\ &= \left\| \begin{bmatrix} U_{k+1} & 0 \\ 0 & \hat{U}_k \end{bmatrix} \left(\begin{bmatrix} B_k \\ \lambda \bar{B}_k \end{bmatrix} y_k^\lambda - \begin{bmatrix} \beta_1 e_1 \\ 0 \end{bmatrix} \right) \right\|_2 \\ &= \left\| \begin{bmatrix} B_k \\ \lambda \bar{B}_k \end{bmatrix} y_k^\lambda - \begin{bmatrix} \beta_1 e_1 \\ 0 \end{bmatrix} \right\|_2 . \end{aligned}$$

Hence, the minimization problem (3.1) is equivalent to computing

$$x_k^\lambda = Z_k y_k^\lambda \qquad \text{with} \qquad y_k^\lambda = \mathrm{argmin}_y \left\| \begin{bmatrix} B_k \\ \lambda \bar{B}_k \end{bmatrix} y - \begin{bmatrix} \beta_1 e_1 \\ 0 \end{bmatrix} \right\|_2 . \tag{3.3}$$

We refer to (3.3) as the projected problem with regularization. The key point here is that the B_k $\bar{B}_k$ λ.

The main idea is that once k is large enough, the projected problem will capture the main features/components of the original problem. To be more precise, let $x_{\lambda^{\text{opt}}}$ denote the Tikhonov solution to the original problem, where λ^{opt} is the optimal regularization parameter. Then there is good hope that—for large enough k—there exists a parameter $\lambda^{k,*}$ such that the corresponding solution $x_k^{\lambda^{k,*}}$ to (3.1) will be a good approximation to $x_{\lambda^{\text{opt}}}$.

Projection-Based General-Form Tikhonov Regularization.

For $k = 1, 2, \ldots$

(a) compute the kth step of the joint bidiagonalization,

(b) compute y_k^λ (and x_k^λ, if desired),

(c) update residual norm and seminorm; check for convergence.

Efficient implementation details are described below. As we will see, step (a) in this algorithm will require an inner iteration to compute orthogonal projections QQ^Tv. In this way, we avoid computing the QR factorization of the stacked matrix explicitly.

We emphasize that for a fixed λ, the solution x_k^λ computed via (3.3) is different from the solution obtained by applying k steps of LSQR to (1.2). In LSQR the stacked matrix $[A^T\ \lambda L^T]^T$ is reduced to a single bidiagonal matrix whose entries depend on λ, and the solution lies in the Krylov subspace

$$\begin{aligned}\mathcal{S}_{k,\text{LSQR}} &= \text{span}\{(A^TA + \lambda^2 L^TL)^i A^Tb\}_{i=0}^{k-1} \\ &= \text{span}\{G^{-T}\left((C_A^TC_A + \lambda^2 S_L^TS_L)(G^TG)^{-1}\right)^i C_A^TP_A^Tb\}_{i=0}^{k-1}.\end{aligned}$$

All the matrices in the above expression are from the GSVD (2.3) of $\{A, L\}$. The subspace $\mathcal{S}_{k,\text{LSQR}}$ is not directly related to the GSVD basis (the columns of G) for the Tikhonov solution unless $L = I$, in which case $\mathcal{S}_{k,\text{LSQR}}$ is spanned by A's right singular vectors. Our algorithm, on the other hand, is based on a bidiagonalization of Q_A, and our solution lies in the subspace

$$\mathcal{S}_{k,\text{new}} = \text{span}\{G(C_A^TC_A)^iC_A^TP_A^Tb\}_{i=0}^{k-1}. \tag{3.4}$$

To see this, note that the solution $x_k = Z_ky_k = R^{-1}V_ky_k$, where V_k is a basis for the Krylov subspace of dimension k generated by Q_A and b (see also Theorem 4.3). However, since $Q_A = P_AC_AW^T$ is the SVD of Q_A (except for a permutation when $p < n$), it follows that this subspace is

$$\text{span}\{W(C_A^TC_A)^iC_A^TP_A^Tb\}_{i=0}^{k-1}.$$

Using the fact that $W = RG$ and $x_k = R^{-1}V_ky_k$, the result follows. Note that the subspace in (3.4) is more directly related to the GSVD solution subspace. Hence, we expect that the iterates living here are a better approximation to the desired solution.

3.2. Joint bidiagonalization. For reasons of efficiency and storage, we do not want to compute the QR factorization in (2.1). Therefore Q_A and Q_L are not available, and it is not possible to execute Bidiag-1 and Bidiag-2 directly. The solution to this dilemma is to use a change of variables in Bidiag-1, together with the results of Theorems 2.1 and 2.2.

The change-of-variables approach we advocate is the approach introduced in [17, pp. 396, 399] for a slightly different form of Lanczos bidiagonalization. We note that

$$Q_A = \begin{bmatrix} I_m\,,\ 0_{m\times p}\end{bmatrix}Q \qquad \text{and} \qquad Q_L = \begin{bmatrix} 0_{p\times m}\,,\ I_p\end{bmatrix}Q$$

and we set

$$\tilde{u}_j = \begin{bmatrix} I_m \\ 0_{p\times m} \end{bmatrix} u_j = \begin{bmatrix} u_j \\ 0_p \end{bmatrix}, \quad \tilde{v}_j = Qv_j, \quad \bar{u}_j = \begin{bmatrix} 0_{m\times p} \\ I_p \end{bmatrix} \hat{u}_j = \begin{bmatrix} 0_m \\ \hat{u}_j \end{bmatrix}, \quad \bar{v}_j = Q\hat{v}_j.$$

Using these relations and multiplying through by Q, $\begin{bmatrix} I_m \,,\, 0_{m\times p} \end{bmatrix}^T$, or $\begin{bmatrix} 0_{p\times m} \,,\, I_p \end{bmatrix}^T$ as appropriate, we obtain modified versions of Bidiag-1 and Bidiag-2, due to Theorem 2.1. The modified algorithms for bidiagonalization of Q_A and Q_L take the following form.

Modified Bidiag-1.
$\beta_1\tilde{u}_1 = \begin{bmatrix} I_m \,,\, 0_{m\times p} \end{bmatrix}^T b$
$\alpha_1\tilde{v}_1 = QQ^T\tilde{u}_1$
For $i = 1, 2, \ldots$
$\quad \beta_{i+1}\tilde{u}_{i+1} = \begin{bmatrix} I_m \,,\, 0_{m\times p} \end{bmatrix}^T \begin{bmatrix} I_m \,,\, 0_{m\times p} \end{bmatrix} \tilde{v}_i - \alpha_i\tilde{u}_i$
$\quad \alpha_{i+1}\tilde{v}_{i+1} = QQ^T\tilde{u}_{i+1} - \beta_{i+1}\tilde{v}_i.$

Modified Bidiag-2.
$\bar{v}_1 = \tilde{v}_1$
$\hat{\alpha}_1\bar{u}_1 = \begin{bmatrix} 0_{p\times m} \,,\, I_p \end{bmatrix}^T \begin{bmatrix} 0_{p\times m} \,,\, I_p \end{bmatrix} \bar{v}_1$
For $i = 1, 2, \ldots$
$\quad \hat{\beta}_i = (\alpha_{i+1}\beta_{i+1})/\hat{\alpha}_i$
$\quad \bar{v}_{i+1} = (-1)^i\tilde{v}_{i+1}$
$\quad \hat{\alpha}_{i+1}\bar{u}_{i+1} = \begin{bmatrix} 0_{p\times m} \,,\, I_p \end{bmatrix}^T \begin{bmatrix} 0_{p\times m} \,,\, I_p \end{bmatrix} \bar{v}_{i+1} - \hat{\beta}_i\bar{u}_i.$

In practice, we merge the two iterations and take advantage of the zero blocks in the matrices and vectors. This leads to the final combined algorithm.

Joint Bidiagonalization Algorithm.
$\beta_1 u_1 = b$
$\alpha_1\tilde{v}_1 = QQ^T \begin{bmatrix} u_1 \\ 0_p \end{bmatrix}$
$\hat{\alpha}_1\hat{u}_1 = \tilde{v}_1(m+1\colon m+p)$
For $i = 1, 2, \ldots$
$\quad \beta_{i+1}u_{i+1} = \tilde{v}_i(1\colon m) - \alpha_i u_i$
$\quad \alpha_{i+1}\tilde{v}_{i+1} = QQ^T \begin{bmatrix} u_{i+1} \\ 0_p \end{bmatrix} - \beta_{i+1}\tilde{v}_i$
$\quad \hat{\beta}_i = (\alpha_{i+1}\beta_{i+1})/\hat{\alpha}_i$
$\quad \hat{\alpha}_{i+1}\hat{u}_{i+1} = (-1)^i\tilde{v}_{i+1}(m+1\colon m+p) - \hat{\beta}_i\hat{u}_i.$

Note that the only computation that involves the matrices A and L is the action of QQ^T on a vector. This is an orthogonal projection that can be computed using the LSQR algorithm, as described by Saunders [16], which [illegible] the need to compute the QR factorization explicitly. With the scaling that we have assumed, if L is reasonably well conditioned, then so is $[A^T\ L^T]$. Also, efficient preconditioners for matrices of this form exist [7], [12]. Hence, we do not expect this step to be overly time consuming.

3.3. Working with the projected problem. Assume that we do not have a priori knowledge of a suited value of the regularization parameter and that we want to try many values of λ at the kth step. It is straightforward to compute the solution y_k^λ of the projected problem (3.3) via the QR factorization

$$\begin{bmatrix} B_k & \beta_1 e_1 \\ \lambda\bar{B}_k & 0 \end{bmatrix} = Q_k^\lambda \begin{bmatrix} R_k^\lambda & f_k^\lambda \\ 0 & h_k^\lambda \end{bmatrix}. \tag{3.5}$$

Here, R_k^λ is an upper bidiagonal matrix of size $k \times k$, while f_k^λ and h_k^λ are vectors of length k and $k+1$, respectively. These quantities can be computed in $O(k)$ flops by means of "strategic" Givens rotations, and Q_k^λ is not needed. Then y_k^λ is computed by solving the upper bidiagonal system

$$R_k^\lambda y_k^\lambda = f_k^\lambda,$$

which also requires only $O(k)$ flops.

Many parameter selection techniques for choosing an appropriate value of λ, such as the L-curve criterion or the discrepancy principle, need estimates of the residual norm and seminorm as functions of λ. Fortunately, these norms can be obtained directly from the projected problem. It is straightforward to show that both the residual norm $\|Ax_k^\lambda - b\|_2$ and the seminorm $\|Lx_k^\lambda\|_2$ can be expressed in terms of the projected problem, as shown below, and therefore also computed in $O(k)$ flops.

THEOREM 3.1.

$$\|Ax_k^\lambda - b\|_2 = \|B_k y_k^\lambda - \beta_1 e_1\|_2 = \left(\|h_k^\lambda\|_2^2 - \lambda^2 \|\bar{B}_k y_k^\lambda\|_2^2\right)^{1/2}$$

$$\|Lx_k^\lambda\|_2 = \|\bar{B}_k y_k^\lambda\|_2.$$

. The first equality for the residual norm follows from (2.5) and (3.2), $x_k^\lambda = Z_k y_k^\lambda$, and $U^T U = I$, while the second equality follows from the relation $\|B_k y_k^\lambda - \beta_1 e_1\|_2^2 + \lambda^2 \|\bar{B}_k y_k^\lambda\|_2^2 = \|h_k^\lambda\|_2^2$. The equality for the seminorm follows from (2.6), $x_k^\lambda = Z_k y_k^\lambda$, and $\widehat{U}^T \widehat{U} = I$. □

Once the final number of iterations has been reached (determining this final number of iterations is a subject for future research), then we need to compute the solution x_k^λ. This step is only needed once. Let $\tilde{V}_k = [\tilde{v}_1, \ldots, \tilde{v}_k]$, where the vectors $\tilde{v}_k \in \mathbb{R}^{m+p}$ are generated by the joint bidiagonalization algorithm. Then

$$\widetilde{V}_k = QV_k = QR(R^{-1}V_k) = \begin{bmatrix} A \\ L \end{bmatrix} Z_k.$$

Multiplying on the right by y_k^λ and using that $y_k^\lambda = (R_k^\lambda)^{-1} f_k^\lambda$, we obtain

$$\begin{bmatrix} A \\ L \end{bmatrix} x_k^\lambda = \tilde{V}_k y_k^\lambda. \tag{3.6}$$

The right-hand side of this system, $\tilde{V}_k y_k^\lambda$, is computed in $O((m+p)k)$ flops, and we compute x_k^λ by solving the reasonably well-conditioned system by means of CGLS or LSQR (possibly using the same preconditioner as in the QQ^T computations).

3.4. Updating issues. Alternatively, we can a priori fix a set of λ-values and derive updating formulas for all the quantities (i.e., solutions, residual norms, and constraint norms) associated with these λ-values, similar to how they are computed in [15] for regularized least squares problems with $L = I$.

Note that if we introduce the matrix $D_k^\lambda = \widetilde{V}_k (R_k^\lambda)^{-1}$, then we can write

$$\begin{bmatrix} A \\ L \end{bmatrix} x_k^\lambda = \widetilde{V}_k y_k^\lambda = s_k^\lambda \qquad \text{with} \qquad s_k^\lambda = \widetilde{V}_k (R_k^\lambda)^{-1} f_k^\lambda = D_k^\lambda f_k^\lambda.$$

Hence, if we can derive a recurrence relation for s_k^λ, then we have an easy way to update the vectors Ax_k^λ and Lx_k^λ as k increases.

The key to deriving the recurrence relations is to observe that the matrix in (3.5) has the following structure:

$$\begin{bmatrix} R_k^\lambda & f_k^\lambda \\ 0 & h_k^\lambda \end{bmatrix} = \left[\begin{array}{cc|c} R_{k-1}^\lambda & r_{k-1,k}^\lambda e_{k-1} & f_{k-1}^\lambda \\ 0 & r_{k,k}^\lambda & \phi_k^\lambda \\ \hline 0 & 0 & h_k^\lambda \end{array}\right];$$

i.e., the leading $(k-1)\times(k-1)$ block of R_k^λ is R_{k-1}^λ, and f_k^λ agrees in its first $k-1$ components with f_{k-1}^λ. Consider first the relation $\widetilde{V}_k = D_k^\lambda R_k^\lambda$:

$$[\,\widetilde{V}_{k-1}\,,\,\tilde{v}_k\,] = [\,D_{k-1}^\lambda\,,\,d_k^\lambda\,]\begin{bmatrix} R_{k-1}^\lambda & r_{k-1,k}^\lambda e_{k-1} \\ 0 & r_{k,k}^\lambda \end{bmatrix}$$

from which we see that $\widetilde{V}_{k-1} = D_{k-1}^\lambda R_{k-1}^\lambda$ and

$$d_k^\lambda = \frac{\tilde{v}_k - d_{k-1}^\lambda r_{k-1,k}^\lambda}{r_{k,k}^\lambda},$$

which is a recurrence in the d_k^λ-vectors that needs only the most recent vector $\tilde{v}_k$. Then, from the relation between the elements of f_k^λ we get

$$s_k^\lambda = D_{k-1}^\lambda f_{k-1}^\lambda + d_k^\lambda \phi_k^\lambda = s_{k-1}^\lambda + \phi_k^\lambda d_k^\lambda.$$

Given these recurrences for d_k^λ and s_k^λ, we also automatically have recurrence relations for the residual vector $b - Ax_k^\lambda$ and the vector Lx_k^λ, since

$$b - Ax_k^\lambda = b - s_k^\lambda(1\colon m), \qquad Lx_k^\lambda = s_k^\lambda(m+1\colon m+p).$$

These quantities can also be important for some parameter-choice methods, such as the one recently proposed in [8] which is based on a statistical analysis of the residual vector for the discrete ill-posed problem.

We note that if the recurrence relations are not used, then we must store the matrix $\widetilde{V}_k$ because it is needed to compute x_k^λ in (3.6). So another advantage of the recurrence relations is that we explicitly have the right-hand side of (3.6) in the form of the vector s_k^λ.

The conclusion is that if a set of λ-values is defined a priori, then we can carry along a set of recursions for these values and select the best of them via a method such as the L-curve criterion—again without having to compute every solution x_k^λ. For example, for each k an L-curve can be determined and the value of the parameter on the "corner" can then be computed. As k increases, the overall behavior of A and L becomes adequately captured in the corresponding projections B_k and $\bar{B}_k$, and the λ-value at the corner should stagnate, signifying that the iterations can be terminated. Then $x_k^{\lambda^{k,*}}$ should be taken as the approximate solution to the original Tikhonov problem, where $\lambda^{k,*}$ corresponds to the corner at the final iteration k. Note that $\lambda^{k,*}$ may not correspond to the optimal parameter for the original Tikhonov problem.

4. Insight. In this section we demonstrate why we can expect our new algorithm to generate a reasonable regularized solution for $k \ll n$. We note that this section is intended only to gain some insight into the properties of the algorithm and not as a rigorous analysis. To do this, we must also consider why the projected problem (i.e., $B_k y \approx \beta_1 e_1$) needs Tikhonov regularization as in (3.3). We note that this section may be skipped on a first reading.

Our argument proceeds as follows. First, we show that Q_A inherits properties of the matrix A in the sense that the singular values of Q_A follow the same pattern as those of A (decaying toward zero, with no gap in the spectrum) and that the singular vectors of Q_A have the property that they increase in frequency as the size of the singular values decrease. We use these facts to deduce that the singular values of B_k approximate the singular values of Q_A in their natural order (i.e., largest to smallest), with the so-called Ritz vectors as approximations of the corresponding singular vectors of Q_A. This means that for k large enough, systems involving B_k must be regularized, hence the need for the regularized projected problem (3.3). Furthermore, we show that since the CS decomposition of $\{Q_A, Q_L\}$ effectively gives the SVDs of Q_A and Q_L (appropriately ordered), our algorithm must be generating approximations of the GSVD of $\{A, L\}$ in an ordering consistent with what is needed to produce a regularized solution (refer to section 2.1); thus, $x_k^{\lambda^{k,*}}$ is a reasonable approximation to $x_{\lambda^{\text{opt}}}$ (although $\lambda^{k,*}$ may not be equal to λ^{opt}).

From the QR factorization in (2.1) we have that $\text{cond}(R) = \text{cond}([A^T \; L^T])$, and it follows from the analysis in [9] that if L is well conditioned, then so is $[A^T \; L^T]$ and thus R. Moreover, since

$$\text{cond}(A) = \text{cond}(Q_A R) \le \text{cond}(Q_A)\,\text{cond}(R) \quad \Rightarrow \quad \text{cond}(Q_A) \ge \frac{\text{cond}(A)}{\text{cond}(R)},$$

we can conclude that the ill conditioning of A must carry over to the matrix Q_A. In fact, we can say something stronger about the actual decay rate of the singular values in Q_A relative to the singular values in A.

LEMMA 4.1. σ_i A c_i C_A Q_A $i = 1, \ldots, n$

$$\sigma_i \le c_i \, \|R\|_2 \le \sigma_i \, \text{cond}(R).$$

. The proof follows immediately from Theorem 2.4 in [9] and $\|R\|_2 = \|[A^T \; L^T]\|_2$. □

Let $p_{A,i}$ denote the ith column of the matrix P_A, which is the left singular matrix of Q_A and also the left generalized singular matrix of A. The behavior of these vectors is closely related to that of the ordinary left singular vectors of A.

THEOREM 4.2. σ_i A $A = \check{U}\Sigma\check{V}^T$ $p_{A,i}$ $\check{U}$

$$p_{A,i} = \check{U}\xi_i, \qquad \text{where} \qquad \xi_i = \check{U}^T p_{A,i}.$$

i $(\xi_i)_j$ ξ_i

$$|(\xi_i)_j| \le \frac{\sigma_i}{\sigma_j}\,\text{cond}(R).$$

. Consider the vector $A^T p_{A,i}$. Using the SVD A, we can write

$$A^T p_{A,i} = \check{V}\Sigma\check{U}^T p_{A,i} = \check{V}\Sigma\xi_i = \sum_{j=1}^{n} \sigma_j \, (\xi_i)_j \, \check{v}_j,$$

and since the absolute value of each element in a vector is bounded by its norm, we have $|\sigma_j\,(\xi_i)_j| \le \|A^T p_{A,i}\|_2$ and thus $|(\xi_i)_j| \le \|A^T p_{A,i}\|_2 \,/\, \sigma_j$. Using the GSVD we can also write

$$A^T p_{A,i} = G^{-T} C_A^T P_A^T p_{A,i} = c_i\,\theta_i,$$

where θ_i is the ith column of G^{-T}, and thus

$$\|A^T p_{A,i}\|_2 = c_i\,\|\theta_i\|_2 \le c_i\,\|G^{-T}\|_2 = c_i\,\|R\|_2 \le \sigma_i\,\mathrm{cond}(R),$$

where we used Lemma 4.1. The result is obtained by combining the two results. ∎

The conclusion is that for a given i, the coefficients $|(\xi_i)_j|$ for $j < i$ cannot be large (because R is well conditioned). Indeed they must be small if σ_i/σ_j is small enough. Hence the vector $p_{A,i}$ is dominated by those left singular vectors $\check{u}_j$ for which $j > i$. As i increases, $p_{A,i}$ will be more and more dominated by singular vectors $\check{u}_j$ corresponding to small singular values σ_j. As the vectors $\check{u}_j$ become more and more oscillatory as j increases, this feature must carry over to the vectors $p_{A,i}$.

If we set $\lambda = 0$ and $w = Rx$, then the relation

$$\min_x \|Ax - b\|_2 = \min_w \|Q_A w - b\|_2$$

shows that we can use our BIDIAG-1 algorithm to compute approximate least squares solutions. Since Q_A inherits the spectral features of A, BIDIAG-1 produces a sequence of matrices B_k whose larger singular values approximate the larger singular values of A, but progressively smaller singular values of A are also picked up. The appearance of these components forces the need for regularization of the projected problem $B_k y_k^\lambda \approx \beta_1 e_1$.

Likewise, we transform the regularization problem (1.2) to the regularization problem

$$\min_w \left\| \begin{bmatrix} Q_A \\ \lambda Q_L \end{bmatrix} w - \begin{bmatrix} b \\ 0 \end{bmatrix} \right\|_2 . \tag{4.1}$$

This justifies the choice $x_k^\lambda = R^{-1} w_k^\lambda$, where w_k^λ is an approximate solution to (4.1). The next theorem shows that in defining $x_k^\lambda = Z_k y_k^\lambda$, where y_k^λ is the solution to the regularized projected problem, we achieve this goal.

THEOREM 4.3. [illegible] x_k^λ [illegible]

$$x_k^\lambda = R^{-1} w_k^\lambda \qquad \text{with} \qquad w_k^\lambda = \min_{w \in \mathcal{K}_k} \left\| \begin{bmatrix} Q_A \\ \lambda Q_L \end{bmatrix} w - \begin{bmatrix} b \\ 0 \end{bmatrix} \right\|_2 ,$$

[illegible] $\mathcal{K}_k$ [illegible]

$$\mathcal{K}_k = \mathrm{span}\{Q_A^T b, Q_A^T Q_A\, Q_A^T b, (Q_A^T Q_A)^2 Q_A^T b, \ldots, (Q_A^T Q_A)^{k-1} Q_A^T b\}.$$

[illegible] Write $w = V_k y$; then

$$\min_{w \in \mathcal{K}_k} \left\| \begin{bmatrix} Q_A \\ \lambda Q_L \end{bmatrix} w - \begin{bmatrix} b \\ 0 \end{bmatrix} \right\| = \min_y \left\| \begin{bmatrix} B_k \\ \lambda \bar{B}_k \end{bmatrix} y - \begin{bmatrix} \beta_1 e_1 \\ 0 \end{bmatrix} \right\|,$$

where $\mathrm{range}(V_k) = \mathcal{K}_k$ is generated by applying BIDIAG-1 to the (ill-posed) problem $\min \|Q_A w - b\|_2$. But by definition of Z_k, $x_k^\lambda = Z_k y_k^\lambda = (R^{-1} V_k) y_k^\lambda = R^{-1} w_k^\lambda$. ∎

The intuitive analysis given above has shown that the Krylov subspace $\mathcal{K}_k$ is a legitimate subspace in which to look for a solution to the regularized problem (1.2). The conclusion is that the columns of Z_k form a well-suited basis in which to construct our regularized solution x_k^λ to (1.1).

5. Numerical results. We conclude with an illustration of the performance of our algorithm. All computations were performed in MATLAB.

Our example is a 2D image deblurring problem which consists of recovering the original $N \times N$ image from an image blurred by the optical system used to record it. The matrix A is a symmetric $N^2 \times N^2$ doubly Toeplitz matrix given by $A = (2\pi\sigma^2)^{-1} T \otimes T$, where T is an $N \times N$ symmetric banded Toeplitz matrix whose first row is

$$\texttt{z = [exp(-([0:band-1].\^{}2)/(2*}\sigma^2\texttt{));zeros(1,N-band)].}$$

The parameter `band` is the half-bandwidth of the matrix T, and the parameter σ controls the width of the underlying Gaussian point spread function

$$h(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(\frac{x^2+y^2}{2\sigma^2}\right)$$

that models the degradation of the image and consequently its smoothness. Thus, a larger σ implies a wider Gaussian and thus a more ill-posed problem. We use $N = 64$ and the parameters `band` $= 16$ and $\sigma = 2$. The condition number of A is $2 \cdot 10^{16}$.

The exact image, represented by x_{true}, is a 64×64 subimage of the image `rice` from MATLAB's Image Processing Toolbox, shown as the left image in Figure 5.1. The noise-free blurred image, represented by b_{true}, is computed as $b_{\text{true}} = Ax_{\text{true}}$. The elements of the noise vector e are normally distributed with zero mean, and the standard deviation is chosen such that $\|e\|_2/\|b_{\text{true}}\|_2 = 10^{-2}$. The noisy right-hand side of our system is then $b = b_{\text{true}} + e$; it is shown as the middle image in Figure 5.1.

For the seminorm $\|Lx\|_2$, we use the matrix

$$L = \begin{bmatrix} I \otimes L_1 \\ L_1 \otimes I \end{bmatrix}, \qquad L_1 = \begin{bmatrix} 1 & -1 & & & \\ & 1 & -1 & & \\ & & \ddots & \ddots & \\ & & & 1 & -1 \end{bmatrix},$$

where $L_1 \in \mathbb{R}^{63\times 64}$ is a discrete approximation to the first derivative operator on a regular grid, with no assumptions on boundary conditions. The two matrices A and L are scaled such that $\|A\|_2 = \|L\|_2 = 1$, and $\text{cond}([A^T\ L^T]) = 6.2$.

We define 30 logarithmically spaced values of λ from 10^{-5} to 1 and run 50 iterations of our joint bidiagonalization algorithm using the updating formulas from section 3.4. The right image in Figure 5.1 shows the optimal reconstruction $x_k^{\lambda^{k,\text{opt}}}$ for $k = 50$ with $\lambda^{k,\text{opt}}$ chosen among the 30 candidates such that $\|x_{\text{true}} - x_k^{\lambda^{k,\text{opt}}}\|_2$ is

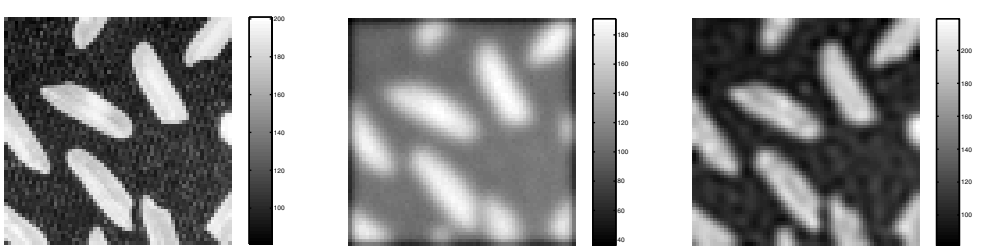


FIG. 5.1. *Exact image x_{true} (left), noisy and blurred image b (middle), and best restoration x_k^{λ} using $L \neq I$ (right).*

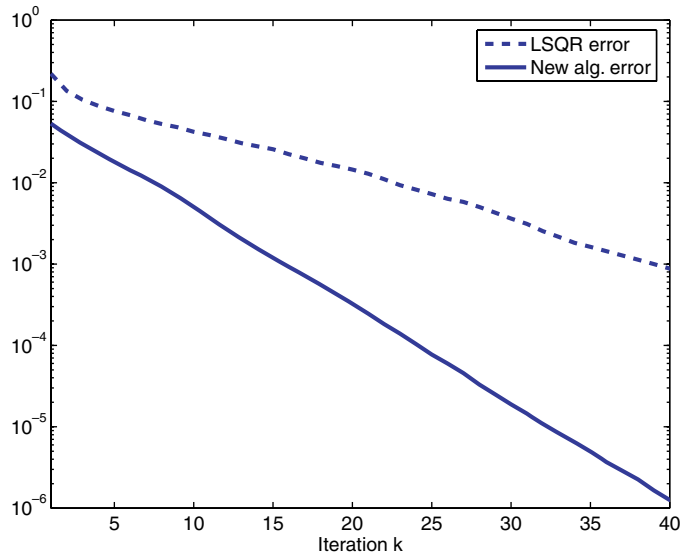


FIG. 5.2. *Error histories as a function of k for a fixed value of λ. The solid line shows the relative error* $\|x_\lambda - x_k^\lambda\|_2/\|x_\lambda\|_2$ *for our new algorithm, while the dotted line shows the relative error* $\|x_\lambda - x_{k,\mathrm{LSQR}}^\lambda\|_2/\|x_\lambda\|_2$ *for LSQR applied to the least squares Tikhonov problem.*

minimum. This image is a very good approximation to the Tikhonov solution which, in turn, is a good approximation to the exact image.

Next, we illustrate the quality of the basis we are using to construct our approximate solution relative to the quality of the basis that is used by standard LSQR for a fixed value of the regularization parameter. We consider the convergence of our approximate Tikhonov solution x_k^λ, for $\lambda = \lambda^{\mathrm{opt}}$ independent of k, to the Tikhonov solution $x_{\lambda^{\mathrm{opt}}}$ (as computed directly through a QR factorization). Recall that λ^{opt} refers to the one of 30 values of λ for which the Tikhonov solution is a minimum. For comparison we also consider the convergence of the approximation $x_{k,\mathrm{LSQR}}^\lambda$ obtained by applying LSQR directly to the Tikhonov least squares problem (again using the same λ). Figure 5.2 shows the relative errors

$$\|x_\lambda - x_k^\lambda\|_2/\|x_\lambda\|_2 \qquad \text{and} \qquad \|x_\lambda - x_{k,\mathrm{LSQR}}^\lambda\|_2/\|x_\lambda\|_2$$

for $\lambda = \lambda^{\mathrm{opt}}$, which is [illegible] k, for iterations $k = 1, 2, \ldots, 40$, where the exact Tikhonov solution x_λ was computed by means of a (computationally tedious) QR factorization. Clearly, our algorithm takes fewer iterations to converge—primarily because the subspace underlying our projection method is better suited for regularization problems than the Krylov subspace underlying LSQR when $L \neq I$. In other words, if one needs an approximate Tikhonov solution with about three correct digits, then our algorithm needs about 15 iterations, while LSQR needs about 40 iterations! We note that, for a fixed λ, the cost of an individual iteration for our algorithm differs from the cost of an individual iteration of LSQR (as our method requires a projection inner iteration, it is more expensive) and that the use of iteration number, as opposed to computational time or complexity, was deliberate in order to illustrate the superior approximation properties of the underlying subspaces in the new approach.

Figure 5.3 shows the "progression" of the L-curve (for the 30 values of λ used in this example) as the number of iterations k increases. We see that after just 10 iterations, the flatter part of the L-curve has become stable, showing that our algo-

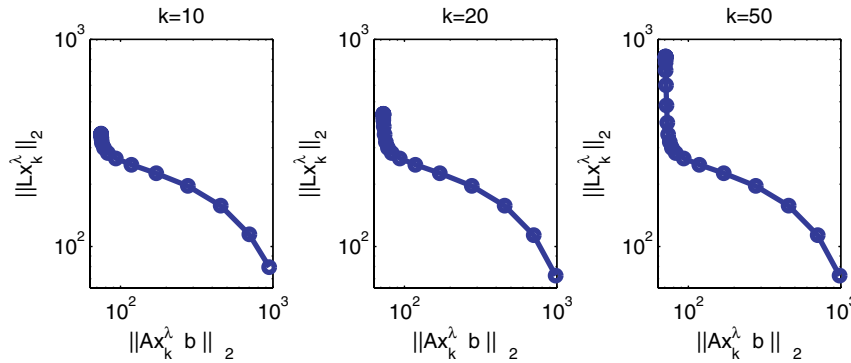


FIG. 5.3. *The "progression" of the L-curves for the solutions x_k^λ as k increases.*

rithm has already captured most of the smooth spectral components of the solution. After 50 iterations, our algorithm has captured so many spectral components that the whole region of the L-curve near the "corner" has become stable, indicating that all the corresponding spectral components have been captured. It is therefore safe to stop the iterations here and use the value of λ chosen by the parameter-choice algorithm applied to the projected problem.

We caution that in showing the graphs in Figure 5.3, our goal is not to advocate the L-curve criterion for choosing the regularization parameter λ. We simply desire to point out that our algorithm provides a cheap mechanism to compute residual norms and seminorms at each k for a set of λ-values. Whether or not the corner of the L-curve corresponds to a near-optimal solution (where optimality is meant in the sense of the solution with minimum 2-norm relative error), the L-curve gives useful qualitative information. That is, the presence of a corner and growth of the vertical component indicates the presence of noise in the basis and thus the need for additional regularization. In this example, the minimal 2-norm solution over the discrete set of λ-values used does in fact correspond to the λ-value at the corner.

6. Conclusion. As has been observed elsewhere in the literature, it is important in some cases to use general-form Tikhonov regularization with $L \neq I$. We presented an iterative algorithm for approximating the solution to the Tikhonov problem in an efficient way which avoids the need to transform to the standard form of the Tikhonov regularization problem and which produces solutions in a subspace which is well suited for general-form Tikhonov regularization. Furthermore, because the regularization is essentially applied to a projected problem, it allows for the possibility of finding the regularization parameter cheaply; that is, λ need not be known a priori. If a set of λ-values is specified a priori, we have given short-term recurrences that allow us to easily update the residual and constraint term for each λ at iterations k, as well as solve for the solution for the desired λ, k pair without storing the Lanczos vectors. Although we have not investigated the potential effects of loss of orthogonality, in our experience with the algorithm so far, it has not been necessary to reorthogonalize at any stage of the algorithm. Finally, we analyzed the regularizing properties of the algorithm and gave an example of its performance for an image deblurring problem.

Acknowledgments. We thank Julianne Chung, Jim Nagy, and Zdeněk Strakoš for their careful reading of an earlier version of the manuscript and for valuable comments and suggestions that helped to improve the paper.

REFERENCES

[1] A. Björk, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, 1996.

[2] M. G. Cox, *Data approximation by splines in one and two independent variables*, in The State of the Art in Numerical Analysis, A. Iserles and M. J. D. Powell, eds., Clarendon Press, Oxford University Press, New York, 1987.

[3] L. Eldén, *A weighted pseudoinverse, generalized singular values, and constrained least squares problems*, BIT, 22 (1982), pp. 487–502.

[4] M. Fedi, P. C. Hansen, and V. Paoletti, *Tutorial: Analysis of depth resolution in potential-field inversion*, Geophys., 70 (2005), pp. A1–A11.

[5] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, MD, 1996.

[6] C. W. Groetsch, *Inverse Problems in the Mathematical Sciences*, Vieweg, Wiesbaden, Germany, 1993.

[7] M. Hanke and C. R. Vogel, *Two-level preconditioners for regularized inverse problems.* I. *Theory*, Numer. Math., 83 (1999), pp. 385–402.

[8] P. C. Hansen, M. Kilmer, and R. H. Kjeldsen, *Exploiting residual information in the parameter choice for discrete ill-posed problems*, BIT, 46 (1996), pp. 41–59.

[9] P. C. Hansen, *Regularization, GSVD and truncated GSVD*, BIT, 29 (1989), pp. 491–504.

[10] P. C. Hansen, *The discrete Picard condition for discrete ill-posed problems*, BIT, 30 (1990), pp. 658–672.

[11] P. C. Hansen, *Regularization tools: A Matlab package for analysis and solution of discrete ill-posed problems*, Numer. Algorithms, 6 (1994), pp. 1–35.

[12] M. Jacobsen, P. C. Hansen, and M. A. Saunders, *Subspace preconditioned LSQR for discrete ill-posed problems*, BIT, 43 (2003), pp. 975–989.

[13] T. K. Jensen and P. C. Hansen, *Smoothing-norm preconditioning for regularizing minimum-residual methods*, SIAM J. Matrix Anal. Appl., 29 (2006), pp. 1–14.

[14] M. E. Kilmer and D. P. O'Leary, *Choosing regularization parameters in iterative methods for ill-posed problems*, SIAM J. Matrix Anal. Appl., 22 (2001), pp. 1204–1221.

[15] C. C. Paige and M. A. Saunders, *LSQR: An algorithm for sparse linear equations and sparse least squares problems*, ACM Trans. Math Software, 8 (1982), pp. 43–71.

[16] M. A. Saunders, *Computing projections with LSQR*, BIT, 37 (1997), pp. 96–104.

[17] H. Zha, *Computing the generalized singular values/vectors of large sparse or structured matrix pairs*, Numer. Math., 72 (1996), pp. 391–417.

SIAM J. SCI. COMPUT.
Vol. 29, No. 1, pp. 331–360
© 2007 Society for Industrial and Applied Mathematics

ANALYSIS OF NUMERICALLY INDUCED OSCILLATIONS IN 2D FINITE-ELEMENT SHALLOW-WATER MODELS PART I: INERTIA-GRAVITY WAVES*

DANIEL Y. LE ROUX†, VIRGILE ROSTAND†, AND BENOIT POULIOT†

Abstract. The numerical approximation of shallow-water models is a delicate problem. For most of the discretization schemes, the coupling between the momentum and the continuity equations usually leads to anomalous dispersion in the representation of fast waves. A dispersion relation analysis is employed here to ascertain the presence and determine the form of spurious modes as well as the dispersive nature of the finite-element Galerkin mixed formulation of the two-dimensional linearized shallow-water equations. Nine popular finite-element pairs are considered using a variety of mixed interpolation schemes. For each pair the frequency or dispersion relation is obtained and analyzed, and the dispersion properties are compared analytically and graphically with the continuous case. It is shown that certain choices of mixed interpolation schemes may lead to significant phase and group velocity errors and spurious solutions in the calculation of fast waves. The $P_1^{NC} - P_1$ and RT0 pairs are identified as a promising compromise, provided the grid resolution is high relative to the Rossby radius of deformation for the RT0 element. The numerical solutions of two test problems to simulate fast waves are in good agreement with the analytical results.

Key words. shallow-water equations, finite-element method, dispersion analysis, surface waves

AMS subject classifications. 35L99, 74S05, 65T99, 74J15

DOI. 10.1137/060650106

1. Introduction.

The shallow-water (SW) equations describe the behavior of a shallow homogeneous incompressible and inviscid fluid layer. They are derived from the depth-averaged Navier–Stokes equations under Boussinesq and hydrostatic pressure assumptions. The SW system is extensively used in environmental studies to model hydrodynamics in lakes, estuaries, coastal regions, and other applications.

Galerkin techniques have gradually evolved to become a popular method for this problem class including the finite-element (FE) [14, 18, 27, 31, 32, 36, 44], the spectral-element [28, 43], the finite-volume (FV) [4, 15, 16, 19, 26], and the discontinuous Galerkin [3, 8, 17, 20, 41] methods. Indeed, unstructured elements offer the enhanced flexibility of using grids of variable sizes, shapes, and orientation for representing the boundaries of complex domains and a natural treatment of boundary conditions. Calculations on graded meshes of unstructured grids are hence standard practice [5, 18, 19, 20, 33, 46, 47].

For most numerical methods, including the finite-difference (FD), FV, and FE schemes, the coupling between the momentum and continuity equations is a delicate problem. One of the issues associated with mixed formulations is the possibility of spurious modes that may arise for certain choices of grids and bases. For example, the piecewise linear Galerkin discretization of the SW equations (for velocity and elevation variables) is usually plagued by spurious oscillations [28, 45]. The appearance of such oscillations is mainly due to an inappropriate placement of variables on the grid and/or a bad choice of approximation function spaces. Improvements have been achieved

*Received by the editors January 16, 2006; accepted for publication (in revised form) July 25, 2006; published electronically February 5, 2007. This work was supported by grants from the Natural Sciences and Engineering Research Council (NSERC).
http://www.siam.org/journals/sisc/29-1/65010.html
†Département de Mathématiques et de Statistique, Université Laval, Québec G1K 7P4, QC, Canada (dleroux@mat.ulaval.ca, vrostand@mat.ulaval.ca, bpouliot@giref.ulaval.ca).

through the use of suitable FD grids [6, 39], a variety of mixed-order FE interpolation schemes [2, 28, 27, 32, 33, 48], and a wave equation formulation [29, 36].

This difficulty with mixed methods is not specific to the SW problem alone but is well known also in other contexts such as primitive variable viscous flows where it has been extensively investigated for the Galerkin FE method [13, 23]. However, the situation in the mixed SW problem is different and, as noted initially in [45] and in subsequent studies [27, 32, 47], an analysis of the dispersion relation for a given formulation will explicitly ascertain the presence and determine the form of spurious modes as well as the dissipative/dispersive nature of a given formulation.

The dispersion relation analysis assumes that the solution of the differential equations can be separated and formulated as periodic in space and time. It differs from the Fourier analysis in that the time variable remains continuous by using the harmonic form of the equations. Consequently, a Fourier expansion is performed only for the nodal unknowns that appear in the discrete equations. The analysis results in a dispersion relation where the magnitude of the temporal frequency is expressed in terms of wave numbers. Such a technique was applied early on to the SW model in [38].

Two-dimensional dispersion analysis of FE SW discretizations using the same linear basis functions for all variables was examined first. One grid configuration is considered in [45], while the influence of grid configuration on the dispersion properties is analyzed in [9, 10, 21] for the mixed primitive variable form and in [37] in the case of the wave equation formulation. Phase behavior of FV SW discretizations has been studied in [24, 42]. To our knowledge, the two-dimensional dispersion relation analysis of the SW equations using the $P_1^{NC} - P_1$ FE pair [35] was the first study analyzing the dispersion relation of a FE scheme using different approximation spaces for the velocity and surface-elevation fields on meshes made up of triangles. The aim of the present study is to extend the two-dimensional dispersion relation analysis of the FE SW equations to a variety of mixed-order interpolation schemes. Because all the aforementioned problems occur primarily in the context of linear formulations, solving linear equations is sufficient for our purpose.

The paper is developed as follows: The linear SW equations and the Galerkin FE discretization schemes are presented in section 2. The discrete operators are computed in section 3 and the dispersion relations are obtained. The latter are compared analytically and graphically with the continuous case in section 4. This dispersion analysis is followed by numerical tests in section 5. Some concluding remarks complete the study.

2. Discretization of the linear SW equations.

2.1. Governing equations. Let Ω be the model domain with boundary Γ. The inviscid linear SW equations are expressed in Cartesian coordinates [30] as

$$\bar{\mathbf{u}}_t + f\,\mathbf{k} \times \bar{\mathbf{u}} + g\,\nabla\bar{\eta} = 0\,, \tag{2.1}$$

$$\bar{\eta}_t + H\,\nabla \cdot \bar{\mathbf{u}} = 0\,, \tag{2.2}$$

where $\bar{\mathbf{u}} = (\bar{u}, \bar{v})$ is the velocity field, $\bar{\eta}$ is the surface elevation with respect to the reference level $z = 0$, g is the gravitational acceleration, $\mathbf{k}$ is a unit vector in the vertical direction, and the mean depth H and the Coriolis parameter f are assumed constant. Note that $\bar{\eta}$ would be the pressure in the Navier–Stokes equations. For a contained flow, (2.1) and (2.2) are solved subject to the no-normal flow boundary condition $\bar{\mathbf{u}} \cdot \mathbf{n} = 0$ on Γ, where $\mathbf{n}$ is the outward pointing normal at the boundary.

For this analysis we seek periodic solutions of (2.1)–(2.2) of the form

$$\bar{\mathbf{u}}(x,y,t) = \mathbf{u}(x,y)e^{i\omega t}, \quad \bar{\eta}(x,y,t) = \eta(x,y)e^{i\omega t}, \tag{2.3}$$

where $\mathbf{u} = (u, v)$, η are amplitudes, and ω is the angular frequency, and we obtain

$$i\omega\,\mathbf{u} + f\,\mathbf{k}\times\mathbf{u} + g\,\nabla\eta = 0, \tag{2.4}$$

$$i\omega\,\eta + H\,\nabla\cdot\mathbf{u} = 0. \tag{2.5}$$

2.2. Spatial discretization.

2.2.1. The weak formulation. We assume $\mathbf{u}$ and η belong to the spaces $\mathbf{V}$ and Q, respectively, with Q and $\mathbf{V}$ being either the square-integrable space $L^2(\Omega)$ or the Sobolev space $H^1(\Omega)$, i.e., the space of functions in $L^2(\Omega)$ whose first derivatives belong to $L^2(\Omega)$. The weak formulation of (2.4) and (2.5) requires the test functions $\boldsymbol{\varphi}$ (whose x- or y-component is formally denoted by φ) and ψ to be sufficiently regular and to belong, respectively, to the same function space as $\mathbf{u}$ and η, such that

$$i\omega\int_\Omega \mathbf{u}\cdot\boldsymbol{\varphi}\,d\Omega + \int_\Omega f(\mathbf{k}\times\mathbf{u})\cdot\boldsymbol{\varphi}\,d\Omega + g\int_\Omega \nabla\eta\cdot\boldsymbol{\varphi}\,d\Omega = 0, \tag{2.6}$$

$$i\omega\int_\Omega \eta\,\psi\,d\Omega + H\int_\Omega \nabla\cdot\mathbf{u}\,\psi\,d\Omega = 0, \tag{2.7}$$

where $d\Omega$ is the area element.

Depending on the regularity of functions $\mathbf{u}$ and η, the terms containing derivatives in (2.6) and (2.7) may be integrated by parts using Green's theorem. In this way, (2.6) is rewritten if needed as

$$i\omega\int_\Omega \mathbf{u}\cdot\boldsymbol{\varphi}\,d\Omega + \int_\Omega f(\mathbf{k}\times\mathbf{u})\cdot\boldsymbol{\varphi}\,d\Omega - g\int_\Omega \eta\,\nabla\cdot\boldsymbol{\varphi}\,d\Omega = 0, \tag{2.8}$$

and for (2.7) we obtain

$$i\omega\int_\Omega \eta\,\psi\,d\Omega - H\int_\Omega \mathbf{u}\cdot\nabla\psi\,d\Omega = 0, \tag{2.9}$$

by letting $\mathbf{u}\cdot\mathbf{n} = 0$ on Γ for all $\mathbf{u}$ belonging to $\mathbf{V}$. Hence, we also have $\boldsymbol{\varphi}\cdot\mathbf{n} = 0$ on Γ.

2.2.2. Galerkin FE discretization. The Galerkin method approximates the solution of (2.6) and (2.7), and eventually (2.8) and (2.9) if integration by parts needs to be performed, in finite-dimensional subspaces. Consider a FE triangulation $\mathcal{T}_h$, of the polygonal domain Ω, where h is a representative meshlength parameter that measures resolution. For triangle $K \in \mathcal{T}_h$, let $P_n(K)$ denote the space of polynomials of degree n on K.

The discrete solutions $\mathbf{u}_h$ and η_h sought belong to finite-dimensional spaces $\mathbf{V}_h$ and Q_h, respectively, whose restrictions on K belong to $P_k(K)\times P_k(K)$ for $\mathbf{u}_h$ and to $P_l(K)$ for η_h. The components of $\mathbf{u}_h$ and η_h are represented over a triangle K_i by interpolating functions $\varphi(x,y)$ of degree k and $\psi(x,y)$ of degree l (in the x- and y-components), respectively, with $\boldsymbol{\varphi}(x,y)$ belonging to $\mathbf{V}_h$ and $\psi(x,y)$ belonging to Q_h. We thus have

$$\mathbf{u}_h = \sum_{j\in S^{\mathbf{u}}_{K_i}} \mathbf{u}_j\,\varphi_j, \quad \eta_h = \sum_{j\in S^{\eta}_{K_i}} = \eta_j\,\psi_j, \tag{2.10}$$

where j represents a node of K_i and $S^{\mathbf{u}}_{K_i}$ and $S^{\eta}_{K_i}$ denote the set of nodes of K_i.

The expansions of $\mathbf{u}_h$ and η_h over the whole domain Ω are then obtained by summing (2.10) over all triangles K_i of the triangulation $\mathcal{T}_h$. Introducing the FE basis leads to a FE statement as in (2.6) and (2.7), and eventually in (2.8) and (2.9), but with $\mathbf{u}, \eta$ replaced by the FE trial functions $\mathbf{u}_h, \eta_h$ and φ, ψ replaced by the corresponding FE test functions. We then decompose the integrals in (2.6) and (2.7) into triangle contributions.

For the purposes of the following analysis we consider a uniform mesh made up of biased right triangles as in the figures of section 3, and h is thus taken as a constant in the x- and y-directions.

2.2.3. The FE pairs. We now introduce the schemes that are used for the spatial discretization. Nine candidate FE pairs for representing velocity and surface elevation are described and evaluated in the remainder of this paper. Conventional FE terminology is adopted to describe the FE pairs. The nomenclature $P_m - P_n$ means that velocity components and surface elevation are represented, respectively, as piecewise-defined polynomials of degree m and n. Enhancements of this basic terminology are introduced as needed.

Common to the first six FE velocity/surface-elevation pairs (as shown in Figures 4.2(d) and 4.3) is a piecewise-linear continuous representation of surface elevation, and they differ from one another in their representation of velocity. The $P_1 - P_1$ pair has velocity colocated at triangle vertices, and the corresponding basis functions are piecewise-linear. The MINI element [7] also has continuous piecewise-linear basis functions at the vertices, but bubble functions are added at the barycenters for velocity in order to stabilize the pair. The $P_1^{NC} - P_1$ pair [27, 35] has velocity nodes at triangle edge midpoints, and linear basis functions are used to approximate the two velocity components on the element's two-triangle support. Since this particular representation of velocity is continuous only across triangle boundaries at midpoint nodes, and discontinuous everywhere else around a triangle boundary, this element is termed nonconforming (NC) in the FE literature. The $P_1 \,\mathrm{iso}\, P_2 - P_1$ element pair [12] has piecewise linear basis functions for velocity on a refined triangulation obtained by dividing each triangle into four subtriangles using the midpoints of triangle sides. There are thus six velocity nodes over each unrefined triangle, the same as for a quadratic approximation of velocity, termed P_2. The designation P_1 in $P_1 \,\mathrm{iso}\, P_2$ denotes linear velocity elements on subtriangles, whereas $\mathrm{iso}\, P_2$ indicates that the nodal placement is that associated with quadratic elements on unrefined triangles. The $P_2 - P_1$ pair [25], also known in the literature as the Taylor–Hood element, has quadratic velocity basis functions. Finally, the $P_0 - P_1$ pair, shown in Figure 4.2(d), has a piecewise-constant representation of velocity.

Common to the last three pairs (shown in Figures 4.2(a)–(c)) is a discontinuous piecewise-constant representation of surface elevation. The RT0 element, also called low-order Raviart–Thomas element [40], is based on flux conservation on element edges and has normal velocity components at triangle midedge points. Finally, the $P_1^{NC} - P_0$ and $P_2 - P_0$ pairs have, respectively, P_1^{NC} and P_2 representations of velocity.

3. Computation of the dispersion relations.

3.1. Matrix computations. For all FE pairs having a piecewise-constant representation of surface-elevation an integration by parts of $\nabla\eta$ in the left-hand side of (2.6) is performed, and hence (2.8) is employed. For the $P_1^{NC} - P_1$ and $P_0 - P_1$ pairs, the term $\nabla \cdot \mathbf{u}$ appearing in (2.7) is integrated by parts to avoid computing u

and v derivatives, and (2.9) is used instead.

After substitution of $\mathbf{u}_h$ and η_h from (2.10) into (2.6) and (2.7), the discrete formulation is obtained. We then need to compute

$$M_{i,j} = \int \boldsymbol{\varphi} \cdot \boldsymbol{\varphi} \; d\Omega, \quad G_{i,j} = \int \nabla \psi_i \cdot \boldsymbol{\varphi} \; d\Omega, \quad D_{i,j} = \int \nabla \cdot \boldsymbol{\varphi} \; \psi_j \, d\Omega, \tag{3.1}$$

where $M_{i,j}, G_{i,j}$, and $D_{i,j}$ represent the mass, gradient, and divergence elementary FE matrices, respectively. When (2.8) is employed instead of (2.6) the elementary gradient matrix is then $-D_{j,i}$, and if (2.9) is used, the elementary divergence matrix becomes $-G_{j,i}$. Assembling the elementary matrices leads to the stencils of Figures 3.1, 3.2, 3.3, 3.4, and 3.5 for the pairs examined here. The symbols • and black arrows indicate the location of velocity and normal velocity nodes, respectively. The arrow points in the direction of the chosen normal. When these symbols are represented in grey this means that velocity and normal velocities have zero values at the corresponding nodes. The symbol ○ indicates nodes for surface elevation and dotted circles mean that the elevation is zero at those nodes.

For the subsequent dispersion analysis we consider two possible types of barycenters, corresponding to lower left and upper right triangles, and three types of faces: horizontal, vertical, and diagonal. Note that for the $P_1^{NC} - P_0$ and $P_2 - P_0$ pairs in Figure 3.2, the x- (resp., y-) component of the surface elevation gradient is zero on horizontal (resp., vertical) faces. Further, in Figure 3.4 for the $P_2 - P_1$ pair, the surface-elevation gradient is zero at vertex nodes.

3.2. The continuous case. The free modes of (2.4)–(2.5) are examined by perturbing about the basic state $u = v = \eta = 0$. Because the governing equations are linear, the solution may be examined by considering the behavior of one Fourier mode. We then seek solutions of (2.4)–(2.5) of the form $(u, v, \eta) = (\tilde{u}, \tilde{v}, \tilde{\eta})\, e^{i(kx+ly)}$, where k and l are the wave numbers in the x- and y-directions, respectively. Substitution into (2.4)–(2.5) leads to a square matrix system for the amplitudes $\tilde{u}, \tilde{v}, \tilde{\eta}$. For a nontrivial solution to exist, the determinant of the matrix must equal zero, and this constraint leads to a relationship between the two wave numbers k and l and the frequency ω. The so-called dispersion relation is then obtained for the frequency

$$\omega \, (\, \omega^2 - f^2 - g \, H \, (\, k^2 + l^2 \,) \,) = 0 \, . \tag{3.2}$$

The first solution $\omega = 0$ is the geostrophic mode, and it would correspond to the slow Rossby mode on a β-plane, while the other two solutions

$$\omega_{AN} = \pm \sqrt{f^2 + g \, H \, (\, k^2 + l^2 \,)} \tag{3.3}$$

correspond to the free-surface gravitational modes with rotational correction. Since ω is purely real, all modes are neutrally stable and neither amplify nor decay.

3.3. The discrete case. For each FE pair, the discrete momentum and continuity equations are obtained from the stencils of Figures 3.1, 3.2, 3.3, 3.4, and 3.5. As previously mentioned, the following analysis will consider a uniform mesh made up of biased right triangles as in the figures of section 3, and h is thus taken as a constant in the x- and y-directions. Because nodal unknowns may be located on different types of nodes, i.e., vertices, faces, and barycenters, selected discrete equations for each type of node are retained. For example, the $P_2 - P_1$ pair leads to considering four discrete momentum equations, one at a typical vertex node and three on the three possible

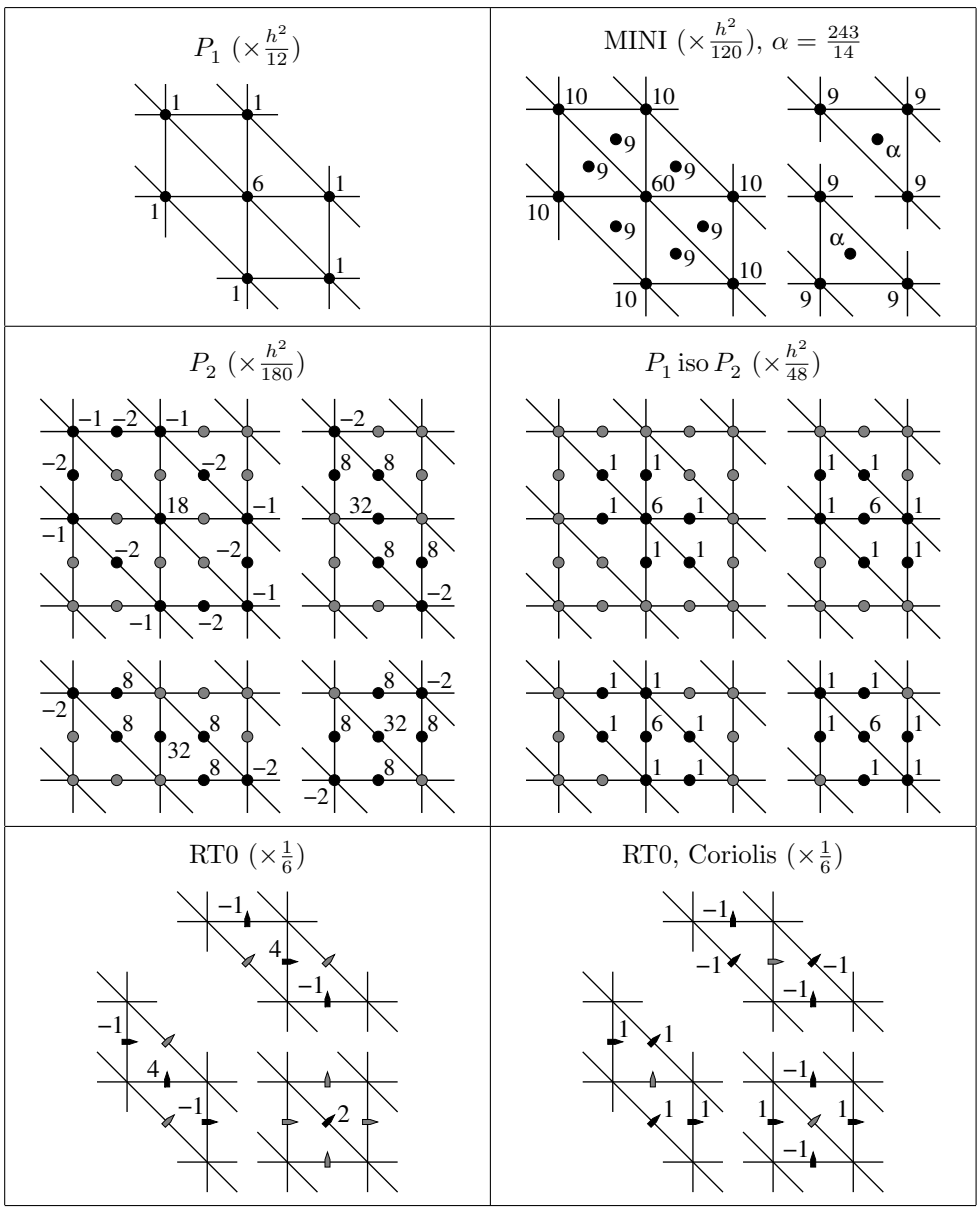


FIG. 3.1. *Mass stencils for several FE discretizations. The symbols ● and black arrows indicate the location of velocity and normal velocity nodes, respectively. The arrow points in the direction of the chosen normal. Grey symbols mean that velocity and normal velocities have zero values at the corresponding nodes.*

types of faces, i.e., horizontal, vertical, and diagonal (written as H, V, and D, respectively, in the following), and only one discrete continuity equation at a typical vertex node (written as S in the following). For the $P_0 - P_1$ pair, two discrete momentum equations are retained at the two possible types of barycenters (corresponding to lower left and upper right triangles, written as C_1 and C_2, respectively, in the following). More details are given in [10, 21] for the $P_1 - P_1$ pair, and in [35] for the $P_1^{NC} - P_1$

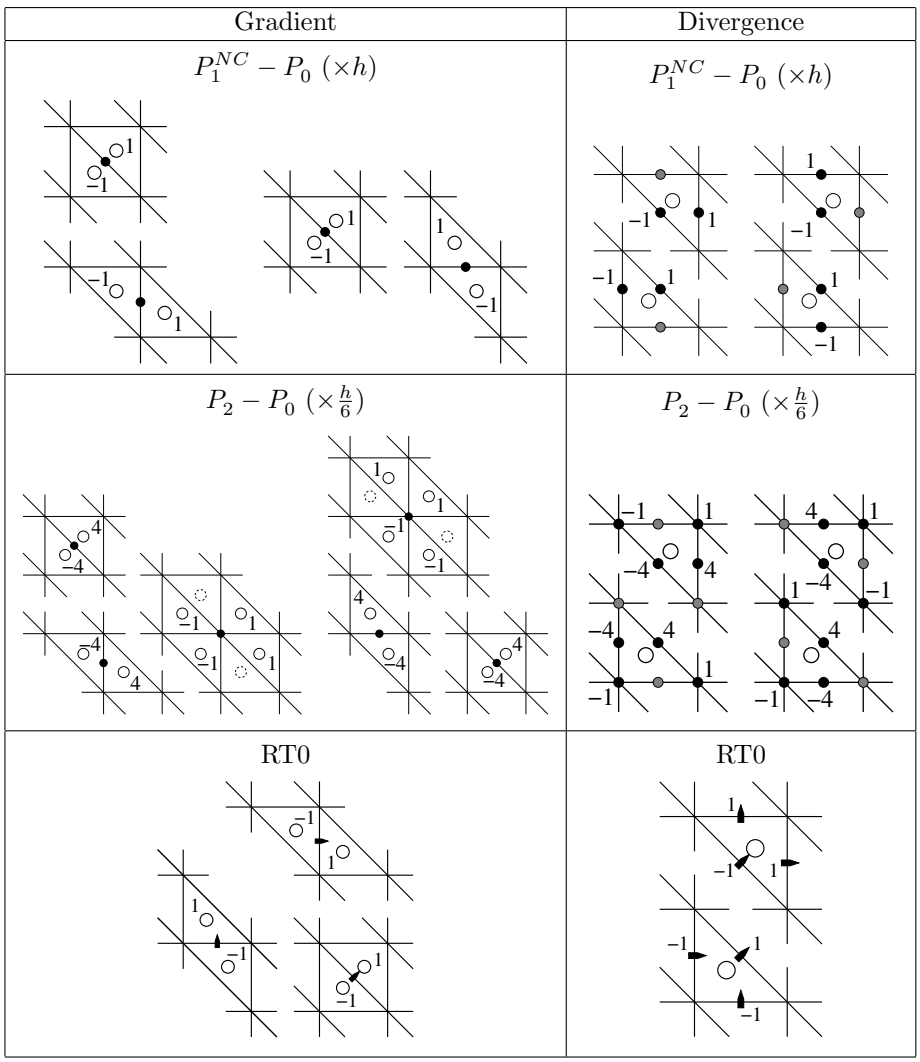


FIG. 3.2. *As for Figure* 3.1 *but for the gradient and divergence stencils of the* $P_1^{NC} - P_0$, $P_2 - P_0$, *and RT*0 *pairs. For the* $P_1^{NC} - P_0$ *and* $P_2 - P_0$ *pairs, the x- and y-components are on the left and right sides, respectively. The symbol* ○ *indicates nodes for surface elevation and dotted circles mean that the elevation is zero at those nodes.*

one, to compute their dispersion relations. For all pairs, the typical nodes belonging to the same set (vertices, faces, midedge points, or barycenters) are hence distributed on a regular grid of size h.

As for the continuum case, the dispersion relation for the discrete scheme is found through a Fourier expansion. The discrete solutions corresponding to $(u_j, v_j, \eta_j) = (\tilde{u}, \tilde{v}, \tilde{\eta})\, e^{i(kx_j + ly_j)}$ are sought at node j ($j = 1, 2, 3 \ldots$), where (u_j, v_j, η_j) are the nodal unknowns that appear in the selected discrete equations and $(\tilde{u}, \tilde{v}, \tilde{\eta})$ are amplitudes. The (x_j, y_j) coordinates are expressed in terms of a distance to a reference node. Substitution into the discrete equations leads to a square matrix system for the Fourier amplitudes. The dispersion relation is then obtained by setting the determinant of the matrix system to zero.

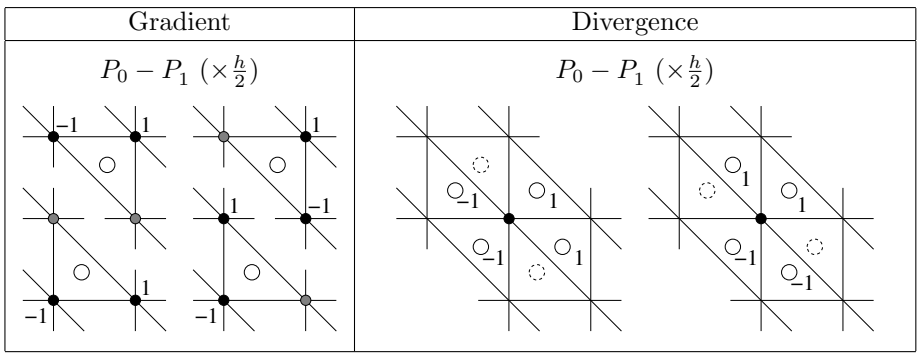


FIG. 3.3. *As for Figure* 3.2 *but for the* $P_0 - P_1$ *pair.*

We let I_n be the $n \times n$ identity matrix, M an $m \times n$ matrix with m and n two positive integers, and $M^* = \overline{M}^T$ the conjugate transpose of M, and we define

$$a = \frac{1}{3}(\,3 + \cos kh + \cos lh + \cos(k-l)h\,), \quad A = \begin{pmatrix} i\omega & -f \\ f & i\omega \end{pmatrix},$$

with $a \geq \frac{1}{2}$ for all k and l.

We now examine the dispersion relations corresponding to the nine pairs that are considered in this study.

3.4. The $P_1^{NC} - P_0$ pair. The selected discrete equations are written in the matrix form

$$\begin{pmatrix} A & 0 & 0 & \\ 0 & A & 0 & -g\,B^* \\ 0 & 0 & A & \\ & HB & & \frac{3}{2}i\omega I_2 \end{pmatrix} \begin{pmatrix} \tilde{\mathbf{u}}_H \\ \tilde{\mathbf{u}}_V \\ \tilde{\mathbf{u}}_D \\ \tilde{\boldsymbol{\eta}} \end{pmatrix} = 0, \tag{3.4}$$

where $\tilde{\boldsymbol{\eta}} = (\tilde{\eta}_{C_1}, \tilde{\eta}_{C_2})$, and

$$B = \frac{3}{h} \begin{pmatrix} 0 & -b_1 & -\overline{b}_3 & 0 & b_2 & b_2 \\ 0 & \overline{b}_1 & b_3 & 0 & -\overline{b}_2 & -\overline{b}_2 \end{pmatrix},$$

with

$$b_1 = e^{i(k-2l)\frac{h}{6}}, \quad b_2 = e^{i(k+l)\frac{h}{6}}, \quad b_3 = e^{i(2k-l)\frac{h}{6}}.$$

Vanishing the above 8×8 determinant leads to

$$\omega_{1,2} = 0, \;\; \omega_{3,4} = \pm f, \;\; \omega_{5,6} = \pm\sqrt{f^2 + 6\frac{gH}{h^2}\,(4-\alpha)}, \;\; \omega_{7,8} = \pm\sqrt{f^2 + 6\frac{gH}{h^2}\,(4+\alpha)},$$

with $\alpha = \sqrt{2(3a + \cos kh + \cos lh)}$. For infinitesimal mesh spacing we obtain

$$\omega_{5,6} = \pm\sqrt{f^2 + gH\left(\frac{9}{4}(k^2+l^2) - \frac{3}{2}kl\right)} + O(h^2), \tag{3.5}$$

$$\omega_{7,8} = \pm 4\sqrt{3}\frac{\sqrt{gH}}{h} + O(h). \tag{3.6}$$

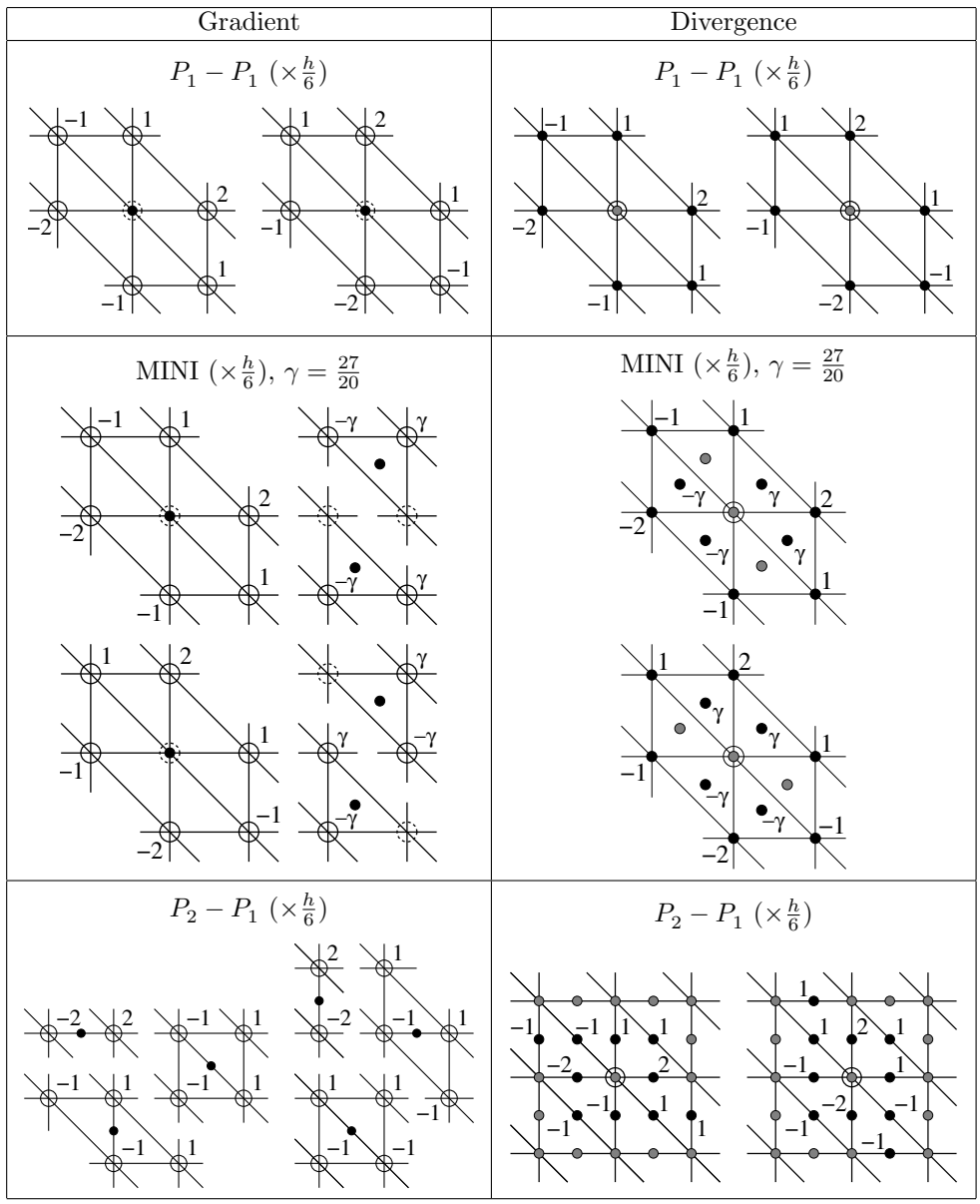


FIG. 3.4. *As for Figure* 3.2 *but for the* $P_1 - P_1$*, MINI and* $P_2 - P_1$ *pairs. For the* $P_1 - P_1$ *and* $P_2 - P_1$ *pairs, the x- and y-components are on the left and right sides, respectively, and they are at the top and bottom, respectively, for MINI.*

Note that neither $\omega_{5,6}$ nor $\omega_{7,8}$ coincide with the continuous solution obtained from (3.2) in the limit as mesh spacing $h \to 0$. The roots $\omega_{7,8}$ correspond to a spurious mode from the $P_1^{NC} - P_0$ discretization scheme. Such a mode of type $O(\frac{1}{h})$ has already been observed in [34], where the one-dimensional SW equations are discretized using the discontinuous Galerkin method. Finally, the roots $\omega_{3,4}$ represent spurious inertial oscillations. It has been shown in [1] for the FD CD-grid and in [35] for the $P_1^{NC} - P_1$ pair that such spurious modes take the form of propagating inertial oscillations and

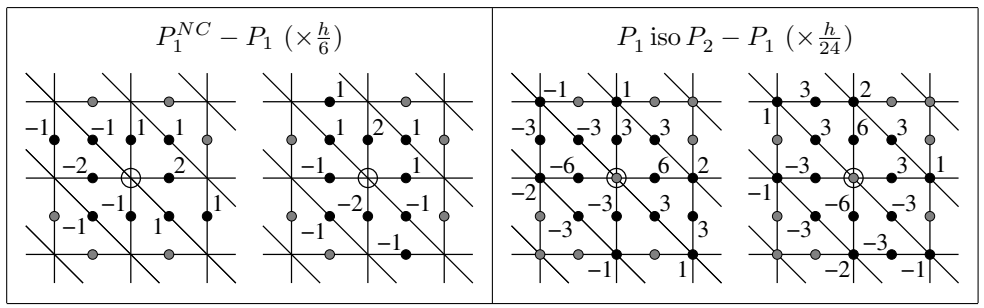


FIG. 3.5. *Divergence stencils for the $P_1^{NC} - P_1$ and P_1 iso $P_2 - P_1$ pairs. For each pair, the x- and y-components are on the left and right sides, respectively.*

have no particular spatial characteristics. They are a consequence of using more velocity nodes than surface-elevation nodes.

3.5. The $P_2 - P_0$ pair. The selected discrete equations lead to

$$\begin{pmatrix} a_1A & a_3A & a_4A & a_2A & \\ a_3A & a_8A & a_7A & a_6A & \\ a_4A & a_7A & a_8A & a_5A & -gB^* \\ a_2A & a_6A & a_5A & a_8A & \\ \multicolumn{4}{c}{HB} & \frac{i\omega}{2}I_2 \end{pmatrix} \begin{pmatrix} \tilde{\mathbf{u}}_S \\ \tilde{\mathbf{u}}_H \\ \tilde{\mathbf{u}}_V \\ \tilde{\mathbf{u}}_D \\ \tilde{\boldsymbol{\eta}} \end{pmatrix} = 0, \tag{3.7}$$

where $\tilde{\boldsymbol{\eta}} = (\tilde{\eta}_{C_1}, \tilde{\eta}_{C_2})$,

$$B = \frac{1}{6h} \begin{pmatrix} -\bar{b}_1 + \bar{b}_2 & -\bar{b}_1 + \bar{b}_5 & 0 & -4\bar{b}_6 & -4\bar{b}_4 & 0 & 4\bar{b}_3 & 4\bar{b}_3 \\ b_1 - b_2 & b_1 - b_5 & 0 & 4b_6 & 4b_4 & 0 & -4b_3 & -4b_3 \end{pmatrix},$$

with $a_1 = \frac{1}{30}(4-a)$, $a_8 = \frac{8}{45}$, and

$$\begin{aligned} a_2 &= -\tfrac{1}{45}\cos\tfrac{(k+l)h}{2}, & a_5 &= \tfrac{4}{45}\cos\tfrac{kh}{2}, & b_1 &= e^{i(k+l)\frac{h}{3}}, & b_4 &= e^{i(2k-l)\frac{h}{6}}, \\ a_3 &= -\tfrac{1}{45}\cos(\tfrac{k-2l}{2})h, & a_6 &= \tfrac{4}{45}\cos\tfrac{lh}{2}, & b_2 &= e^{-i(2k-l)\frac{h}{3}}, & b_5 &= e^{i(k-2l)\frac{h}{3}}, \\ a_4 &= -\tfrac{1}{45}\cos\tfrac{(2k-l)h}{2}, & a_7 &= \tfrac{4}{45}\cos\tfrac{(k-l)h}{2}, & b_3 &= e^{-i(k+l)\frac{h}{6}}, & b_6 &= e^{-i(k-2l)\frac{h}{6}}. \end{aligned}$$

Vanishing the above 10×10 determinant leads to a polynomial in ω of the form

$$(\omega^2 - f^2)^2 \left(c_6\,\omega^6 + c_4\,\omega^4 + c_2\,\omega^2 + c_0 \right) = 0,$$

where c_0, c_2, c_4, and c_6 are functions of k and l. We then obtain

$$\omega_{1,2} = \pm O(h), \quad \omega_{3,4,5,6} = \pm f \text{ (double root)}, \quad \omega_{7,8} = \pm O(1), \quad \omega_{9,10} = \pm O\left(\frac{1}{h}\right).$$

We have found that $c_0 \neq 0$, and hence $\omega = 0$ is no longer a solution to the dispersion relation, and the possibility of damped or growing slow modes occurs. However, we note the appearance of modes $\omega_{1,2} = \pm O(h)$ which are expected to be close to zero in the limit as mesh spacing $h \to 0$. As for the $P_1^{NC} - P_0$ pair, the roots $\omega_{3,4,5,6}$ represent

spurious inertial oscillations. They are again due to an imbalance in the number of degrees of freedom between velocity and surface-elevation nodes. We have also found, as for the $P_1^{NC} - P_0$ pair, that the frequencies corresponding to inertia-gravity modes, here $\omega_{7,8}$, do not coincide with the continuous solution obtained from (3.2) in the limit as mesh spacing $h \to 0$. Finally, the frequencies $\omega_{9,10}$ again correspond to a spurious mode from the $P_2 - P_0$ discretization scheme, as for the $P_1^{NC} - P_0$ pair.

3.6. The RT0 pair. The discrete equations lead to the following system:

$$\begin{pmatrix} A_{\mathrm{RT0}} & -g\,B^* \\ HB & i\omega\frac{h^2}{2}I_2 \end{pmatrix} \begin{pmatrix} \tilde{\mathbf{J}} \\ \tilde{\boldsymbol{\eta}} \end{pmatrix} = 0, \tag{3.8}$$

where $\tilde{\mathbf{J}} = (\tilde{J}_H, \tilde{J}_D, \tilde{J}_V)$ and J_p is the velocity flux through the edge containing node p (here H, D, or V), with $J_p = \tilde{J}_p\, e^{i(kx_p + ly_p)}$. We also have $\tilde{\boldsymbol{\eta}} = (\tilde{\eta}_{C_1}, \tilde{\eta}_{C_2})$,

$$A_{\mathrm{RT0}} = \frac{1}{3}\begin{pmatrix} 2i\omega & a_2 f & -a_3(i\omega - f) \\ -a_2 f & i\omega & a_1 f \\ -a_3(i\omega + f) & -a_1 f & 2i\omega \end{pmatrix},\quad B = \begin{pmatrix} -b_1 & b_2 & -b_3 \\ \bar{b}_1 & -\bar{b}_2 & \bar{b}_3 \end{pmatrix},$$

and

$$\begin{aligned} a_1 &= \cos\tfrac{kh}{2}, \qquad a_2 = \cos\tfrac{lh}{2}, \qquad a_3 = \cos\tfrac{(k-l)h}{2}, \\ b_1 &= e^{i(k-2l)\frac{h}{6}}, \quad b_2 = e^{i(k+l)\frac{h}{6}}, \quad b_3 = e^{-i(2k-l)\frac{h}{6}}. \end{aligned}$$

For the 5×5 determinant to vanish we obtain $\omega = 0$ and four additional roots corresponding to gravity modes. For infinitesimal mesh spacing we have

$$\omega_1 = 0, \quad \omega_{2,3} = \omega_{AN} + O(h^2), \quad \omega_{4,5} = \pm 6\frac{\sqrt{gH}}{h} + O(h).$$

Note that only $\omega_{2,3}$ coincide with the continuous solution obtained from (3.2) in the limit as mesh spacing $h \to 0$, while $\omega_{4,5}$ presumably correspond to spurious modes from the RT0 discretization scheme.

3.7. The $P_0 - P_1$ pair. The selected discrete equations are written in the matrix form

$$\begin{pmatrix} A & 0 & \\ 0 & A & -gB^* \\ & HB & ia\omega \end{pmatrix} \begin{pmatrix} \tilde{\mathbf{u}}_{C_1} \\ \tilde{\mathbf{u}}_{C_2} \\ \tilde{\eta}_S \end{pmatrix} = 0, \tag{3.9}$$

where $B = \frac{1}{h}\begin{pmatrix} b_1 & b_2 & -\bar{b}_1 & -\bar{b}_2 \end{pmatrix}$, with

$$b_1 = e^{i(k+l)\frac{h}{3}}(1 - e^{-ikh}), \quad b_2 = e^{i(k+l)\frac{h}{3}}(1 - e^{-ilh}).$$

Vanishing the above 5×5 determinant leads to

$$\omega_1 = 0, \quad \omega_{2,3} = \pm f, \quad \omega_{4,5} = \pm\sqrt{f^2 + 4\frac{gH}{h^2}\left(\frac{2 - \cos kh - \cos lh}{a}\right)}.$$

For infinitesimal mesh spacing the frequencies $\omega_{4,5}$ coincide with the continuous solution obtained from (3.2) in the limit as mesh spacing $h \to 0$, and we have $\omega_{4,5} = \omega_{AN} + O(h^2)$. Again, the roots $\omega_{2,3}$ represent spurious inertial oscillations.

3.8. The $P_1 - P_1$ pair. The selected discrete equations lead to

$$\begin{pmatrix} aA & -g\,B^* \\ H\,B & ia\omega \end{pmatrix} \begin{pmatrix} \tilde{\mathbf{u}}_S \\ \tilde{\eta}_S \end{pmatrix} = 0, \tag{3.10}$$

where $B = \frac{2i}{3h} \begin{pmatrix} b_1 & b_2 \end{pmatrix}$, with

$$b_1 = 2\sin kh + \sin lh + \sin(k-l)h, \quad b_2 = \sin kh + 2\sin lh - \sin(k-l)h.$$

For a nontrivial solution to exist, the 3×3 determinant of the coefficient matrix above must vanish. This condition implies

$$\omega_1 = 0, \quad \omega_{2,3} = \pm\sqrt{f^2 + \frac{4}{9}\frac{gH}{h^2}\frac{(b_1^2+b_2^2)}{a^2}}.$$

For infinitesimal mesh spacing we obtain $\omega_{2,3} = \omega_{AN} + O(h^4)$. In the case $f = 0$, corresponding to the propagation of pure gravity waves, we may have $\omega_{2,3} = 0$ if $b_1 = b_2 = 0$. This occurs, in particular, when $kh = lh = \pi$ (wavelength $2h$) or when $kh = -lh = \frac{2\pi}{3}$ (wavelength $3h$). The waves of length $2h$ and $3h$ hence lead to $\omega_{2,3} = 0$ and do not propagate, but they are trapped within the model grid. They are numerical artifacts introduced by the spatial discretization scheme and describe oscillations of wavelength $2h$ and $3h$. Such waves are identified as spurious surface-elevation modes corresponding to physical eigenmodes which have their phase speed reduced to zero by the numerical scheme and appear as stationary oscillations [45, 21]. They can cause an accumulation of energy in the smallest-resolvable scale, leading to noisy solutions.

3.9. The MINI pair. If a cubic bubble is employed, the discrete equations lead to the following system:

$$\begin{pmatrix} aA & \overline{a}_1 A & a_1 A & \\ a_1 A & a_2 A & 0 & -g\,B^* \\ \overline{a}_1 A & 0 & a_2 A & \\ & HB & & ia\omega \end{pmatrix} \begin{pmatrix} \tilde{\mathbf{u}}_S \\ \tilde{\mathbf{u}}_{C_1} \\ \tilde{\mathbf{u}}_{C_2} \\ \tilde{\eta}_S \end{pmatrix} = 0, \tag{3.11}$$

where $B = \frac{1}{h} \begin{pmatrix} b_1 & b_2 & b_3 & b_4 & -\overline{b}_3 & -\overline{b}_4 \end{pmatrix}$, with

$$a_1 = \tfrac{3}{20} e^{i(k+l)\frac{h}{6}} \left(2\cos\tfrac{(k-l)h}{2} + e^{\frac{-i}{2}(k+l)h}\right), \quad a_2 = \tfrac{81}{280},$$
$$b_1 = \tfrac{2i}{3}(2\sin kh + \sin lh + \sin(k-l)h), \qquad b_2 = \tfrac{2i}{3}(2\sin lh + \sin kh + \sin(l-k)h),$$
$$b_3 = \tfrac{9}{20} e^{i(k+l)\frac{h}{3}}(1 - e^{-ikh}), \qquad b_4 = \tfrac{9}{20} e^{i(k+l)\frac{h}{3}}(1 - e^{-ilh}).$$

For the 7×7 determinant to vanish we obtain

$$\omega_1 = 0, \quad \omega_{2,3,4,5} = \pm f \text{ (double root)}, \quad \omega_{6,7} = O(1).$$

For infinitesimal mesh spacing we have $\omega_{6,7} = \omega_{AN} + O(h^2)$, and the roots $\omega_{2,3,4,5}$ represent spurious inertial oscillations.

The MINI discretization offers the particularity that equations associated with bubble nodes on an element depend only on the values of the solution on that element. This permits us to eliminate the bubble unknowns at the element level through a

procedure called static condensation [22]. The linear system (3.11) reduces to a 3×3 one after static condensation, and for the corresponding 3×3 determinant to vanish we exactly obtain the solutions ω_1 and $\omega_{6,7}$. Because the factor $(\omega^2 - f^2)^{-2}$ arises during the procedure, the solutions $\omega_{2,3,4,5}$ also need to be considered. Finally, we have also performed the calculations and numerical experiments with a linear bubble, instead of a cubic one, and insignificant differences have been observed.

3.10. The $P_1^{NC} - P_1$ pair. The selected discrete equations are written in the matrix form

$$\begin{pmatrix} A & 0 & 0 & \\ 0 & A & 0 & -g\,B^* \\ 0 & 0 & A & \\ & HB & & \frac{3}{2}ia\omega \end{pmatrix} \begin{pmatrix} \tilde{\mathbf{u}}_H \\ \tilde{\mathbf{u}}_V \\ \tilde{\mathbf{u}}_D \\ \tilde{\eta}_S \end{pmatrix} = 0, \tag{3.12}$$

where $B = \frac{1}{h} \begin{pmatrix} b_1 & b_2 & b_3 & b_4 & b_5 & b_6 \end{pmatrix}$, with

$$b_1 = 2i \sin \tfrac{kh}{2}, \quad b_2 = 2i \sin \tfrac{lh}{2} \cos \tfrac{(k-l)h}{2}, \quad b_3 = 2i \sin \tfrac{kh}{2} \cos \tfrac{(k-l)h}{2},$$
$$b_4 = 2i \sin \tfrac{lh}{2}, \quad b_5 = 2i \sin \tfrac{kh}{2} \cos \tfrac{lh}{2}, \qquad b_6 = 2i \sin \tfrac{lh}{2} \cos \tfrac{kh}{2}.$$

Vanishing the above 7×7 determinant leads to

$$\omega_1 = 0, \quad \omega_{2,3,4,5} = \pm f \text{ (double root)}, \quad \omega_{6,7} = \pm \sqrt{f^2 + 4 \frac{gH}{h^2} \alpha},$$

with

$$\alpha = \sin^2 \frac{kh}{2} + \sin^2 \frac{lh}{2} + \frac{2}{3a} \left[\sin^4 \frac{kh}{2} + \sin^4 \frac{lh}{2} \right].$$

The above frequencies coincide with those found in [35], where temporally discretized SW equations are considered, in the limit as time step $\Delta t \to 0$. For infinitesimal mesh spacing we obtain $\omega_{6,7} = \omega_{AN} + O(h^4)$.

3.11. The P_1 iso $P_2 - P_1$ pair. The selected discrete equations lead to

$$\begin{pmatrix} 3A & a_1 A & a_2 A & a_3 A & \\ a_1 A & 3A & a_3 A & a_2 A & \\ a_2 A & a_3 A & 3A & a_1 A & -gB^* \\ a_3 A & a_2 A & a_1 A & 3A & \\ & HB & & & 12ia\omega \end{pmatrix} \begin{pmatrix} \tilde{\mathbf{u}}_S \\ \tilde{\mathbf{u}}_H \\ \tilde{\mathbf{u}}_V \\ \tilde{\mathbf{u}}_D \\ \tilde{\eta}_S \end{pmatrix} = 0, \tag{3.13}$$

where $B = \frac{1}{h} \begin{pmatrix} b_1 & b_2 & b_3 & b_4 & b_5 & b_6 & b_7 & b_8 \end{pmatrix}$, with

$$a_1 = \cos \tfrac{kh}{2}, \quad a_2 = \cos \tfrac{lh}{2}, \quad a_3 = \cos \tfrac{(k-l)h}{2}, \quad b_4 = 6i(\sin \tfrac{kh}{2} - \sin \tfrac{(k-2l)h}{2}),$$
$$b_1 = 2i(2 \sin kh + \sin lh + \sin(k-l)h), \qquad b_5 = 6i(\sin \tfrac{lh}{2} + \sin \tfrac{(2k-l)h}{2}),$$
$$b_2 = 2i(2 \sin lh + \sin kh - \sin(k-l)h), \qquad b_7 = 6i(\sin \tfrac{(k-l)h}{2} + \sin \tfrac{(k+l)h}{2}),$$
$$b_3 = 12i \sin \tfrac{kh}{2}, \quad b_6 = 12i \sin \tfrac{lh}{2}, \qquad b_8 = 6i(\sin \tfrac{(k+l)h}{2} - \sin \tfrac{(k-l)h}{2}).$$

For the 9×9 determinant to vanish we obtain

$$\omega_1 = 0, \quad \omega_{2,3,4,5,6,7} = \pm f \text{ (triple root)}, \quad \omega_{8,9} = O(1).$$

For infinitesimal mesh spacing we have $\omega_{8,9} = \omega_{AN} + O(h^2)$.

3.12. The $P_2 - P_1$ pair. The discrete equations lead to the following system:

$$\begin{pmatrix} a_1A & a_4A & a_5A & a_3A & & \\ a_4A & a_2A & a_8A & a_7A & & -gB^* \\ a_5A & a_8A & a_2A & a_6A & & \\ a_3A & a_7A & a_6A & a_2A & & \\ & & HB & & \frac{1}{2}ia\omega & \end{pmatrix} \begin{pmatrix} \tilde{\mathbf{u}}_S \\ \tilde{\mathbf{u}}_H \\ \tilde{\mathbf{u}}_V \\ \tilde{\mathbf{u}}_D \\ \tilde{\eta}_S \end{pmatrix} = 0, \tag{3.14}$$

where $B = \frac{1}{h}\begin{pmatrix} 0 & 0 & b_1 & b_2 & b_3 & b_4 & b_5 & b_6 \end{pmatrix}$, with $a_1 = \frac{1}{30}(4-a)$, $a_2 = \frac{8}{45}$, and

$$\begin{aligned} a_3 &= \tfrac{-1}{45}\cos\tfrac{(k+l)h}{2}, & a_4 &= \tfrac{-1}{45}\cos\tfrac{(k-2l)h}{2}, & a_5 &= \tfrac{-1}{45}\cos\tfrac{(2k-l)h}{2}, \\ a_6 &= \tfrac{4}{45}\cos\tfrac{kh}{2}, & a_7 &= \tfrac{4}{45}\cos\tfrac{lh}{2}, & a_8 &= \tfrac{4}{45}\cos\tfrac{(k-l)h}{2}, \\ b_1 &= \tfrac{2i}{3}\sin\tfrac{kh}{2}, & b_2 &= \tfrac{2i}{3}\sin\tfrac{lh}{2}\cos\tfrac{(k-l)h}{2}, & b_3 &= \tfrac{2i}{3}\sin\tfrac{kh}{2}\cos\tfrac{(k-l)h}{2}, \\ b_4 &= \tfrac{2i}{3}\sin\tfrac{lh}{2}, & b_5 &= \tfrac{2i}{3}\sin\tfrac{kh}{2}\cos\tfrac{lh}{2}, & b_6 &= \tfrac{2i}{3}\sin\tfrac{lh}{2}\cos\tfrac{kh}{2}. \end{aligned}$$

Vanishing the above 9×9 determinant leads to

$$\omega_1 = 0, \quad \omega_{2,3,4,5,6,7} = \pm f \text{ (triple root)}, \quad \omega_{8,9} = O(1).$$

For infinitesimal mesh spacing we obtain $\omega_{8,9} = \omega_{AN} + O(h^2)$.

3.13. Summary of discrete frequencies. The previous results are summarized in Table 3.1, where n is the degree of the dispersion relation, and hence the total number of discrete frequencies, for the nine FE schemes examined in section 3.

TABLE 3.1
Number of frequencies of type $\omega = 0$, $O(1)$, $O(\frac{1}{h})$, $O(h)$, $\pm f$, solutions of the dispersion relation of degree n for the nine FE schemes examined in section 3.

FE pair	n	$\omega = 0$	$O(1)$	$O(1) \to \omega_{AN}$ when $h \to 0$	$O(\frac{1}{h})$	$O(h)$	f	$-f$
$P_1^{NC} - P_0$	8	2	2	no	2	0	1	1
$P_2 - P_0$	10	0	2	no	2	2	2	2
RT0	5	1	2	yes	2	0	0	0
$P_0 - P_1$	5	1	2	yes	0	0	1	1
$P_1 - P_1$	3	1	2	yes	0	0	0	0
MINI	7	1	2	yes	0	0	2	2
$P_1^{NC} - P_1$	7	1	2	yes	0	0	2	2
P_1 iso $P_2 - P_1$	9	1	2	yes	0	0	3	3
$P_2 - P_1$	9	1	2	yes	0	0	3	3

We mention whether frequencies of type $O(1)$, corresponding to inertia-gravity modes, coincide with the analytical solution ω_{AN} in the limit as mesh parameter $h \to 0$. This is the case for all FE pairs, except for the $P_1^{NC} - P_0$ and $P_2 - P_0$ schemes, i.e., when n is even. We also notice the presence of spurious frequencies of type $O(\frac{1}{h})$ for the $P_1^{NC} - P_0$, $P_2 - P_0$, and RT0 pairs, i.e., when a piecewise-constant representation of the surface elevation is employed. Such solutions have already been observed in [34], where the one-dimensional SW equations are discretized using the discontinuous Galerkin method. The slow mode corresponding to $\omega = 0$ is not present

for the $P_2 - P_0$ pair; however, a mode of type $O(h)$ appears instead, while $\omega = 0$ is a double root for the $P_1^{NC} - P_0$ pair. Finally, the solutions $\pm f$ are propagating spurious inertial oscillations that have no particular spatial characteristics. They are a consequence of using more velocity nodes than surface-elevation nodes [35] and are present for all schemes except for the $P_1 - P_1$ and RT0 pairs.

4. Analysis of the dispersion relations. We now analyze the computed frequencies obtained from the dispersion relations for the nine FE pairs examined in section 3. We first consider the gravity wave limit, where $f = 0$, and will then comment on the cases $f \neq 0$ and other modes.

4.1. Gravity wave limit. We let $f = 0$, and examine the computed frequencies, denoted by ω_{CP}, corresponding to pure gravity modes of type $O(1)$, as functions of kh and lh. From (3.3) and the results of section 3, we determine the analytical and computed phase speeds, denoted by c_{AN} and c_{CP}, respectively,

$$c_{AN} \equiv \frac{\omega_{AN}}{\sqrt{k^2 + l^2}} = \pm\sqrt{gH}, \quad c_{CP} \equiv \frac{\omega_{CP}}{\sqrt{k^2 + l^2}}.$$

We then obtain the phase speed ratio of the computed phase speed to the analytical one, denoted by r_{PH}, with

$$r_{PH} \equiv \frac{|c_{CP}|}{|c_{AN}|} = \frac{|\omega_{CP}|}{|\omega_{AN}|} = \frac{|\omega_{CP}|}{\sqrt{gH(k^2 + l^2)}}. \tag{4.1}$$

Note that we should have $r_{PH} = 1$ in the absence of numerical dispersion. We show r_{PH} as a surface function depending on kh and lh, and along the selected axes OX, OY, OD1, and OD2 (vertical section), in Figures 4.2 and 4.3, respectively, for the FE schemes previously examined. As in [21], the values of kh and lh vary over their complete domain $[-\pi, \pi]$. However, the phase speed ratio along selected axes is plotted only on $[0, \pi]$, and the solution is then deduced on $[-\pi, 0]$ by symmetry through the origin. The directions OX, OY, OD1, and OD2 are shown in Figure 4.1.

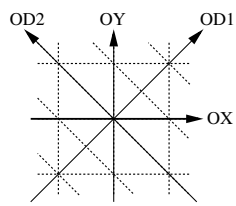


FIG. 4.1. *Definition of the directions OX, OY, OD*1*, and OD*2 *on a uniform mesh made up of biased right triangles.*

The directions OX and OY correspond to waves travelling in the x- and y-directions, for $lh = 0$ and $kh = 0$, respectively, while the directions OD1 and OD2 correspond to waves travelling along the diagonal axes, for $kh = lh$ and $lh = -kh$, respectively.

For the $P_1^{NC} - P_0$ and $P_2 - P_0$ pairs in Figure 4.2, r_{PH} is unexpectedly greater than 1 for small values of $|kh|$ and $|lh|$. Indeed, depending on the direction the wave numbers tend to zero, the phase speed ratio ranges from 1.2 to 1.7 for the $P_1^{NC} - P_0$ pair and from 1.8 to 2.5 for the $P_2 - P_0$ one. We notice that for both pairs r_{PH} decreases for increasing values of $|kh|$ and $|lh|$, implying that gravity modes propagate

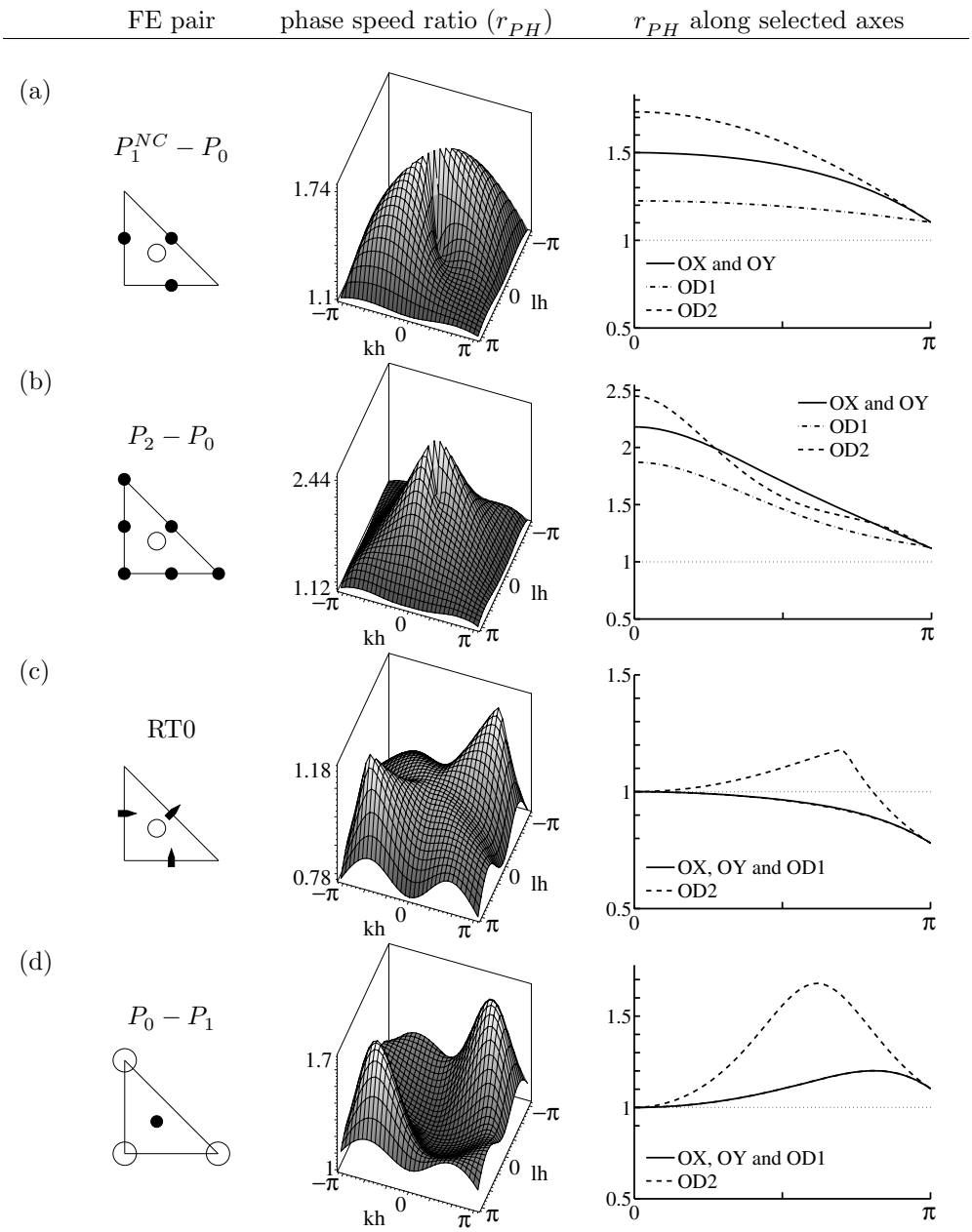


FIG. 4.2. *The phase speed ratio is plotted as a surface function and along selected axes (OX, OY, OD*1*, and OD*2*) for the* (a) $P_1^{NC} - P_0$, (b) $P_2 - P_0$, (c) *RT*0*, and* (d) $P_0 - P_1$ *pairs.*

more rapidly for small wave numbers. Consequently, both pairs poorly represent the propagation of pure gravity waves.

For the $P_1 - P_1$ pair in Figure 4.3, we obtain $r_{PH} \leq 1$ for all values of kh and lh. Hence, the waves are travelling slower than expected, and this is particularly true for large values of $|kh|$ and $|lh|$. We have $r_{PH} = 0$ in the OX, OY, and OD1 directions at the following values of (kh, lh), $(0, \pm\pi)$, $(\pm\pi, 0)$, $(\pm\pi, \pm\pi)$ (waves of length $2h$),

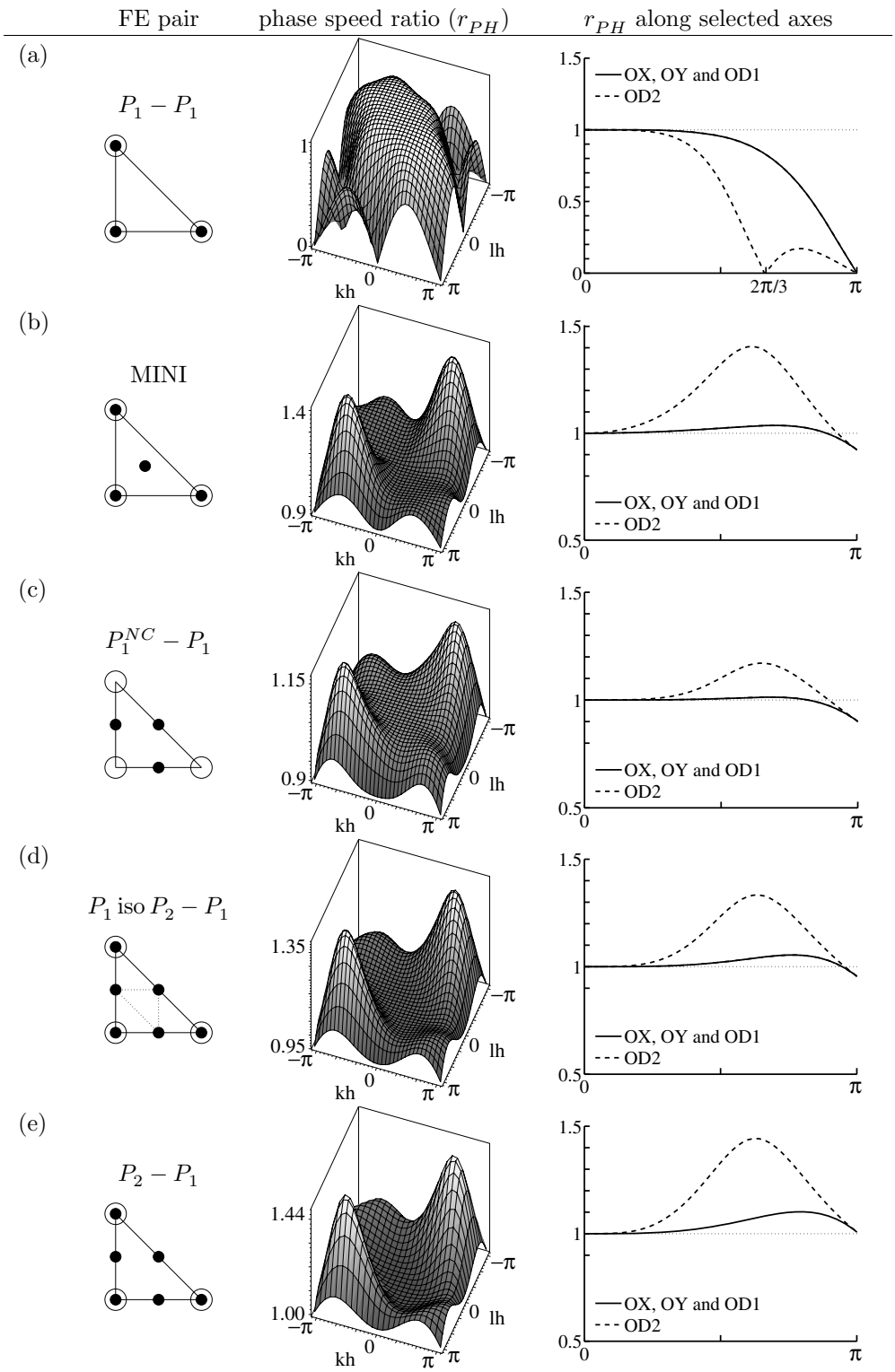


FIG. 4.3. *As for Figure* 4.2 *but for the* (a) $P_1 - P_1$, (b) *MINI,* (c) $P_1^{NC} - P_1$, (d) P_1 *iso* $P_2 - P_1$, *and* (e) $P_2 - P_1$ *pairs.*

and also in the OD2 direction for $|kh| = |lh| = \frac{2\pi}{3}$ (waves of length $3h$). This is a consequence of having zero frequency for waves of length $2h$ and $3h$, as shown previously in section 3.8. Such waves do not propagate and describe oscillations of wavelengths $2h$ and $3h$. They are spurious surface-elevation modes introduced by the spatial discretization scheme. Hence, the $P_1 - P_1$ pair is usually not used to solve the SW system, unless a wave equation formulation or a stabilization procedure is employed [18, 36].

For the remaining FE pairs in Figures 4.2 and 4.3, we have r_{PH} close to 1 for small values of $|kh|$ and $|lh|$ in all directions. However, for increasing values of $|kh|$ and $|lh|$, corresponding to wavelengths ranging from $2h$ to $4h$, r_{PH} departs moderately from 1 in the OX, OY, and OD1 directions for the RT0 element (waves decelerate) and the $P_0 - P_1$ pair (waves accelerate). In the OD2 direction, except for wavelengths ranging from $2h$ to $3h$, r_{PH} strongly departs from 1 for the $P_0 - P_1$, MINI, P_1 iso $P_2 - P_1$, and $P_2 - P_1$ pairs, and moderately for the RT0 and $P_1^{NC} - P_1$ pairs, for increasing values of $|kh|$ and $|lh|$, and the waves will accelerate.

From (3.3) and the results of section 3, we now determine the analytical and computed group velocities, $\mathbf{c}_{g,AN}$ and $\mathbf{c}_{g,CP}$, respectively, i.e., the velocity at which the energy is carried by the waves, with

$$\mathbf{c}_{g,AN} = \left(\frac{\partial \omega_{AN}}{\partial k}, \frac{\partial \omega_{AN}}{\partial l} \right), \quad \mathbf{c}_{g,CP} = \left(\frac{\partial \omega_{CP}}{\partial k}, \frac{\partial \omega_{CP}}{\partial l} \right).$$

We also calculate the directional derivatives $\mathbf{c}_{g,AN} \cdot \mathbf{d}$ and $\mathbf{c}_{g,CP} \cdot \mathbf{d}$, where $\mathbf{d}$ is a unit vector in the OX, OY, OD1, or OD2 directions. For the continuous case and the nine FE pairs of section 3, the group velocities (left panels) and the directional derivatives normalized by the factor $\sqrt{gH}$ (right panels) are shown in Figures 4.4, 4.5, and 4.6.

For the $P_1^{NC} - P_0$ and $P_2 - P_0$ pairs in Figure 4.4, the normalized directional derivatives $\mathbf{c}_{g,CP} \cdot \mathbf{d}$ unexpectedly depart from 1 in all directions for small values of $|kh|$ and $|lh|$. Hence, $\mathbf{c}_{g,CP}$ is not consistent with the analytic solution for both pairs as kh and lh tend to zero, and large inaccuracies can be expected in the speed of energy propagation. For the $P_1 - P_1$ pair in Figure 4.4, the group velocity is zero for waves of length $3h$ in the OX, OY, and OD1 directions, and for wavelengths ranging from $2h$ to $3h$ the group velocity has the wrong sign. In the OD2 direction, because the group velocity is zero three times for waves longer than $2h$, it has alternatively right and wrong signs. This is a consequence of having zero frequency for waves of length $2h$ and $3h$, as shown in section 3.8.

For the remaining pairs of Figures 4.5 and 4.6, the group velocity is consistent with the analytic solution as kh and lh are small. However, at the following values of (kh, lh), $(0, \pm\pi)$, $(\pm\pi, 0)$, $(\pm\pi, \pm\pi)$ (waves of length $2h$), we have $\mathbf{c}_{g,CP} = \mathbf{0}$ contrarily to the analytic case, and hence although waves of length $2h$ are propagating, the associated energy is not. For the RT0, $P_0 - P_1$, and MINI pairs in Figure 4.5 and for the $P_2 - P_1$ pair in Figure 4.6, we observe that $\mathbf{c}_{g,CP} \cdot \mathbf{d}$ has a negative (wrong) sign in the OD2 direction for wavelengths greater than $2h$ and smaller than $3h$. Finally, for all pairs in the OD2 direction, the magnitude of the group velocity reaches a peak ranging from 1.5 (for the $P_1^{NC} - P_1$ and RT0 pairs) to 2.5 (for the $P_0 - P_1$ pair) times the analytical value for wavelengths close to $4h$, leading to inaccuracies if such waves are generated and sustained in a numerical model.

4.2. Inertia-gravity waves and other modes. We now assume $f \neq 0$ and note $\lambda \equiv \sqrt{gH}/f$, the Rossby radius of deformation. Within a numerical factor, it is the distance covered by a wave travelling at the speed $\sqrt{gH}$ during one inertial period

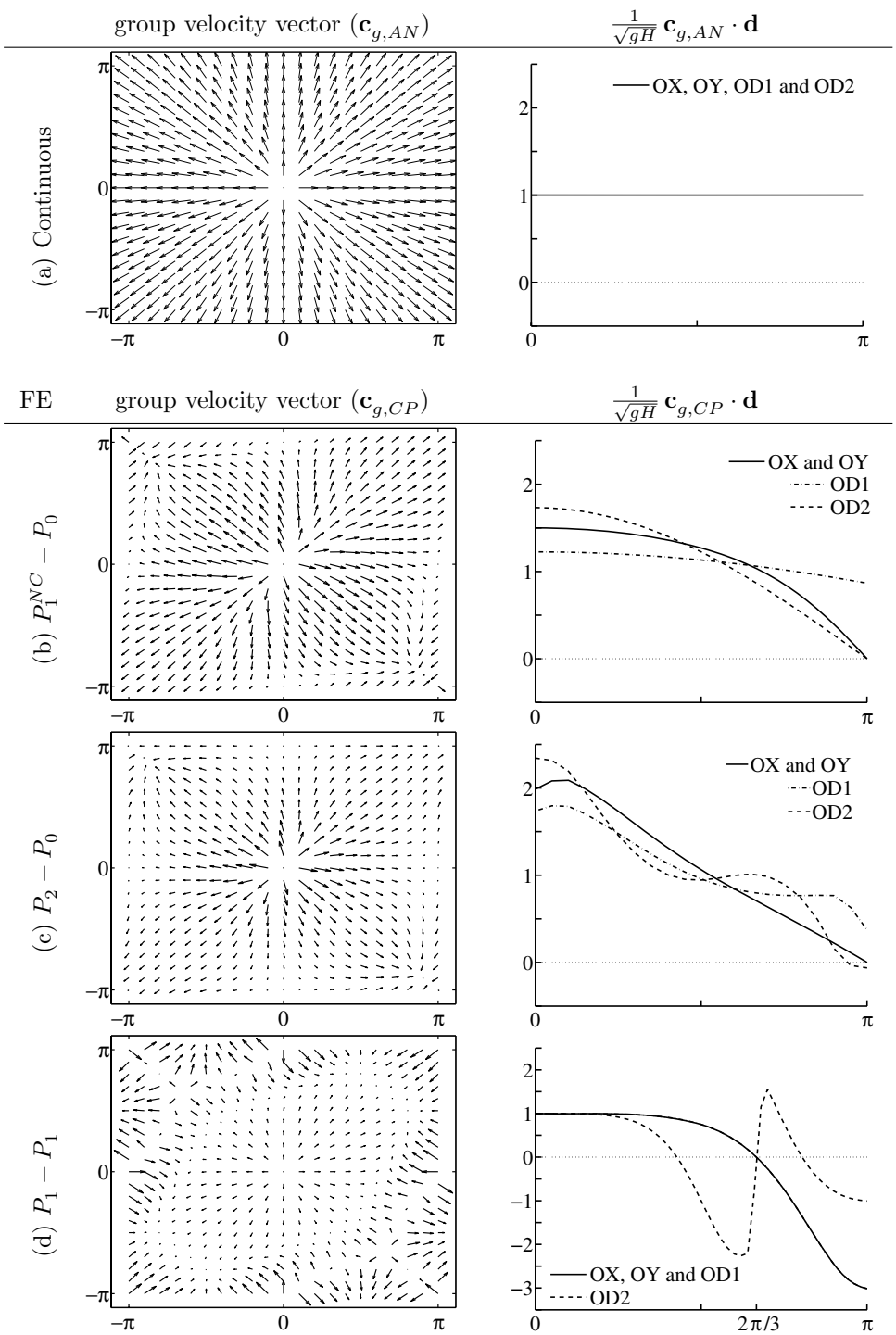


FIG. 4.4. *The group velocity vector (left panels) where the Cartesian coordinate axes are kh and lh, and the normalized directional derivative* $\frac{1}{\sqrt{gH}}\,\mathbf{c}_g\cdot\mathbf{d}$ *(right panels), where* $\mathbf{d}$ *is a unit vector in the OX, OY, OD*1*, or OD*2 *direction, for* (a) *the continuous case and the* (b) $P_1^{NC}-P_0$, (c) P_2-P_0, *and* (d) P_1-P_1 *pairs.*

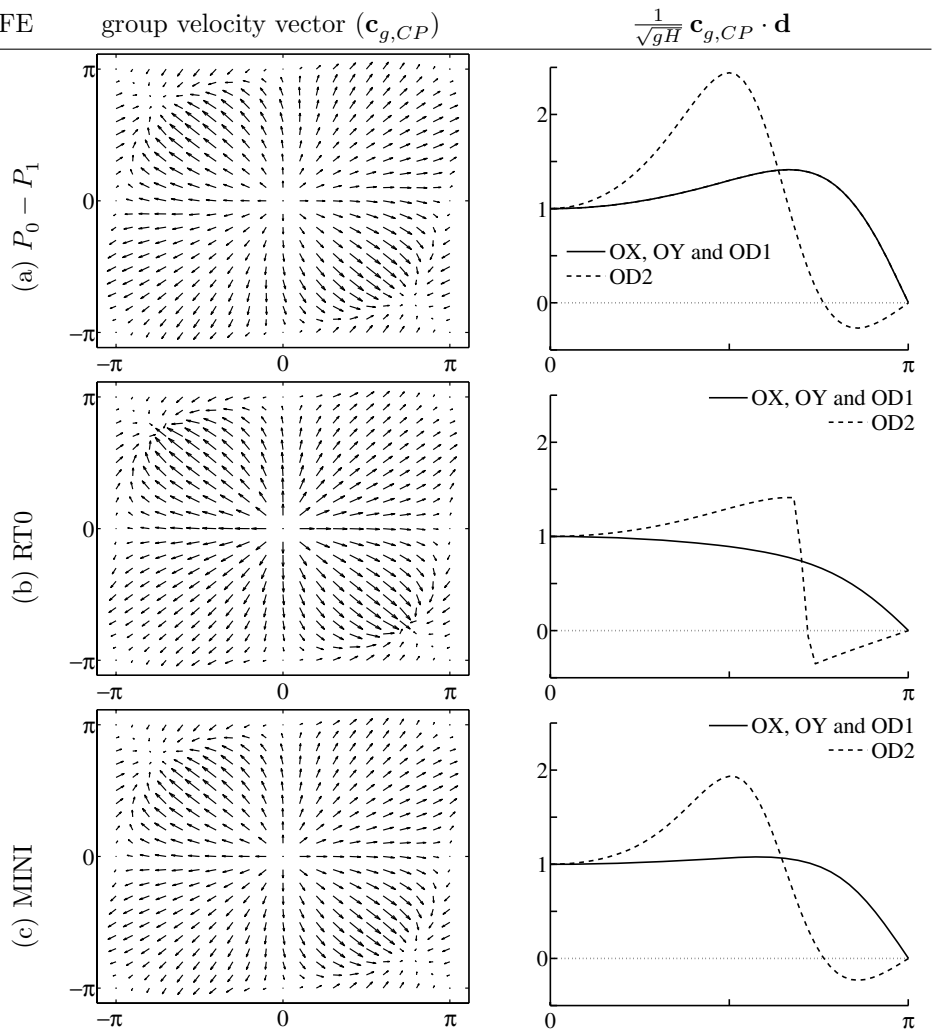


FIG. 4.5. *As for Figure* 4.4 *but for the* (a) $P_0 - P_1$, (b) *RT*0, *and* (c) *MINI pairs.*

$(2\pi/f)$. Using (3.3) and the results of section 3 we now obtain the phase speed ratio

$$r^f_{PH} \equiv \frac{|c_{CP}|}{|c_{AN}|} = \frac{|\omega_{CP}|}{|\omega_{AN}|} = \frac{f\sqrt{1+\left(\frac{\lambda}{h}\right)^2 t(kh,lh)}}{f\sqrt{1+\left(\frac{\lambda}{h}\right)^2 (k^2h^2+l^2h^2)}}, \tag{4.2}$$

where ω_{CP} corresponds to an inertia-gravity mode of type $O(1)$, as in (4.1) but now with $f \neq 0$, and $t(kh, lh)$ is a trigonometric expression depending on kh and lh.

For the $P_1 - P_1$, $P_0 - P_1$, $P_1^{NC} - P_1$, P_1 iso $P_2 - P_1$, MINI, and $P_2 - P_1$ pairs, a realistic positive value for f would have little effect on the plots of Figures 4.2 to 4.6 when $\lambda/h \gg 1$ (high resolution), as $t(kh, lh)$ is bounded in (4.2) for $0 \leq kh \leq \pi$ and $0 \leq lh \leq \pi$. Hence, we have $r^f_{PH} \simeq r_{PH}$ for $\lambda/h \gg 1$. In the case $\lambda/h \ll 1$ (coarse resolution), we deduce from (4.2) that $r^f_{PH} \simeq 1$. However, the results obtained for r_{PH} in Figures 4.2 to 4.6 are roughly preserved but at smaller scales.

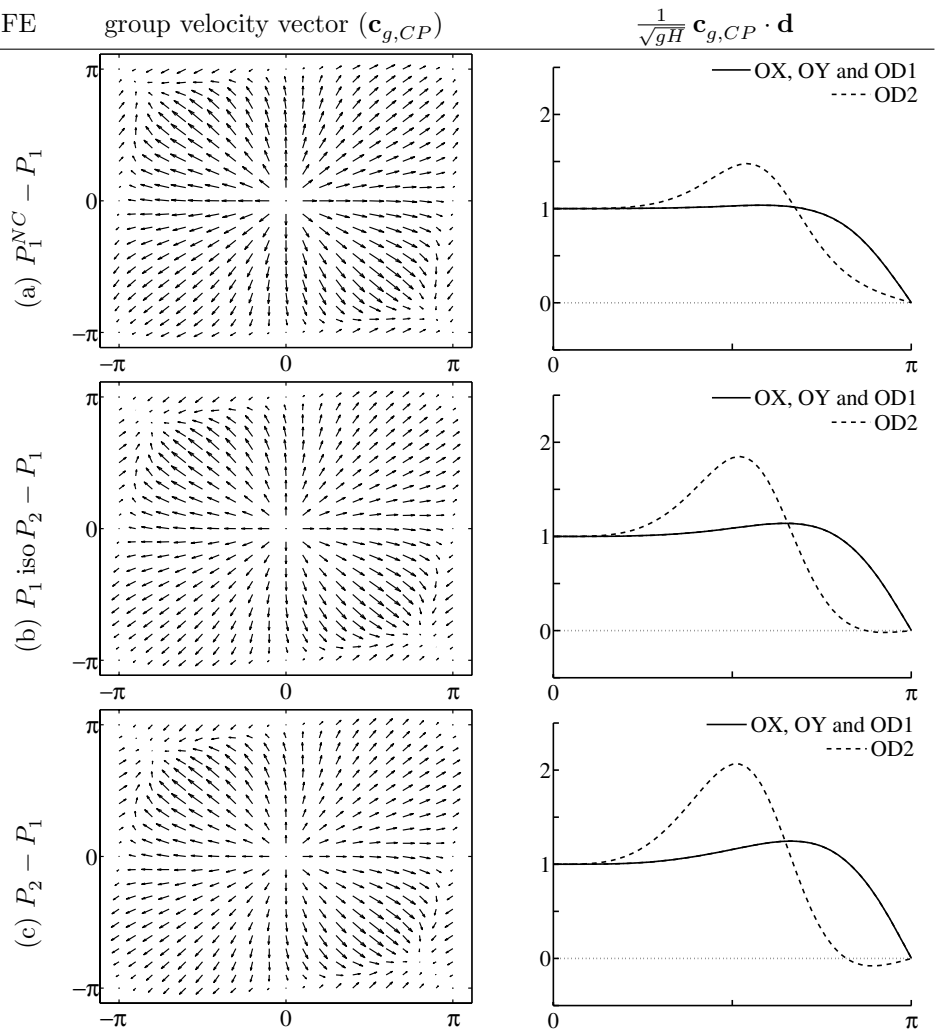


FIG. 4.6. *As for Figure* 4.4 *but for the* (a) $P_1^{NC}-P_1$, (b) P_1 *iso* P_2-P_1, *and* (c) P_2-P_1 *pairs.*

The remaining $P_1^{NC}-P_0$, P_2-P_0, and RT0 pairs, as mentioned in Table 3.1, lead to discrete frequencies of type $O(\frac{1}{h})$, and additionally in $O(h)$ for the P_2-P_0 pair. The nondimensional frequency ω_{CP}/f, corresponding to modes in $O(1)$, $O(\frac{1}{h})$, and $O(h)$ along selected axes (OX, OY, OD1, and OD2), is plotted for these pairs in Figures 4.7 and 4.8 and is compared with the continuous case. We consider λ/h ranging from 1/10 (coarse resolution) to 2 (high resolution).

For the $P_1^{NC}-P_0$ and P_2-P_0 pairs in Figure 4.7, the mode of type $O(\frac{1}{h})$ is a spurious mode appearing as an approximate mirror image of the mode in $O(1)$. For the P_2-P_0 pair, the mode in $O(h)$ is trapped close to zero, and this is especially true when the Rossby radius is resolved (high resolution). Because the P_2-P_0 pair is the only one in Table 3.1 to have a mode in $O(h)$ and no mode corresponding to the geostrophic mode $\omega=0$ in (3.2), we suggest considering the mode in $O(h)$ as a spurious one which coincides with the geostrophic mode as mesh spacing $h\to 0$. Recall that the geostrophic mode would correspond to the slow Rossby mode on a

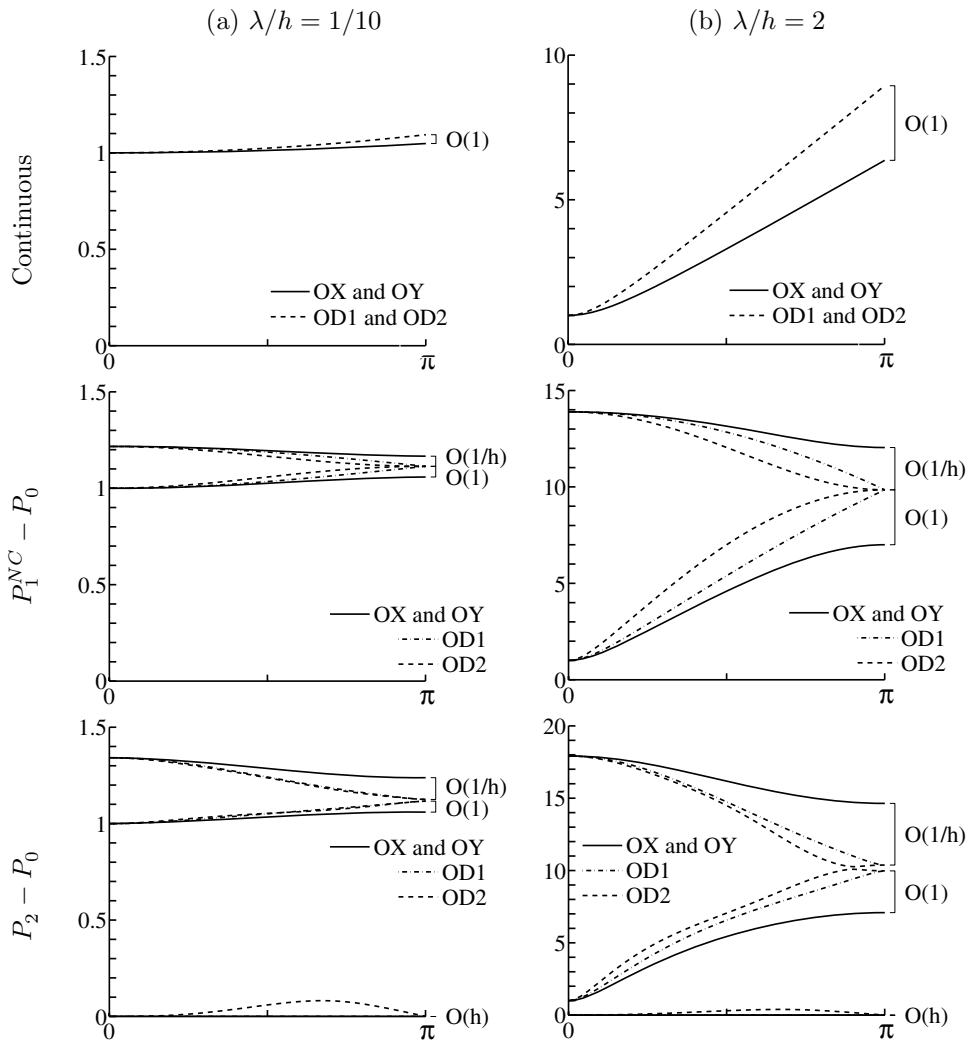


FIG. 4.7. *The nondimensional frequency ω_{CP}/f, corresponding to modes in $O(1)$, $O(\frac{1}{h})$, and $O(h)$ along selected axes (OX, OY, OD1, and OD2), for the continuous case and the $P_1^{NC} - P_0$ and $P_2 - P_0$ pairs. We note $\lambda \equiv \sqrt{gH}/f$, the radius of deformation (Rossby radius), and two cases are considered, namely, $\lambda/h = 1/10$ (coarse resolution) and $\lambda/h = 2$ (high resolution).*

β-plane.

For the RT0 pair in Figure 4.8 the mode in $O(\frac{1}{h})$ behaves quite differently from the corresponding ones for the $P_1^{NC} - P_0$ and $P_2 - P_0$ pairs in Figure 4.7. Indeed, for values of λ/h below 0.2 (obtained numerically), the mode in $O(\frac{1}{h})$ is much closer to the continuous solution in Figure 4.8 than the $O(1)$ one. Moreover, the mode in $O(1)$ is uniformly zero for all kh and lh in the inertial limit ($g = 0$, i.e., $\lambda/h = 0$), but it tends to draw closer to the continuous solution as the values of λ/h progressively increase. However, in order for the mode in $O(1)$ to be monotonic in kh and lh, as for the continuous one, λ/h has to be chosen greater than 0.45 (obtained numerically). The modes in $O(\frac{1}{h})$ and in $O(1)$ have no common intersection except locally, when $\lambda/h = 1/6$ and $kh = lh = 0$, as shown in Figure 4.8(c). Note the mode in $O(\frac{1}{h})$

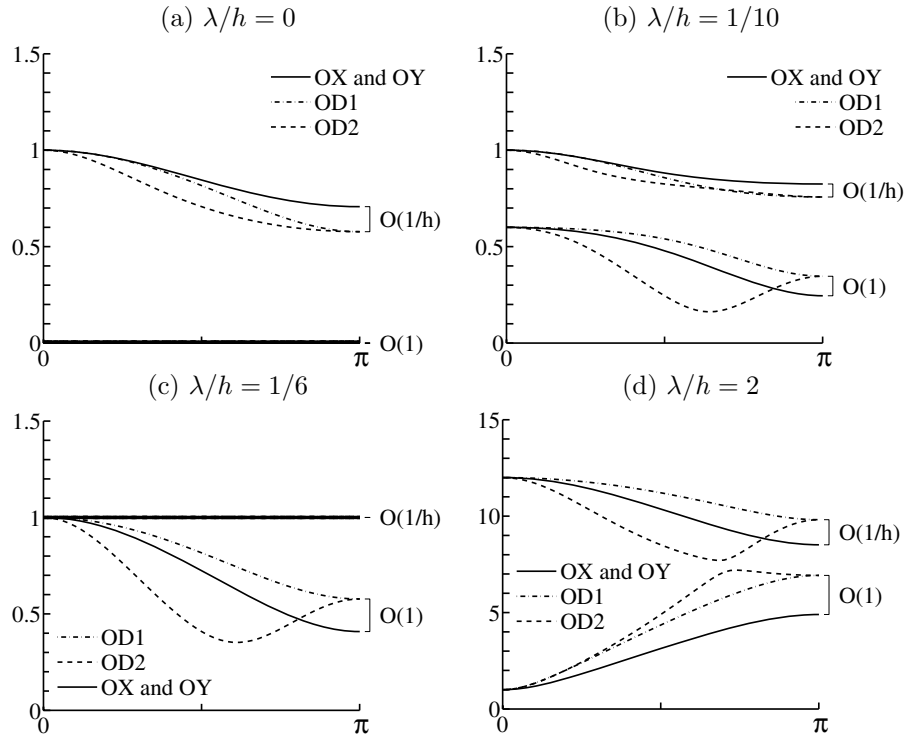


FIG. 4.8. *As for Figure* 4.7 *but for the RT*0 *element in the cases* $\lambda/h = 0$, 1/10, 1/6, *and* 2.

is uniformly one for all kh and lh in the case $\lambda/h = 1/6$, and as λ/h progressively increases (above 1/6), it appears as an approximate mirror image of the mode in $O(1)$, i.e., as a spurious mode. Finally, in the OD2 direction both the $O(\frac{1}{h})$ and $O(1)$ modes exhibit nonmonotonic curves for a wide range of λ/h.

5. Numerical results. The results of two tests using (2.1) and (2.2) are now presented. The first test is performed in [11] for FD grids. It is reproduced here to examine the consequences of the distortion observed in Figure 4.8 in the dispersion relation of the RT0 element for small values of λ/h (coarse resolution).

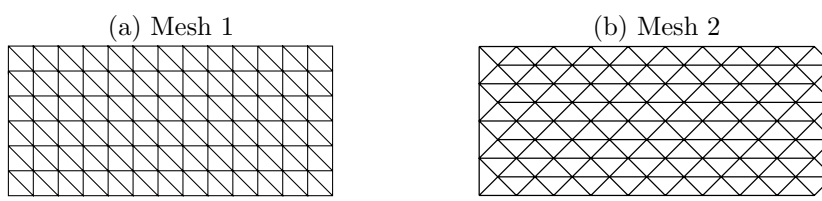


FIG. 5.1. *A window of* (a) *Mesh* 1, *made up of biased right isosceles triangles, and* (b) *Mesh* 2, *obtained from Mesh* 1 *by rotation.*

The main point of interest is to verify whether or not the RT0 discretization properly simulates the geostrophic adjustment process, in which the dispersion of the inertia-gravity waves leads to the establishment of a geostrophic balance.

The square domain extent is 5500 km × 5500 km, and it is discretized using biased right isosceles triangles (as in Figure 5.1(a)) with a uniform node spacing $h = 220$

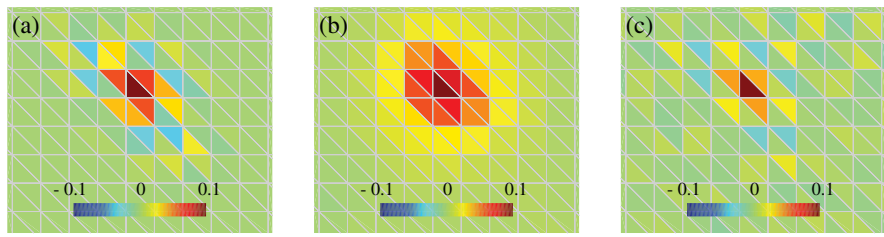


FIG. 5.2. *Window of the surface-elevation field around the point mass source* (0.1 m) *for the RT*0 *pair; after* 1 *day of simulation and* (a) $\lambda/h = 1/6$ *(coarse resolution),* (b) $\lambda/h = 2$ *(high resolution);* (c) *after* 10 *days of simulation and* $\lambda/h = 1/12$ *(coarse resolution).*

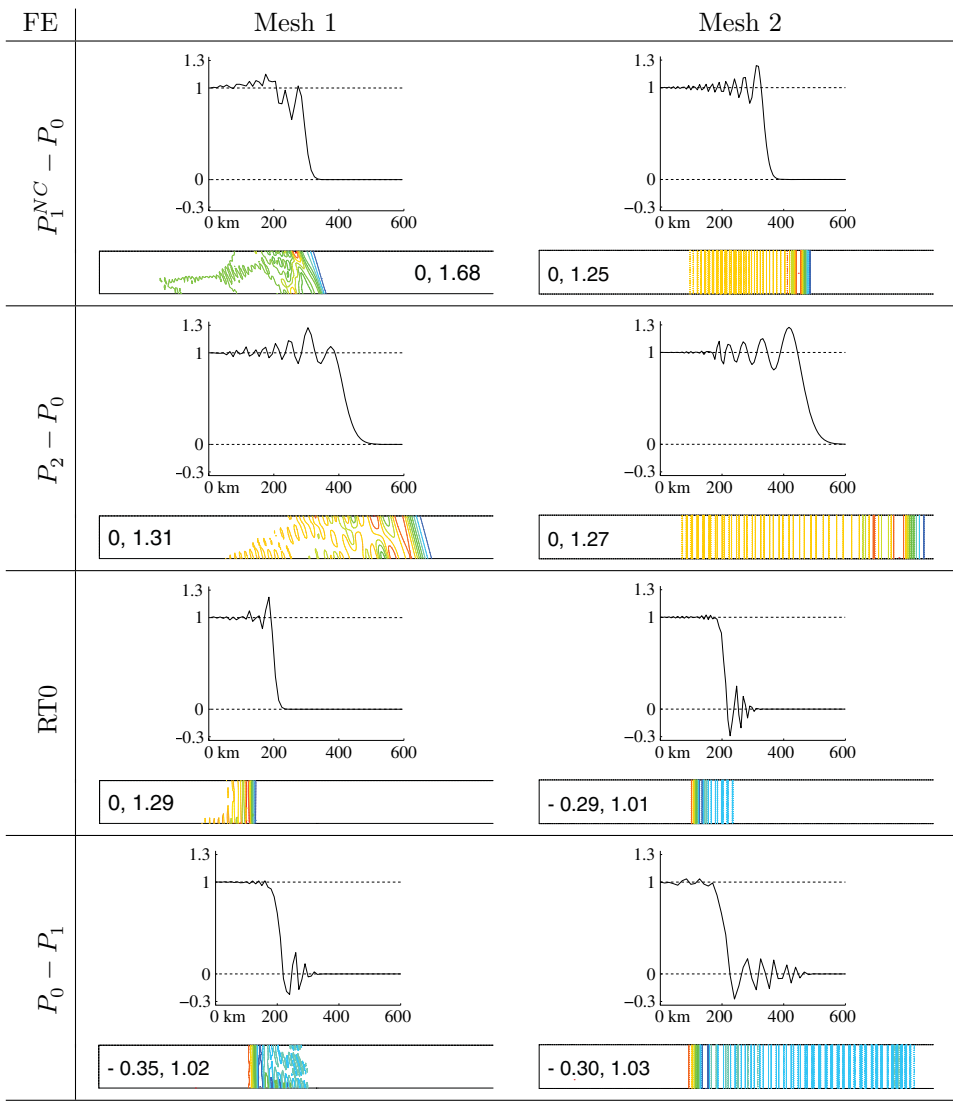


FIG. 5.3. *Vertical cross-sections (top) and isolines (bottom, over* 2550 km*) of the elevation field at times* 2000 s *and* 10000 s *of the propagation, respectively, for the* $P_1^{NC} - P_0$, $P_2 - P_0$, *RT*0, *and* $P_0 - P_1$ *pairs. For the isolines, the minimum and maximum values are specified in each panel.*

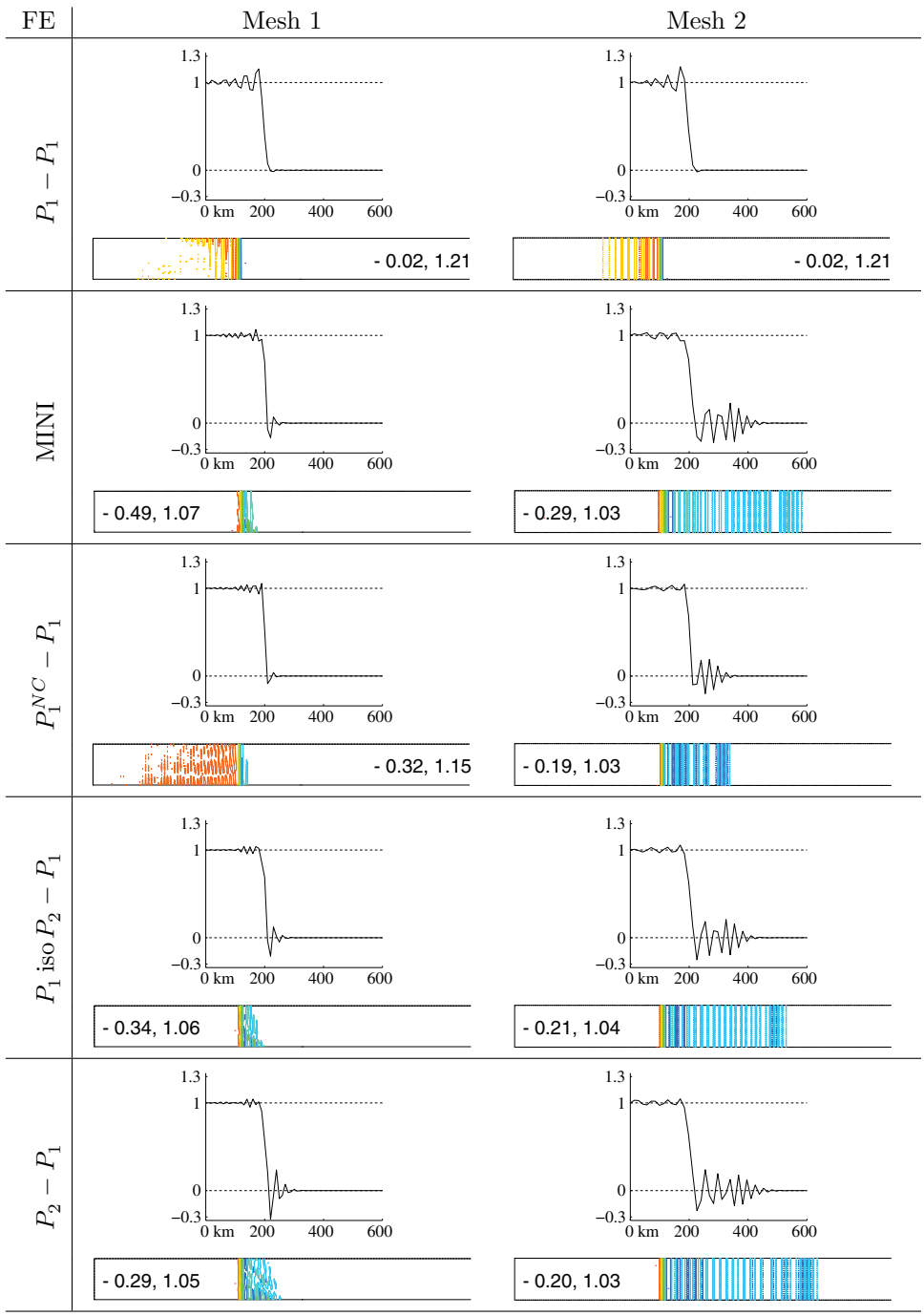


FIG. 5.4. *As for Figure* 5.3 *for the* $P_1 - P_1$, *MINI,* $P_1^{NC} - P_1$, P_1 *iso* $P_2 - P_1$, *and* $P_2 - P_1$ *pairs.*

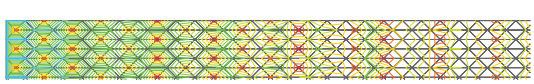


FIG. 5.5. *Isolines of the surface elevation on a window of Mesh* 2 *for the* $P_1 - P_1$ *pair at time* 10000 s *by using a linear bathymetry. As expected, a spurious mode of wavelength* $3h$ *is observed.*

km, leading to a 25×25 mesh. The fluid is initially at rest, and zero normal velocity is specified at the lateral boundaries. A point mass source and a point mass sink of 0.1 m and -0.1 m, respectively, are prescribed at fixed locations $12\,h$ apart at each time step; otherwise the initial surface elevation field is uniform. The Coriolis parameter is held constant and evaluated close to $45°$ N, with $f = 10^{-4}\,\mathrm{s}^{-1}$, and the time step is 450 s. The surface-elevation field is shown after one day of simulation, which is approximately the time required for geostrophic adjustment to occur, in Figures 5.2(a) and (b). In Figure 5.2(a), for $\lambda/h = 1/6$ (coarse resolution) with $\sqrt{gH} = 11/3\,\mathrm{m\,s}^{-1}$, and also for $\lambda/h \leq 1/2$, the RT0 scheme exhibits a checkerboard-like pattern of noise in the elevation field around the mass source and sink points, shown by an irregular alternation of positive and negative height areas (that are constant per triangle). Note that such a pattern is not observed for the other pairs. Due to the symmetry, only the result around the point mass source is shown. We believe the checkerboard-like appearance is probably due to the distortion observed in Figure 4.8 in the dispersion relation for small values of λ/h (coarse resolution). In Figure 5.2(b) for $\lambda/h = 2$ (high resolution) with $\sqrt{gH} = 44\,\mathrm{m\,s}^{-1}$, the surface-elevation distribution exhibits a regular and expected pattern as for the other pairs, except for the $P_1 - P_1$ one (due to the spurious surface-elevation mode). We also conducted the experiment over 10 days, and the result is shown in Figure 5.2(c) for $\lambda/h = 1/12$ (coarse resolution) with $\sqrt{gH} = 11/6\,\mathrm{m\,s}^{-1}$. The checkerboard-like pattern is more apparent than in Figure 5.2(a), particularly when considering the alternation of blue and yellow triangles surrounding the four centered ones (red and orange) in the OX, OY, and OD2 directions. Note that in all cases the gravitational Courant number is small (≤ 0.1) and the use of a forward Euler and a Crank–Nicolson time stepping schemes yields essentially the same results.

In the second test, (2.1) and (2.2) are solved using $f = 0$. The purpose of the experiment is to validate the analytical results obtained in section 4.1 on two meshes, Mesh 1 and Mesh 2, shown in Figure 5.1. Mesh 1 is made up of biased right triangles and Mesh 2 is obtained from Mesh 1 by a rotation of $\pi/4$. Such meshes are used in order to simulate gravity-wave propagation: along the OX and OY directions for Mesh 1 and along the OD2 direction for Mesh 2. The domain extent is 7000 km $\times$ 280 km for Mesh 1 and $5000\sqrt{2}$ km $\times$ $200\sqrt{2}$ km for Mesh 2. In both cases the resolution is $h = 10$ km, along the x- or y-axis for Mesh 1 and along the diagonal for Mesh 2. The fluid is initially at rest and zero normal velocity is specified at the lateral boundaries, except at the western one. The conditions $\eta = 1$ m and $u = 0.1\,\mathrm{m\,s}^{-1}$ (i.e., $u = \sqrt{\frac{g}{H}} \times 1$ m) are prescribed at each time step at the western boundary, with $H = 1000$ m and $g = 10\,\mathrm{m\,s}^{-2}$, i.e., $\sqrt{gH} = 100\,\mathrm{m\,s}^{-1}$. Note that the velocity has been imposed weakly at the western boundary for the $P_0 - P_1$ pair. The time step is set to 10 s, and the gravitational Courant number is thus 0.1.

In Figures 5.3 and 5.4 the vertical cross-sections (on the x, z-plane) and the isolines of the wave propagation are displayed for graphical convenience, after 2000 s and 10000 s, respectively, on the two meshes. For all pairs, except the $P_1^{NC} - P_0$ and $P_2 - P_0$ ones, the wave front propagates at the analytical speed $100\,\mathrm{m\,s}^{-1}$. This is because r_{PH} is close to 1 for small values of $|kh|$ and $|lh|$ in all directions as shown in Figures 4.2 and 4.3. However, the maximum values observed for r_{PH} in the OX and OD2 directions are 1.5 and 1.73, respectively, for the $P_1^{NC} - P_0$ pair, and 2.18 and 2.45, respectively, for the $P_2 - P_0$ one. For the $P_1^{NC} - P_0$ and $P_2 - P_0$ pairs, the wave front speeds are thus expected to be $150\,\mathrm{m\,s}^{-1}$ and $218\,\mathrm{m\,s}^{-1}$, respectively, in the OX direction, and $173\,\mathrm{m\,s}^{-1}$ and $245\,\mathrm{m\,s}^{-1}$, respectively, in the OD2 direction. These analytical values are in good agreement with the computed ones, observed in

Figures 5.3 and 5.4. Indeed, for the $P_1^{NC} - P_0$ and $P_2 - P_0$ pairs, the computed wave front speeds are $141\,\mathrm{m\,s^{-1}}$ and $209\,\mathrm{m\,s^{-1}}$, respectively, on Mesh 1, and $172\,\mathrm{m\,s^{-1}}$ and $242\,\mathrm{m\,s^{-1}}$, respectively, on Mesh 2, with a mesurement error of about $2\,\mathrm{m\,s^{-1}}$. The loss of symmetry observed in the front propagation for the $P_1^{NC} - P_0$ and $P_2 - P_0$ pairs on Mesh 1 is likely due to a faster propagation in the OD2 direction. This explains the observed southeast orientation of the front and hence its partial propagation in the OD1 direction. Consequently, due to the results of Figure 4.2 in the OD1 direction, the computed wave speed is slightly smaller than the analytic one on Mesh 1. On Mesh 2, because the faster (dominant) propagation is along the OD2 direction, the front is not tilted and the analytical and computed values nearly coincide.

There is also a good agreement between the numerical dispersion observed in Figures 4.2 and 4.3 and the simulation results shown in Figures 5.3 and 5.4, except for the $P_1^{NC} - P_0$ and $P_2 - P_0$ pairs due to the incorrect position of the front. Indeed, in the OX, OY, and OD1 directions we have $r_{PH} \simeq 1$ for the $P_1^{NC} - P_1$ pair in Figure 4.3(c), and a small amount of numerical dispersion is observed in Figure 5.4 on Mesh 1. For the MINI, P_1 iso $P_2 - P_1$, $P_2 - P_1$, and $P_0 - P_1$ pairs in the previous directions, r_{PH} progressively increases from 1.03 (MINI) up to 1.2 ($P_0 - P_1$) as observed in Figures 4.2 and 4.3. Hence, we have $|c_{CP}| \geq |c_{AN}|$ and increasing dispersion is noticeable upstream the front in Figures 5.3 and 5.4 for these pairs on Mesh 1. For the RT0 and $P_1 - P_1$ pairs in the previous directions, we have $r_{PH} \leq 1$ in Figures 4.2 and 4.3, and hence $|c_{CP}| \leq |c_{AN}|$, leading to dispersion effects downstream the front as shown in Figures 5.3 and 5.4 on Mesh 1. In the OD2 direction, we still have $r_{PH} \leq 1$ for the $P_1 - P_1$ pair, and oscillations are again present downstream the front on Mesh 2 as shown in Figure 5.4. For the RT0 pair we now have $r_{PH} \geq 1$ in the OD2 direction (except for small wavelengths), and the dispersion effects are noticeable upstream the front in Figure 5.3 on Mesh 2. For the $P_1^{NC} - P_1$, P_1 iso $P_2 - P_1$, MINI, $P_2 - P_1$, and $P_0 - P_1$ pairs in the OD2 direction, r_{PH} progressively increases from 1.15 ($P_1^{NC} - P_1$) up to 1.7 ($P_0 - P_1$) as observed in Figures 4.2 and 4.3. We thus have $|c_{CP}| \geq |c_{AN}|$, and increasing dispersion is noticeable for these pairs upstream the front in Figures 5.3 and 5.4 on Mesh 2.

Spurious elevation modes are not observed in Figure 5.4 for the $P_1 - P_1$ pair. This is because a constant value of H is used. However, when $H = 1000$ m if $x \leq 400$ km, when $H = 800$ m if $x \geq 600$ km, and by assuming H is linear if 400 km $\leq x \leq 600$ km, a spurious surface elevation mode of wavelength $3h$ is triggered, as shown in Figure 5.5 on Mesh 2. This agrees well with the result predicted by theory in Figure 4.3(a) in the OD2 direction.

Finally, the second test has also been performed on unstructured meshes. Two main features are observed for all pairs: the numerical dispersion upstream the front is more significant than on Mesh 1 but less than on Mesh 2, and additional noise is generated downstream the front. Further, the loss of symmetry observed in the front propagation in Figure 5.3 for the $P_1^{NC} - P_0$ and $P_2 - P_0$ is only marginally present.

6. Conclusions. This appears to be the first study of the dispersion relation and spurious mode behavior for FE solutions of the two-dimensional linearized SW equations based on the examination of a variety of mixed FE pairs. For each pair the frequency wave number or dispersion relation is obtained and analyzed, and the dispersion properties are compared analytically and graphically with the continuous case to illustrate the main points of interest. Two numerical tests, concerning the propagation of gravity waves, are performed. In the first experiment, the simulation of the geostrophic adjustment process, the RT0 element exhibits a checkerboard-like pattern of noise in the elevation field when the grid resolution is low relative to the

Rossby radius of deformation. Such a pattern has not been observed for the other pairs. The second test illustrates the analytical dispersion results by simulating the propagation of pure gravity waves in a long channel. In particular, for the $P_1^{NC} - P_0$ and $P_2 - P_0$ pairs, the wave front propagates much faster than the analytical speed. These results can be compared with those in [32] for the geostrophic balance. We also notice the presence of spurious frequencies of type $O(\frac{1}{h})$ for the FE pairs having a piecewise-constant representation of the surface elevation. Such solutions have already been observed in [34], where the one-dimensional SW equations are discretized using the discontinuous Galerkin method, and they will be analyzed in a further study.

The preceding analysis illustrates how phase and group velocity accuracy can help in the selection of a spatial discretization scheme. In particular, the $P_1^{NC} - P_1$ and RT0 pairs have been identified as a promising compromise for the discretization of the inviscid linear SW equations, provided the grid resolution is high relative to the Rossby radius of deformation for the RT0 element. However, because of its restrictive nature, due to the use of constant values for h and H, such an analysis should be only one step of the selection process. The use of nonconstant h and H will be analyzed in subsequent studies. Further, the implementation of a time stepping technique can change the relative accuracy and merits of two spatial discretization schemes.

REFERENCES

[1] A. J. Adcroft, C. N. Hill, and J. C. Marshall, *A new treatment of the Coriolis terms in C-grid models at both high and low resolutions*, Mon. Wea. Rev., 127 (1999), pp. 1928–1936.

[2] V. Agoshkov, E. Ovtchinnikov, V. Pennati, D. Ambrosi, and F. Saleri, *Finite element, finite volume, and finite differences approximation to the shallow water equations*, in Finite Elements in Fluids, K. Morgan, E. Oñate, J. Periaux, J. Peraire, and O. C. Zienkiewicz, eds., Pineridge Press, Barcelona, Spain, 1993, pp. 1001–1009.

[3] V. Aizinger and C. Dawson, *A discontinuous Galerkin method for two-dimensional flow and transport in shallow water*, Adv. in Water Res., 25 (2002), pp. 67–84.

[4] F. Alcrudo and P. Garcia-Navarro, *A high-resolution Godunov-type scheme in finite volumes for the 2D shallow water equations*, Int. J. Numer. Methods Fluids, 16 (1993), pp. 489–505.

[5] K. Anastasiou and C. T. Chan, *Solution of the 2D shallow water equations using the finite volume method on unstructured triangular meshes*, Int. J. Numer. Methods Fluids, 24 (1997), pp. 1225–1245.

[6] A. Arakawa and V. R. Lamb, *Computational design of the basic dynamical processes of the UCLA general circulation model*, Methods Comput. Phys., 17 (1977), pp. 173–265.

[7] D. N. Arnold, F. Brezzi, and M. Fortin, *A stable finite element for the Stokes equations*, Calcolo, 21 (1984), pp. 337–344.

[8] D. N. Arnold, F. Brezzi, B. Cockburn, and L. D. Marini, *Unified analysis of discontinuous Galerkin methods for elliptic problems*, SIAM J. Numer. Anal., 39 (2002), pp. 1749–1779.

[9] J. H. Atkinson, J. J. Westerink, and J. M. Hervouet, *Similarities between the wave equation and the quasi-bubble solutions to the shallow water equations*, Int. J. Numer. Methods Fluids, 45 (2004), pp. 689–714.

[10] J. H. Atkinson, J. J. Westerink, and R. A. Luettich, Jr., *Two dimensional dispersion analyses of finite element approximations to the shallow water equations*, Int. J. Numer. Methods Fluids, 45 (2004), pp. 715–749.

[11] M. L. Batteen and Y. J. Han, *On the computational noise of finite-difference schemes used in ocean models*, Tellus, 33 (1981), pp. 387–396.

[12] M. Bercovier and O. Pironneau, *Error estimates for the finite element method solution of the Stokes problem in the primitive variables*, Numer. Math., 33 (1979), pp. 211–224.

[13] F. Brezzi and M. Fortin, *Mixed and Hybrid Finite Element Methods*, Springer Ser. Comput. Math. 15, Springer-Verlag, Berlin, 1991.

[14] G. F. Carey, ed., *Finite Element Modeling of Environmental Problems*, John Wiley and Sons, Chichester, UK, 1995.

[15] S. Chippada, C. N. Dawson, M. L. Martinez, and M. F. Wheeler, *A Godunov-type finite volume method for the system of shallow water equations*, Comput. Methods Appl. Mech.

Engrg., 151 (1998), pp. 105–129.
[16] B. Choi, M. Iskandarani, J. C. Levin, and D. B. Haidvogel, *A spectral finite-volume method for the shallow-water equations*, Mon. Wea. Rev., 132 (2004), pp. 1777–1791.
[17] B. Cockburn, E. Karniadakis, and C. W. Shu, eds., *Discontinuous Galerkin Methods—Theory, Computation, and Applications*, Lect. Notes Comput. Sci. Engrg. 11, Springer-Verlag, Berlin, 2000.
[18] S. Danilov, G. Kivman, and J. Schröter, *A finite-element ocean model: Principles and evaluation*, Ocean Modelling, 6 (2004), pp. 125–150.
[19] C. Dawson and S. Chippada, *Numerical modeling of shallow water flows with wetting and drying boundaries by a finite volume method*, in Proceedings of the 1998 Annual Conference on Mission Earth: Modeling and Simulation of the Earth System, Western MultiConference, 1998, pp. 9–14.
[20] C. Dawson and J. Proft, *Coupled discontinuous and continuous Galerkin finite element methods for the depth-integrated shallow-water equations*, Comput. Methods Appl. Mech. Engrg., 193 (2004), pp. 289–318.
[21] M. G. G. Foreman, *A two-dimensional dispersion analysis of selected methods for solving the linearized shallow-water equations*, J. Comput. Phys., 56 (1984), pp. 287–323.
[22] L. P. Franca and C. Farhat, *On the limitation of bubble functions*, Comput. Methods Appl. Mech. Engrg., 117 (1994), pp. 225–230.
[23] V. Girault and P. A. Raviart, *Finite Element Methods for Navier–Stokes Equations*, Springer Ser. Comput. Math. 5, Springer-Verlag, Berlin, 1986.
[24] C. M. Gossard and R. L. Kolar, *Phase behavior of a finite volume shallow water algorithm*, in Proceedings of the CMWR XIII Volume 2: Computational Methods, Surface Water Systems, and Hydrology, L. R. Bentley et al., eds., A. A. Balkema, Rotterdam, 2000, pp. 921–928.
[25] P. Hood and C. Taylor, *Navier–Stokes equations using mixed interpolation*, in Finite Elements in Flow Problems, J. T. Oden, O. C. Zienkiewicz, R. H. Gallagher, and C. Taylor, eds., UAH Press, Huntsville, AL, 1974, pp. 121–132.
[26] M. Horritt, *Development and testing of a simple 2D finite volume model of sub-critical shallow water flow*, Int. J. Numer. Methods Fluids, 44 (2004), pp. 1231–1255.
[27] B. L. Hua and F. Thomasset, *A noise-free finite-element scheme for the two-layer shallow-water equations*, Tellus, 36 (1984), pp. 157–165.
[28] M. Iskandarani, D. Haidvogel, and J. Boyd, *A staggered spectral finite-element model for the shallow-water equations*, Int. J. Numer. Methods Fluids, 20 (1995), pp. 393–414.
[29] I. P. Kinnmark and W. G. Gray, *A two-dimensional analysis of the wave equation model for finite element tidal computations*, Internat. J. Numer. Methods Engrg., 20 (1984), pp. 369–383.
[30] P. H. LeBlond and L. A. Mysak, *Waves in the Ocean*, Elsevier, Amsterdam, 1978.
[31] C. Le Provost and P. Vincent, *Finite Element for Modeling Ocean Tides*, in Tidal Hydrodynamics, B. Parker, ed., John Wiley and Sons, New York, 1991, pp. 41–60.
[32] D. Y. Le Roux, A. Staniforth, and C. A. Lin, *Finite elements for shallow-water equation ocean models*, Mon. Wea. Rev., 126 (1998), pp. 1931–1951.
[33] D. Y. Le Roux, C. A. Lin, and A. Staniforth, *A semi-implicit semi-Lagrangian finite-element shallow-water ocean model*, Mon. Wea. Rev., 128 (2000), pp. 1384–1401.
[34] D. Y. Le Roux and G. F. Carey, *Stability/dispersion analysis of the discontinuous Galerkin linearized shallow-water system*, Int. J. Numer. Methods Fluids, 48 (2005), pp. 325–347.
[35] D. Y. Le Roux, *Dispersion relation analysis of the $P_1^{NC} - P_1$ finite-element pair in shallow-water models*, SIAM J. Sci. Comput., 27 (2005), pp. 394–414.
[36] D. R. Lynch and W. G. Gray, *A wave-equation model for finite-element tidal computations*, Comput. Fluids, 7 (1979), pp. 207–228.
[37] R. Mullen and T. Belytschko, *Dispersion analysis of finite element semidiscretizations of the 2-dimensional wave equation*, Internat. J. Numer. Methods Engrg., 18 (1982), pp. 11–29.
[38] G. W. Platzman, *Some response characteristics of finite element tidal models*, J. Comput. Phys., 40 (1981), pp. 36–63.
[39] D. A. Randall, *Geostrophic adjustment and the finite-difference shallow-water equations*, Mon. Wea. Rev., 122 (1994), pp. 1371–1377.
[40] P. Raviart and J. Thomas, *A mixed finite element method for 2nd order elliptic problems*, in Mathematical Aspects of the Finite Element Methods, Lecture Notes in Math. 606, I. Galligani and E. Magenes, eds., Springer-Verlag, Berlin, 1977, pp. 292–315.
[41] D. Schwanenberg and J. Köngeter, *A discontinuous Galerkin method for the shallow-water equations with source terms*, in Discontinuous Galerkin Methods—Theory, Computation,

and Applications, B. Cockburn, E. Karniadakis, and C. W. Shu, eds., Lect. Notes Comput. Sci. Engrg. 11, Springer-Verlag, Berlin, 2000, pp. 419–424.

[42] C. M. Szpilka and R. L. Kolar, *Numerical analogs to Fourier and dispersion analysis: Development, verification, and application to the shallow water equations*, Adv. in Water Res., 26 (2003), pp. 649–662.

[43] M. Taylor, J. Tribbia, and M. Iskandarani, *The spectral element method for the shallow water-equations on the sphere*, J. Comput. Phys., 130 (1997), pp. 92–108.

[44] R. A. Walters and R. T. Cheng, *Accuracy of an estuarine hydrodynamic model using smooth elements*, Water Resources Res., 16 (1980), pp. 187–195.

[45] R. A. Walters and G. F. Carey, *Analysis of spurious oscillation modes for the shallow-water and Navier–Stokes equations*, Comput. Fluids, 11 (1983), pp. 51–68.

[46] R. A. Walters, *A three-dimensional finite-element model for coastal and estuarine circulation*, Continent. Shelf Res., 12 (1992), pp. 83–102.

[47] R. A. Walters and V. Casulli, *A robust, finite-element model for hydrostatic surface water flows*, Commun. Numer. Meth. Engrg., 14 (1998), pp. 931–940.

[48] R. T. Williams and O. C. Zienkiewicz, *Improved finite-element forms for the shallow-water wave equations*, Int. J. Numer. Methods Fluids, 1 (1981), pp. 81–97.

SIAM J. SCI. COMPUT.
Vol. 29, No. 1, pp. 361–375
© 2007 Society for Industrial and Applied Mathematics

ORDER-OPTIMAL PRECONDITIONERS FOR IMPLICIT RUNGE–KUTTA SCHEMES APPLIED TO PARABOLIC PDES*

K.-A. MARDAL†, T. K. NILSSEN‡, AND G. A. STAFF†

Abstract. In this paper we show that standard preconditioners for parabolic PDEs discretized by implicit Euler or Crank–Nicolson schemes can be reused for higher-order fully implicit Runge–Kutta time discretization schemes. We prove that the suggested block diagonal preconditioners are order-optimal for A-stable and irreducible Runge–Kutta schemes with invertible coefficient matrices. The theoretical investigations are confirmed by numerical experiments.

Key words. preconditioners, Runge–Kutta schemes, parabolic PDE

AMS subject classifications. 65, 65M55

DOI. 10.1137/05064093X

1. Introduction.

Let Ω be a bounded polygonal domain in $\mathbb{R}^d$, with $d = 2$ or 3, and boundary $\partial\Omega$. In this paper we will consider a parabolic PDE of the form

$$\frac{\partial u}{\partial t} = \Delta u + f \quad \text{in } \Omega,\ t > 0, \tag{1}$$

$$u = 0 \quad \text{on } \partial\Omega,\ t \geq 0, \tag{2}$$

$$u = u^0 \quad \text{in } \Omega,\ t = 0. \tag{3}$$

Equations (1)–(3) are discretized in space, by a finite element method, which gives a system of ODEs to be solved,

$$M\frac{du_h}{dt} = -Ku_h + f_h, \quad t > 0, \tag{4}$$

$$u_h = u_h^0, \quad t = 0, \tag{5}$$

where M is the mass matrix and K is the stiffness matrix. Higher-order time discretization schemes based on fully implicit Runge–Kutta schemes are rarely used when discretizing (4)–(5), due to the additional cost of solving a larger and more complicated system of equations. The lack of efficient solution algorithms for such systems has prevented these algorithms from widespread use for PDEs, despite the many appealing properties of such schemes. However, the lack of efficient algorithms applies not only to PDEs, but to ODEs in general—a fact we emphasize by quoting as follows from some standard ODE textbooks:

*Received by the editors September 22, 2005; accepted for publication (in revised form) August 29, 2006; published electronically February 5, 2007.
http://www.siam.org/journals/sisc/29-1/64093.html
†Department of Scientific Computing, Simula Research Laboratory, 1325 Lysaker, Norway (kent-and@simula.no, gunnaran@simula.no).
‡Department of Scientific Computing, Scandpower Petroleum Technology, 2027 Kjeller, Norway (trygve.nilssen@scandpowerpt.com).

(1) [illegible] [8, p. 118].

(2) [illegible] [2, pp. 103, 104].

In this paper we will see how one can reuse fast solution algorithms developed for low-order time discretizations, in the case of a parabolic PDE.

When (4)–(5) are discretized by lower-order time discretization schemes such as implicit Euler, we arrive at the following sequence of linear systems to be solved at each time step:

$$(M + \delta t K) u_h^n = M u^{n-1} + \delta t f_h^n, \tag{6}$$

where δt is the time stepping parameter. Order-optimal solution algorithms for this system have been found for most spatial discretization methods. The goal in this paper is to reuse such preconditioners for fully implicit Runge–Kutta schemes. In that way the method will have implementational similarities to BDF (backward differentiation formulae) and DIRK (diagonally implicit Runge–Kutta) methods, but still will have the stability and convergence of the chosen fully implicit Runge–Kutta method. Furthermore, we will prove that these preconditioners are order optimal with respect to δt and h and demonstrate their efficiency by several numerical experiments. Order-optimality in s, where s is the number of stage values in the Runge–Kutta schemes, is not obtained. However, we demonstrate that the s-dependency is weak compared to the accuracy gained. Furthermore, we have in [19] numerically demonstrated that this s-dependency can be reduced by generalizing a Gauss–Seidel preconditioner with certain coefficients determined by a simple optimization principle.

Except for BDF and DIRK methods, which require only the solution of systems of the form (6), the authors know of only a few other works on solution algorithms for higher-order time discretization schemes in the context of PDEs (see [5], [7], [11], [12], [13], [14], and [15]). References [12] and [13] present an inexact block LU preconditioner when using the W-transformation on the Runge–Kutta schemes. Block Jacobi and Gauss–Seidel preconditioners for generalized Adams methods are studied in [11]. In [15], we discussed preconditioners for higher-order Padé formulations of parabolic PDEs, where the matrix was a polynomial of $(M + \delta t K)$-like matrices, of the form $\Pi_{i=1}^{n}(M + a_i \delta t K)^i$. In this work, we instead consider linear systems of $(M + \delta t K)$-like matrices.

In the recent work [14], multigrid methods for Runge–Kutta methods were studied. A key idea in this work was to employ block smoothers, simultaneously updating all unknowns related to a spatial grid point. A convergence rate independent of the number of quadrature nodes of the Runge–Kutta scheme, s, was obtained, but the smoothers require the solution of $s \times s$ systems at each spatial grid point. Our approach is different in the sense that we employ standard preconditioners (e.g., multigrid with pointwise smoothers or domain decomposition) for (6) on each of the blocks in the block matrix arising from the Runge–Kutta discretization.

The order-optimality of the method proposed here is proved by viewing the differential operator as an isomorphism in appropriate spaces. This gives an upper bound on the condition number of the preconditioned system, and therefore an upper bound on the number of iterations. Other works taking the same approach are [1], [9], and [17].

The preconditioners discussed in this paper can be (and have been) implemented as a block diagonal matrix of standard preconditioners. Practical details concerning

the implementation of block preconditioners (in the C++ library Diffpack) can be found in [16]. More numerical experiments can be found in [19]. Here, numerical experiments also in three dimensions verify the order-optimality. Furthermore, it is shown numerically that block Gauss–Seidel preconditioners can improve the efficiency of the preconditioner.

The remaining part of the paper is organized as follows. In section 2 we review some useful concepts. In section 3 we prove that the continuous version of the time stepping operator for Runge–Kutta methods is an isomorphism bounded independent of δt, when using the proper interpolation spaces. This enables us to construct order-optimal preconditioners in the discrete case in section 4. Section 5 presents some numerical experiments.

2. Preliminaries. Let $H^1 = H^1(\Omega)$ be the Sobolev space with first-order derivatives in $L^2 = L^2(\Omega)$, with norm $\|\cdot\|_1$. The space H_0^1 is the closure of C_0^∞ in H^1, with norm $\|\cdot\|_{H_0^1} = |\cdot|_1$, which is the L_2 norm of the first-order derivatives. The $|\cdot|_1$ norm is equivalent to the standard H^1 norm on H_0^1. The space H^{-1} is the dual space of H_0^1. Vector valued functions and spaces are denoted by boldface symbols. The inner product of both scalars, vectors, and matrices, as well as the duality pairing of H_0^1 and H^{-1} and their corresponding vector spaces, are denoted by $(\cdot,\cdot)$, and the corresponding norm is denoted by $\|\cdot\|$. The operators Δ and ∇ are applied to both scalars and vectors, where the scalar versions are standard and the vector versions are defined as

$$(\Delta \boldsymbol{u})_i = \Delta u_i \quad \text{and } (\nabla \boldsymbol{u})_{i,j} = \frac{\partial u_i}{\partial x_j} \quad \text{for } 1 \le i \le s,\ 1 \le j \le d,$$

where u_i is the ith component of $\boldsymbol{u}$. Notice that $\nabla \boldsymbol{u}$ is an $s \times d$ matrix. The spaces involving time are defined by

$$\|u\|_{L^2(0,T;X)} = \left(\int_0^T \|u(t)\|_X^2 dt \right)^{1/2},$$

where $\|\cdot\|_X$ is the spatial norm in the space X (which is either H_0^1, H^{-1}, or L^2). The weak form of (1)–(3) is as follows:

Find $u \in L^2(0,T;H_0^1)$ with $\frac{\partial u}{\partial t} \in L^2(0,T;H^{-1})$ such that

$$\left(\frac{\partial u}{\partial t}, v\right) + (\nabla u, \nabla v) = (f, v) \quad \forall v \in H_0^1, \quad t > 0, \tag{7}$$

$$u = u_0, \quad t = 0. \tag{8}$$

Similarly, the finite element formulation is defined by seeking an approximation $u_h(t)$ in a finite element subspace $V_h \subset H_0^1$ by the following:

Find $u_h \in L^2(0,T;V_h)$ with $\frac{\partial u_h}{\partial t} \in L^2(0,T;V_h)$ such that

$$\left(\frac{\partial u_h}{\partial t}, v\right) + (\nabla u_h, \nabla v) = (f, v) \quad \forall v \in V_h, \quad t > 0,$$

$$u = u_0, \quad t = 0.$$

This can be written as a linear system of ODEs,

$$I_h \frac{\partial u_h}{\partial t} - \Delta_h u_h = f_h,$$

where I_h is the identity operator (the mass matrix M) and Δ_h is the Δ operator (the negative stiffness matrix $-K$) on V_h. The right-hand side f_h is the L_2-projection of f on V_h (often approximated by the mass matrix times a vector with the values of f evaluated at nodal points). We may write this system as

$$\frac{\partial u_h}{\partial t} - I_h^{-1}\Delta_h u_h = I_h^{-1} f_h,$$

and the eigenvalues of $-I_h^{-1}\Delta_h$ are real and positive.

The unknown $u_h(t)$ is approximated by a Runge–Kutta method,

$$u_h^n = u_h^{n-1} + \delta t \sum_{i=1}^{s} b_i u_{h,i}^n,$$

where

$$I_h u_{h,i}^n = \Delta_h \left(u_h^{n-1} + \delta t \sum_{j=1}^{s} a_{ij} u_{h,j}^n \right) + f_h(t_{n-1} + c_i \delta t), \quad 1 \le i \le s, \tag{9}$$

where $u_{h,i}^n$ is the intermediate stage value approximating the time derivative of u at a given quadrature node (we use the slightly confusing variable name $u_{h,i}^n$ for the time derivatives of u, since $u_{h,i}^n$ is the unknown that appears in our linear system). This system can be written as

$$(\boldsymbol{I}^{s,s} \otimes I_h - \delta t A \otimes \Delta_h)\boldsymbol{u}_h^n = -\boldsymbol{I}^{s,s} \otimes \Delta_h \boldsymbol{u}_h^{n-1} + \boldsymbol{f}_h^n,$$

where $\boldsymbol{f}_h^n = (f_{h,1}^n, \ldots, f_{h,s}^n)^T$, $f_{h,i}^n = f_h(t_{n-1} + c_i \delta t)$; $\boldsymbol{u}_h^n$ is the vector of intermediate stage values $\boldsymbol{u}_h^n = (u_{h,1}^n, \ldots, u_{h,s}^n)^T$; $\boldsymbol{I}^{s,s}$ is the identity matrix in $\mathbb{R}^{s,s}$; A, b, and c are the Runge–Kutta coefficients, weights, and nodes; and $\otimes$ is the Kronecker tensor product,

$$A \otimes B = \begin{pmatrix} a_{11}B & \ldots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{n1}B & \ldots & a_{nn}B \end{pmatrix}.$$

A Runge–Kutta scheme is A-stable if the stability function

$$R(z) = 1 + z b^T (I - zA)^{-1} \mathbf{1}$$

satisfies $|R(z)| \le 1$ when $\mathrm{Re}(z) < 0$, where $\mathbf{1} \in \mathbb{R}^s$ is a vector of ones. In this paper we will assume that the Runge–Kutta methods are A-stable, irreducible, and have invertible A, such as Gauss, LobattoIIIC, RadauIA, and RadauIIA; cf., e.g., [8]. We will also assume that the diagonal entries of A are positive. We refer to [6] for a description of parabolic PDEs and to [20] for corresponding discretization methods.

We use the common definition of an order-optimal preconditioner, which is that $\mathcal{B}_k$ is an order-optimal preconditioner for $\mathcal{A}_k$, with respect to the parameter k, given that the condition number of $\mathcal{B}_k\mathcal{A}_k$ is bounded independently of the parameter k, and that the evaluation and storage of $\mathcal{B}_k$ are similar to that of $\mathcal{A}_k$. In this paper we will show that the considered preconditioners are order optimal with respect to both the parameters h and δt.

Finally, we will need the intersection and sum of Hilbert spaces. If X and Y are Hilbert spaces, both continuously contained in some larger Hilbert space, then the intersection $X \cap Y$ and the sum $X + Y$ are Hilbert spaces, and the norms are defined as

$$\|z\|_{X\cap Y} = (\|z\|_X^2 + \|z\|_Y^2)^{1/2}$$

and

$$\|z\|_{X+Y} = \inf_{\substack{z=x+y \\ x\in X,\, y\in Y}} (\|x\|_X^2 + \|y\|_Y^2)^{1/2}.$$

Furthermore, if $X \cap Y$ is dense in both X and Y, then $(X \cap Y)^* = X^* + Y^*$. We refer to [4, Chapter 2] for these results. We will use the space $L^2 \cap \delta t H_0^1$ and its dual space $L^2 + \frac{1}{\delta t} H^{-1}$, and their vector spaces, $\boldsymbol{L}^2 \cap \delta t \boldsymbol{H}_0^1$ and $\boldsymbol{L}^2 + \frac{1}{\delta t} \boldsymbol{H}^{-1}$, with dimension s. We remark that for $\delta t > 0$, $L^2 \cap \delta t H_0^1$ is equal to H^1 as a set, but as $\delta t \to 0$, the norm degenerates to L^2. Since the spaces will appear many times, we have adopted the following shorthand notation:

$$V = L^2 \cap \delta t H_0^1,$$

$$V^* = L^2 + \frac{1}{\delta t} H^{-1},$$

$$\boldsymbol{V} = \boldsymbol{L}^2 \cap \delta t \boldsymbol{H}_0^1,$$

$$\boldsymbol{V}^* = \boldsymbol{L}^2 + \frac{1}{\delta t} \boldsymbol{H}^{-1}.$$

3. An existence result. The purpose of this paper is to introduce preconditioners for Runge–Kutta schemes applied to parabolic PDEs that are uniform in both space and time (but not in the number of stages). To do this we will employ the spaces introduced in the previous section to describe the spatial problem to be solved at each time step.

We start with the following implicit Euler time discretization of (1) to explain the problem:

Find $u^n \in V$, such that

$$a^E(u^n, v) = b^n(v) \quad \forall v \in V,$$

where

$$a^E(u^n, v) = (u^n, v) + \delta t(\nabla u^n, \nabla v),$$

$$b^n(v) = (u^{n-1}, v) + \delta t(f^n, v).$$

It then follows that

$$a^E(u, v) \le c_1^E \|u\|_V \|v\|_V,$$

$$a^E(u, u) \ge \frac{1}{c_2^E} \|u\|_V^2.$$

From the Lax–Milgram theorem (or the Riesz representation theorem since a^E is symmetric) we know that there exists a unique solution in V, depending continuously on b^n in V^*. Hence, if we let $\mathcal{A}^E$ be defined by

$$(\mathcal{A}^E u, v) = a^E(u, v),$$

where

$$\mathcal{A}^E = I - \delta t \Delta,$$

then $\mathcal{A}^E$ is an isomorphism mapping V to V^* bounded independently of δt, i.e.,

$$\|\mathcal{A}^E\|_{\mathcal{L}(V,V^*)} \le c_1^E \quad \text{and} \quad \|(\mathcal{A}^E)^{-1}\|_{\mathcal{L}(V^*,V)} \le c_2^E.$$

In contrast, if we had considered $\mathcal{A}^E$ as a mapping from H_0^1 to H^{-1}, then c_1^E and c_2^E would depend on δt.

Similarly, the implicit Runge–Kutta method (i.e., the continuous version of (9)) can be written in the following form:

Find $\boldsymbol{u}^n \in \boldsymbol{V}$, such that

$$(\boldsymbol{u}^n, \boldsymbol{v}) + \delta t(A \otimes \Delta \boldsymbol{u}^n, \boldsymbol{v}) = (-\boldsymbol{I}^{s,s} \Delta \boldsymbol{u}^{n-1}, \boldsymbol{v}) + \delta t(\boldsymbol{f}^n, \boldsymbol{v}) \quad \forall \boldsymbol{v} \in \boldsymbol{V}. \tag{10}$$

The solution u^n is obtain by

$$u^{n+1} = u^n + \delta t \sum_{i=1}^{s} b_i u_i^n.$$

Equation (10) may be written as

$$\mathcal{A}\boldsymbol{u} = \boldsymbol{f},$$

where

$$\mathcal{A} = I - \delta t A \otimes \Delta.$$

In this section we will show that $\mathcal{A} : \boldsymbol{V} \to \boldsymbol{V}^*$ is an isomorphism,

$$\|\mathcal{A}\|_{\mathcal{L}(\boldsymbol{V},\boldsymbol{V}^*)} \le c_1 \quad \text{and} \quad \|\mathcal{A}^{-1}\|_{\mathcal{L}(\boldsymbol{V}^*,\boldsymbol{V})} \le c_2, \tag{11}$$

where both c_2 and c_1 are independent of δt. The associated bilinear form is

$$\begin{aligned} a(\boldsymbol{u}, \boldsymbol{v}) &= (\mathcal{A}\boldsymbol{u}, \boldsymbol{v}) = ((I - \delta t A \otimes \Delta)\boldsymbol{u}, \boldsymbol{v}) \\ &= (\boldsymbol{u}, \boldsymbol{v}) - \delta t(A\Delta \boldsymbol{u}, \boldsymbol{v}) \\ &= (\boldsymbol{u}, \boldsymbol{v}) + \delta t(A\nabla \boldsymbol{u}, \nabla \boldsymbol{v}). \end{aligned}$$

We remark that $a(\cdot, \cdot)$ is neither positive nor symmetric for a general A arising from a Runge–Kutta scheme. Therefore we cannot use the theorems of Riesz or Lax and Milgram. To be able to prove that $\mathcal{A}$ is an isomorphism, we need the theorem of Babuska and Aziz [3], which in our setting reads as follows.

THEOREM 3.1. *A* $\boldsymbol{V}$ $\boldsymbol{V}^*$

- c_1 δt

$$|a(\boldsymbol{u},\boldsymbol{v})| \le c_1 \|\boldsymbol{u}\|_{\boldsymbol{V}} \|\boldsymbol{v}\|_{\boldsymbol{V}} \quad \forall \boldsymbol{u},\boldsymbol{v} \in \boldsymbol{V}. \tag{12}$$

- c_2 δt (inf-sup)

$$\sup_{\boldsymbol{v}\in\boldsymbol{V}} \frac{a(\boldsymbol{u},\boldsymbol{v})}{\|\boldsymbol{v}\|_{\boldsymbol{V}}} \ge \frac{1}{c_2}\|\boldsymbol{u}\|_{\boldsymbol{V}} \quad \forall \boldsymbol{u} \in \boldsymbol{V}. \tag{13}$$

- $\boldsymbol{v} \in \boldsymbol{V}$ $\boldsymbol{u} \in \boldsymbol{V}$

$$a(\boldsymbol{u},\boldsymbol{v}) \neq 0. \tag{14}$$

. See [3]. □

Hence, it remains to prove (12)–(14). The first two properties are proven in the following lemmas.

LEMMA 3.1. c_1 δt

$$|a(\boldsymbol{u},\boldsymbol{v})| \le c_1 \|\boldsymbol{u}\|_{\boldsymbol{V}} \|\boldsymbol{v}\|_{\boldsymbol{V}}.$$

. To show (12) we see that

$$\begin{aligned}
a(\boldsymbol{u},\boldsymbol{v}) &= ((I - \delta t A \otimes \Delta)\boldsymbol{u},\boldsymbol{v}) \\
&= (\boldsymbol{u},\boldsymbol{v}) - \delta t \sum_{i,j\le s} a_{ij} \left(\Delta u_i, v_j\right) \\
&= (\boldsymbol{u},\boldsymbol{v}) + \delta t \sum_{i,j\le s} a_{ij} \left(\nabla u_i, \nabla v_j\right) \\
&\le a_{max} \left(\|\boldsymbol{u}\|\|\boldsymbol{v}\| + \delta t \sum_{i,j\le s} \|\nabla u_i\|\|\nabla v_j\| \right) \qquad (15) \\
&\le c_1 \left(\|\boldsymbol{u}\|\|\boldsymbol{v}\| + \delta t \|\nabla \boldsymbol{u}\|\|\nabla \boldsymbol{v}\| \right) \qquad (16) \\
&\le c_1 \left(\|\boldsymbol{u}\|^2 + \delta t \|\nabla \boldsymbol{u}\|^2 \right)^{\frac{1}{2}} \left(\|\boldsymbol{v}\|^2 + \delta t \|\nabla \boldsymbol{v}\|^2 \right)^{\frac{1}{2}} \\
&= c_1 \|\boldsymbol{u}\|_V \|\boldsymbol{v}\|_V,
\end{aligned}$$

where (15) follows by the Cauchy–Schwarz inequality and the definition

$$a_{max} = \max\left(\max_{ij} |a_{ij}|, 1 \right),$$

and (16) comes from the fact that

$$\begin{aligned}
\sum_{i,j\le s} \|\nabla u_i\|\|\nabla v_j\| &= \sum_{i\le s} \|\nabla u_i\| \sum_{j\le s} \|\nabla v_j\| \\
&\le s \|\nabla \boldsymbol{u}\|\|\nabla \boldsymbol{v}\|, \qquad (17)
\end{aligned}$$

where (17) follows from the equivalence ℓ^1 and ℓ^2 norms on $\mathbb{R}^s$. □

To prove the inf-sup condition we introduce a family of matrices which we will refer to as weakly positive definite. These matrices are defined as follows.

DEFINITION 3.1. [illegible] $A \in \mathbb{R}^{s,s}$ [illegible] $C \in \mathbb{R}^{s,s}$ [illegible]

$$x^T C x > 0 \quad \forall x \in \mathbb{R}^s, \tag{18}$$

$$x^T C A x > 0 \quad \forall x \in \mathbb{R}^s. \tag{19}$$

LEMMA 3.2. [illegible]

The proof of this lemma can be found in [18]. Weakly positive matrices can be seen as a generalization of the diagonally stable matrices discussed in, e.g., [10].

In the following calculations we need the existence of a positive number α such that

$$x^T C x \geq \alpha \|x\|^2 \quad \forall x \in \mathbb{R}^s, \tag{20}$$

$$x^T C A x \geq \alpha \|x\|^2 \quad \forall x \in \mathbb{R}^s. \tag{21}$$

This follows since C and CA are positive definite (not necessarily symmetric). To see that (18) leads to (20), let $x = \frac{y}{\|y\|}$ and use linearity to obtain the equivalent statement that $x^T C x > \alpha$ for $\|x\| = 1$. Furthermore, since $x^T C x$ is a continuous function of x and the set defined by $\|x\| = 1$ is compact, $x^T C x$ will have a lower bound, and therefore (20) follows from (18). Similarly, (21) follows from (19).

In the next lemma we use that the Runge–Kutta coefficient matrix A is weakly positive definite; i.e., there exists a positive definite matrix C such that CA is positive definite. By Lemma 3.2 this is true as long as the arguments of the eigenvalues of A differ from π. However, for A-stable irreducible Runge–Kutta methods with an invertible A a stronger result is known, namely, that the real parts of the eigenvalues of A are positive. To see this, notice that the stability function of a Runge–Kutta method is given by

$$R(z) = \frac{\det(I - zA + z\mathbf{1}b^T)}{\det(I - zA)}.$$

Irreducibility gives that the fraction cannot be reduced. Assume that there exists an eigenvalue μ of A with $Re(\mu) < 0$. Then the stability function will have a pole in $z = 1/\mu$, which also lies in the left half plane. This contradicts with the A-stability, which requires that $|R(z)| \leq 1$ for all z in the left half plane. Furthermore, A is invertible; therefore zero is not an eigenvalue. We therefore conclude that the real part of the eigenvalues must be positive.

LEMMA 3.3. [illegible] c_2 [illegible] δt [illegible]

$$\sup_{\boldsymbol{v} \in \boldsymbol{V}} \frac{a(\boldsymbol{u}, \boldsymbol{v})}{\|\boldsymbol{v}\|_{\boldsymbol{V}}} \geq \frac{1}{c_2} \|\boldsymbol{u}\|_{\boldsymbol{V}} \quad \forall \boldsymbol{u} \in \boldsymbol{V}.$$

[illegible]. Define

$$\alpha = \min \left(\min_{\|x\|=1} x^T C x, \min_{\|x\|=1} x^T C A x \right).$$

Given $\boldsymbol{u} \in \boldsymbol{V}$, let $\boldsymbol{v} = C^T \boldsymbol{u}$, where C is constructed such that we have (20)–(21) with A being the Runge–Kutta coefficient matrix. Then,

$$\begin{aligned}
\sup_{\boldsymbol{v}\in\boldsymbol{V}} \frac{(\mathcal{A}\boldsymbol{u},\boldsymbol{v})_{L^2}}{\|\boldsymbol{u}\|_{\boldsymbol{V}}\|\boldsymbol{v}\|_{\boldsymbol{V}}} &\geq \frac{(\mathcal{A}\boldsymbol{u}, C^T\boldsymbol{u})_{L^2}}{\|\boldsymbol{u}\|_{\boldsymbol{V}}\|C^T\boldsymbol{u}\|_{\boldsymbol{V}}} \\
&= \frac{(C\mathcal{A}\boldsymbol{u},\boldsymbol{u})_{L^2}}{\|\boldsymbol{u}\|_{\boldsymbol{V}}\|C^T\boldsymbol{u}\|_{\boldsymbol{V}}} \\
&= \frac{(C\boldsymbol{u},\boldsymbol{u})_{L^2} - \delta t((CA\otimes\Delta)\boldsymbol{u},\boldsymbol{u})_{L^2}}{\|\boldsymbol{u}\|_{\boldsymbol{V}}\|C^T\boldsymbol{u}\|_{\boldsymbol{V}}} \\
&\geq \frac{1}{\|C\|}\frac{(C\boldsymbol{u},\boldsymbol{u})_{L^2} + \delta t(CA\nabla\boldsymbol{u},\nabla\boldsymbol{u})_{L^2}}{\|\boldsymbol{u}\|_{\boldsymbol{V}}\|\boldsymbol{u}\|_{\boldsymbol{V}}} \\
&\geq \frac{\alpha}{\|C\|}\frac{\|\boldsymbol{u}\|_{L^2}^2 + \delta t\|\nabla\boldsymbol{u}\|_{L^2}^2}{\|\boldsymbol{u}\|_{\boldsymbol{V}}^2} \\
&= \frac{\alpha}{\|C\|} \\
&> 0,
\end{aligned}$$

where $\|C\|$ is the L^2 matrix norm. Hence, $c_2 = \frac{\|C\|}{\alpha}$ in (13). □

Finally, (14) follows by an argument similar to the proof of (13) in Lemma 3.3. First, we notice that A^T and A share the same set of eigenvalues, and therefore A^T is also weakly positive definite. By Lemma 3.2 we now have that there exists a D such that both D and DA^T are positive definite. Further let $\boldsymbol{v}$ be given and let $\boldsymbol{u} = D^T\boldsymbol{v}$. Then we have

$$\begin{aligned}
a(\boldsymbol{u},\boldsymbol{v}) = a(D^T\boldsymbol{v},\boldsymbol{v}) &= ((I - A\otimes\Delta)D^T\boldsymbol{v},\boldsymbol{v}) = ((D^T - AD^T\otimes\Delta)\boldsymbol{v},\boldsymbol{v}) \\
&= (\boldsymbol{v}, D\boldsymbol{v}) + (\nabla\boldsymbol{v}, DA^T\nabla\boldsymbol{v}) > 0.
\end{aligned}$$

We therefore conclude that $\mathcal{A}$ is an isomorphism (11).

4. The preconditioner. The preconditioner for the continuous operator is constructed based on the weighted Sobolev spaces. The discrete preconditioner can be viewed as an operator acting on discrete subspaces. Other works taking the same approach are [1], [9], and [17].

Let $\mathcal{B} : \boldsymbol{V}^* \to \boldsymbol{V}$ be

$$\mathcal{B} = (I - \delta t\,\mathrm{diag}(A)\otimes\Delta)^{-1}, \tag{22}$$

where we assume that the diagonal of A is always strictly positive. It follows from the definition of $\boldsymbol{V}$ and $\boldsymbol{V}^*$ that

$$\|\mathcal{B}\|_{\mathcal{L}(\boldsymbol{V}^*,\boldsymbol{V})} \leq d_1 \quad \text{and} \quad \|\mathcal{B}^{-1}\|_{\mathcal{L}(\boldsymbol{V},\boldsymbol{V}^*)} \leq d_2, \tag{23}$$

where d_1 and d_2 are independent of δt. Hence,

$$\mathcal{B}\mathcal{A} : \boldsymbol{V} \to \boldsymbol{V}$$

and

$$\|\mathcal{BA}\|_{\mathcal{L}(\boldsymbol{V},\boldsymbol{V})} \le c_1 d_1 \quad \text{and} \quad \|(\mathcal{BA})^{-1}\|_{\mathcal{L}(\boldsymbol{V},\boldsymbol{V})} \le c_2 d_2. \tag{24}$$

This means that the condition number of the continuous operator is bounded, i.e.,

$$\kappa(\mathcal{BA}) = \|\mathcal{BA}\|_{\mathcal{L}(\boldsymbol{V},\boldsymbol{V})}\|(\mathcal{BA})^{-1}\|_{\mathcal{L}(\boldsymbol{V},\boldsymbol{V})} \le c_1 d_1 c_2 d_2. \tag{25}$$

In the following we study the condition number in the discrete case. Let Ω_h be a triangulation of Ω. Furthermore, let V_h be a finite element space $V_h \subset H_0^1$, while $\boldsymbol{V}_h = (V_h)^s \subset \boldsymbol{H}_0^1$ is the corresponding vector finite element space. The discrete counterparts of $\mathcal{A}$ and $\mathcal{B}$ are defined by

$$(\mathcal{A}_h \boldsymbol{u}_h, \boldsymbol{v}_h) = (\boldsymbol{u}_h, \boldsymbol{v}_h) + \delta t (A\nabla \boldsymbol{u}_h, \nabla \boldsymbol{v}_h) \qquad \forall \boldsymbol{u}_h, \boldsymbol{v}_h \in \boldsymbol{V}_h, \tag{26}$$

$$(\mathcal{B}_h^{-1} \boldsymbol{u}_h, \boldsymbol{v}_h) = (\boldsymbol{u}_h, \boldsymbol{v}_h) + \delta t \,\mathrm{diag}(A)(\nabla \boldsymbol{u}_h, \nabla \boldsymbol{v}_h) \qquad \forall \boldsymbol{u}_h, \boldsymbol{v}_h \in \boldsymbol{V}_h. \tag{27}$$

Since $\boldsymbol{V}_h$ is a subspace of $\boldsymbol{V}$ we get that

$$\kappa(\mathcal{B}_h \mathcal{A}_h) \le \kappa(\mathcal{BA}). \tag{28}$$

Let $\tilde{B}_i$ be a cheap approximation of $B_i = (I - \delta t a_{ii} \Delta_h)^{-1}$ constructed by, e.g., multigrid or domain decomposition. This is the standard preconditioner which is used for the implicit Euler step. It is then well known that $\tilde{B}_i$ may be constructed such that it is symmetric and spectrally equivalent to B_i, i.e.,

$$c_3(\tilde{B}_i^{-1} v, v) \le ((I - \delta t a_{ii} \Delta_h)v, v) \le c_4(\tilde{B}_i^{-1} v, v) \quad \forall v \in V_h. \tag{29}$$

Let c_3 and c_4 be chosen such that (29) is valid for all i. Let further the full preconditioner be denoted

$$\tilde{\mathcal{B}}_h = \begin{pmatrix} \tilde{B}_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \tilde{B}_s \end{pmatrix}.$$

We have that the condition number fulfills the Cauchy–Schwarz-like property $\kappa(CD) \le \kappa(C)\kappa(D)$ for two matrices C and D, since

$$\kappa(CD) = \|CD\|\|(CD)^{-1}\| \le \|C\|\|D\|\|C^{-1}\|\|D^{-1}\| = \kappa(C)\kappa(D).$$

Therefore we get

$$\kappa(\tilde{\mathcal{B}}_h \mathcal{A}_h) \le \kappa(\tilde{\mathcal{B}}_h \mathcal{B}_h^{-1}) \kappa(\mathcal{B}_h \mathcal{A}_h).$$

By using (25) and (29), we get

$$\kappa(\tilde{B}_h \mathcal{A}_h) \le \frac{c_1 d_1 c_2 d_2 c_4}{c_3}. \tag{30}$$

5. Numerical experiments. In this section we test our preconditioner in a series of numerical experiments. We test three families of fully implicit Runge–Kutta schemes: Gauss, Radau, and Lobatto. The different families have different properties when it comes to the order of the approximation and the stability. We will give only a brief description here and refer the reader to [8] for a more thorough description.

By using Gauss quadrature rules, we get the implicit Runge–Kutta Gauss methods. They have a global error estimate of $\mathcal{O}(\delta t^{2s})$, where s is the number of quadrature points. Notice that the single-node Gauss method is the implicit midpoint method and is equal to the Crank–Nicolson scheme in our example.

Using Radau quadrature, which requires one end point among the quadrature nodes, the global error estimate is $\mathcal{O}(\delta t^{2s-1})$. These methods will be named RadauIA and RadauIIA, where the start or the end point is among the quadrature nodes, respectively. Notice that implicit Euler is the single-node method, where the end point is the quadrature node.

Finally, one can require that both the start and end points are quadrature nodes, which is called Lobatto quadrature. There exist three different subfamilies of the Lobatto scheme, where the coefficient matrix A differs among them. They have different properties regarding, e.g., stability, but they all have a global error estimate of $\mathcal{O}(\delta t^{2s-2})$. Two of the subfamilies have one explicit step. These methods are not studied here since they have a singular A-matrix. We will consider only the LobattoIIIC methods.

All the methods are A-stable, but only the Radau methods and the LobattoIIIC methods are L-stable, which is an attractive property for stiff problems, such as a semidiscretized heat equation. In addition, the RadauIIA and LobattoIIIC methods are stiffly accurate.

It is well known from standard ODE theory that in the case of very stiff problems, where $\delta t \to 0$ while $z = \lambda \delta t \to \infty$, where λ is an eigenvalue of $I_h^{-1}\Delta_h$, the order of the implicit Runge–Kutta schemes is reduced. This is known as B-convergence and is more thoroughly described in [8]. This does, however, affect only the order of convergence for the Runge–Kutta scheme and not the performance of our preconditioner. In our numerical tests, we did not experience order reduction.

5.1. The condition number. First, we want to verify numerically the order-optimality of the preconditioner with respect to the spatial discretization parameter h and the time step δt. This is done for a three-node RadauIIA method. We solve a 1D problem, where linear finite elements are used in space. The preconditioner is the inverse of the submatrices on the diagonal. The results are presented in Table 1 and Figure 1.

TABLE 1

The condition number $\kappa(\mathcal{BA})$ for the 1D problem (1)–(3) using linear finite elements in space, and using the three-node RadauIIA method in time. The preconditioner is constructed by inverting the diagonal blocks exactly.

$\delta t/h$	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}
0.1	8.23	13.40	14.93	15.32	15.42	15.44	15.45	15.45
0.05	5.49	11.79	14.44	15.19	15.38	15.43	15.44	15.45
0.02	2.90	8.55	13.14	14.82	15.29	15.41	15.44	15.45
0.01	1.91	5.79	11.37	14.23	15.13	15.37	15.43	15.44
0.005	1.43	3.58	8.89	13.18	14.82	15.29	15.41	15.44
0.002	1.16	1.99	5.28	10.72	13.96	15.05	15.35	15.42
0.001	1.08	1.47	3.24	8.08	12.71	14.67	15.25	15.40

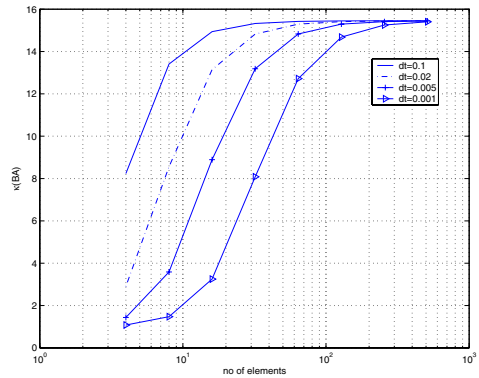


FIG. 1. *The condition number* $\kappa(\mathcal{BA})$ *for the* 1*D problem* (1)–(3) *using linear finite elements in space, and using the three-node RadauIIA method in time. The preconditioner is inverted exactly.*

If the time step δt is sufficiently small compared to the spatial discretization parameter h, the Helmholtz operator is close to the mass matrix, and the condition number is small. For a sufficiently small spatial discretization compared to the time step, the condition number $\kappa(\mathcal{BA})$ of the preconditioned system seems to reach an asymptotic value of about 15.4. The next section concerns the asymptotic value for this and other schemes in both one and two dimensions using multigrid preconditioners. In [19] we have also shown experiments in three dimensions.

The results clearly confirm that the preconditioner is order optimal with respect to both the spatial discretization parameter h and the time step δt.

5.2. Multigrid preconditioning. We will use multigrid to approximate the preconditioner. All computations will be done on a domain $\Omega = (0,1)^d$, where d is the number of spatial dimensions. A sequence of meshes is constructed by uniform refinement of a 2 or 2×2 partition of the domain Ω. The preconditioner $\tilde{\mathcal{B}}$ is computed using a standard V-cycle with a symmetric Gauss–Seidel smoother. Gaussian elimination is used as the coarse grid solver.

We estimate the condition number using one multigrid V-cycle as the preconditioner. The results are shown in Table 2 for both one and two dimensions. The discretization parameters δt and h are chosen such that the condition number is close to the asymptotic value. In our case $\delta t = 0.1$ and $h = 2^{-9}$ have been appropriate. The results for multigrid in two dimensions are better than the results for multigrid in one dimension, and this seems a bit strange. We have tested this multigrid method also in the case of a Poisson equation, and similar results were obtained here. However, it is probably possible to tune the parameters in the 1D multgrid method such that it performs better, but we have not done this, since the results are good.

The single stage methods are included as a point of reference. In two dimensions, the multigrid preconditioner is much better than in one, and the condition numbers are roughly 10% higher than the exact 1D case.

Clearly our preconditioner is not optimal with respect to the number of quadrature nodes in the Runge–Kutta scheme. In fact, it seems that the growth in the condition number with respect to s is slightly above quadratic. But remember that an increase in the number of nodes by one implies an increase in the order of two. So an increase in the number of iterations may be subordinate to the decrease in the number of required

TABLE 2
Estimates of the condition number $\kappa(\mathcal{BA})$ for various implicit Runge–Kutta schemes.

Method	Exact 1D	MG 1D	MG 2D
Gauss 1 node (Implicit midpoint)	1.0	1.9	1.1
Gauss 2 nodes	4.8	9.1	5.2
Gauss 3 nodes	11.8	22.0	12.8
Gauss 4 nodes	22.4	41.2	24.2
Gauss 5 nodes	37.2	67.8	40.0
Gauss 6 nodes	56.6	102.0	60.4
RadauIA 1 node	1.0	1.9	1.1
RadauIA 2 nodes	1.5	2.6	1.5
RadauIA 3 nodes	7.2	13.6	7.8
RadauIIA 1 node (Implicit Euler)	1.0	1.9	1.1
RadauIIA 2 nodes	6.8	12.9	7.4
RadauIIA 3 nodes	15.4	29.0	16.8
RadauIIA 4 nodes	27.1	50.1	29.3
RadauIIA 5 nodes	41.2	75.2	44.3
RadauIIA 6 nodes	57.5	104	61.5
LobattoIIIC 2 nodes	1.3	1.9	1.4
LobattoIIIC 3 nodes	11.2	21.4	12.2
LobattoIIIC 4 nodes	21.6	40.6	23.5

time steps. This is what we will investigate in the next example. Notice also that in [19] we studied Gauss–Seidel preconditioners, which have far better performance with respect to s.

5.3. Iteration count for a test problem. Finally, we compare the actual CPU time for a given test problem. We solve (1)–(3), with a source term f such that the exact solution is

$$u(x,y,t) = \sin(\omega_x x)\sin(\omega_y y)\sin(\omega_t \pi t)e^{xy}, \quad (\omega_x,\omega_y,\omega_t) = (\pi,\pi,20.5\pi).$$

The high number of oscillations in time is to generate a certain degree of complexity in time. We solve the problem from $T = 0.0$ to $T = 1.0$ on the unit square. In space we discretize using linear finite elements. Both the element size h and the time step δt are chosen such that the error is of order 10^{-5}, measured in the following discrete $L^2(0,T;L^2)$ norm:

$$\|e\|_{L^2(0,T,L^2)} = \left(\sum_{k=0}^{n} e(t_k)^T M e(t_k)\,\delta t\right)^{1/2},$$

where $e(t_k)$ is the error vector at nodal points at time t_k, M is the mass matrix, and n is the number of time steps. In space we discretized using 2^{16} bilinear finite elements over the entire domain.

The preconditioner is one geometric multigrid V-cycle, and the linear system is solved using the GMRES method with restart after five search vectors (for RadauIIA and Gauss with one node we used conjugated gradients) with the stopping criterion that $\|r_k\| < 10^{-7}$. The result is displayed in Table 3. Notice that the residual is evaluated only before the restart. This means that the system is possibly overiterated, but the computational time is in general smaller due to the high cost of evaluating the residual every iteration.

TABLE 3

The total computational time measured in minutes (wct), the time spent to solve the linear system (lst), and the average number of iterations (for all time steps) for solving the heat equations (1)–(3) *for schemes with various numbers of stages. The discretization parameters are chosen such that the error from the discretizations is approximately* 10^{-5}. *The preconditioner is approximated using one multigrid V-cycle. For comparison, the same simulations are also done without a preconditioner.*

		With MG prec			No prec		
Method	δt	wct	iter	lst	wct	iter	lst
Gauss 1 node	1/5000	87.4	2.9	22.7	92.8	18.4	27.50
Gauss 2 nodes	1/142	11.0	14.7	6.4	169	1184	163.6
Gauss 3 nodes	1/47	8.5	24.4	5.7	280	3565	623
Gauss 4 nodes	1/26	8.8	34.8	6.3	—	>5000	—
RadauIIA 1 node	5.0e-7	3e4	3	9085	2e4	2	1607
RadauIIA 2 nodes	1/440	34.5	15	20.2	277	607	262
RadauIIA 3 nodes	1/68	14.0	30	10.0	488	3684	817
RadauIIA 4 nodes	1/30	12.5	46.8	9.8	—	>5000	—
LobattoIIIC 2 nodes	1/5000	387.5	14.7	227.5	530	105	515
LobattoIIIC 3 nodes	1/140	33.5	37.1	25.5	953	2902	938
LobattoIIIC 4 nodes	1/43	23.4	64.5	19.5	—	>5000	—

The results are computed on a Linux machine with an AMD Athlon 64 2.2GHz processor with 2GB RAM using the C++ library Diffpack.

The number of iterations stated is an average over all the iterations. The three-node Gauss scheme is the fastest, while the four-node Gauss is almost equally fast. In general, the higher-order schemes are more efficient than the lower-order schemes for all three classes of Runge–Kutta schemes.

The cost of doing a preconditioned GMRES iteration is approximately 2.5 the cost of doing an unpreconditioned iteration. The cost of evaluating the right-hand side for each time step is equal to several preconditioned iterations due to the complexity of the function f. This is noticeable for the schemes in which the number of required iterations is low, where the time to solve the linear systems (lst) is much smaller than the total solution time (wct).

Note that this is only an indication that high-order Runge–Kutta methods may be efficient compared to low-order methods. But which solver is the most efficient depends on several properties such as the regularity of the solution, the required accuracy, the implementation, and others.

6. Final remarks. In this paper we have shown that the systems arising from fully implicit Runge–Kutta schemes applied to parabolic equations can be efficiently preconditioned with a block preconditioner. The block preconditioner is block diagonal and has blocks that are standard elliptic preconditioners. It is well known that such preconditioners are order optimal when constructed by, e.g., multigrid or domain decomposition methods. The proposed preconditioner is proven to be order optimal when constructed from an order-optimal preconditioner for a backward Euler scheme.

In several numerical experiments we have demonstrated that the condition numbers for the preconditioned systems are bounded. We have also seen that higher-order methods are beneficial, when using efficient preconditioners, even for problems with relatively fast dynamics and modest accuracy requirements.

Acknowledgments. The authors are grateful to Professors Brynjulf Owren and Syvert P. Nørsett of the Department of Mathematical Science, Norwegian University of Science and Technology, Norway, for many useful discussions.

REFERENCES

[1] D. N. Arnold, R. S. Falk, and R. Winther, *Preconditioning discrete approximations of the Reissner–Mindlin plate model*, Math. Model. Numer. Anal., 31 (1997), pp. 517–557.

[2] U. M. Ascher and L. R. Petzold, *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, SIAM, Philadelphia, 1998.

[3] I. Babuska and A. K. Aziz, *Survey lectures on the mathematical foundations of the finite element method*, in The Mathematical Foundations of the Finite Element Method with Applications to Partial Differential Equations, A. K. Aziz, ed., Academic Press, New York, 1972, pp. 1–359.

[4] J. Bergh and J. Löfström, *Interpolation Spaces*, Springer-Verlag, Berlin, 1976.

[5] J. H. Bramble and P. H. Sammon, *Efficient higher order single step methods for parabolic problems: Part* I, Math. Comp., 35 (1980), pp. 655–677.

[6] L. C. Evans, *Partial Differential Equations*, Grad. Stud. Math. 19, American Mathematical Society, Providence, RI, 1998.

[7] B. Gustafsson and W. Kress, *Deferred correction methods for initial value problems*, BIT, 41 (2001), pp. 986–995.

[8] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations* II. *Stiff and Differential-Algebraic Problems*, Springer-Verlag, Berlin, 1996.

[9] E. Haug and R. Winther, *A domain embedding preconditioner for the Lagrange multiplier system*, Math. Comp., 69 (1999), pp. 65–82.

[10] H. Hu, *A projective method for rescaling a diagonally stable matrix to be positive definite*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 1255–1263.

[11] F. Iavernaro and F. Mazzia, *Solving ordinary differential equations by generalized Adams methods: Properties and implementation techniques*, Appl. Numer. Math., 28 (1998), pp. 107–126.

[12] L. O. Jay, *Inexact simplified Newton iterations for implicit Runge-Kutta methods*, SIAM J. Numer. Anal., 38 (2000), pp. 1369–1388.

[13] L. O. Jay and T. Braconnier, *A parallelizable preconditioner for the iterative solution of implicit Runge-Kutta type methods*, J. Comput. Appl. Math., 111 (1999), pp. 63–76.

[14] J. Van lent and S. Vandewalle, *Multigrid methods for implicit Runge–Kutta and boundary value method discretizations of PDEs*, SIAM J. Sci. Comput., 27 (2005), pp. 67–92.

[15] K.-A. Mardal and T. K. Nilssen, *Reuse of standard preconditioners for higher-order time discretizations of parabolic PDEs*, J. Numer. Math., 14 (2006), pp. 103–122.

[16] K.-A. Mardal, J. Sundnes, H. P. Langtangen, and A. Tveito, *Block preconditioning and systems of PDEs*, in Advanced Topics in Computational Partial Differential Equations—Numerical Methods and Diffpack Programming, Lect. Notes Comput. Sci. Eng. 33, H. P. Langtangen and A. Tveito, eds., Springer-Verlag, Berlin, 2003, pp. 199–236.

[17] K.-A. Mardal and R. Winther, *Uniform preconditioners for the time dependent Stokes problem*, Numer. Math., 98 (2004), pp. 305–327.

[18] T. K. Nilssen, *Weakly Positive Definite Matrices*, Research Report 2005–07, Simula Research Laboratory, Lysaker, Norway, 2005. Available online at http://www.simula.no/departments/scientific/publications/Nilssen.2005.1.

[19] G. A. Staff, K.-A. Mardal, and T. K. Nilssen, *Preconditioning of fully implicit Runge-Kutta schemes for parabolic PDEs*, Model. Identif. Control, 27 (2006), pp. 109–123.

[20] V. Thomée, *Galerkin Finite Element Methods for Parabolic Problems*, Springer-Verlag, Berlin, 1997.

SIAM J. SCI. COMPUT.
Vol. 29, No. 1, pp. 376–396
© 2007 Society for Industrial and Applied Mathematics

A SEMICLASSICAL COUPLED MODEL FOR THE TRANSIENT SIMULATION OF SEMICONDUCTOR DEVICES*

PHILIPPE BECHOUCHE† AND LAURENT GOSSE‡

Abstract. We consider the approximation of a microelectronic device corresponding to an $n^+ - n - n^+$ diode consisting of a channel flanked on both sides by two highly doped regions. This is modeled through a system of equations: ballistic for the channel and drift-diffusion elsewhere. The overall coupling stems from the Poisson equation for the self-consistent potential. We propose an original numerical method for its processing, being realizable, explicit in time, and nonnegativity preserving on the density. In particular, the boundary conditions at the junctions express the continuity of the current and do not destabilize the general scheme. Efficiency is shown by presenting results on test cases of some practical interest.

Key words. drift-diffusion equation, Schrödinger–Poisson equation, WKB ansatz, open quantum system, Robin boundary condition

AMS subject classifications. 76W05, 65J10

DOI. 10.1137/060655262

1. Introduction and modeling.

1.1. Preliminaries. We develop a fast, efficient, and stable approach to 1D semiconductor component computations directly processing any kind of kinetic equation; only moment-type approximations are to be used, in "high-field regime," in order to cope with "submicronic standards."

It is a well-known fact nowadays that a whole hierarchy of mathematical models ranges from quantum mechanics to transport equations and eventually leads to "fluid-like" models; see, e.g., [7, 20, 33]. Two main tools exist for performing the first step, namely, WKB and Wigner measures methods; see [3, 5, 6, 30, 31, 32]; the correct way to interpret the WKB classical system is through a Vlasov equation; see [22, 26]. It is then at this "mesoscopic scale" that one introduces collisions between electrons and impurities (i.e., phonons, quantum representations of the crystal's vibrations) which thermalize the carriers population. This is rendered through a collision operator put on the right-hand side of the kinetic equation, which is now meant to relax toward a local equilibrium called by analogy with Boltzmann's theory of rarefied gases, [14]. The passage from kinetic models to coarser ones always stems from prescribing a peculiar dependence of the kinetic density in its velocity variable; such a simplification can sometimes be justified by means of convenient time/space rescalings; consult [33] for a survey.

At this level, we take advantage of former results, mainly [13, 11, 2]; in these papers, coexistence of several qualitatively different areas has been clearly evidenced inside a standard n^+nn^+ diode [10], under reasonable voltage biases. To each of these regions can be associated a particular rescaling of the kinetic transport model;

*Received by the editors March 27, 2006; accepted for publication (in revised form) August 30, 2006; published electronically February 5, 2007. The work of both authors was partially supported by the EEC project HYKE (Hyperbolic and kinetic equations) under grant HPRN-CT-2002-00282.
http://www.siam.org/journals/sisc/29-1/65526.html

†Departamento de Matemática Aplicada, Universidad de Granada, Facultad de Ciencias, 18071 Granada, Spain (phbe@ugr.es).

‡Istituto per le Applicazioni del Calcolo (sezione di Bari), Via G. Amendola 122, 70126 Bari, Italy (l.gosse@ba.iac.cnr.it).

roughly speaking, heavily doped zones (the [illegible] and the [illegible]) exhibit "low-field" diffusive behavior, whereas inside the [illegible] "high-field" ballistic transport takes place. To some extent, a schematic but acceptable picture goes as follows: carriers move inside the channel according to monokinetic solutions of the Vlasov–Poisson equation (because they accelerate, no shock is likely to appear, at least at steady state), whereas they are "nearly Maxwellian" inside the source/drain regions, where their position densities can be computed by means of a drift-diffusion approximation. We obviously expect to observe stabilization of transient dynamics leading to convergence onto some steady time-asymptotic regime (see, e.g., [15, 34] for direct steady-state computations). We stress that our goal here is to devise a time-marching strategy.

Such a program clearly raises the issue of time-stepping compatibility—and this whatever the boundary conditions on each side of the channel. Indeed, any drift-diffusion equation, being of parabolic type, asks for a CFL restriction like $\Delta t = O(\Delta x^2)$, whereas the ballistic regime is essentially of a hyperbolic nature (signals travel at finite speed), thus requiring $\Delta t = O(\Delta x)$ instead. Of course, imposing the smallest value of Δt everywhere, despite ensuring stability, would result in an overwhelming smearing of the sharp profiles inside the channel. We thus selected a mixed implicit/explicit numerical strategy as described in section 4.1. The channel is flanked on each side by low-field areas, and thus deriving appropriate boundary conditions connecting these regions was necessary to keep oscillations from developing. It has been observed that Dirichlet conditions are not conclusive as they do not allow for the scheme to relax toward a steady state; however, imposing the current's continuity at the junctions leads to a nonlinear Robin boundary condition [1, 17, 18, 19] for the drift-diffusion equations: it also remains stable and easy to implement; see section 4.3. Obviously Dirichlet conditions are imposed on each side of the device and for the Poisson equation.

The remainder of the paper is devoted to numerical experiments as an outcome of the aforementioned general framework. A careful rescaling of a well-known GaAs device has been performed in section 3 and practical results are displayed in section 5. In particular, we observe stabilization in time toward a steady profile together with a satisfying residue decay. We also show the way current settles inside the device for large times; a reasonably constant current is the main goal in order to achieve reliable intensity/voltage diagrams (see especially Figure 6 in section 6, where an original derivation leading to a real improvement of ballistic currents is proposed). We are not aware of any time-marching scheme able to stabilize onto steady states with perfectly flat currents for such models. Presently, we found that Ohm's law, stipulating a linear dependence between currents and biases, is satisfied as soon as potential drops are strong enough. This was to be somehow expected as we developed the present approach for a high-field setting in which low biases do not really fit.

1.2. From quantum models to kinetic equations. We briefly recall the different levels for the modeling of charge carriers; the finest one here is nonrelativistic quantum mechanics. One considers thus a mixed state, i.e., a statistical repartition of quantum states $\psi_\ell \in \mathbb{C}$ indexed by the (quantized) energy levels solution of the 1D Schrödinger–Poisson equation in the parabolic band approximation (m^* stands for the effective mass of the electron in the material under consideration):

$$i\hbar\partial_t\psi_\ell + \frac{\hbar^2}{2m^*}\partial_{xx}\psi_\ell = eU(t,x)\psi_\ell, \qquad \varepsilon_s\partial_{xx}U = e(\varrho - D). \tag{1.1}$$

The wave function $\psi_\ell(t,x)$ corresponds to an energy level E_ℓ of the homogeneous Hamiltonian; $\hbar$, e, and m^* stand for Planck's constant, the charge, and the effective mass of the electron in the considered material, respectively. The Hartree equation renders a mean-field approximation of the Coulombian forces, with ε_0 being the dielectric permittivity of the medium. D is the given doping concentration and $\varrho(t,x)$ is the electrons' density, which reads

$$\varrho(t,x) = \sum_\ell F(E_\ell)|\psi_\ell(t,x)|^2,$$

with, e.g., $F(E) = (1+\exp((E-E_F)/k_B\theta_0))^{-1}$ in the case of Fermi–Dirac statistics, where E_F is usually called the Fermi energy.[1] θ_0 is the ambient temperature, and k_B is the Boltzmann constant. The sum ranges therefore over the whole spectrum of the Hamiltonian consisting of an orthonormal basis of L^2; hence (1.1) is indeed a system of equations all coupled by the Poisson term U. With the next level of modeling, the Wigner transform comes into play:

$$w(t,x,p) = \frac{1}{2\pi}\int_{\mathbb{R}} z\left(t, x+\frac{\hbar y}{2}, x-\frac{\hbar y}{2}\right)\exp(iyp).dy,$$

where

$$z(t,x,y) = \sum_\ell F(E_\ell)\psi_\ell(t,x)\overline{\psi_\ell(t,y)}.$$

One observes at once that

$$\varrho(t,x) = \int_{\mathbb{R}} w(t,x,p).dp = z(t,x,x).$$

The Wigner transform solves the (pseudodifferential) Wigner equation,

$$\partial_t w + \frac{p}{\hbar m^*}\partial_x w - e\Theta[U]f = 0,\ \Theta[V] = \frac{i}{\hbar}\left\{U\left(t, x+\frac{\hbar}{2i}\partial_x\right) - U\left(t, x-\frac{\hbar}{2i}\partial_x\right)\right\},$$

completed by the Poisson equation as in (1.1). Passing to the (semi)classical limit $\hbar \to 0$, $w \rightharpoonup f(t,x,\xi) \geq 0$ in the sense of measures [31], and the pseudodifferential operator Θ becomes the standard Vlasov term $\partial_x U.\partial_\xi f$. It is usually at this intermediate level that electron-phonon collisions are introduced in the models (see, however, [3, 23]) by means of an integral operator on the right-hand side:

$$\partial_t f + \xi\partial_x f + \frac{e}{m^*}\partial_x U\partial_\xi f = \mathcal{Q}(f), \qquad \varepsilon_s\partial_{xx}U = e(\varrho - D), \qquad E = -\partial_x U. \tag{1.2}$$

We shall not enter the details of $\mathcal{Q}(f)$ since different models exist such as BGK (Bhatnagar–Gross–Krook) and Fokker–Planck; however, they share the interesting property of admitting the same Maxwellian distribution, a fact that will be extensively used in the next section. At this level of description the electron and current densities are the moments of f:

$$\varrho(t,x) = \int_{\mathbb{R}} f(t,x,\xi)d\xi, \qquad J(t,x) = \int_{\mathbb{R}} \xi f(t,x,\xi)d\xi.$$

[1] It corresponds to an energy level for which the occupation probability is exactly one half.

2. Low- and high-field scalings inside an n^+nn^+ device. Our 1D problem combines two distinct zones according to the importance which is given to the phenomena rendered through $\mathcal{Q}$ in (1.2); purely ballistic transport means $\mathcal{Q} \equiv 0$. However, we begin with the opposite case and mainly follow [11] in our presentation. Let us first recall some conventions: $L > 0$ stands for the characteristic device's length, whereas $\varrho_0 \geq 0$ and $\theta_0 = k_B T/m^* \geq 0$ for an average value of the doping profile $D(x)$ and the lattice temperature. $U_{th} = k_B T/e$ is the thermal potential. Four (possibly) different velocity magnitudes appear: $v_{th} = \sqrt{\theta_0}$ is the [illegible]. If τ_0 stands for an average value of the relaxation times,[2] $\tau(x)$ inside the device, $v_{relax} = L/\tau_0$, is the [illegible]. Similarly, if $[\Phi]$ is the bias applied at the contacts, one defines the reference magnitude for the [illegible] as

$$v_{drift} = \tau_0 \frac{e[\Phi]}{m^* L}, \tag{2.1}$$

whereas the [illegible] reads

$$v_{bal} = \sqrt{\frac{2e[\Phi]}{m^*}}.$$

In a 1D framework, it is reasonable to render collisions by mean of a linear relaxation time operator; thus we can start from the following kinetic equation to describe the dynamic of the electrons in the device:

$$\partial_t f + \xi \partial_x f - \frac{e}{m^*} E(t,x) \partial_\xi f = \frac{1}{\tau}\Big(\mathcal{M}(\theta_0)\varrho(x) - f\Big), \tag{2.2}$$

where the Maxwellian reads

$$\mathcal{M}(\theta_0) = \frac{1}{\sqrt{2\pi\theta_0}} \exp\left(-\frac{\xi^2}{2\theta_0}\right) \overset{\theta_0 \to 0}{\to} \delta,$$

where δ stands for the Dirac mass in zero; the electric field $E(t,x)$ is coupled to the Poisson equation by

$$\varepsilon_s \partial_{xx} U = e(\varrho(x) - D(x)), \qquad E(t,x) = -\partial_x U. \tag{2.3}$$

[illegible] 1. Starting from here, we assume that junctions are rendered through sharp interfaces inside the device. Some quantities, such as the temperature, for example, will obviously be discontinuous across them. This modeling has already been widely used; see, for instance, [2, 8, 11, 12, 15].

2.1. Drift-diffusion approximation. It is convenient to normalize the doping concentration and the potential bias: $D(x) = \varrho_0 \hat{D}(\hat{x})$ and $U(t,x) = [\Phi]\hat{U}(\hat{t},\hat{x})$, where we make use of rescaled variables defined as follows:

$$x = L\hat{x}, \qquad \xi = v_{th}\hat{\xi}, \qquad t = v_{drift}\frac{\hat{t}}{L}, \qquad \tau(x) = \tau_0 \hat{\tau}(\hat{x}).$$

Moreover the kinetic density can be rewritten as

$$f(t,x,\xi) = \frac{\varrho_0}{v_{th}} \hat{f}(\hat{t},\hat{x},\hat{\xi}).$$

[2]For instance, in the case when $\mathcal{Q}$ is given by a BGK model.

The main assumption in this approximation is $v_{drift} \ll v_{th} \ll v_{relax}$; hence two small parameters naturally appear:

$$\epsilon = \frac{v_{th}}{v_{relax}}, \qquad \eta = \frac{v_{drift}}{v_{th}}, \tag{2.4}$$

which we suppose to be of the same order of magnitude.

Then "low-field" is precisely meant to be the limit $\eta \simeq \epsilon \to 0$; thus (2.2) can be rewritten as

$$\epsilon \partial_t f + \xi \partial_x f - \frac{e}{m^*} E(t,x) \partial_\xi f = \frac{1}{\epsilon \tau} (\mathcal{M}(1)\varrho - f). \tag{2.5}$$

This regime describes the flow in highly doped areas (source and drain); (2.5) admits asymptotic expansions which allow us to compute the two first moments of f in closed form. Since collisions are dominant, drift velocity is much smaller than thermal velocity, and the mean free path can be considered infinitesimal, i.e., $o(1)$. Hilbert expansion leads to

$$\partial_t \varrho + \partial_x J = 0, \qquad J = -d_n(x) \partial_x \varrho - \mu_n(x) E \varrho. \tag{2.6}$$

There, $\mu_n(x) = e\tau(x)/m^*$ is the electron mobility and $d_n(x) = \tau(x)\theta_0 = \mu_n U_{th} \geq 0$ appears to be a diffusion coefficient. In this regime, it is tacitly assumed that the solution of (2.5) is "nearly Maxwellian", i.e.,

$$f(t,x,\xi) = \varrho(t,x)\mathcal{M}_{\theta_0}(\xi) + o(\epsilon).$$

2.2. Shur–Eastman/WKB ansatz for ballistic regime. This is somehow the opposite of the drift-diffusion approximation; we consider now the following variables:

$$x = L\hat{x}, \qquad \xi = v_{bal}\hat{\xi}, \qquad t = v_{bal}\frac{\hat{t}}{L}, \qquad \tau(x) = \tau_0 \hat{\tau}(\hat{x}).$$

The kinetic density reads, accordingly,

$$f(t,x,\xi) = \frac{\varrho_0}{v_{bal}} \hat{f}(\hat{t},\hat{x},\hat{\xi}).$$

The main assumption in this approximation is $v_{th} \ll v_{bal} \simeq v_{drift}$, which leads to the limit $\beta = v_{th}/v_{bal} \to 0$, while $\alpha = v_{drift}/v_{bal} = O(1)$. Thus

$$\partial_t f + \xi \partial_x f + \frac{e}{m^*} \partial_x U . \partial_\xi f = \frac{1}{2\alpha\tau} \mathcal{Q}(f), \qquad \theta_0 \to 0, \quad \partial_{xx} U = \gamma(\varrho - D). \tag{2.7}$$

Clearly, neither Hilbert nor Chapman–Enskog expansions are likely to lead to a diffusion-like equation. Instead, one obtains formally a Vlasov–Poisson system for which the kinetic density is of the [illegible] form at zero temperature (see [35] and [2, section 5]). This can be approximated by the so-called "WKB system" as in [25],

$$\partial_t \varrho + \partial_x \left(\frac{\partial_x \varphi}{m^*} \varrho \right) = 0, \qquad \partial_t \varphi + \frac{1}{2m^*} |\partial_x \varphi|^2 + eU = 0, \tag{2.8}$$

assuming that the wave functions behave like $\sqrt{\varrho}\exp(i\varphi/\hbar)$, or equivalently, that the solution of (2.7) reads $f(t,x,\xi) = \varrho(t,x)\delta(\xi - \partial_x\varphi(t,x))$; see [31]. Of course,

$u = \frac{1}{m^*}\partial_x\varphi$ is meant to remain smooth[3] and this holds in particular for the accelerating particles $\partial_x u(t,x) \geq 0$ inside the channel. In section 5 of [2], Arnold et al. suggest that the distribution f should be of the type $\delta(\xi - E\tau)$, which boils down to assuming that $u(t,x) \simeq \tau(x)E(t,x)$, at least after some time. This is indeed what we observe in the numerical results (in constrast with [11], where E and u look quite different at steady state); see section 5.4.

3. Rescaling the device. First, let us remark that the device does not have the same physical characteristics in the channel (ballistic zone), where there is a low doping concentration relative to the source and the drain (drift-diffusion zone). This implies that the electron mobility is higher in the channel, as in the source and drain. The same occurs for the relaxation time in the same proportion. More precisely, let μ_1 be the mobility in the source and drain and μ_2 in the channel. We have

$$\frac{\mu_1}{\mu_2} = \frac{\tau_1}{\tau_2} = \frac{v_2}{v_1} \tag{3.1}$$

if we suppose that τ_1 (resp., τ_1) is proportional to L/v_1 (resp., to L/v_2). This means that if we want to see phenomena evolve at the same velocity, we have to scale the time in this proportion. Another way to explain this is if we scale the drift-diffusion equation first (with mobility μ_1); then we have to scale the ballistic equation with the same scaling obtained in the drift-diffusion equation but with the mobility μ_2.

3.1. Rescaling the drift-diffusion equation. According to (2.6), the physical drift-diffusion model reads, for $x \in [0,L]$, $t \geq 0$,

$$\varepsilon_s \partial_{xx} U = e(\varrho - D), \qquad \partial_t \varrho = \partial_x \left(\mu_1 U_{th} \partial_x \varrho - \mu_1 \varrho \partial_x U\right). \tag{3.2}$$

The first equation in (3.2) is Poisson's completed by a continuity equation. In the device's variables, the current density is given by $J(t,x) = -d_1 \partial_x \varrho + \mu_1 \varrho \partial_x U$. The first part of the current is the diffusion term ($d_1 = \mu_1 U_{th}$) and the second part is the drift term. The general way to process (3.2) starts by rescaling space and time variables according to the size of the device, and to some characteristic velocity which is to be chosen for convenience:

$$\hat{x} = \frac{x}{L}, \quad \hat{t} = \frac{t}{L/v_1}. \tag{3.3}$$

We can deduce an adimensionalized density ρ and a potential $\hat{U}$ as follows:

$$\varrho(t,x) = \varrho_0 \hat{\rho}(\hat{t},\hat{x}), \qquad U(t,x) = U_0 \hat{U}(\hat{t},\hat{x}). \tag{3.4}$$

The doping concentration is obviously treated accordingly:

$$D(x) = \varrho_0 \hat{D}(\hat{x}).$$

Here ϱ_0 stands for a mean density inside the device (obtained by, e.g., the doping concentration's arithmetic or geometric average) and U_0 is a reference value for the potential. We first rescale the Poisson equation so as to obtain

$$\frac{\varepsilon_s U_0}{e\varrho_0 L^2} \frac{\partial^2 \hat{U}}{\partial \hat{x}^2} = \hat{\rho} - \hat{D}.$$

[3] If not, breakup occurs and multibranch solutions are to be sought instead, as in [22, 26, 29].

If we denote by $[\Phi_0]$ the applied bias, the scaled [illegible] is given by

$$\gamma = \frac{\varepsilon_s[\Phi_0]}{e\varrho_0 L^2}.$$

At this level, we decide to choose $U_0 = \frac{e\varrho_0 L^2}{\varepsilon_s} = \gamma[\Phi_0]$ in order to scale the coefficient to 1. Thus the scaled Poisson equation can be rewritten as

$$\frac{\partial^2 \hat{U}}{\partial \hat{x}^2} = \hat{\rho} - \hat{D}. \tag{3.5}$$

Let us now rescale the continuity equation in (3.2); the outcome reads

$$\frac{\partial \hat{\rho}}{\partial \hat{t}} = \frac{\partial}{\partial \hat{x}} \left(\frac{\mu_1 U_{th}}{v_1 L} \frac{\partial \hat{\rho}}{\partial \hat{x}} - \frac{\mu_1 U_0}{v_1 L} \hat{\rho} \frac{\partial \hat{U}}{\partial \hat{x}} \right). \tag{3.6}$$

A relevant quantity is the ratio between the diffusion and drift current: here it is U_{th}/U_0. We observe that in general, it is not possible to simultaneously scale to unit the scaled Debye length, the drift and diffusion terms. So in the present framework, we chose to scale to unit the scaled Debye length and the scaled drift term. Let us choose v_1 such that the drift velocity coefficient scales to 1, i.e., $\frac{\mu_1 U_0}{v_1 L} = 1$. In such a way (3.6) transforms into

$$\frac{\partial \hat{\rho}}{\partial \hat{t}} = \frac{\partial}{\partial \hat{x}} \left(\frac{U_{th}}{U_0} \frac{\partial \hat{\rho}}{\partial \hat{x}} - \hat{\rho} \frac{\partial \hat{U}}{\partial \hat{x}} \right). \tag{3.7}$$

The scaled current density therefore reads

$$J(\hat{t}, \hat{x}) = -\frac{U_{th}}{U_0} \frac{\partial \hat{\rho}}{\partial \hat{x}} + \hat{\rho} \frac{\partial \hat{U}}{\partial \hat{x}}, \tag{3.8}$$

and (3.7)–(3.8) should be completed with the rescaled potential equation (3.5). A short computation with the numerical values given below shows that for high concentrations of doping, the drift term is more important than the diffusion term; for a concentration of doping of $10^{19}/m^3$, both terms are of the same order of magnitude, and for a lower concentration of doping, the diffusion term is dominant.

3.2. Rescaling the ballistic equation. Let us now scale the (2.8) with the same scaling (3.3) in space and time but with v_2 instead of v_1 and (3.4) for the density and potential. Moreover we assume that the phase scales in the following way:

$$\varphi(t, x) = \varphi_0 \hat{\varphi}(\hat{t}, \hat{x}). \tag{3.9}$$

With this scaling, (2.8) become

$$\frac{\partial \hat{\rho}}{\partial \hat{t}} + \frac{\partial}{\partial \hat{x}} \left(\frac{\varphi_0}{m^* L v_2} \frac{\partial \hat{\varphi}}{\partial \hat{x}} \hat{\rho} \right) = 0,$$

$$\frac{\partial}{\partial \hat{t}} \frac{\partial}{\partial \hat{x}} \left(\frac{\varphi_0 \hat{\varphi}}{m^* L v_2} \right) + \frac{1}{2} \frac{\partial}{\partial \hat{x}} \left(\frac{\varphi_0^2}{(m^*)^2 L^2 v_2^2} |\partial_{\hat{x}} \hat{\varphi}|^2 \right) + \frac{e U_0}{m^* v_2^2} \frac{\partial \hat{U}}{\partial \hat{x}} = 0.$$

Note that we have derived w.r.t. $\hat{x}$ the second equation. In order to work with the same time scale as in the drift-diffusion case, we have to choose v_2 such that (3.1) is verified. Let us set $u = \frac{\varphi_0}{m^* L v_2}\partial_{\hat{x}}\hat{\varphi}$. We can choose φ_0 such that $\frac{\varphi_0}{m^* L v_2} = 1$. We set for the scaled current $\hat{J} = \hat{\rho}u$. Then the scaled WKB equation (2.8) becomes

$$\frac{\partial \hat{\rho}}{\partial \hat{t}} + \frac{\partial}{\partial \hat{x}}(\hat{\rho}u) = 0, \qquad \frac{\partial}{\partial \hat{t}}u + \frac{1}{2}\frac{\partial}{\partial \hat{x}}u^2 + \frac{eU_0}{m^* v_2^2}\frac{\partial \hat{U}}{\partial \hat{x}} = 0.$$

From now on, we shall always work with scaled quantities; thus it makes sense to remove the "hats."

. . . . 2 (numerical values). Let us take the same test case as in [4, 11], that is, $m^* = 0.065 \times 9.109 \times 10^{-31} Kg$, $e = 1.602 \times 10^{-19} C$, $k_B = 1.38 \times 10^{-23} J/K$, $\varepsilon_s = 13.2 \times 8.85418 \times 10^{-12} F/m$, $T_0 = 300K$, $L = 0.8 \times 10^{-6} m$. Moreover in the classical zone, we take for the doping concentration $D = 10^{24}/m^3$, and for the electron mobility $\mu_1 = \tau_1 e/m^* = 0.75$. In the ballistic zone, $D = 2 \times 10^{21}/m^3$ and $\mu_2 = \tau_2 e/m^* = 4$. These values imply that $\varrho_0 = 5 \times 10^{23}/m^3$, $U_0 = \frac{e\varrho_o L^2}{\varepsilon_s} = 438.6$. Concerning the velocity, $v_1 = \frac{\mu_1 U_0}{L} = 4.111 \times 10^8 m/s$ which implies $v_2 = v_1\mu_1/\mu_2 = 7.71 \times 10^7 m/s$. The scaled diffusion coefficient is $d_1 = \frac{U_{th}}{U_0} = 5.9 \times 10^{-5}$ (and the scaled drift coefficient is $\mu = 1$). Finally we have $eU_0/(m^* v_2^2) = 0.2$.

3.3. The kinetic equation. We finally rewrite the kinetic model (2.2); let us rescale the variables as follows:

$$x = L\hat{x}, \qquad \xi = v_{th}\hat{\xi}, \qquad t = v\frac{\hat{t}}{L}, \qquad \tau(x) = \tau_0 \hat{\tau}(\hat{x}),$$

where v is a velocity that is to be set according to the scaling which has been chosen. Moreover the adimensioned kinetic density can be rewritten as

$$f(t, x, \xi) = \frac{\varrho_0}{v_{th}}\hat{f}(\hat{t}, \hat{x}, \hat{\xi}).$$

As usual, we define an adimensioned density ρ and a potential Φ as follows:

$$\varrho(t, x) = \varrho_0 \rho(\hat{t}, \hat{x}), \qquad U(t, x) = \Phi_0 \Phi(\hat{t}, \hat{x}). \tag{3.10}$$

With this notation the adimensioned form of (2.2) reads

$$\partial_{\hat{t}}\hat{f} + \left(\frac{v_{th}}{v}\right)\hat{\xi}\cdot\partial_{\hat{x}}\hat{f} - \left(\frac{U_0}{U_{th}}\frac{v_{th}}{v}\right)\partial_{\hat{x}}\Phi\cdot\partial_{\hat{\xi}}\hat{f} = \frac{v_{relax}}{v}\Big(\mathcal{M}(\theta_0)\hat{\varrho}(\hat{x}) - \hat{f}\Big). \tag{3.11}$$

According to Remark 2, v can be chosen to be, e.g., between $100 v_{th}$ and $1000 v_{th}$.

4. A staggered scheme for the coupled model. Hereafter we always consider working with the rescaled equations derived in the preceding section.

4.1. A semi-implicit discretization. We now aim at describing in full detail the numerical recipe we propose for solving the following adimensionalized problems:

$$\text{(CL)} \qquad \begin{cases} \partial_t \varrho + \partial_x J = 0, \\ J = -d\partial_x \varrho - \varrho \partial_x U, \ \varrho(t, x = 0) = 1, \end{cases} \qquad x \in]0, x_L[;$$

$$\text{(B)} \qquad \begin{cases} \partial_t \varrho + \partial_x J = 0, \\ \partial_t u + u\partial_x u + \partial_x U = 0, \quad J = \varrho u, \end{cases} \qquad x \in]x_L, x_R[;$$

$$\text{(CR)} \qquad \begin{cases} \partial_t \varrho + \partial_x J = 0, \\ J = -d\partial_x \varrho - \varrho \partial_x U, \ \varrho(t, x = 1) = 1, \end{cases} \qquad x \in]x_R, 1[,$$

where all these PDEs are coupled through the Poisson equation:

$$\text{(P)} \qquad \begin{cases} -\partial_{xx}U = \varrho - D(x), \\ U(t, x=0) = 0, \quad U(t, x=1) = U_0 \le 0, \end{cases} \qquad x \in]0,1[.$$

We are given initial data $\varrho(t=0,.) = \varrho_0 \ge 0$, $D(x) \ge 0$ and $u(t=0,.)$; we usually take zero. We shall develop implementation of boundary conditions in $x = x_L, x_R$, rendering the junctions in section 3.3. First, let us say that (CL) and (CR) will be solved implicitly, whereas (B) will be processed explicitly in time. Another important feature is that ϱ and u will not be known at the same abscissae (hence the use of "staggered"). The computational domain is the half-stripe $\mathbb{R}^+ \times]0,1[$, which is parametrized by $\Delta x = \frac{1}{N} > 0$; we define

$$x_j = j\Delta x = \frac{j}{N}, \qquad j = 0, \dots, N.$$

We also define the following domains:

$$\begin{cases} \mathcal{B} = \{x_Q, x_{Q+1}, \dots, 1 - x_Q\}, \qquad x_Q = x_q + \frac{\Delta x}{2}, 1 - x_Q = x_{N-q} - \frac{\Delta x}{2}, \\ \mathcal{C}_L = \{x_0, x_2, \dots, x_q\}, \qquad \mathcal{C}_R = \{x_{N-q}, \dots, x_N\}. \end{cases}$$

Thus $\vec{\varrho}_{n=0} = (\varrho_0(x_j))_{j=0,\dots,N} \in \mathbb{R}_+^{N+1}$. The boundary conditions read, at the discrete level, $\varrho_{0,n} = \varrho(t_n, 0)$ and $\varrho_{N,n} = \varrho(t_n, x_N) = \varrho(t_n, 1)$.

4.2. Choosing the time-step Δt. We select a standard centered discretization for (CL) and (CR) that reads (n the time index, j the space one)

$$\begin{aligned} &\varrho_{j,n+1} - \Delta t \tfrac{d(\varrho_{j+1,n+1} - \varrho_{j,n+1}) - d(\varrho_{j,n+1} - \varrho_{j-1,n+1})}{\Delta x^2} \\ &\qquad = \varrho_{j,n} + \tfrac{\Delta t}{\Delta x}\Big(\Big\langle \mathbf{F}_{j+\frac12,n}, \mathbf{R}_{j+\frac12,n} \Big\rangle_{\mathbb{R}^2} - \Big\langle \mathbf{F}_{j-\frac12,n}, \mathbf{R}_{j-\frac12,n} \Big\rangle_{\mathbb{R}^2}\Big), \end{aligned} \tag{4.1}$$

where the following notations have been used:

$$\mathbf{F}_{j+\frac12,n} := \left\{ \max\left(0, \frac{U_{j+1,n} - U_{j,n}}{\Delta x}\right), \min\left(0, \frac{U_{j+1,n} - U_{j,n}}{\Delta x}\right) \right\},$$

$$\mathbf{R}_{j+\frac12,n} := \{\varrho_{j,n}, \varrho_{j+1,n}\}.$$

Clearly, $\langle , \rangle_{\mathbb{R}^2}$ stands for the standard $\mathbb{R}^2$ scalar product. At this stage, we tacitly assume that $\vec{\varrho}_n$ is known, and we already have a way to produce the two borderline values $\varrho_{q+1,n+1}$ and $\varrho_{N-q-1,n+1}$. Thus, the scheme boils down to inverting a tri-diagonal $q \times q$ matrix, reading

$$\mathbf{M}_q = \begin{pmatrix} 1 + \frac{\alpha d\Delta t}{\Delta x^2} & -\frac{d\Delta t}{\Delta x^2} & 0 & \cdots & \\ -\frac{d\Delta t}{\Delta x^2} & 1 + \frac{2d\Delta t}{\Delta x^2} & -\frac{d\Delta t}{\Delta x^2} & 0 & \\ 0 & \ddots & \ddots & \ddots & \\ & & 0 & -\frac{d\Delta t}{\Delta x^2} & 1 + \frac{\beta d\Delta t}{\Delta x^2} \end{pmatrix},$$

where α, β can be 1 or 2 depending on whether (4.1) is meant to approximate (CL) or (CR) because of the boundary conditions. Anyway, it is clearly unconditionally invertible since it is strictly diagonally dominant; thus the discretization (4.1) will

remain stable as long as its right-hand side will be a convex combination, i.e., for a time-step $\Delta t = O(\Delta x)$. In order to produce $\varrho_{q+1,n+1}$ and $\varrho_{N-q-1,n+1}$, we solve (B) by means of an explicit time-marching scheme already used in [24, 25], except that we set it up with a well-balanced strategy, as in [21] for the handling of the electric field. More precisely, from [24],

$$\varrho_{j,n+1} = \varrho_{j,n} - \frac{\Delta t}{\Delta x}\Big(\Big\langle \mathbf{A}_{j+\frac12,n}, \mathbf{R}_{j+\frac12,n}\Big\rangle_{\mathbb{R}^2} - \Big\langle \mathbf{A}_{j-\frac12,n}, \mathbf{R}_{j-\frac12,n}\Big\rangle_{\mathbb{R}^2}\Big), \tag{4.2}$$

with $j = q, \dots, N-q-1$ and

$$\mathbf{A}_{j+\frac12,n} := \left\{\max\left(0, u_{j+\frac12,n}\right), \min\left(0, u_{j+\frac12,n}\right)\right\}.$$

Now the velocity values $u_{j+\frac12,n}$ are sought on the staggered computational grid by means of a simple, well-balanced upwind scheme:

$$u_{j+\frac12,n+1} = u_{j+\frac12,n} - \frac{\Delta t}{\Delta x}\Big(\mathcal{F}(u_{j+\frac12,n}, \tilde u_{j+\frac32,n}) - \mathcal{F}(\tilde u_{j-\frac12,n}, u_{j+\frac12,n})\Big). \tag{4.3}$$

$\mathcal{F}(u,v) = \frac12(\max(u,0)^2 + \min(v,0)^2)$ is the standard upwind scheme, and the modified values $\tilde u_{j-\frac12,n}$, $\tilde u_{j+\frac32,n}$ are computed according to the steady-state equation $u\partial_x u + \partial_x U = 0$ far from resonance points; see the appendix and [21] for details. We know from [21, 24, 25] that this approach for solving (B) will be stable for a time-step $\Delta t = O(\Delta x)$.

Hence the algorithm proceeds as follows: Knowing all $\vec\varrho_n$ and $u_{j+\frac12,n}$ ($j = q, \dots, N-q-1$) at some time $t^n = n\Delta t$, we first update $u_{j+\frac12,n+1}$ inside $\mathcal{B}$ with (4.3) in order to deduce $\varrho_{j,n+1}$ by means of (4.2). Then, knowing $\varrho_{q+1,n+1}$ and $\varrho_{N-q-1,n+1}$, we can invert the implicit scheme (4.1) as soon as we have chosen a correct boundary condition to render the junction between the diffusive/ballistic zones.

4.3. A (nonlinear) Robin boundary condition. Let us mention first that junctions have been put at staggered abscissae, $x_{q+\frac12}, x_{N-q-1-\frac12}$, which correspond to the velocity unknowns. At these points, imposing Dirichlet boundary conditions leads to some kind of inconsistency because of a somewhat redundant definition of the current J between, e.g., (CL) and (B). Numerically, it produced some spurious oscillations eventually leading to general instability. Moreover in order to observe a constant current everywhere inside the device at steady state, it sounds reasonable to impose its value at the junctions between diffusive and ballistic areas. That is, at $x_{q+\frac12}$,

$$u_{q+\frac12,n+1} = -d\frac{\varrho_{q+1,n+1} - \varrho_{q,n+1}}{\Delta x \varrho_{q+\frac12,n+1}} - \frac{U_{q+1,n+1} - U_{q,n+1}}{\Delta x},$$

which implies

$$-d\frac{\varrho_{q+1,n+1} - \varrho_{q,n+1}}{\Delta x} = \varrho_{q+\frac12,n+1}\Big(u_{q+\frac12,n+1} + \frac{U_{q+1,n+1} - U_{q,n+1}}{\Delta x}\Big),$$

which we can approximate by means of

$$\begin{aligned} -d\frac{\varrho_{q+1,n+1} - \varrho_{q,n+1}}{\Delta x} \\ = \big(\theta\varrho_{q+1,n+1} + (1-\theta)\varrho_{q,n+1}\big)\Big(u_{q+\frac12,n+1} + \frac{U_{q+1,n+1} - U_{q,n+1}}{\Delta x}\Big), \end{aligned} \tag{4.4}$$

for $\theta \in [0,1]$. Hence one recognizes on the left-hand side of (4.4) the unknown flux which appears in (4.1) and the $\mathbf{M}_q$ matrix, and the right-hand side contains only values which can be deduced from the outcome of the explicit schemes used inside $\mathcal{B}$. Actually, only the average value $\varrho_{q+\frac{1}{2},n+1}$ has to be arbitrarily chosen as, e.g., the arithmetic average $\frac{1}{2}(\varrho_{q+1,n+1}+\varrho_{q,n+1})$, $\varrho_{q,n+1}$, or $\varrho_{q+1,n+1}$, corresponding to $\theta=\frac{1}{2}$, $\theta=0$, and $\theta=1$, respectively.

REMARK 3. This choice of boundary conditions corresponds to solving the following problem for the drift-diffusion equation (CL):

$$\begin{cases} \partial_t \varrho + \partial_x J = 0, & J = -d\partial_x \varrho - \varrho \partial_x U, \qquad x \in]0, x_L[, \\ \varrho(t, x=0) = D(x=0), & -\partial_x \ln(\varrho)(t, x=x_L) = u + \partial_x U, \end{cases}$$

where $u(t, x=x_L)$ is deduced from (B). This is a mixed (nonlinear) Robin-type boundary condition for a strictly parabolic equation. The well-posedness theory for problems of this flavor is quite recent; it started with the paper by Evans [17] and was then developed further in [1, 18, 19].

Of course, we presented everything in $x = x_{q+\frac{1}{2}}$, but the right junction $x = x_{N-q-1-\frac{1}{2}}$ is completely equivalent (with maybe a different choice of θ, however).

4.4. Some stability results. First of all, we wish to state a result from [19] which we adapt to our situation involving only a uniformly elliptic linear operator.

THEOREM 4.1 (see Goldstein and Goldstein [19]). Let $\beta_j(\xi)$ [illegible] be two nondecreasing Lipschitz continuous functions on $\mathbb{R}$ [illegible] and consider the problem

$$\begin{cases} \partial_t u = \partial_{xx} u, & x, t \in (0,1) \times \mathbb{R}^+, \\ (-1)^j \partial_x u(., x) = \beta_j(u), & x = j \in \{0,1\}, \\ u(t=0, x) = f(x) \in L^1(0,1), & x \in (0,1), \end{cases}$$

[illegible] u [illegible] $(0,1) \times [0,T]$ [illegible] $T > 0$ [illegible] $u(t,.) \in L^\infty(0,1)$ and $\partial_t u(t,.) \in L^2(0,1)$.

Clearly in order to be complete, a full study of the coupled problem (even if neglecting the Poisson coupling equation) would be necessary before concluding the stability of the whole model given by (CL)–(B)–(CR). However, the former theorem is a stepping stone in this direction. An interesting feature of our numerical approach (4.1)–(4.2) lies in a nonnegativity-preserving property for the density.

THEOREM 4.2. Let $\theta = 0$, (4.4) $\max_{j,n}(|u_{j+\frac{1}{2},n}| + |U_{j+1,n+1} - U_{j,n+1}|/\Delta x)\Delta t \leq \Delta x$ and $\vec{\varrho}_n \in (\mathbb{R}_+)^N$ [illegible] (4.1)–(4.2) [illegible] $\vec{\varrho}_{n+1} \in (\mathbb{R}_+)^N$.

Actually, we never used (4.1)–(4.2) with $\theta = 0$ because it appears to be the less accurate choice. However, we were unable to prove Theorem 4.2 for $\theta > 0$.

PROOF. The proof proceeds in three steps, namely, the explicit part, the implicit part, and the occurrences at the junctions.

- We first observe that, thanks to the prescribed CFL condition, the explicit parts of both schemes (4.1) and (4.2) turn out to be convex combinations of the values showing up on their right-hand sides. This has been precisely shown in previous papers; see, e.g., Proposition 4.6 in [24] or section 4.3 in [25].
- The implicit part of (4.1) is given by $\mathbf{M}_q$ which is an M-matrix in the terminology of [9], that is, its entries satisfy

$$m_{i,i} > 0, \qquad m_{i,j\neq i} \leq 0, \qquad |m_{i,i}| > \sum_{j \neq i} |m_{i,j}|.$$

So let us suppose we solve $\mathbf{M}_q \vec{x} = \vec{b}$, where $\vec{b}$ has nonnegative components; it is a classical fact that $\vec{x}$ will have only nonnegative components too. For completeness we recall its (quick) proof: Let i_0 correspond to the smallest component of $\vec{x}$, $x_{i_0} := \min(\vec{x})$. There holds

$$m_{i_0,i_0} x_{i_0} = b_{i_0} - \sum_{j \neq i_0} m_{i_0,j} x_j \geq b_{i_0} - x_{i_0} \sum_{j \neq i_0} m_{i_0,j},$$

which implies

$$x_{i_0} \geq \frac{b_{i_0}}{m_{i_0,i_0} + \sum_{j \neq i_0} m_{i_0,j}} \geq 0.$$

- It remains to observe that choosing $\theta = 0$ in (4.4) means that the boundary condition shows up only in the implicit part of (4.1). Let us focus on the left junction: Inside $\mathbf{M}_q$, we have $\alpha = 2$ and $\beta = 1$. Condition (4.4) gives us another term which reads $u_{q+\frac{1}{2},n+1} + \frac{U_{q+1,n+1} - U_{q,n+1}}{\Delta x}$, but thanks to the CFL condition, we know that $\mathbf{M}_q$ remains diagonal dominant even on this line. Hence the preceding computation guarantees the nonnegativity-preserving property, and we are done. □

These computations are based on convexity arguments; hence they somehow open the way to compactness for the whole scheme. However, we do not yet have any precise theory at the continuous level for (CL)–(B)–(CR). This might constitute a direction for further investigation.

4. Actually the choice of the parameter θ is quite a delicate task when considering its importance in stabilizing the numerical scheme on a steady-state regime with flat currents (as can be seen on the numerical figures). Here is what we implemented in order to produce the numerical results of section 5: First, we distinguished between the left junction where the solution is Lipschitz continuous, and the right, where it generally involves a discontinuity. Then we observed that it was not possible to code directly $\partial_x \ln(\varrho) = u + \partial_x U$, simply because it would require applying the change of variable $v = \ln(\varrho)$ while solving the drift-diffusion equation, which is not easy. Hence it becomes necessary to linearize the $\ln(.)$ function, and this is where the $\theta \in [0,1]$ parameter comes into play through the midpoint rule. In the case when this linearization occurs in a smooth location, $\theta = \frac{1}{2}$ gives a second-order accuracy and is therefore sufficient. In the case when a discontinuity shows up, the upwind value is often the right choice, that is, the weight should go to the point from where the signal comes from (generally, it is the left side).

5. Numerical validation with doping $5 \times 10^{22} m^{-3}$. In this section, we display some numerical results at steady state for a standard GaAs device as already considered in [11, 2, 10, 35]. We took a discontinuous doping profile equal to 1 inside $x \in [0, 0.25[\cup]0.75, 1]$ and equal to 0.002 elsewhere. We iterated the marching schemes presented in section 3.2 until residues on $\vec{\varrho}_n$ decayed lower than 10^{-15}. The resulting parameters are $d_1 = 0.00118$ and $eU_0/(m^* v_2^2) = 4$. We stress that neither drift-diffusion nor mixed currents are flat in the whole computational domain at steady state; so in order to derive the current–voltage relations, we computed arithmetic averages of the results coming from both discretizations; we shall, however, propose a more interesting derivation in section 5.5 by relying on well-balanced ideas. 255 points have been used to sample the $[0,1]$ interval, and the time step is chosen adaptively in order to minimize the mixed scheme's numerical diffusion. The main difference

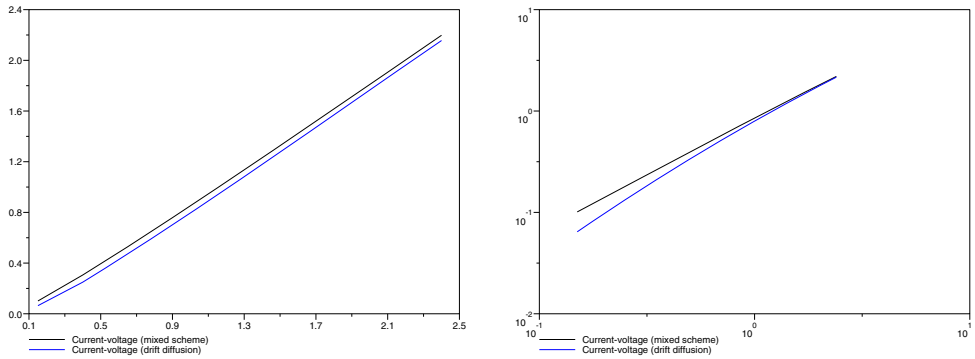


FIG. 1. *Current–voltage relation with doping* $5 \times 10^{22} m^{-3}$ *(left, normal scale; right, log scale).*

between both curves in Figure 1 lies in slightly lower values for the drift-diffusion currents. Apart from that, both approaches appear to agree. However, we are about to see that despite the current–voltage relations may look similar, the densities and velocities are rather different.

. . . . 5. It may seem reasonable to think about directly simulating (2.2), at least to validate the outcome of our mixed scheme since in one dimension, the computational cost still can be considered reasonable. Indeed, performing the rescalings as in section 3 leads to the following problem:

$$v\partial_t f + v_{th}\xi\partial_x f + v_{th}\frac{U_0}{U_{th}}\partial_x U\partial_\xi f = v_{relax}(\mathcal{M}(\theta_0)\varrho - f),$$

supplemented by (3.5) and with v being a free parameter. Actually, it is also possible to evaluate all the parameters involved: $\epsilon \simeq 0.09$, $\eta \simeq 0.3$ (tending to zero in section 3.1) and $\beta \simeq 0.1$, $\alpha \simeq 2$ (being $o(1)$ and $O(1)$, respectively, in section 3.2). However, there is at least one obstacle in the way: the problem still involves two different time scales and two different Maxwellian distributions. Hence a direct upwind discretization of (3.11) is unstable because some boundary conditions are needed at each junction: a way out may consist of numerically implementing formulas (16)–(17) in [8] (see also [16] for related questions). The schemes should also be asymptotic preserving in the sense of [28, 27]. Finally, we notice from [12] that direct kinetic simulations deliver results quite similar to drift-diffusion ones.

5.1. Weak bias: $[\Phi] = -0.2$. Essentially, the main difference between the mixed approach and the global drift-diffusion solution is that the last one always produces continuous solutions, whereas jumps show up in the mixed approach. This is clearly demonstrated by the results in Figure 2, which hold even if the corresponding diffusion coefficient turns out to be relatively small. Velocities and densities turn out to be rather different for both approaches, especially when compared to the Poisson potential and the resulting electric field. We can observe that our prescribed boundary condition for the junction works fine for continuous solutions, but is unable to produce a flat current close to a shock. This can come from the fact that it involves an average value, $\varrho_{q+\frac{1}{2},n+1}$, which should perhaps be derived more precisely. However, outside the junctions, currents look satisfying.

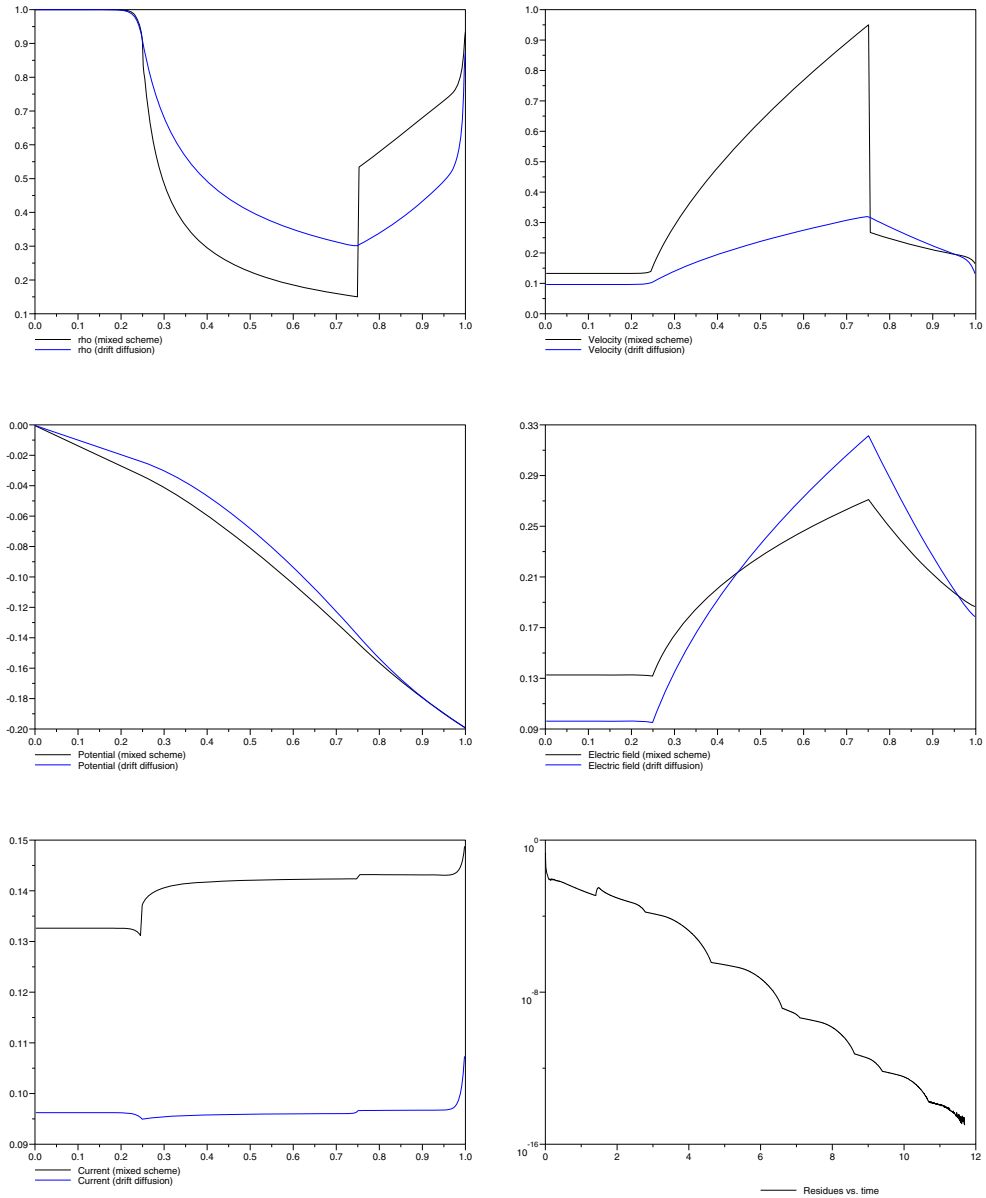


FIG. 2. *Carriers density, velocity, potential, electric field, current, and residues for an applied bias of* $[\Phi] = -0.2$.

5.2. Moderate bias: $[\Phi] = -0.35$. In this case, the higher bias accelerates electrons; the outcome of both schemes is presented in Figure 3. The residue decay is satisfying and the steady-state currents are reasonably flat. Densities and velocities are again very different for both approaches, quantitatively and qualitatively.

5.3. Higher bias: $[\Phi] = -0.5$. In this last case, the higher bias tends to flatten more the "mixed current" (see Figure 4) despite some sort of boundary layer on the

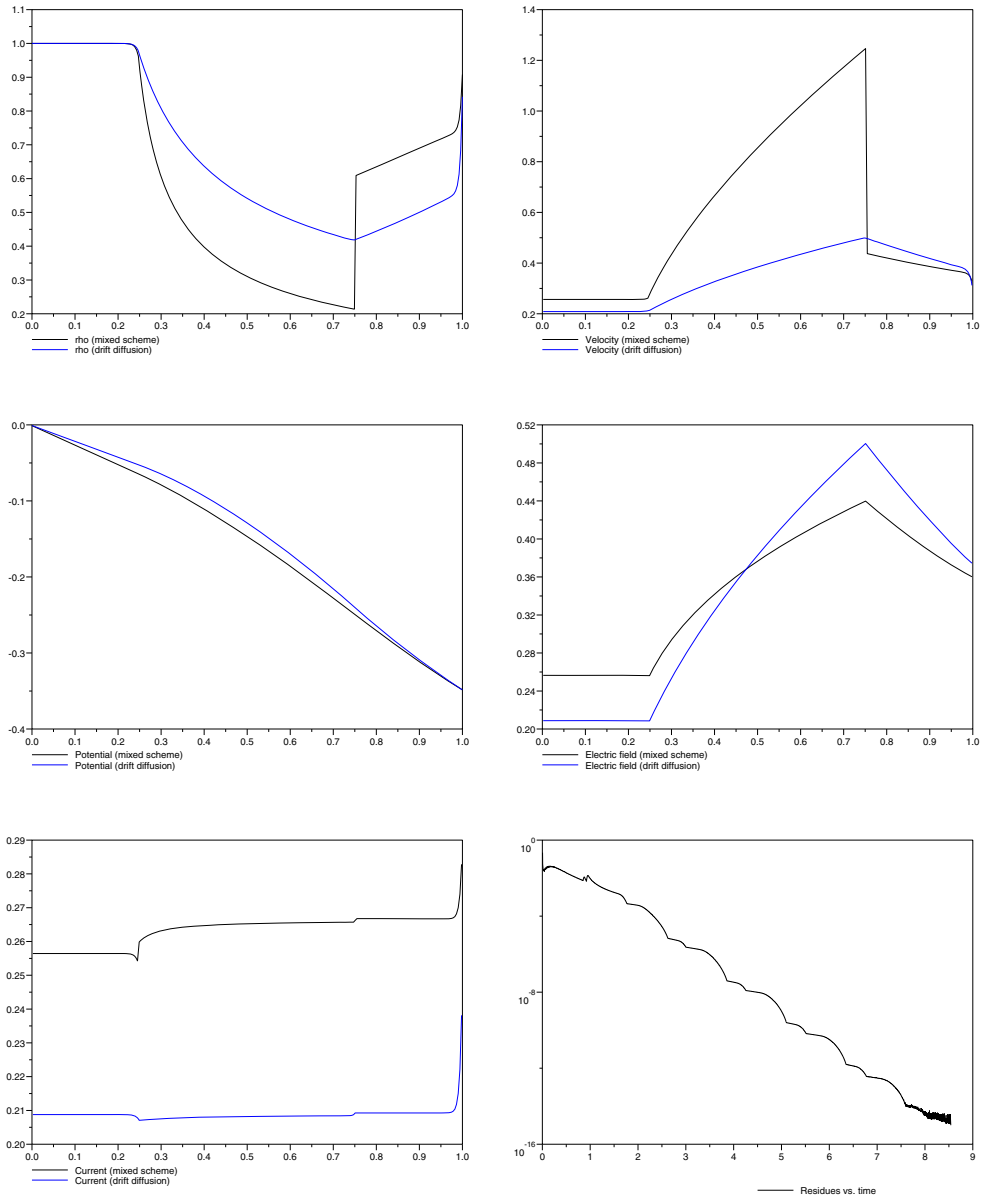


FIG. 3. *Carriers density, velocity, potential, electric field, current, and residues for an applied bias of* $[\Phi] = -0.35$.

right side. The potentials and the electric fields are quite similar for both approaches even if velocities and densities are still very different. Convergence in time is obtained in half the time it asked for with $[\Phi] = -0.2$.

5.4. Electric field and velocities inside the device. For completeness, we finally plot the fraction $u(x)/E(x)$ at steady state since this quantity measures the conductivity as a function of the applied bias; see Figure 5. We observe that,

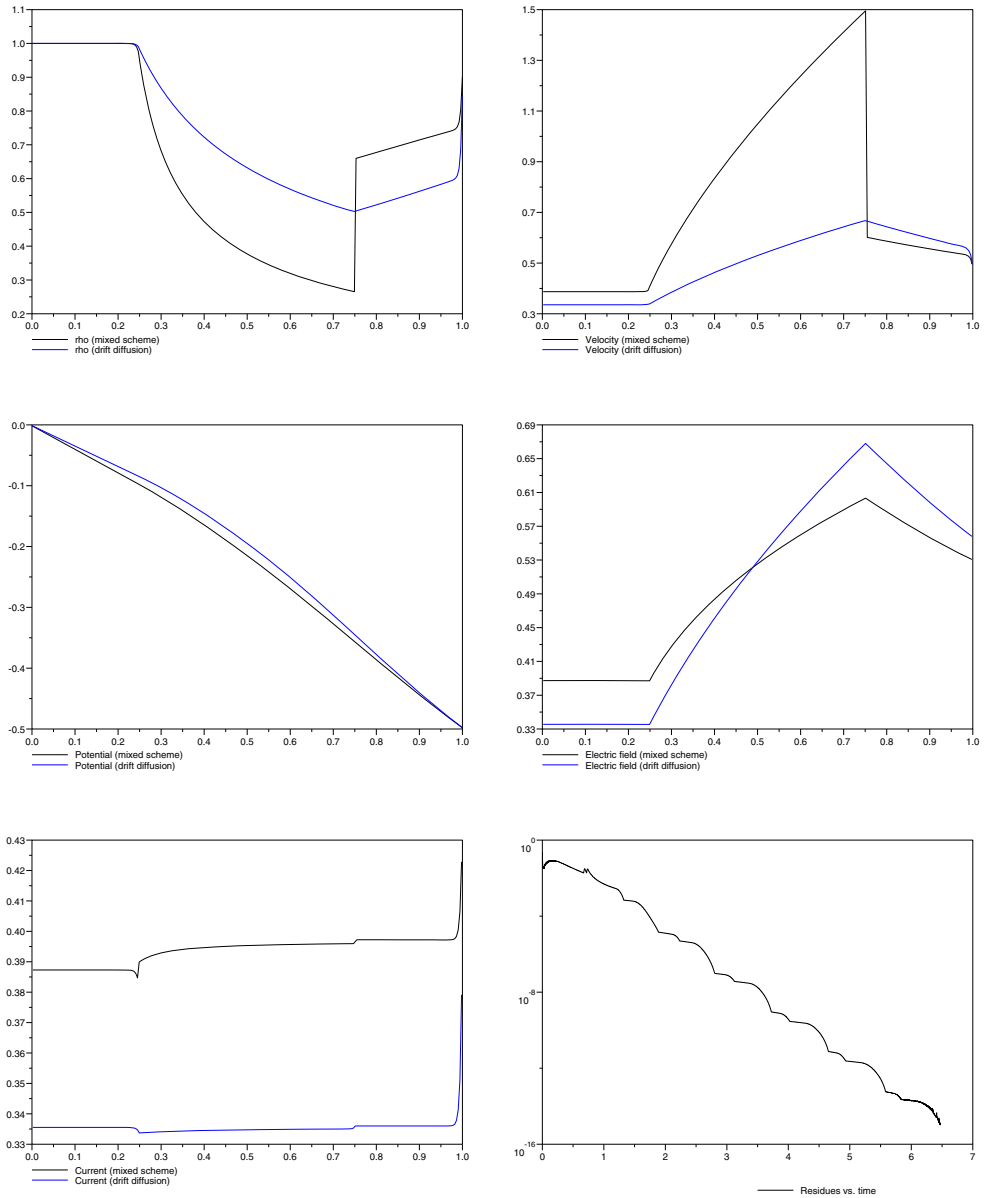


FIG. 4. *Carriers density, velocity, potential, electric field, current, and residues for an applied bias of* $[\Phi] = -0.5$.

independently of the bias, this fraction is practically constant in the zones where the drift-diffusion algorithm is used (this reflects the smallness of the diffusion coefficient for high doping). However, it falls down strongly inside the channel even if increasing the bias improves things. Junctions are slightly visible, but the corresponding discontinuities do not seem to have strong effects on the global picture.

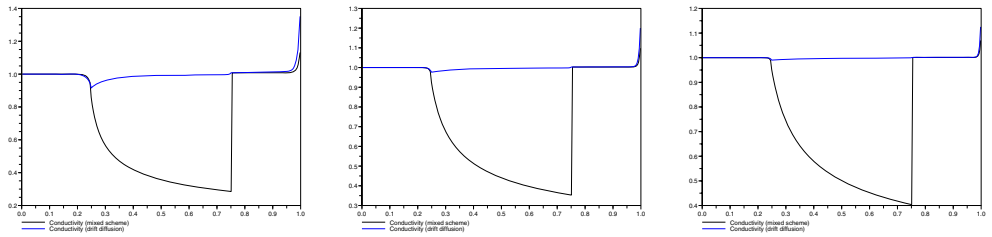


FIG. 5. *Steady-state fraction* $u(x)/E(x)$ *for applied biases* $[\Phi] = -0.2, -0.35, -0.5$ *(left to right).*

5.5. A different way to compute currents in the device. We finally propose a new way to compute steady-state currents inside the device, owing to the fact that a well-balanced discretization is used to solve the ballistic part (B) of the problem; see the appendix for details. Actually, assuming (B) is stationary, one derives easily that the following equality holds:

$$J_{bal}(x) = \varrho(x)u(x) = \varrho(x)\sqrt{-2(U(x) - U(x_L)) + u(x_L)^2}. \tag{5.1}$$

In the same way, one actually can compute stationary drift-diffusion:

$$J_{DD} = -\varrho(d\partial_x \ln(\varrho) + \partial_x U). \tag{5.2}$$

In Figure 6, we compare the outcome of the usual arithmetic averages with formulas (5.1)–(5.2). One clearly sees that inside the ballistic channel, the resulting currents are , which to the authors' knowledge has never been obtained before. Even the drift-diffusion currents are rendered in a slightly better way: for $[\Phi] = -0.35$, there is no jump around $x = x_R$. Nevertheless, this is not meant to improve the treatment of the junctions where small discontinuities still appear. We believe that only a discretization of type (6.1) can solve this issue.

6. Conclusion and outlook. We presented in this paper a first attempt to develop a robust mixed approach to transient semiconductor computations relying on a simple time-marching strategy. The two main obstructions to circumvent were the choice of a time step which would ensure the compatibility of both the parabolic (drift-diffusion) and hyperbolic (ballistic) areas, and the derivation of a stable boundary condition at the junctions between the source/drain and the channel. For completeness, we display here the current–voltage relation for a doping value of $10^{24}m^{-3}$, see Figure 7. With a diffusion coefficient so low, both curves look very similar.

There exist several ways of extending such an approach: one could be the inclusion of the energy bands, which is more accurate than the simple "effective mass" approximation made in (1.1). Indeed, such a simplification makes the whole model very close to the so-called "Drude theory of conduction" (see [3]), which is known to be quite crude. In order to improve the overall accuracy, an efficient algorithm to compute the Bloch decomposition in one dimension has already been studied in [26]. It may also be interesting to use the sophisticated "multibranch" routines of [22, 26] in this present context to achieve more precise simulations inside the ballistic channel. A routine able to handle phonons has been recently proposed in [23].

From the perspective of the "well-balanced" aspect of things, the big step would be to succeed in working with a good set of variables which would prevent us from

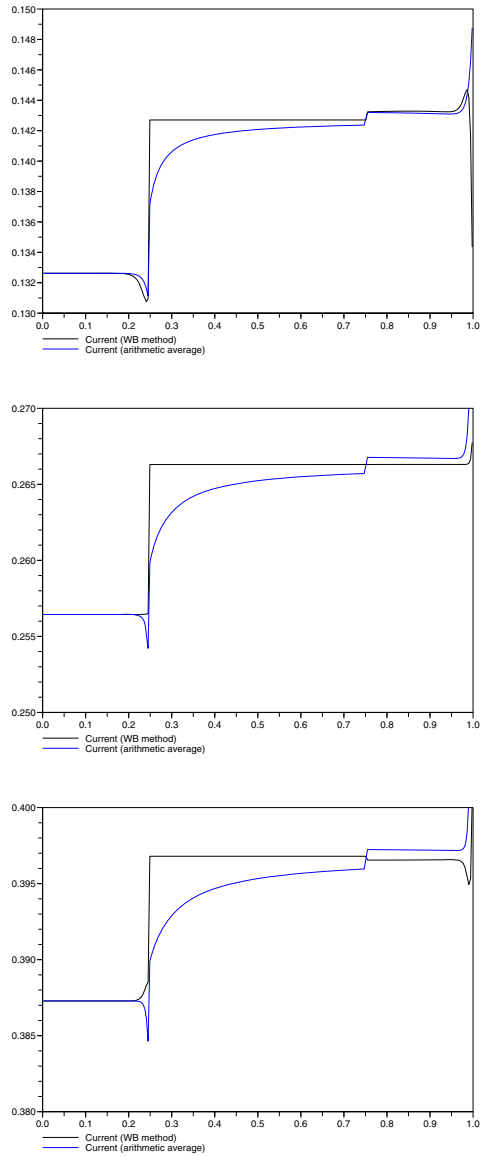


FIG. 6. *Steady-state currents for the mixed scheme with* $[\Phi] = -0.2, -0.35, -0.5$ *(top to bottom).*

linearizing at the junctions (recall the delicate choice of the θ parameter; see Remarks 3 and 4). The first idea could be to replace the ϱ variable in (CL) and (CR) with $z = \ln \varrho$. However, the price to pay is to make these drift-diffusion problems nonlinear; indeed, in the z variable the equations become

$$\text{(6.1)} \qquad \begin{cases} \partial_t z + d\big(\partial_{xx} z + |\partial_x z|^2\big) + \partial_{xx} U + \partial_x U \ \partial_x z = 0, \\ \exp(z)(\cdot, x = 0) = 1, \quad \partial_x z(\cdot, x = x_L) = u + \partial_x U, \end{cases}$$

where the left junction is rendered through the last boundary condition. However, the nonlinear term is first order; hence it asks only for a mild CFL restriction.

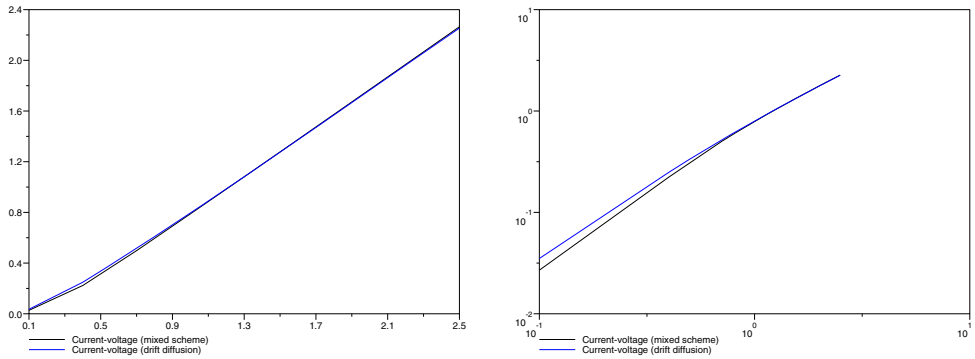


FIG. 7. *Current–voltage relation with strong doping (left, normal scale; right, log scale).*

Appendix. Well-balanced scheme for (B) in a nutshell. In this short section, we follow [21] in order to quickly explain how to manage the forced Burgers equation appearing in (B) in such a way to approximate steady states as best as possible. Let us suppose that $u(t,.) > 0$ holds (this appears to be completely fair in the kind of steady regimes we have in mind to reproduce); we can rewrite the scalar balance law the following way:

$$\partial_t u + \partial_x\Big(\frac{u^2}{2} + U(t,x)\Big) = 0.$$

This clearly shows that steady states with $u \neq 0$ are given by the conservation law $\frac{u^2}{2} + U \equiv C^t \in \mathbb{R}$. At this point, it is easy to include the Poisson potential U in the upwind fluxes in order to maintain this property; thus we obtain the modified fluxes,

$$\mathcal{F}(u_l, u_r; U_l, U_r) = \frac{1}{2}\Big(\max(\tilde{u}_l, 0)^2 + \min(\tilde{u}_r, 0)^2\Big),$$

with the upwinded values,

$$\tilde{u}_l = \sqrt{u_l^2 - 2(U_r - U_l)} \geq 0, \qquad \tilde{u}_r = -\sqrt{u_r^2 + 2(U_r - U_l)} \leq 0.$$

Acknowledgments. The second author warmly greets the Mathematics Department of University of Granada with thanks for its hospitality (and Prof. J. Soler in particular).

REFERENCES

[1] J. ARRIETA, A. CARVALHO, AND A. RODRÍGUEZ-BERNAL, *Parabolic problems with nonlinear boundary conditions and critical nonlinearities*, J. Differential Equations, 156 (1999), pp. 376–406.

[2] A. ARNOLD, J. A. CARRILLO, I. GAMBA, AND C. W. SHU, *Low and high field scaling limits for the Vlasov- and Wigner-Poisson-Fokker-Planck systems*, Transport Theory Statist. Phys., 30 (2001), pp. 121–153.

[3] N. W. ASHCROFT AND N. D. MERMIN, *Solid-State Physics*, Holt, Rinehart, and Winston, New York, 1976.

[4] H. U. Baranger and J. W. Wilkins, *Ballistic structure in the electron distribution function of small semiconducting devices: General features and specific trends*, Phys. Rev. B, 36 (1987), pp. 1487–1502.
[5] P. Bechouche, N. Mauser, and F. Poupaud, *Semiclassical limit for the Schrödinger-Poisson equation in a crystal*, Comm. Pure Appl. Math., 54 (2001), pp. 851–890.
[6] N. Ben Abdallah and P. Degond, *The Child–Langmuir law for the Boltzmann equation of semiconductors*, SIAM J. Math. Anal., 26 (1995), pp. 364–398.
[7] N. Ben Abdallah and P. Degond, *On a hierarchy of macroscopic models for semi-conductors*, J. Math. Phys., 37 (1995), pp. 3306–3333.
[8] N. Ben Abdallah, P. Degond, and I. Gamba, *Coupling one-dimensional time-dependent classical and quantum transport models*, J. Math. Phys., 43 (2002), pp. 1–24.
[9] A. Berman and R. Plemmons, *Nonnegative Matrices in the Mathematical Sciences*, Academic Press, New York, 1979.
[10] K. Blotekjaer, *Transport equations for electrons in two-valley semiconductors*, IEEE Trans. Electron. Devices, 17 (1970), pp. 38–47.
[11] J. A. Carrillo, I. Gamba, and C. W. Shu, *Computational macroscopic approximations to the 1-D relaxation time kinetic system for semiconductors*, Phys. D, 146 (2000), pp. 289–306.
[12] C. Cercignani, I. Gamba, G. W. Jerome, and C. W. Shu, *Device benchmark comparisons via kinetic, hydrodynamic and high-field models*, Comput. Methods Appl. Mech. Engrg., 181 (2000), pp. 381–392.
[13] C. Cercignani, I. M. Gamba, and C. D. Levermore, *A drift-collision balance for a Boltzmann–Poisson system in bounded domains*, SIAM J. Appl. Math., 61 (2001), pp. 1932–1958.
[14] P. Degond, *Mathematical modelling of microelectronics semiconductor devices*, in Some Current Topics on Nonlinear Conservation Laws, AMS/IP Stud. Adv. Math. 15, L. Zhiao and Z. Xin, eds., AMS, Providence, RI, 2000, pp. 77–110.
[15] P. Degond and A. El Ayyadi, *A coupled Schrödinger drift-diffusion model for quantum semiconductor device simulations*, J. Comput. Phys., 191 (2002), pp. 222–259.
[16] P. Degond, S. Jin, and L. Mieussens, *A smooth transition model between kinetic and hydrodynamic equations*, J. Comput. Phys., 209 (2005), pp. 665–694.
[17] L. C. Evans, *Regularity properties for the heat equation subject to nonlinear boundary constraints*, Nonlinear Anal., 1 (1977), pp. 593–602.
[18] A. Favini, G. Goldstein, J. Goldstein, and S. Romanelli, *Nonlinear boundary conditions for nonlinear second order operators on $C[0,1]$*, Arch. Math., 76 (2001), pp. 391–400.
[19] G. Goldstein and J. Goldstein, *Smoothing for nonlinear parabolic equations with nonlinear boundary conditions*, J. Math. Anal. Appl., 213 (1997), pp. 422–443.
[20] F. Golse and F. Poupaud, *Limite fluide des équations de Boltzmann des semi-conducteurs pour une statistique de Fermi-Dirac*, Asymptotic Anal., 6 (1992), pp. 135–160.
[21] L. Gosse, *A well-balanced flux splitting scheme designed for hyperbolic systems of conservation laws with source terms*, Comput. Math. Appl., 39 (2000), pp. 135–159.
[22] L. Gosse, *Using K-branch entropy solutions for multivalued geometric optics computations*, J. Comput. Phys., 180 (2002), pp. 155–182.
[23] L. Gosse, *Multiphase semiclassical approximation of the one-dimensional harmonic crystal.* I. *The periodic case*, J. Phys. A, 39 (2006), pp. 10509–10521.
[24] L. Gosse and F. James, *Numerical approximation of linear one-dimensional conservation equations with discontinuous coefficients*, Math. Comp., 69 (2000), pp. 987–1015.
[25] L. Gosse and F. James, *Convergence results for an inhomogeneous system arising in various high frequency approximations*, Numer. Math., 90 (2002), pp. 721–753.
[26] L. Gosse and P. A. Markowich, *Multiphase semiclassical approximation of an electron in a one-dimensional crystalline lattice* – I. *Homogeneous problems*, J. Comput. Phys., 197 (2004), pp. 387–417.
[27] L. Gosse and G. Toscani, *Space localization and well-balanced schemes for discrete kinetic models in diffusive regimes*, SIAM J. Numer. Anal., 41 (2003), pp. 641–658.
[28] S. Jin, *Efficient asymptotic-preserving (AP) schemes for some multiscale kinetic equations*, SIAM J. Sci. Comput., 21 (1999), pp. 441–454.
[29] S. Jin, X. T. Li, J. G. Wohlbier, and J. H. Booske, *An Eulerian method for computing multi-valued solutions of the Euler-Poisson equations and applications to wave breaking in klystrons*, Phys. Rev. E, 70 (2004), 016502.
[30] J. B. Keller, *Semiclassical mechanics*, SIAM Rev., 27 (1985), pp. 485–504.
[31] P.-L. Lions and T. Paul, *Sur les mesures de Wigner*, Rev. Mat. Iberoamericana, 9 (1993), pp. 553–618.

[32] P. Markowich, C. A. Ringhofer, and C. Schmeiser, *Semiconductor Equations*, Springer, New York, 1990.
[33] F. Poupaud, *Mathematical theory of kinetic equations for transport modelling in semiconductors*, in Advances in Kinetic Theory and Computing, Ser. Adv. Math. Appl. Sci. 22, B. Perthame, ed., World Sci. Publ., River Edge, NJ, 1994, pp. 141–168.
[34] C. Ringhofer, *Computational methods for semiclassical and quantum transport in semiconductor devices*, Acta Numer., 3 (1997), pp. 485–521.
[35] M. S. Shur and L. F. Eastman, *Near ballistic transport in GaAs devices at 77K*, Solid State Electr., 24 (1991), pp. 11–18.

SIAM J. SCI. COMPUT.
Vol. 29, No. 1, pp. 397–414
© 2007 Society for Industrial and Applied Mathematics

TIME-STEP-SIZE-INDEPENDENT CONDITIONING AND SENSITIVITY TO PERTURBATIONS IN THE NUMERICAL SOLUTION OF INDEX THREE DIFFERENTIAL ALGEBRAIC EQUATIONS*

CARLO L. BOTTASSO†, OLIVIER A. BAUCHAU‡, AND ALBERTO CARDONA§

Abstract. We propose a simple preconditioning for the equations of motion of constrained mechanical systems in index three form. The scaling transformation is applied to the displacement-velocity-multiplier and to the reduced displacement-multiplier forms. The analysis of the transformed system shows that conditioning and sensitivity to perturbations become independent of the time step size, as in the case of well-behaved ordinary differential equations. The new scaling transformation is simple to implement and does not require the rewriting of the system equations as other approaches do. The theoretical analysis is confirmed by numerical examples.

Key words. differential algebraic equations, constraints, Lagrange multipliers, multibody systems, high index

AMS subject classification. 65L05

DOI. 10.1137/050638503

1. Introduction.

It is well known that the amplification of small errors and perturbations in the solution of high index differential algebraic equations (DAEs) causes severe numerical difficulties. For example, Petzold and Lötstedt have shown in [1] that index three DAEs for constrained mechanical (multibody) systems are severely ill conditioned for small time step sizes when discretized using backward differentiation formulas (BDFs). Their analysis indicates that, unless corrective actions are taken, the condition number of the iteration matrix is $\mathcal{O}(h^{-3})$. Furthermore, errors propagate in the displacement, velocity, and multiplier fields at rates which are shown to be $\mathcal{O}(h^{-1})$, $\mathcal{O}(h^{-2})$, and $\mathcal{O}(h^{-3})$, respectively. A related perturbation analysis that deals with the propagation of the error in the satisfaction of the constraint equations is described in Arnold [2], which also considers systems with friction. These results indicate that errors will grow very rapidly as the time step size is reduced, preventing in practice the use of time refinement procedures, and imposing a very tight tolerance on the solution of the nonlinear discrete equations. Furthermore, the ill conditioning of the linear system does not favor the use of iterative solvers.

Several methods have been proposed in the literature to address these problems. For example, the system can be recast in index two form, by replacing the displacement level constraints with the velocity level ones. While the index two form is more robust to perturbations [2], this approach suffers from the well-known drift effect, which calls for further adhoc corrective actions.

*Received by the editors August 20, 2005; accepted for publication (in revised form) August 7, 2006; published electronically February 13, 2007.
http://www.siam.org/journals/sisc/29-1/63850.html

†Dipartimento di Ingegneria Aerospaziale, Politecnico di Milano, Via La Masa 34, 20156 Milano, Italy (carlo.bottasso@polimi.it, http://www.aero.polimi.it/∼bottasso).

‡D. Guggenheim School of Aerospace Engineering, Georgia Institute of Technology, 270 Ferst Drive, Atlanta, GA 30332-0150 (olivier.bauchau@ae.gatech.edu, http://www.ae.gatech.edu/∼obauchau).

§Cimec-Intec, Universidad Nacional del Litoral – Conicet, Güemes 3450, 3000 Santa Fe, Argentina (acardona@intec.unl.edu.ar).

An alternative approach is to rewrite the governing equations so as to include, together with the position level constraints, their derivatives, as in the Gear, Leimkuhler, and Gupta (GGL) [3] and embedded projection (EP) [4] methods. These methods, however, require additional multipliers which imply a somewhat increased problem size and, hence, a possibly higher computational cost.

Petzold and Lötstedt [1] discuss a simple scaling transformation of the index three governing equations which yields a condition number $\mathcal{O}(h^{-2})$ and an improvement of one order in the errors for all solution fields. Although the sensitivity to perturbations is reduced with respect to the unscaled problem, difficulties can still be expected in practice.

In this paper we propose a solution to the conditioning and error propagation problems that is based on a left and right preconditioning. The left preconditioning amounts to a scaling of the dynamic equilibrium and constraint equations, while the right preconditioning amounts to a scaling of the unknowns. The proposed procedure is trivially implemented in an existing code and does not require the rewriting of the equations of motion as in the GGL and EP methods.

The new scaling transformation is applied first to the equations in the standard displacement-velocity-multiplier form, using a scaling of the sole Lagrange multipliers. A similar scaling was previously considered in Cardona and Géradin [5] in the context of the Hilbert, Hughes, and Taylor (HHT) scheme. The analysis of the present paper shows that errors propagate in the displacement, velocity, and multiplier fields at rates equal to $\mathcal{O}(h^0)$, $\mathcal{O}(h^{-1})$, and $\mathcal{O}(h^0)$, respectively, while the condition number is $\mathcal{O}(h^{-1})$. Next, it is shown that if the velocities are scaled together with the Lagrange multipliers, one can achieve perfect time step size independence ($\mathcal{O}(h^0)$) for error propagation and conditioning.

Finally, we analyze the displacement-multiplier form, obtained by static condensation of the velocities at each time step. This form of the equations is more computationally interesting than the full form for applications denoted by a large number of degrees of freedom, since it leads to the solution of smaller linear problems. Even in this case, the preconditioning of the reduced form leads to perfect time step size independence for both the conditioning and the error propagation in the displacement-multiplier solution variables.

These results indicate that index three DAEs are as easy to integrate as well-behaved ordinary differential equations (ODEs), once they are recast in one of the $\mathcal{O}(h^0)$ forms.

The paper is organized as follows. In section 2 we formulate the equations of motion of multibody systems in index three three-field form for BDF-type schemes, and we derive their sensitivity to perturbations and conditioning. Preconditioning by scaling of the discretized equations is presented in section 3; the left preconditioning strategy is detailed in section 3.1, while the left and right preconditioning is described in section 3.2. The classical pendulum problem in section 3.3 numerically confirms the results of the analysis and concludes this section. The two-field form of the problem is discussed in section 4, the left preconditioning is described in section 4.2, the left and right preconditioning is given in section 4.2, and the pendulum problem verifies the analysis in section 4.3. Finally, a more complex contact-impact problem for a flexible multibody system is presented in section 5 to show the importance of scaling in complex problems involving time step size refinement. The paper closes with section 6, which reports the main conclusions and findings of this work.

2. Sensitivity to perturbations and conditioning of multibody system equations. The governing DAEs for multibody systems are written as

$$\boldsymbol{u}' = \boldsymbol{v}, \tag{1a}$$

$$\boldsymbol{v}' = \boldsymbol{f} + \boldsymbol{G}\boldsymbol{\lambda}, \tag{1b}$$

$$\boldsymbol{\phi} = \boldsymbol{0}, \tag{1c}$$

where (1a) are the kinematic equations, (1b) represent the equations of dynamic equilibrium, $\boldsymbol{u}$ are generalized coordinates, $\boldsymbol{v}$ are generalized velocities, $\boldsymbol{\lambda}$ are Lagrange multipliers that enforce the constraints (1c), and finally $\boldsymbol{G}^T = \boldsymbol{\phi}_{,\boldsymbol{u}}$ is the constraint Jacobian. Following [1], we have considered for simplicity an identity mass matrix in (1b) without loss of generality.

The solution of (1) is sought by means of a k-step BDF method [6], where temporal derivatives in (1a) and (1b) are replaced with the following difference approximation:

$$(\bullet)'_n = \frac{1}{h}\sum_{i=0}^{k} \alpha_i (\bullet)_{n-i}. \tag{2}$$

Although the analysis here is restricted to BDF methods, similar results and conclusions can be reached for other integration schemes for DAEs. For example, the case of the Newmark family of integrators is analyzed in Bottasso, Dopico, and Trainelli [7].

Using a Newton-type method, at each time step one solves a linear system of equations in the form

$$\boldsymbol{A}\boldsymbol{z} = \boldsymbol{b}, \tag{3}$$

where the iteration matrix is

$$\boldsymbol{A} = h\boldsymbol{J}_n = \begin{bmatrix} \alpha_0\boldsymbol{I} & -h\boldsymbol{I} & \boldsymbol{0} \\ h\boldsymbol{X} & \alpha_0\boldsymbol{I} + h\boldsymbol{Y} & -h\boldsymbol{G} \\ h\boldsymbol{G}^T & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}, \tag{4}$$

with

$$\boldsymbol{X} = -\boldsymbol{f}_{,\boldsymbol{u}} - \boldsymbol{G}_{,\boldsymbol{u}}\boldsymbol{\lambda}, \tag{5a}$$

$$\boldsymbol{Y} = -\boldsymbol{f}_{,\boldsymbol{v}}. \tag{5b}$$

The three block rows of this matrix correspond to the kinematic, equilibrium, and constraint equations of system (1), while the three block columns correspond to the $\boldsymbol{u}$, $\boldsymbol{v}$, and $\boldsymbol{\lambda}$ variables, respectively.

Petzold and Lötstedt [1] show that, by using Gaussian elimination with partial pivoting, the accuracy in the ith component of the solution can be estimated as

$$|\Delta z_i| \leq r\varepsilon \sum_j |(\boldsymbol{A}^{-1})_{ij}| \parallel \boldsymbol{A} \parallel_\infty \parallel \boldsymbol{z} + \Delta\boldsymbol{z} \parallel_\infty, \tag{6}$$

where r is an unknown coefficient, ε is the machine accuracy, and

$$(\boldsymbol{A} + \Delta\boldsymbol{A})(\boldsymbol{z} + \Delta\boldsymbol{z}) = \boldsymbol{b}, \tag{7}$$

with

$$\parallel \Delta\boldsymbol{A} \parallel_\infty \leq r\varepsilon \parallel \boldsymbol{A} \parallel_\infty . \tag{8}$$

The leading terms in the inverse of the iteration matrix (4) were derived in [1] as

$$\boldsymbol{A}^{-1} = (h\boldsymbol{J}_n)^{-1} = \frac{1}{\alpha_0} \begin{bmatrix} \boldsymbol{I} - \boldsymbol{T} & \gamma(\boldsymbol{I} - \boldsymbol{T})\boldsymbol{R}^{-1} & \gamma^{-1}\boldsymbol{R}^{-1}\boldsymbol{G}\boldsymbol{S} \\ -\gamma^{-1}\boldsymbol{T} & (\boldsymbol{I} - \boldsymbol{T})\boldsymbol{R}^{-1} & \gamma^{-2}\boldsymbol{R}^{-1}\boldsymbol{G}\boldsymbol{S} \\ -\gamma^{-2}\boldsymbol{S}\boldsymbol{G}^T & -\gamma^{-1}\boldsymbol{S}\boldsymbol{G}^T\boldsymbol{R}^{-1} & \gamma^{-3}\boldsymbol{S} \end{bmatrix}, \tag{9}$$

where $\gamma = h/\alpha_0$ and

$$\boldsymbol{R} = \boldsymbol{I} + \gamma\boldsymbol{Y} + \gamma^2\boldsymbol{X}, \tag{10a}$$

$$\boldsymbol{S} = (\boldsymbol{G}^T\boldsymbol{R}^{-1}\boldsymbol{G})^{-1}, \tag{10b}$$

$$\boldsymbol{T} = \boldsymbol{R}^{-1}\boldsymbol{G}\boldsymbol{S}\boldsymbol{G}^T. \tag{10c}$$

Therefore, using (6) and (9), we conclude that the roundoff errors in the solution are

$$\Delta u_i = \mathcal{O}(h^{-1}), \tag{11a}$$

$$\Delta v_i = \mathcal{O}(h^{-2}), \tag{11b}$$

$$\Delta \lambda_i = \mathcal{O}(h^{-3}). \tag{11c}$$

This shows that, as $h \to 0$, the accuracy in the Lagrange multipliers deteriorates quickly. Furthermore, it is readily verified that

$$\| \boldsymbol{A} \|_\infty = \mathcal{O}(h^0), \tag{12a}$$

$$\| \boldsymbol{A}^{-1} \|_\infty = \mathcal{O}(h^{-3}). \tag{12b}$$

Hence, the condition number $C = \| \boldsymbol{A} \|_\infty \| \boldsymbol{A}^{-1} \|_\infty$ becomes

$$C = \mathcal{O}(h^{-3}), \tag{13}$$

and the iteration matrix becomes severely ill conditioned for small h. The $\mathcal{O}(h^{-3})$ dependence of roundoff errors and conditioning on the time step size pose limitations to the practical use of variable step size solvers and time refinement procedures.

3. Preconditioning by scaling.

3.1. Left preconditioning. As suggested in [1], this situation can be improved by considering a left diagonal scaling transformation of the equations. The scaled system can be written as

$$\widetilde{\boldsymbol{A}}\boldsymbol{z} = \widetilde{\boldsymbol{b}}, \tag{14}$$

where

$$\widetilde{\boldsymbol{A}} = \boldsymbol{D}\boldsymbol{A}, \tag{15a}$$

$$\widetilde{\boldsymbol{b}} = \boldsymbol{D}\boldsymbol{b}, \tag{15b}$$

and $\boldsymbol{D}$ is a diagonal scaling matrix defined as

$$\boldsymbol{D} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & h^{-1}\boldsymbol{I} \end{bmatrix}. \tag{16}$$

The inverse of the iteration matrix is in this case

$$\widetilde{\boldsymbol{A}}^{-1} = \frac{1}{\alpha_0}\begin{bmatrix} \boldsymbol{I}-\boldsymbol{T} & \gamma(\boldsymbol{I}-\boldsymbol{T})\boldsymbol{R}^{-1} & \alpha_0\boldsymbol{R}^{-1}\boldsymbol{G}\boldsymbol{S} \\ -\gamma^{-1}\boldsymbol{T} & (\boldsymbol{I}-\boldsymbol{T})\boldsymbol{R}^{-1} & \alpha_0\gamma^{-1}\boldsymbol{R}^{-1}\boldsymbol{G}\boldsymbol{S} \\ -\gamma^{-2}\boldsymbol{S}\boldsymbol{G}^T & -\gamma^{-1}\boldsymbol{S}\boldsymbol{G}^T\boldsymbol{R}^{-1} & \alpha_0\gamma^{-2}\boldsymbol{S} \end{bmatrix}, \tag{17}$$

and, using (6) again, we find

$$\Delta u_i = \mathcal{O}(h^0), \tag{18a}$$

$$\Delta v_i = \mathcal{O}(h^{-1}), \tag{18b}$$

$$\Delta \lambda_i = \mathcal{O}(h^{-2}). \tag{18c}$$

The situation is improved with respect to the original (unscaled) system, but the error in the Lagrange multipliers still grows rapidly as $h \to 0$ even for the left preconditioned problem. Similarly, we find $\| \widetilde{\boldsymbol{A}} \|_\infty = \mathcal{O}(h^0)$ and $\| \widetilde{\boldsymbol{A}}^{-1} \|_\infty = \mathcal{O}(h^{-2})$, which leads to the conclusion that the condition number is

$$C = \mathcal{O}(h^{-2}), \tag{19}$$

which is again not a favorable behavior.

3.2. Left and right preconditioning. A further improvement on the situation expressed by (18) can be obtained by considering a left and right preconditioning, i.e., a scaling of the equations together with a scaling of the unknowns. In particular, since the Lagrange multipliers are affected by the largest roundoff errors according to (18), the governing equations are here rewritten in the form

$$\boldsymbol{u}' = \boldsymbol{v}, \tag{20a}$$

$$\boldsymbol{v}' = \boldsymbol{f} + s\boldsymbol{G}\hat{\boldsymbol{\lambda}}, \tag{20b}$$

$$\boldsymbol{\phi} = \boldsymbol{0}, \tag{20c}$$

where $\hat{\boldsymbol{\lambda}} = \boldsymbol{\lambda}/s$ are scaled Lagrange multipliers, and the scaling factor is $s = \mathcal{O}(h^{-2})$. The idea is now to solve the problem in terms of the scaled multipliers $\hat{\boldsymbol{\lambda}}$ and generalized coordinates $\boldsymbol{u}$ and velocities $\boldsymbol{v}$, and to recover the values of $\boldsymbol{\lambda}$ once at convergence.

Consider first for simplicity $s = 1/h^2$. The linear iteration problem for the left and right preconditioned system can be written as

$$\hat{\boldsymbol{A}}\hat{\boldsymbol{z}} = \hat{\boldsymbol{b}}, \tag{21}$$

where

$$\hat{\boldsymbol{A}} = \boldsymbol{D}_L\boldsymbol{A}\boldsymbol{D}_R, \tag{22a}$$

$$\hat{\boldsymbol{z}} = \boldsymbol{D}_R^{-1}\boldsymbol{z}, \tag{22b}$$

$$\hat{\boldsymbol{b}} = \boldsymbol{D}_L\boldsymbol{b}. \tag{22c}$$

The left diagonal scaling matrix $\boldsymbol{D}_L$ is defined as

$$\boldsymbol{D}_L = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & h\boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & h^{-1}\boldsymbol{I} \end{bmatrix}. \tag{23}$$

The scaling of the third block rows is the same as in [1], while the scaling of the second block rows is needed to ensure $\| \hat{\boldsymbol{A}} \|_\infty = \mathcal{O}(h^0)$, which is beneficial for the condition number. The right diagonal transformation $\boldsymbol{D}_R$ which scales the Lagrange multipliers is defined as

$$\boldsymbol{D}_R = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & h^{-2}\boldsymbol{I} \end{bmatrix}. \tag{24}$$

This gives the following form for the inverse of the iteration matrix:

$$\hat{\boldsymbol{A}}^{-1} = \frac{1}{\alpha_0} \begin{bmatrix} \boldsymbol{I}-\boldsymbol{T} & \alpha_0^{-1}(\boldsymbol{I}-\boldsymbol{T})\boldsymbol{R}^{-1} & \alpha_0 \boldsymbol{R}^{-1}\boldsymbol{G}\boldsymbol{S} \\ -\gamma^{-1}\boldsymbol{T} & \alpha_0^{-1}\gamma^{-1}(\boldsymbol{I}-\boldsymbol{T})\boldsymbol{R}^{-1} & \alpha_0\gamma^{-1}\boldsymbol{R}^{-1}\boldsymbol{G}\boldsymbol{S} \\ -\alpha_0^2 \boldsymbol{S}\boldsymbol{G}^T & -\alpha_0 \boldsymbol{S}\boldsymbol{G}^T\boldsymbol{R}^{-1} & \alpha_0^3 \boldsymbol{S} \end{bmatrix}. \tag{25}$$

Using (6), the roundoff errors for the three solution fields are in this case

$$\Delta u_i = \mathcal{O}(h^0), \tag{26a}$$

$$\Delta v_i = \mathcal{O}(h^{-1}), \tag{26b}$$

$$\Delta \hat{\lambda}_i = \mathcal{O}(h^0). \tag{26c}$$

Notice that the roundoff errors in the displacement field and in the scaled Lagrange multipliers are now insensitive to the time step size, while the condition number is

$$C = \mathcal{O}(h^{-1}). \tag{27}$$

The Lagrange multipliers are recovered at convergence of the Newton process from the scaled multipliers using $\boldsymbol{\lambda} = s\hat{\boldsymbol{\lambda}}$. The scaled multipliers will be affected by an error due to machine accuracy and to the Newton termination tolerance, error which is $\mathcal{O}(h^0)$ for the results above. Since s is $\mathcal{O}(h^{-2})$, then $\boldsymbol{\lambda}$ will be affected by an error $\mathcal{O}(h^{-2})$ due to the recovery process. However, the reduced accuracy of the multipliers cannot affect the iterative Newton process, since this is formulated in terms of the scaled variables. Hence, the Newton iterations can be carried out until as tight a tolerance as necessary has been achieved.

A more general form for the multiplier scaling factor can be derived based on a dimensional argument. Recall that in general the dynamic equilibrium equations (1b) express the balance of inertial forces, elastic forces, constraint reactions, and externally applied forces. To reflect the different nature of these forces, s is selected as

$$s = k_{\text{ave}} + \frac{m_{\text{ave}}}{h^2}, \tag{28}$$

where k_{ave} represents an average stiffness term for the system and m_{ave} an average mass term. To understand the dimensional effect of this scaling, let u_{ave} denote a representative displacement of the system, $F_{\text{ave}}^{\text{elas}}$ a representative value of the elastic forces, and finally $F_{\text{ave}}^{\text{iner}}$ a representative value of the inertial forces. For large time step sizes, $s \approx k_{\text{ave}}$, and $\hat{\boldsymbol{\lambda}} \approx \boldsymbol{\lambda}/k_{\text{ave}}$. Then the ratio of the constraint reactions and of the elastic forces is approximately $s\boldsymbol{G}\hat{\boldsymbol{\lambda}}/F_{\text{ave}}^{\text{elas}} \approx k_{\text{ave}}\boldsymbol{G}\hat{\boldsymbol{\lambda}}/(k_{\text{ave}}u_{\text{ave}}) \approx \boldsymbol{G}\hat{\boldsymbol{\lambda}}/u_{\text{ave}}$; this means that the scaled Lagrange multipliers will be roughly of the same order of magnitude as the displacements, balancing the magnitude of all the unknowns of the problem. For very small time step sizes, $s \approx m_{\text{ave}}/h^2$, and $\hat{\boldsymbol{\lambda}} \approx \boldsymbol{\lambda}/(m_{\text{ave}}/h^2)$. It then

follows that $s\boldsymbol{G}\hat{\boldsymbol{\lambda}}/F_{\text{ave}}^{\text{iner}} \approx (m_{\text{ave}}\boldsymbol{G}\hat{\boldsymbol{\lambda}}/h^2)/(m_{\text{ave}}u_{\text{ave}}/h^2) \approx \boldsymbol{G}\hat{\boldsymbol{\lambda}}/u_{\text{ave}}$; this means that even in this case the scaled Lagrange multipliers will be roughly of the same order of magnitude as the displacements, balancing again the magnitude of all the unknowns of the problem.

The roundoff errors and the conditioning of the left and right scaled system are much improved with respect to the unscaled or the left scaled ones. Yet, the velocity field still shows a mild dependence on the time step size, which in turn induces the same behavior in the conditioning. This effect can be eliminated by considering a scaling of the unknowns that includes, together with the scaling of the Lagrange multipliers, also a scaling of the velocities, as suggested by Arnold [8]. This amounts to the following redefinition of the right diagonal transformation $\boldsymbol{D}_R$:

$$\boldsymbol{D}_R = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & h^{-1}\boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & h^{-2}\boldsymbol{I} \end{bmatrix}. \tag{29}$$

The inverse of the iteration matrix is readily found as

$$\hat{\boldsymbol{A}}^{-1} = \frac{1}{\alpha_0} \begin{bmatrix} \boldsymbol{I}-\boldsymbol{T} & \alpha_0^{-1}(\boldsymbol{I}-\boldsymbol{T})\boldsymbol{R}^{-1} & \alpha_0\boldsymbol{R}^{-1}\boldsymbol{G}\boldsymbol{S} \\ -\alpha_0\boldsymbol{T} & (\boldsymbol{I}-\boldsymbol{T})\boldsymbol{R}^{-1} & \alpha_0^2\boldsymbol{R}^{-1}\boldsymbol{G}\boldsymbol{S} \\ -\alpha_0^2\boldsymbol{S}\boldsymbol{G}^T & -\alpha_0\boldsymbol{S}\boldsymbol{G}^T\boldsymbol{R}^{-1} & \alpha_0^3\boldsymbol{S} \end{bmatrix}, \tag{30}$$

which is not dependent on h. Hence, the roundoff errors in all solution variables and the conditioning are $\mathcal{O}(h^0)$.

3.3. Numerical example. We consider the problem of the pendulum, with equations

$$u_x' = v_x, \tag{31a}$$
$$u_y' = v_y, \tag{31b}$$
$$v_x' = \lambda u_x, \tag{31c}$$
$$v_y' = \lambda u_y - 1, \tag{31d}$$
$$0 = \frac{1}{2}(u_x^2 + u_y^2 - 1), \tag{31e}$$

where u_x, u_y are the Cartesian coordinates of the point mass, and v_x, v_y the mass velocity components. Bar length, point mass, and acceleration of gravity are all equal to 1. The point mass is initially at rest with $u_x = 1$ and $u_y = 0$ and falls under the action of gravity. The integration of (31) is performed using the two-step BDF formula until time $t = 1 \cdot 10^{-3}$, using the time step sizes $h = \{1 \cdot 10^{-4}, 1 \cdot 10^{-5}, 1 \cdot 10^{-6}, 1 \cdot 10^{-7}, 1 \cdot 10^{-8}\}$.

At each time step, the nonlinear discrete equations are solved using Newton method. At the jth Newton iteration of time step n, we solve the linear system of equations

$$\boldsymbol{A}_n^j \boldsymbol{z}_n^j = \boldsymbol{b}_n^j, \tag{32}$$

which yields the corrections $\boldsymbol{z}_n^j$. Typically, the norm of the corrections will decrease at each Newton step, until the accumulation of roundoff errors leads to a saturation value. If further iterations are carried out, one typically observes an oscillation of the correction norm around its saturation value. An example of this behavior is given in

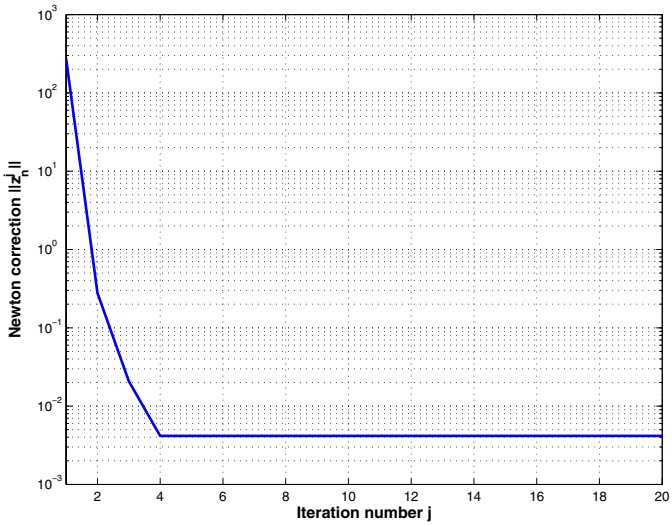


FIG. 1. *Convergence and saturation for the norm of the Newton corrections* $\boldsymbol{z}_n^j$ *plotted versus the Newton iteration* j *for the scheme without preconditioning and* $h = 1 \cdot 10^{-7}$.

Figure 1, which shows $\| \boldsymbol{z}_n^j \|$ versus the Newton iteration j for the scheme without preconditioning and $h = 1 \cdot 10^{-7}$. In this case, the saturation is reached at the fourth iteration, after which the correction norm has very small oscillations around a saturation value of approximatively $4 \cdot 10^{-3}$. To quantify this effect of roundoff errors, in this work iterations are arrested when the Newton corrections stop decreasing, i.e., when we detect the condition

$$\| \boldsymbol{z}_n^{j+1} \| \geq \| \boldsymbol{z}_n^j \| . \tag{33}$$

Hence, the magnitude of the last decreasing Newton correction gives an indication of the tightest achievable convergence of the Newton iterations, which cannot be further improved no matter how many iterations one carries out.

At first, we consider the scaling of the sole Lagrange multipliers based on the simple choice $s = 1/h^2$, and we use the definition of $\boldsymbol{D}_R$ given in (24). Figure 2 shows the maximum throughout each simulation of the 2-norm of the last decreasing Newton corrections versus the time step size, i.e., $\max_n \| \boldsymbol{z}_n^j \|$ versus h. The results obtained with the left preconditioned problem are shown with the $\square$ symbol, while the ones of the left and right preconditioning are shown with the $\circ$ symbol. Notice that with left preconditioning alone it is in fact impossible to solve the nonlinear iteration with a tight tolerance for small values of the time step.

Figure 3 shows the maximum throughout each simulation of the absolute value of the last decreasing Newton corrections by field type, i.e., $\max_n \max(|z^j_{u_x,n}|, |z^j_{u_y,n}|)$, $\max_n \max(|z^j_{v_x,n}|, |z^j_{v_y,n}|)$, $\max_n |z^j_{\lambda,n}|$. These results are in accordance with the analysis of the previous sections.

Finally, Figure 4 shows the condition number C versus h. The condition number is computed by means of the singular value decomposition of the iteration matrix at convergence, i.e., at the last decreasing Newton iteration. Here again, a substantial improvement is observed for the left and right preconditioning with respect to the left preconditioning alone.

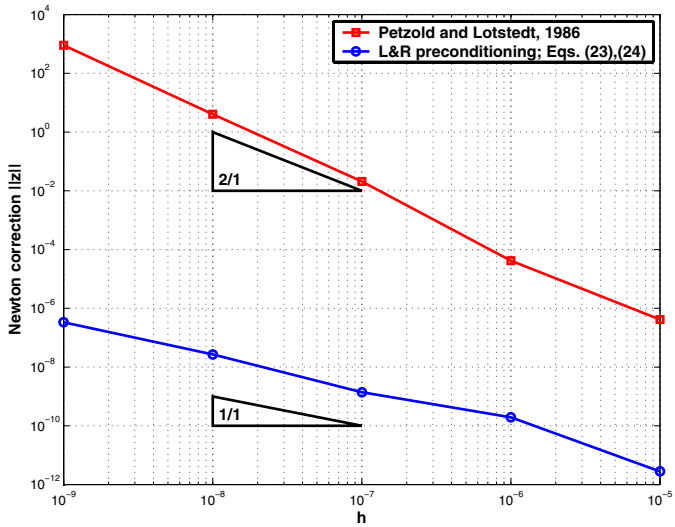


FIG. 2. *Last decreasing Newton corrections versus time step size.*

Next, we consider the previous scaling of the multipliers together with the scaling of the velocities, with $\boldsymbol{D}_R$ as given in (29). Figure 5 shows the maximum throughout each simulation of the 2-norm of the last decreasing Newton corrections versus the time step size, while Figure 6 shows the condition number C versus h. In both cases, the solution of the left and right preconditioned problem does not depend on h, as predicted by the analysis.

4. Displacement based form. For large scale applications denoted by many degrees of freedom, the displacement-velocity-multiplier form discussed in the previous section leads to the solution of large linear systems. For example, this is the case when the governing DAEs (1) are obtained by spatial discretization using the finite element method of a flexible mechanical system. A substantial performance increase can be obtained by implementing the integration scheme in displacement-multiplier form, which is simply achieved by statically eliminating the velocity unknowns at each time step. In fact, considering again the k-step BDF method, velocities at time t_n are computed from the kinematic equations (1a) using the difference approximation (2) as

$$\boldsymbol{u}'_n = \boldsymbol{v}_n = \frac{1}{h}\sum_{i=0}^{k} \alpha_i \boldsymbol{u}_{n-i}. \tag{34}$$

Inserting this expression into the accelerations of the dynamic equilibrium equations (1b), we get

$$\boldsymbol{v}'_n = \frac{1}{h}\sum_{i=0}^{k} \alpha_i \boldsymbol{v}_{n-i} = \frac{1}{h}\left(\frac{\alpha_0}{h}\sum_{i=0}^{k} \alpha_i \boldsymbol{u}_{n-i} + \sum_{i=1}^{k} \alpha_i \boldsymbol{v}_{n-i}\right). \tag{35}$$

Using Newton's method, one solves the reduced system of linear equations

$$\boldsymbol{B}\boldsymbol{y} = \boldsymbol{c}, \tag{36}$$

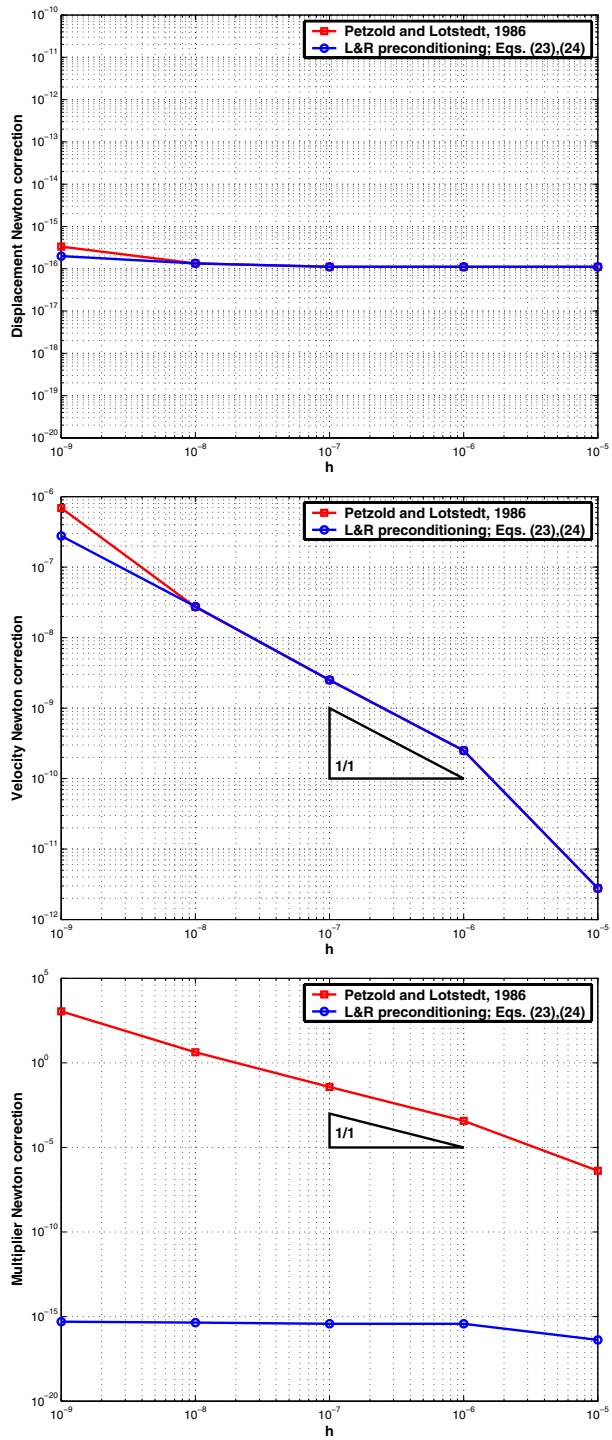


FIG. 3. *Last decreasing Newton corrections by field type versus time step size. Top: Displacements. Center: Velocities. Bottom: Lagrange multipliers.*

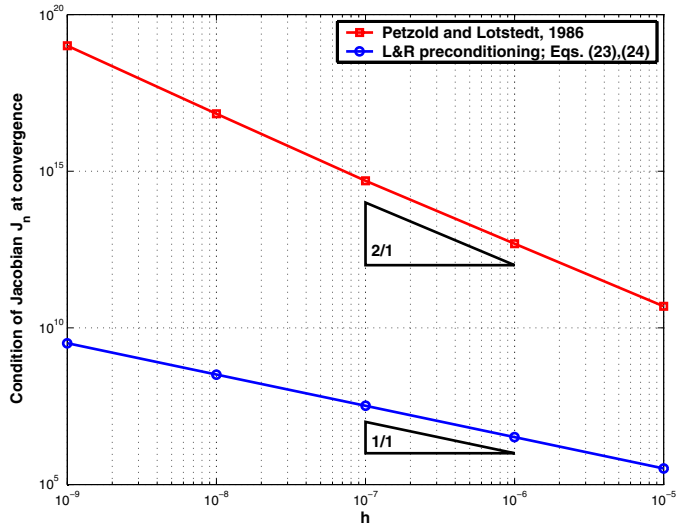


FIG. 4. *Condition number of Jacobian at convergence versus time step size.*

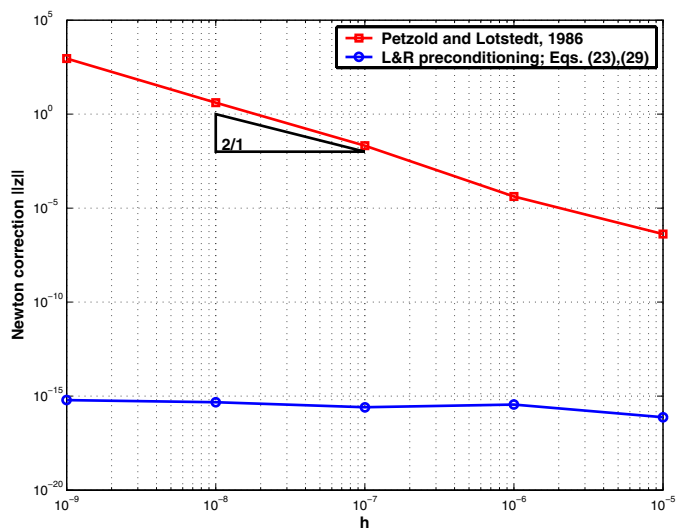


FIG. 5. *Last decreasing Newton corrections versus time step size. Left and right preconditioning based on* (23) *and* (29).

where the iteration matrix is in this case

$$\boldsymbol{B} = h\boldsymbol{J}_n = \begin{bmatrix} \alpha_0 \boldsymbol{U} & -h\boldsymbol{G} \\ h\boldsymbol{G}^T & \boldsymbol{0} \end{bmatrix}, \tag{37}$$

with

$$\boldsymbol{U} = \gamma^{-1}\boldsymbol{I} + \boldsymbol{Y} + \gamma \boldsymbol{X}. \tag{38}$$

The first block rows of this matrix correspond to the equilibrium equations in displacement-based form, while the second block rows correspond to the constraint

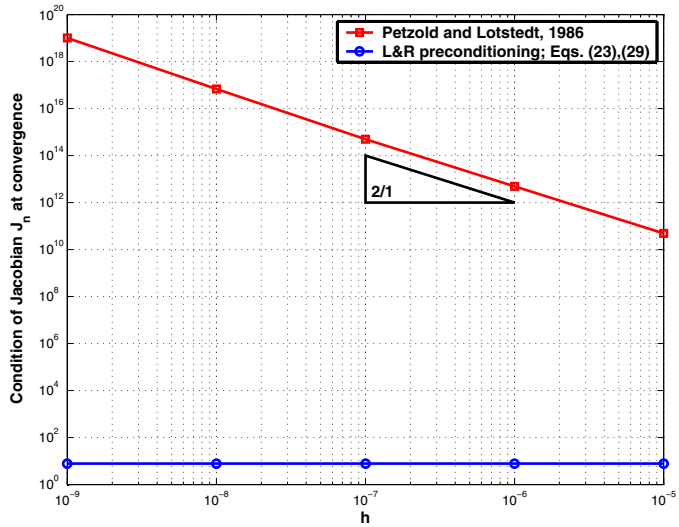


FIG. 6. *Condition number of Jacobian at convergence versus time step size. Left and right preconditioning based on* (23) *and* (29).

conditions. The two block columns correspond to the $\boldsymbol{u}$ and $\boldsymbol{\lambda}$ variables, respectively. Once at convergence, velocities are recovered using the vector relationship (34).

The inverse of the iteration matrix is

$$\boldsymbol{B}^{-1} = \frac{1}{\alpha_0} \begin{bmatrix} (\boldsymbol{I} - \boldsymbol{W})\boldsymbol{U}^{-1} & \gamma^{-1}\boldsymbol{U}^{-1}\boldsymbol{G}\boldsymbol{V} \\ -\gamma^{-1}\boldsymbol{V}\boldsymbol{G}^T\boldsymbol{U}^{-1} & \gamma^{-2}\boldsymbol{V} \end{bmatrix}, \tag{39}$$

where

$$\boldsymbol{V} = (\boldsymbol{G}^T\boldsymbol{U}^{-1}\boldsymbol{G})^{-1}, \tag{40a}$$

$$\boldsymbol{W} = \boldsymbol{U}^{-1}\boldsymbol{G}\boldsymbol{V}\boldsymbol{G}^T. \tag{40b}$$

Using (6), and considering that

$$\lim_{h\to 0} \boldsymbol{U}^{-1} = \mathcal{O}(h), \tag{41a}$$

$$\lim_{h\to 0} \boldsymbol{V} = \mathcal{O}(h^{-1}), \tag{41b}$$

the roundoff errors in the primary solution variables are found to be

$$\Delta u_i = \mathcal{O}(h^{-1}), \tag{42a}$$

$$\Delta \lambda_i = \mathcal{O}(h^{-3}), \tag{42b}$$

while the condition number C becomes

$$C = \mathcal{O}(h^{-4}); \tag{43}$$

again these are very unfavorable behaviors with respect to the time step size.

4.1. Left preconditioning. Even in this case, the simple left preconditioning of Petzold and Lötstedt [1] gives a modest benefit with respect to the unpreconditioned problem. In fact, defining the left scaling transformation as

$$\boldsymbol{D} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & h^{-1}\boldsymbol{I} \end{bmatrix}, \tag{44}$$

we find

$$\widetilde{\boldsymbol{B}}^{-1} = \frac{1}{\alpha_0} \begin{bmatrix} (\boldsymbol{I} - \boldsymbol{W})\boldsymbol{U}^{-1} & \alpha_0 \boldsymbol{U}^{-1}\boldsymbol{G}\boldsymbol{V} \\ -\gamma^{-1}\boldsymbol{V}\boldsymbol{G}^T\boldsymbol{U}^{-1} & \alpha_0\gamma^{-1}\boldsymbol{V} \end{bmatrix}, \tag{45}$$

which gives for the propagation of errors

$$\Delta u_i = \mathcal{O}(h^0), \tag{46a}$$

$$\Delta \lambda_i = \mathcal{O}(h^{-2}), \tag{46b}$$

while the conditioning is

$$C = \mathcal{O}(h^{-3}). \tag{47}$$

4.2. Left and right preconditioning. The left and right scaling matrices for the reduced problem are simply obtained from (21)–(24) and are defined as

$$\boldsymbol{D}_L = \begin{bmatrix} h\boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & h^{-1}\boldsymbol{I} \end{bmatrix} \tag{48}$$

and

$$\boldsymbol{D}_R = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & h^{-2}\boldsymbol{I} \end{bmatrix}, \tag{49}$$

where we have considered again the simple choice $s = 1/h^2$. This gives

$$\hat{\boldsymbol{B}}^{-1} = \frac{1}{\alpha_0} \begin{bmatrix} \alpha_0^{-1}\gamma^{-1}(\boldsymbol{I} - \boldsymbol{W})\boldsymbol{U}^{-1} & \alpha_0 \boldsymbol{U}^{-1}\boldsymbol{G}\boldsymbol{V} \\ -\alpha_0 \boldsymbol{V}\boldsymbol{G}^T\boldsymbol{U}^{-1} & \alpha_0^3\gamma\boldsymbol{V} \end{bmatrix}, \tag{50}$$

which yields time-step-size-independent sensitivity to perturbations of the primary variables

$$\Delta u_i = \mathcal{O}(h^0), \tag{51a}$$

$$\Delta \hat{\lambda}_i = \mathcal{O}(h^0), \tag{51b}$$

and a time-step-size-independent conditioning, i.e.,

$$C = \mathcal{O}(h^0), \tag{52}$$

which is the best possible behavior that can be expected of a numerical method.

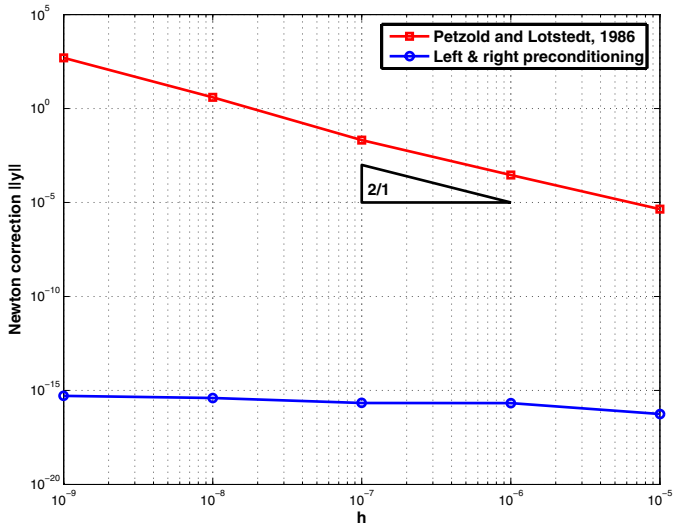


FIG. 7. *Last decreasing Newton corrections versus time step size for the displacement-multiplier form.*

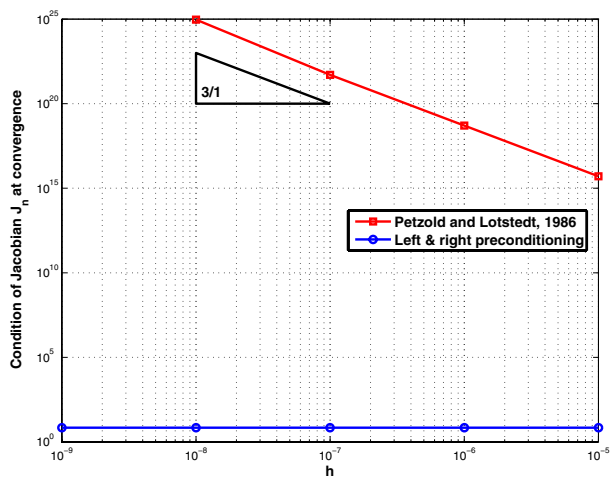


FIG. 8. *Condition number of Jacobian at convergence versus time step size for the displacement-multiplier form.*

4.3. Numerical example. We consider again the model problem (31), using the same data. At the jth Newton iteration of time step n, we solve the linear system of equations

$$\boldsymbol{B}_n^j \boldsymbol{y}_n^j = \boldsymbol{c}_n^j, \tag{53}$$

which yields the corrections $\boldsymbol{y}_n^j$. As in the previous case, iterations are arrested only when the Newton corrections stop decreasing.

The results of the analysis are confirmed by Figures 7 and 8. In particular, the former shows the maximum throughout each simulation of the 2-norm of the last decreasing Newton corrections versus the time step size, $\max_n \parallel \boldsymbol{z}_n^j \parallel$ versus

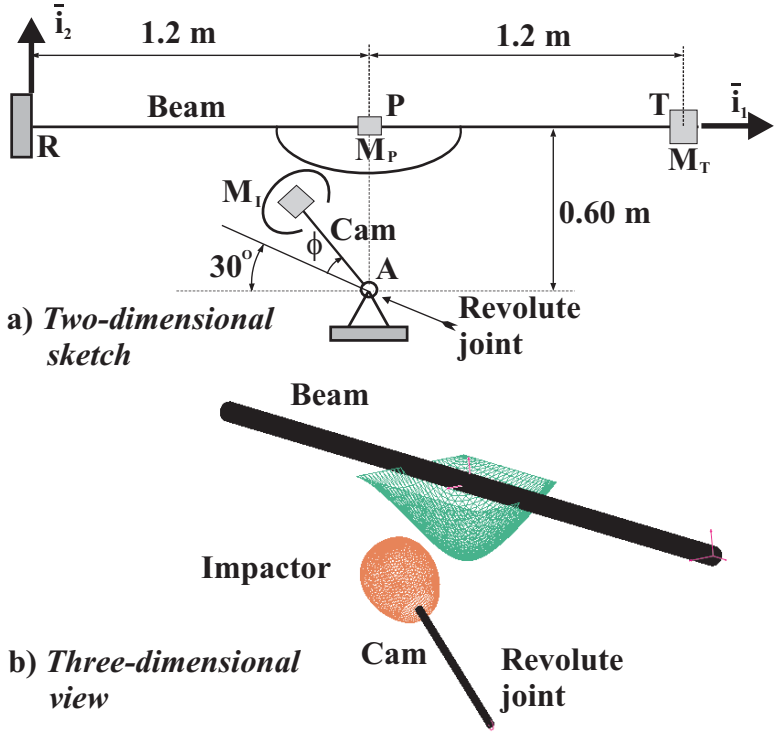


FIG. 9. *Configuration of the contact problem.*

h, while the latter reports the condition number C versus h. For the left and right preconditioning, exact time step size independence is achieved.

5. Numerical example: The need for scaling in contact-impact analysis of flexible multibody systems. In this section we consider a more challenging problem involving a flexible multibody system subjected to unilateral contact conditions. In general, to accurately determine the contact events and for resolving the large gradients in the interactional forces at contact, time step size refinement procedures must be used for contact problems. However, whenever small values of the time steps are used, formulations in index three form will suffer from the problems discussed in the previous sections.

Figure 9(a) depicts a cam of length 0.4 m connected to the ground at point A by means of a revolute joint. The relative rotation in the joint is prescribed to be $\phi(t) = \phi_0(1-\cos\Omega t)$, where $\phi_0 = \pi/3$ rad, and $\Omega = 2\pi/3$ rad/sec. A rigid impactor of total mass $M_I = 4$ kg is attached at the tip of the cam. A flexible beam of length 2.4 m is clamped to the ground at point R, and a point mass $M_T = 40$ kg is attached at its free end, point T. Finally, a rigid body of total mass $M_P = 4$ kg is attached at the middle point P of the beam. As the impactor moves, it contacts the outer surface of this rigid body, as shown in Figure 9(b). The physical properties of the flexible beam are axial stiffness $EA = 44000$ kN, bending stiffnesses $I_{22} = 300$ and $I_{33} = 23$ kN·m^2, torsional stiffness $GJ = 28$ kN·m^2, and mass per unit span $m = 1.6$ kg/m. The bending stiffnesses of the cam are $I_{22} = 23$, $I_{33} = 300$ kN·m^2, and the axial stiffness, torsional stiffness, and mass per unit span are the same as those of the beam.

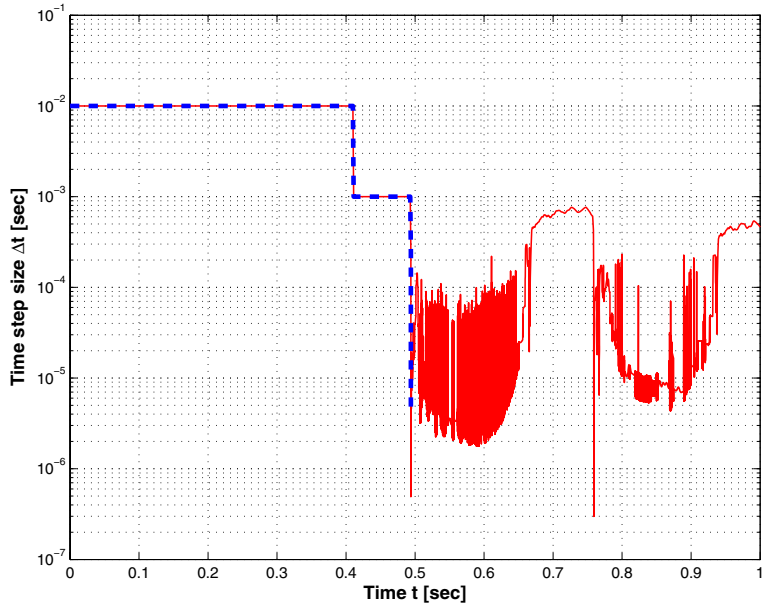


FIG. 10. *Time step size as a function of time.* $m_{\text{ave}} = 0$*: Thick dashed line.* $m_{\text{ave}} = 1$*: Thin solid line.*

When the two bodies come in contact, a contact model and friction model are used to define the normal and tangential contact forces, respectively. A linear spring with a stiffness constant $k = 2$ GN/m in parallel with a viscous damper of constant $\mu = 10^{-3}$ were used to evaluate the normal contact force. The friction force was computed with the help of the LuGre model, using the following parameters: $\sigma_0 = 10^5$ m^{-1}, $\sigma_1 = \sigma_2 = 0$ sec/m, $v_s = 10^{-3}$ m/sec, $\mu_k = 0.30$, $\mu_s = 0.30$, and $\gamma = 2$. More details about this modeling technique and the definition of these parameters can be found in Bauchau and Ju [9].

The simulation was conducted for a 1 sec duration. From $t = 0$ to 0.4 sec a constant time step size, $h = 10^{-2}$ sec, was used and at that point, an automated time step size adaptivity algorithm was activated [10]; the desired local error was set to $\varepsilon_{\text{loc}} = 10^{-7}$. In light of (28), the scaling factor was selected as $s = k_{\text{ave}} + m_{\text{ave}}/h^2$, where $k_{\text{ave}} = 10^3$ and $m_{\text{ave}} = 0$ or 1. When $m_{\text{ave}} = 0$, the scaling is independent of the time step size, whereas it becomes time step dependent when $m_{\text{ave}} = 1$.

For the problem at hand, large time step sizes were used before contact occurs, i.e., when $t < 0.494$ sec. This portion of the simulation is not challenging and both scaling factors performed equally well. At time $t \approx 0.494$ the time step size was drastically reduced by the adaptivity algorithm to decrease the large contact force gradients that would result from using a constant time step. Figure 10 depicts the time step size as a function of time during the simulation, for both $m_{\text{ave}} = 0$ and 1. As should be expected from the analysis developed in this paper, when $m_{\text{ave}} = 0$, the simulation fails to converge as soon as the time step size becomes small, whereas for $m_{\text{ave}} = 1$, convergence is achieved at each time step, despite the complex dynamic behavior of the system after impact. Figure 11 shows the corresponding history of the scaling factor. Note that scaling factors of the order of 10^{13} were needed at the

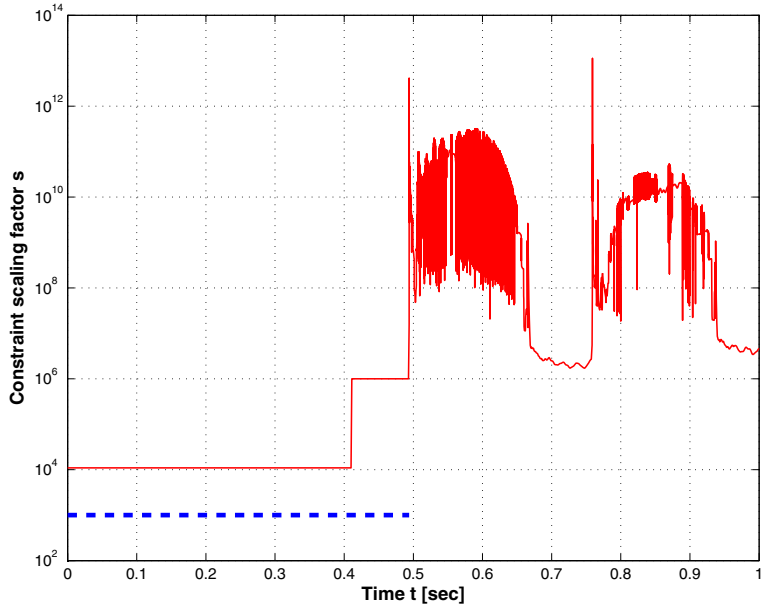


FIG. 11. *Constraint scaling factor* s. $m_{\text{ave}} = 0$*: Thick dashed line.* $m = 1$*: Thin solid line.*

instant of contact; after impact, the highly vibratory response of the system required small time steps and the scaling factor was in the range of $s \in [10^8,\ 10^{11}]$.

6. Conclusions. We have proposed a simple scaling transformation for the index three DAEs describing constrained multibody dynamical systems. The approach amounts to a left and right preconditioning of the iteration matrix, which has the goal of making the solution less sensitive to the propagation of perturbations and of improving the condition number.

Using the left preconditioning, the constraint equations are divided by the time step size as already proposed by [1], while the dynamic equilibrium equations are multiplied by the same quantity. The right preconditioning amounts to a definition of a set of modified unknowns: modified multipliers are obtained by scaling with the square of the time step size of the original Lagrange multipliers, while modified velocities are obtained by scaling with the time step size.

The new scaling was applied to the displacement-velocity-multiplier and to the reduced displacement-multiplier forms. In both cases, we obtain the remarkable results that both error propagations and conditioning are $\mathcal{O}(h^0)$; i.e., the numerical solution of index three DAEs behaves as in the case of regular ODEs.

The simple problem of the pendulum was used to numerically confirm the results of the analysis, while a more challenging problem dealing with the contact-impact analysis of a flexible multibody system was used to illustrate the effect of scaling when using time refinement procedures.

Although other approaches are available to deal with the problem of ill conditioning and error propagation for high index DAEs, the present approach has the possible advantages of being trivial to implement and of not requiring a rewriting of the equations of motion, and furthermore it does not introduce additional unknowns.

REFERENCES

[1] L. Petzold and P. Lötstedt, *Numerical solution of nonlinear differential equations with algebraic constraints* II: *Practical implications*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 720–733.

[2] M. Arnold, *A perturbation analysis for the dynamical simulation of mechanical multibody systems*, Appl. Numer. Math., 18 (1995), pp. 37–56.

[3] C. Gear, B. Leimkuhler, and G. Gupta, *Automatic integration of Euler–Lagrange equations with constraints*, J. Comput. Appl. Math., 12–13 (1985), pp. 77–90.

[4] M. Borri, L. Trainelli, and A. Croce, *The embedded projection method: A general index reduction procedure for constrained system dynamics*, Comput. Methods Appl. Mech. Engrg., 195 (2006), pp. 6974–6992.

[5] A. Cardona and M. Géradin, *Numerical integration of second order differential-algebraic systems in flexible mechanism dynamics*, in Computer-Aided Analysis of Rigid and Flexible Mechanical Systems, M. F. O. Seabra Pereira and J. A. C. Ambrósio, eds., Kluwer Academic Publishers, Dordrecht, The Netherlands, 1994.

[6] C. Gear, *Simultaneous numerical solution of differential/algebraic equations*, IEEE Trans. Circuits Systems, 18 (1971), pp. 89–95.

[7] C. L. Bottasso, D. Dopico, and L. Trainelli, *On the optimal scaling of index three DAEs in multibody dynamics*, in Proceedings of the 3rd European Conference on Computational Mechanics, Solids, Structures, and Coupled Problems in Engineering, C. A. Mota Soares et al., eds., Lisbon, Portugal, 2006.

[8] M. Arnold, *Private communication*, 2005.

[9] O. A. Bauchau and C. K. Ju, *Modeling friction phenomena in flexible multibody dynamics*, Comput. Methods Appl. Mech. Engrg., 195 (2006), pp. 6909–6924.

[10] O. A. Bauchau, *Computational schemes for flexible, nonlinear, multibody systems*, Multibody Syst. Dyn., 2 (1998), pp. 169–225.

SIAM J. SCI. COMPUT.
Vol. 29, No. 1, pp. 415–439
© 2007 Society for Industrial and Applied Mathematics

ON PRECONDITIONED ITERATIVE METHODS FOR BURGERS EQUATIONS*

ZHONG-ZHI BAI†, YU-MEI HUANG‡, AND MICHAEL K. NG‡

Abstract. We study the Newton method and a fixed-point method for solving the system of nonlinear equations arising from the Sinc–Galerkin discretization of the Burgers equations. In each step of the Newton method or the fixed-point method, a structured subsystem of linear equations is obtained and needs to be solved numerically. In this paper, preconditioning techniques are applied to solve such linear subsystems. The bounds for eigenvalues of the preconditioned matrices are derived and numerical examples are given to illustrate the effectiveness of the proposed methods. We also find that a combination of the Newton/fixed-point iteration with the preconditioned GMRES method is quite efficient for the Sinc–Galerkin discretization of the Burgers equations.

Key words. Burgers equation, Sinc–Galerkin discretization, Toeplitz-like matrices, preconditioners, GMRES method

AMS subject classifications. 65F10, 65F15, 65T10

DOI. 10.1137/060649124

1. Introduction.

We consider an iterative solution of the system of nonlinear equations

$$\mathbf{F}(\mathbf{u}) \equiv B\mathbf{u} + C\Psi(\mathbf{u}) - \mathbf{b} = 0, \tag{1.1}$$

where B and C are n-by-n matrices, $\mathbf{b}$ is a given n-vector, and $\Psi : \mathbb{R}^n \to \mathbb{R}^n$, with

$$\Psi(\mathbf{u}) = (\psi_1(u_1), \psi_2(u_2), \ldots, \psi_n(u_n))^T \quad \text{and} \quad \mathbf{u} = (u_1, u_2, \ldots, u_n)^T, \tag{1.2}$$

is a continuous diagonal mapping defined on the open ball

$$\mathcal{U}_\delta := \{u \in \mathbb{R}^n \mid \|\mathbf{u}\| < \delta\}.$$

See [19] for general background and applications about the system of mildly nonlinear equations.

Nonlinear systems such as (1.1)–(1.2) may also arise from the Sinc–Galerkin discretization of the Burgers equation [15]:

(1.3)
$$\begin{cases} \mathcal{P}^{(2)}u(x,t) \equiv \dfrac{\partial u}{\partial t}(x,t) + u(x,t)\dfrac{\partial u}{\partial x}(x,t) - \varepsilon\dfrac{\partial^2 u}{\partial x^2}(x,t) = f(x,t), & a < x < b,\ t \geq 0, \\ u(a,t) = \gamma(t) \quad \text{and} \quad u(b,t) = \delta(t), & t \geq 0, \\ u(x,0) = g(x), & a \leq x \leq b. \end{cases}$$

*Received by the editors January 5, 2006; accepted for publication (in revised form) September 19, 2006; published electronically February 15, 2007.

http://www.siam.org/journals/sisc/29-1/64912.html

†State Key Laboratory of Scientific/Engineering Computing, Institute of Computational Mathematics and Scientific/Engineering Computing, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, P.O. Box 2719, Beijing 100080, The People's Republic of China (bzz@lsec.cc.ac.cn). This author's research was supported by The National Basic Research Program (2005CB321702), The China NNSF National Outstanding Young Scientist Foundation (10525102), and The National Natural Science Foundation (10471146), The People's Republic of China.

‡Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Hong Kong (ymhuang@math.hkbu.edu.hk, mng@math.hkbu.edu.hk). The research of the third author was supported in part by RGC grants 7046/03P, 7035/04P, and 7035/05P, and HKBU FRGs.

In the Sinc–Galerkin method, we approximate u by

$$\widehat{u} = \sum_{j=-m_t-1}^{n_t} \sum_{i=-m_x-1}^{n_x+1} u_{ij}\chi_i(x)\theta_j(t),$$

where the functions $\chi_i(x)$ and $\theta_j(t)$ are bases in the spatial space and the temporal space, respectively, with

$$\chi_i(x) = \begin{cases} \dfrac{b-x}{b-a}, & i = -m_x - 1, \\ \mathcal{S}(i, h_x) \circ \phi_x(x), & -m_x \le i \le n_x, \\ \dfrac{x-a}{b-a}, & i = n_x + 1, \end{cases}$$

and

$$\mathcal{S}(i, h_x) \circ \phi_x(x) := \operatorname{sinc}\left[\frac{\phi_x(x) - ih_x}{h_x}\right],$$

$$\theta_j(t) = \begin{cases} \dfrac{t+1}{t^2+1}, & j = -m_t - 1, \\ \mathcal{S}(j, h_t) \circ \phi_t(t), & -m_t \le j \le n_t; \end{cases}$$

see [15] for details. Here, $\phi_x(x)$ and $\phi_t(t)$ are the restrictions of the conformal mapping $\phi_z(z)$ onto the real intervals (a, b) and $(0, +\infty)$, respectively, with $\phi_z(z)$ a mapping from a simply connected domain $\mathcal{D}$ onto

$$\mathcal{D}_d := \{z \mid z = x + \imath y, |y| < d, d > 0\},$$

with $\imath$ the imaginary unit. We remark that the first and second derivatives of $\phi_z(z)$ with respect to the variable z will be denoted as $\phi_z^{'}(z)$ and $\phi_z^{''}(z)$, respectively.

The Galerkin method enables us to determine

$$u_{ij}, \quad -m_x - 1 \le i \le n_x + 1, \quad -m_t - 1 \le j \le n_t,$$

through solving the system of nonlinear equations

$$\langle \mathcal{P}^{(2)}\widehat{u} - f, \mathcal{S}_{kl} \rangle = 0,$$

where the inner product is defined as

$$\langle f, g \rangle = \int_0^{\infty} \int_a^b f(x,t)g(x,t)\omega_x(x)\omega_t(t)dxdt,$$

with $\omega_x(x)$ and $\omega_t(t)$ being two weighting functions with respect to the spatial and the temporal variables, respectively, and $\mathcal{S}_{kl} = \mathcal{S}(k, h_x)\mathcal{S}(l, h_t)$; see [15].

The most distinctive feature of the Sinc basis is that it can lead to exponential decreasing rate of the error. Moreover, the decreasing rate is maintained when the solution of the boundary value problem has boundary singularities.

In the form of (1.1)–(1.2), the resulting Sinc–Galerkin system can be described as follows; see [15] for more details. Now, the mapping Ψ is given by

$$\Psi(\mathbf{u}) = (u_1^2, u_2^2, \ldots, u_n^2)^T, \quad \text{with} \quad \mathbf{u} = (u_1, u_2, \ldots, u_n)^T, \tag{1.4}$$

the matrices B and C are given by

$$B = (\varepsilon(T_x^{(2)} + D_x^{(1)}T_x^{(1)} + T_x^{(1)}D_x^{(1)} + D_x^{(2)})) \otimes Q_t + Q_x \otimes (D_t^{(3)}T_t^{(1)} + T_t^{(1)}D_t^{(3)} + D_t^{(4)}), \tag{1.5}$$

and

$$C = (D_x^{(3)}T_x^{(1)} + T_x^{(1)}D_x^{(3)} + D_x^{(4)}) \otimes Q_t, \tag{1.6}$$

where $T_z^{(i)}$ $(i = 1, 2$ and $z \in \{x, t\})$ are $(m_z + n_z + 1)$-by-$(m_z + n_z + 1)$ Toeplitz matrices, with

$$T_z^{(1)} = \begin{bmatrix} 0 & -1 & \frac{1}{2} & \cdots & \frac{(-1)^{m_z+n_z}}{m_z+n_z} \\ 1 & & & & \vdots \\ -\frac{1}{2} & & \ddots & & \frac{1}{2} \\ \vdots & & & & -1 \\ \frac{(-1)^{m_z+n_z}}{m_z+n_z} & \cdots & -\frac{1}{2} & 1 & 0 \end{bmatrix}, \tag{1.7}$$

$$T_z^{(2)} = \begin{bmatrix} \frac{\pi^2}{3} & -2 & \frac{2}{2^2} & \cdots & \frac{(-1)^{m_z+n_z}2}{(m_z+n_z)^2} \\ -2 & & & & \vdots \\ \frac{2}{2^2} & & \ddots & & \frac{2}{2^2} \\ \vdots & & & & -2 \\ \frac{(-1)^{m_z+n_z}2}{(m_z+n_z)^2} & \cdots & \frac{2}{2^2} & -2 & \frac{\pi^2}{3} \end{bmatrix}, \tag{1.8}$$

and $D_z^{(i)}$ and Q_z $(i = 1, 2, 3, 4$ and $z \in \{x, t\})$ are $(m_z + n_z + 1)$-by-$(m_z + n_z + 1)$ diagonal matrices, with

$$D_z^{(1)} = \frac{h_z}{2} \cdot \operatorname{diag}\left[\left\{-\frac{\phi_z''(z)}{(\phi_z'(z))^2} - \frac{2\omega_z'(z)}{\phi'(z)\omega_z(z)}\right\}_{z=-m_z}^{n_z}\right], \tag{1.9}$$

$$D_z^{(2)} = \frac{h_z^2}{2} \cdot \operatorname{diag}\left[\left\{-\frac{\omega_z''(z)}{(\phi_z'(z))^2\omega_z(z)}\right\}_{z=-m_z}^{n_z}\right], \tag{1.10}$$

$$D_z^{(3)} = \frac{h_z}{2} \cdot \operatorname{diag}\left[\{-\omega_z(z)\}_{z=-m_z}^{n_z}\right], \tag{1.11}$$

$$D_z^{(4)} = \frac{h_z^2}{2} \cdot \operatorname{diag}\left[\left\{-\frac{\omega_z'(z)}{\phi_z'(z)}\right\}_{z=-m_z}^{n_z}\right], \tag{1.12}$$

and

$$Q_z = \operatorname{diag}\left[\left\{\frac{\omega_z(z)}{\phi_z'(z)}\right\}_{z=-m_z}^{n_z}\right]. \tag{1.13}$$

There are two classes of methods for computing the solution of the system of nonlinear equations (1.1). One casts this problem as a general nonlinear system and uses a nonlinear solver such as the Newton method [15] to solve it, and the other considers this problem as a special system of nonlinear equations

$$B\mathbf{u} = \mathbf{b} - C\mathbf{u} \odot \mathbf{u},$$

where $\odot$ is the entrywise multiplication, and uses the fixed-point method to solve it. It has been shown in [15] that the spectral radius of the inverse of the coefficient matrix B is bounded above by $1/(4\varepsilon)$. Evidently, as ε decreases, the iterative method used to solve this nonlinear system deteriorates. See also [1, 2, 3, 4, 8, 9, 10, 21, 22] and references therein for sequential and parallel matrix splitting iteration methods for solving the system of nonlinear equations (1.1).

For both Newton and fixed-point iterations employed to solve the system of nonlinear equations (1.1), at each step we need to solve a subsystem of linear equations of the form

$$(B + CD)\mathbf{z} = \mathbf{r}, \tag{1.14}$$

where D is a diagonal matrix. A straightforward application of the Gaussian elimination to the linear system (1.14) will result in an algorithm of $\mathcal{O}(n^6)$ complexity, where n is the size of the matrix $T_z^{(i)}$ $(i = 1, 2)$, $D_z^{(i)}$ $(i = 1, 2, 3, 4)$, or Q_z, $z \in \{x, t\}$. For an n-by-n Toeplitz linear system, fast direct solvers of complexity $\mathcal{O}(n^2)$ have been developed; see, for instance, Levinson [14]. However, there does not exist fast direct solver for Toeplitz-plus-diagonal linear systems yet, since the displacement rank of a Toeplitz-plus-diagonal matrix can take any value between 0 and n. Hence, fast direct Toeplitz solvers that are based on small displacement rank matrices are not applicable to the Toeplitz-plus-diagonal linear systems or to the block Toeplitz-like-plus-diagonal linear systems. For details about displacement ranks, we refer the reader to Kailath and Sayed [13].

However, we note that for any n^2-vector $\mathbf{q}$, the matrix-vector product $(B+CD)\mathbf{q}$ can be computed in $\mathcal{O}(n^2 \log n)$ operations. In fact, for $i = 1, 2$ and $z \in \{x, t\}$ the matrix-vector product $T_z^{(i)}\mathbf{q}$ can be obtained by first embedding $T_z^{(i)}$ into a $2n$-by-$2n$ circulant matrix and then using [illegible] (FFTs) to compute it [12]. Since for all $i \in \{1, 2, 3, 4\}$ and $z \in \{x, t\}$, $D_z^{(i)}$ is a diagonal matrix, the matrix-vector product $D_z^{(i)}\mathbf{q}$ can be computed in $\mathcal{O}(n^2)$ operations. Thus, iterative methods such as GMRES [20] can be employed to economically solve the linear system (1.14). Usually, in order to accelerate the convergence speed of GMRES, we need to precondition the linear system (1.14) by an approximate matrix with respect to the coefficient matrix $B + CD$. Therefore, to solve the original linear system, we turn to solving the preconditioned linear system instead; see [17, 7] and the references therein.

The outline of this paper is as follows. In section 2, we demonstrate some basic properties about the system of nonlinear equations (1.1). In section 3, we construct preconditioners for the coefficient matrix of the linear system (1.14); in addition, we study preconditioning properties of the preconditioned matrix and show that its eigenvalues are tightly and uniformly bounded in a rectangular on the complex plane independent of the size of the matrix. Several numerical examples are used to show the effectiveness of the proposed methods in section 4. Finally, in section 5, we end this paper with some concluding remarks.

2. Preliminary results. In this section, we give a few theoretical properties of the system of nonlinear equations (1.1).

THEOREM 2.1. [illegible] $B \in \mathbb{R}^{n\times n}$ [illegible] $\Psi : \mathbb{R}^n \to \mathbb{R}^n$ [illegible]

$$\|\Psi(\mathbf{u})\| \le \gamma \|\mathbf{u}\|^p \quad \forall \mathbf{u} \in \mathcal{U}_\delta,$$

[illegible] γ [illegible] $p \ge 2$ [illegible] $\|\cdot\|$ [illegible]
[illegible] $\mathcal{U}_\delta$ [illegible] δ [illegible]

$$\alpha = \|B^{-1}C\| \qquad \beta = \|B^{-1}b\|,$$

[illegible]

$$\alpha\beta^{p-1}\gamma < (p-1)^{p-1}p^{-p}.$$

[illegible] $\mathbf{u}_* \in \mathcal{U}_\delta$ [illegible] (1.1), [illegible] $\mathbf{u}_* \in \mathcal{U}_{\delta_0}$
[illegible] δ_0 [illegible]

$$\alpha\gamma t^p - t + \beta = 0.$$

[illegible]. Define

$$g(t) = \alpha\gamma t^p - t + \beta$$

and

$$t_0 = (\alpha\gamma p)^{-\frac{1}{p-1}}.$$

Then we have

$$\begin{cases} g(0) = \beta > 0, & g(t_0) = (\alpha\gamma)^{-\frac{1}{p-1}}[(\alpha\beta^{p-1}\gamma)^{\frac{1}{p-1}} - (p-1)p^{-\frac{p}{p-1}}] < 0, \\ g'(t_0) = 0, & g'(t)(t-t_0) > 0 \quad \forall t \in (0,+\infty). \end{cases}$$

It follows that $g(t)$ has two positive roots; we denote the smallest one as δ_0. Therefore, $g(t) \ge 0$ for $t \in [0, \delta_0]$. Since $\mathbf{u}_* \in \mathcal{U}_\delta$ is a solution of the system of nonlinear equations (1.1), it holds that

$$B\mathbf{u}_* + C\Psi(\mathbf{u}_*) = \mathbf{b},$$

or equivalently,

$$\mathbf{u}_* + B^{-1}C\Psi(\mathbf{u}_*) = B^{-1}\mathbf{b}.$$

Therefore,

$$\|B^{-1}b\| \ge \|\mathbf{u}_*\| - \|B^{-1}C\|\|\Psi(\mathbf{u}_*)\| \ge \|\mathbf{u}_*\| - \gamma\|B^{-1}C\|\|\mathbf{u}_*\|^p,$$

i.e.,

$$\alpha\gamma\|\mathbf{u}_*\|^p - \|\mathbf{u}_*\| + \|B^{-1}b\| \ge 0.$$

We immediately know that this inequality holds when $\|\mathbf{u}_*\| \le \delta_0$ or $\mathbf{u}_* \in \mathcal{U}_{\delta_0}$. ∎

COROLLARY 2.2. 2.1 $\Psi(\mathbf{u}) = (u_1^2, u_2^2, \ldots, u_n^2)^T$ $\mathbf{u} = (u_1, u_2, \ldots, u_n)^T$ $\gamma = 1$

$$\delta_0 = \frac{1}{2\alpha}\left[1 - \sqrt{1 - 4\alpha\beta}\right],$$

$\alpha\beta < \frac{1}{4}$

THEOREM 2.3. $B \in \mathbb{R}^{n\times n}$ $\alpha = \|B^{-1}C\|$ $\beta = \|B^{-1}b\|$ $\|\cdot\|$ $\Psi : \mathbb{R}^n \to \mathbb{R}^n$

(i) $\|\Psi(\mathbf{u})\| \le \gamma\|\mathbf{u}\|^p \ \forall \mathbf{u} \in \mathcal{U}_\delta$ $\mathcal{U}_\delta$ δ γ $p \ge 2$

$$\alpha\beta^{p-1}\gamma < (p-1)^{p-1}p^{-p};$$

(ii) $\|\Psi(\mathbf{u}) - \Psi(\mathbf{v})\| \le \tau\|\mathbf{u} - \mathbf{v}\| \ \forall \mathbf{u}, \mathbf{v} \in \mathcal{U}_{\delta_0}$ $\tau > 0$ δ_0

$$\alpha\gamma t^p - t + \beta = 0.$$

$\alpha\tau < 1$ (1.1) $\mathbf{u}_* \in \mathcal{U}_{\delta_0}$. Define $\mathbf{G} : \mathcal{U}_{\delta_0} \subset \mathbb{R}^n \to \mathbb{R}^n$ as

$$\mathbf{G}(\mathbf{u}) = B^{-1}b - B^{-1}C\Psi(\mathbf{u}).$$

Then $\mathbf{G}$ is a contraction mapping defined on $\mathcal{U}_{\delta_0}$ as

$$\|\mathbf{G}(\mathbf{u}) - \mathbf{G}(\mathbf{v})\| = \|B^{-1}C[\Psi(\mathbf{u}) - \Psi(\mathbf{v})]\| \le \alpha\tau\|\mathbf{u} - \mathbf{v}\|.$$

Therefore, $\mathbf{G}$ has a unique fixed point $\mathbf{u}_*$ in $\mathcal{U}_{\delta_0}$. It then follows from Theorem 2.1 that the system of nonlinear equations (1.1) has a unique solution $\mathbf{u}_*$ in $\mathcal{U}_{\delta_0}$. □

COROLLARY 2.4. 2.3 $\alpha\beta < \frac{1}{4}$ $\Psi(\mathbf{u}) = (u_1^2, u_2^2, \ldots, u_n^2)^T$ $\mathbf{u} = (u_1, u_2, \ldots, u_n)^T$

$$\alpha\tau \le 2\alpha\delta_0 = 1 - \sqrt{1 - 4\alpha\beta} < 1.$$

$$B\mathbf{u} + C\mathbf{u} \odot \mathbf{u} = \mathbf{b}$$

$\mathbf{u}_*$ $\mathcal{U}_{\delta_0}$

3. The preconditioners. In this section, we will solve the system of nonlinear equations

$$B\mathbf{u} + C\mathbf{u} \odot \mathbf{u} = \mathbf{b} \tag{3.1}$$

by the Newton or the fixed-point iteration method.

3.1. The Newton and fixed-point iterations. Let $\Omega \in \mathbb{R}^{n\times n}$ be a diagonal matrix such that $B + C\Omega$ is nonsingular. Then we consider the following Newton iteration for solving the system of nonlinear equations (3.1):

$$\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} - (B + 2C\,\mathrm{diag}(\mathbf{u}^k))^{-1}(B\mathbf{u}^{(k)} + C\mathbf{u}^{(k)} \odot \mathbf{u}^{(k)} - \mathbf{b}), \quad k = 0, 1, 2, \ldots.$$

If $\mathbf{u}_* \in \mathcal{U}_{\delta_0}$ is a solution of the nonlinear system (3.1), then it holds that $\{\mathbf{u}^{(k)}\} \subset \mathcal{U}_{\delta_0}$ and $\lim_{k\to\infty} \mathbf{u}^{(k)} = \mathbf{u}_*$, with the convergence speed being at least quadratic, provided $\mathbf{u}^{(0)}$ is in a neighborhood of $\mathbf{u}_*$.

The fixed-point iteration for the system of nonlinear equations (3.1) is as follows:

$$(B + C\Omega)\mathbf{u}^{(k+1)} = C[\Omega\mathbf{u}^{(k)} - \mathbf{u}^{(k)} \odot \mathbf{u}^{(k)}] + \mathbf{b}, \quad k = 0, 1, 2, \ldots,$$

or equivalently,

$$\mathbf{u}^{(k+1)} = (B + C\Omega)^{-1}C[\Omega\mathbf{u}^{(k)} - \mathbf{u}^{(k)} \odot \mathbf{u}^{(k)}] + (B + C\Omega)^{-1}\mathbf{b}, \quad k = 0, 1, 2, \ldots.$$

If $\mathbf{u}_* \in \mathcal{U}_{\delta_0}$ is a solution of the nonlinear system (3.1), then it holds that $\{\mathbf{u}^{(k)}\} \subset \mathcal{U}_{\delta_0}$ and $\lim_{k\to\infty} \mathbf{u}^{(k)} = \mathbf{u}_*$, provided $\mathbf{u}^{(0)}$ is in a neighborhood of $\mathbf{u}_*$ and

$$\rho((B + C\Omega)^{-1}C(\Omega - 2\ \mathrm{diag}(\mathbf{u}_*))) < 1,$$

where $\mathrm{diag}(\mathbf{u}_*)$ is the diagonal matrix with its ith diagonal entries being $[\mathbf{u}_*]_i$, and $\rho(\cdot)$ denotes the spectral radius of the corresponding matrix. See [19] for details.

3.2. Construction of the preconditioners. For both fixed-point and Newton iterations, at each step we need to solve a subsystem of linear equations of the form

$$A\mathbf{z} \equiv (B + CD)\mathbf{z} = \mathbf{r}. \tag{3.2}$$

This kind of linear system may be efficiently solved by the preconditioned GMRES method. The key point here is constructing a high-quality preconditioner M for the coefficient matrix A by making use of its concrete properties and special structure.

Consider the system of nonlinear equations (1.1) with the function $\Psi(\mathbf{u})$ as given in (1.4) and the matrices B and C as given in (1.5) and (1.6), respectively, where $T_z^{(i)}$ and $D_z^{(i)}$ ($i = 1, 2, 3, 4$ and $z \in \{x, t\}$), and Q_z ($z \in \{x, t\}$) are defined as in (1.7)–(1.13). Let $\Omega = \mathrm{diag}(\omega_{\mathbf{i}})$ be a positive definite matrix such that $D := I \otimes \Omega$ is an approximation to the Jacobian of $\Psi(\mathbf{u})$, i.e., $D\mathbf{u} \approx \boldsymbol{\Psi}(\mathbf{u})$. Then the target matrix under consideration is

$$\begin{aligned} A &= B + CD \\ &= \varepsilon(T_x^{(2)} + D_x^{(1)}T_x^{(1)} + T_x^{(1)}D_x^{(1)} + D_x^{(2)}) \otimes Q_t \\ &\quad + Q_x \otimes (D_t^{(3)}T_t^{(1)} + T_t^{(1)}D_t^{(3)} + D_t^{(4)}) \\ &\quad + (D_x^{(3)}T_x^{(1)} + T_x^{(1)}D_x^{(3)} + D_x^{(4)}) \otimes (Q_t\Omega). \end{aligned} \tag{3.3}$$

By utilizing the special structure of the matrix A, we can construct its preconditioner M as

$$\begin{aligned} M &= \hat{B} + \hat{C}D \\ &= \varepsilon(B_x^{(2)} + D_x^{(1)}B_x^{(1)} + B_x^{(1)}D_x^{(1)} + D_x^{(2)}) \otimes Q_t \\ &\quad + Q_x \otimes (D_t^{(3)}B_t^{(1)} + B_t^{(1)}D_t^{(3)} + D_t^{(4)}) \\ &\quad + (D_x^{(3)}B_x^{(1)} + B_x^{(1)}D_x^{(3)} + D_x^{(4)}) \otimes (Q_t\Omega), \end{aligned} \tag{3.4}$$

where

$$\hat{B} = (\varepsilon(B_x^{(2)} + D_x^{(1)}B_x^{(1)} + B_x^{(1)}D_x^{(1)} + D_x^{(2)})) \otimes Q_t + Q_x \otimes (D_t^{(3)}B_t^{(1)} + B_t^{(1)}D_t^{(3)} + D_t^{(4)})$$

and

$$\hat{C} = (D_x^{(3)}B_x^{(1)} + B_x^{(1)}D_x^{(3)} + D_x^{(4)}) \otimes Q_t,$$

and, for $z \in \{x, t\}$,

$$B_z^{(1)} = \text{tridiag}\left[\frac{1}{2}, 0, -\frac{1}{2}\right] \quad \text{and} \quad B_z^{(2)} = \text{tridiag}\,[-1, 2, -1]$$

are tridiagonal approximations to $T_z^{(1)}$ and $T_z^{(2)}$, respectively.

We remark that the preconditioner M is a block tridiagonal matrix, and is usually of mild size because, compared with the finite-difference system, the Sinc–Galerkin system need not be very large in order to achieve the same discretization accuracy [16, 18]. Therefore, for any given vector $\mathbf{r}$, the generalized residual equation $M\mathbf{w} = \mathbf{r}$ involved in the preconditioned GMRES iteration method can be solved in $\mathcal{O}(N_x N_t^2)$ or $\mathcal{O}(N_x^2 N_t)$ operations by using a variety of linear solvers such as the sparse direct methods, where $N_z = m_z + n_z + 1$, with $z \in \{x, t\}$; see also [6, 5] and the references therein.

3.3. Analysis of the preconditioning matrix. In this subsection, we will demonstrate the positive definiteness of the matrix A defined in (3.3) and its preconditioning matrix M defined in (3.4). In addition, we will derive precise bounds for the eigenvalues of the preconditioned matrix $M^{-1}A$ by making use of the generalized Bendixson theorem established in [7].

To this end, in what follows we use $(\cdot)^*$ to denote the conjugate transpose of either a vector or a square matrix. For a given square matrix X, we use $\mathcal{H}(X)$ and $\mathcal{S}(X)$ to denote, respectively, its Hermitian and skew-Hermitian parts [6], and use $\lambda(X)$ to denote its spectral set. In particular, when X is Hermitian or real symmetric, we use $\lambda_{\max}(X)$ to represent its largest eigenvalue.

THEOREM 3.1. $D_z^{(i)}$ $(i = 2, 4$ $z \in \{x, t\})$ $\mathcal{H}(A)$ $\mathcal{H}(M)$ [illegible] 1 A M [illegible]. The Hermitian and the skew-Hermitian parts of A and M are

$$\mathcal{H}(A) = \frac{1}{2}(A + A^*) = \varepsilon(T_x^{(2)} + D_x^{(2)}) \otimes Q_t + Q_x \otimes D_t^{(4)} + D_x^{(4)} \otimes (Q_t\Omega),$$

$$\mathcal{S}(A) = \frac{1}{2}(A - A^*) = \varepsilon(D_x^{(1)}T_x^{(1)} + T_x^{(1)}D_x^{(1)}) \otimes Q_t + Q_x \otimes (D_t^{(3)}T_t^{(1)} + T_t^{(1)}D_t^{(3)}) + (D_x^{(3)}T_x^{(1)} + T_x^{(1)}D_x^{(3)}) \otimes (Q_t\Omega),$$

[1] A matrix is positive definite if its Hermitian part is positive definite. Note that a positive definite matrix is not necessarily Hermitian; see [6, 5].

and

$$\begin{aligned}\mathcal{H}(M) &= \frac{1}{2}(M + M^*)\\ &= \varepsilon(B_x^{(2)} + D_x^{(2)}) \otimes Q_t + Q_x \otimes D_t^{(4)} + D_x^{(4)} \otimes (Q_t\Omega),\end{aligned}$$

$$\begin{aligned}\mathcal{S}(M) &= \frac{1}{2}(M - M^*)\\ &= \varepsilon(D_x^{(1)}B_x^{(1)} + B_x^{(1)}D_x^{(1)}) \otimes Q_t + Q_x \otimes (D_t^{(3)}B_t^{(1)} + B_t^{(1)}D_t^{(3)})\\ &\quad +(D_x^{(3)}B_x^{(1)} + B_x^{(1)}D_x^{(3)}) \otimes (Q_t\Omega).\end{aligned}$$

Because the diagonal matrices $D_z^{(i)}$ ($i = 2, 4$ and $z \in \{x, t\}$) are positive semidefinite, and the Toeplitz matrices $T_x^{(2)}$ and the diagonal matrices Q_z ($z \in \{x, t\}$) are symmetric positive definite [12], we know that $\mathcal{H}(A)$ is symmetric positive definite. Therefore, A is a positive definite matrix and, thus, is nonsingular.

By applying the same arguments to the preconditioning matrix M, we can immediately show that M is positive definite and nonsingular, too. □

The following [illegible], established in [7], is essential for us to derive a rectangular domain for bounding the eigenvalues of the preconditioned matrix $M^{-1}A$.

THEOREM 3.2 ([7, Theorem 2.4]). [illegible] $A, M \in C^{n\times n}$ [illegible] n [illegible] $v \in C^n \setminus \{0\}$ $v^*\mathcal{H}(A)v \neq 0$ [illegible] $v^*\mathcal{H}(M)v \neq 0$ [illegible] $h(v)$ $f_A(v)$ [illegible] $f_M(v)$ [illegible]

$$h(v) = \frac{v^*\mathcal{H}(A)v}{v^*\mathcal{H}(M)v}, \quad f_A(v) = \frac{1}{i} \cdot \frac{v^*\mathcal{S}(A)v}{v^*\mathcal{H}(A)v}, \quad \text{and} \quad f_M(v) = \frac{1}{i} \cdot \frac{v^*\mathcal{S}(M)v}{v^*\mathcal{H}(M)v}.$$

[illegible] γ_1 [illegible] γ_2 [illegible]

$$\gamma_1 \leq h(v) \leq \gamma_2 \quad \forall\, v \in C^n \setminus \{0\},$$

[illegible] η [illegible] μ [illegible]

$$-\eta \leq f_A(v) \leq \eta \quad \text{and} \quad -\mu \leq f_M(v) \leq \mu \quad \forall\, v \in C^n \setminus \{0\}.$$

[illegible] $\eta\mu \leq 1$ [illegible]

$$\begin{cases} \frac{(1-\eta\mu)\gamma_1}{1+\mu^2} \leq \ \mathrm{Re}(\lambda(M^{-1}A)) \leq (1+\eta\mu)\gamma_2, \\ -(\eta+\mu)\gamma_2 \leq \ \mathrm{Im}(\lambda(M^{-1}A)) \leq (\eta+\mu)\gamma_2. \end{cases}$$

[illegible] $\mathrm{Re}(\cdot)$ [illegible] $\mathrm{Im}(\cdot)$ [illegible]

In [7] the following estimates about the bounds of the function $h(v)$ have been derived.

LEMMA 3.3 (see [7, Lemma 4.2]). [illegible] $D_z^{(i)}(i = 2, 4$ [illegible] $z \in \{x, t\})$ [illegible]

$$1 \leq \frac{v^*\mathcal{H}(A)v}{v^*\mathcal{H}(M)v} \leq \frac{\pi^2}{4} \quad \forall v \in \mathbb{C}^n \setminus \{0\}.$$

To obtain the bounds for the functions $f_A(v)$ and $f_M(v)$, we need to establish the following estimates.

LEMMA 3.4. [illegible] $D_z^{(i)}(i = 2, 4 \quad z \in \{x, t\}), Q_z(z \in \{x, t\})$ [illegible] Ω [illegible] $z \in \{x, t\}$ [illegible] $N_z = m_z + n_z + 1$ [illegible]

$$d_z^{(1)} = \max_{1 \le j \le N_z} \{|[D_z^{(1)}]_{jj}|, \ |[D_z^{(3)}]_{jj}|\}, \quad d_z^{(2)} = \min_{1 \le j \le N_z} \{|[D_z^{(2)}]_{jj}|, \ |[D_z^{(4)}]_{jj}|\},$$

$$\mu^{(1)} = \max_{z \in \{x,t\}} \left\{ \frac{2\pi d_z^{(1)}}{\sqrt{(\pi^2 + d_z^{(2)}) d_z^{(2)}}}, \quad \frac{2\pi d_z^{(1)}}{d_z^{(2)}} \right\},$$

$$\mu^{(2)} = \max_{z \in \{x,t\}} \left\{ d_z^{(1)} \left(\sqrt{1 + \frac{4}{d_z^{(2)}}} - 1 \right), \quad \frac{2 d_z^{(1)}}{d_z^{(2)}} \right\}.$$

[illegible] $v \in \mathbb{C}^n \setminus \{0\}$ [illegible]

$$\left| \frac{v^*[(D_x^{(1)} T_x^{(1)} + T_x^{(1)} D_x^{(1)}) \otimes Q_t] v}{v^*[(T_x^{(2)} + D_x^{(2)}) \otimes Q_t] v} \right| \le \mu^{(1)}, \tag{3.5}$$

$$\left| \frac{v^*[(D_x^{(1)} B_x^{(1)} + B_x^{(1)} D_x^{(1)}) \otimes Q_t] v}{v^*[(B_x^{(2)} + D_x^{(2)}) \otimes Q_t] v} \right| \le \mu^{(2)}, \tag{3.6}$$

$$\left| \frac{v^*[Q_x \otimes (D_t^{(3)} T_t^{(1)} + T_t^{(1)} D_t^{(3)})] v}{v^*[Q_x \otimes D_t^{(4)}] v} \right| \le \mu^{(1)}, \tag{3.7}$$

$$\left| \frac{v^*[Q_x \otimes (D_t^{(3)} B_t^{(1)} + B_t^{(1)} D_t^{(3)})] v}{v^*[Q_x \otimes D_t^{(4)}] v} \right| \le \mu^{(2)}, \tag{3.8}$$

$$\left| \frac{v^*[(D_x^{(3)} T_x^{(1)} + T_x^{(1)} D_x^{(3)}) \otimes (Q_t \Omega)] v}{v^*[D_x^{(4)} \otimes (Q_t \Omega)] v} \right| \le \mu^{(1)}, \tag{3.9}$$

$$\left| \frac{v^*[(D_x^{(3)} B_x^{(1)} + B_x^{(1)} D_x^{(3)}) \otimes (Q_t \Omega)] v}{v^*[D_x^{(4)} \otimes (Q_t \Omega)] v} \right| \le \mu^{(2)}. \tag{3.10}$$

[illegible] We first prove (3.5) and (3.6). Because $D_x^{(1)} T_x^{(1)} + T_x^{(1)} D_x^{(1)}$ is a skew-symmetric matrix and $T_x^{(2)} + D_x^{(2)}$ is a symmetric positive definite matrix, for all

$v \neq 0$, by direct computations we can obtain

$$\begin{aligned}
&\left| \frac{v^*[(D_x^{(1)}T_x^{(1)} + T_x^{(1)}D_x^{(1)}) \otimes Q_t]v}{v^*[(T_x^{(2)} + D_x^{(2)}) \otimes Q_t]v} \right| \\
&\quad \leq \max_{v \neq 0} \left\{ \left| \frac{\frac{1}{\imath} v^*[(D_x^{(1)}T_x^{(1)} + T_x^{(1)}D_x^{(1)}) \otimes Q_t]v}{v^*[(T_x^{(2)} + D_x^{(2)}) \otimes Q_t]v} \right| \right\} \\
&\quad \leq \left| \lambda_{\max} \left(\frac{1}{\imath} (T_x^{(2)} + D_x^{(2)})^{-1/2} (D_x^{(1)}T_x^{(1)} + T_x^{(1)}D_x^{(1)}) (T_x^{(2)} + D_x^{(2)})^{-1/2} \right) \right| \\
&\quad \leq \left\| (T_x^{(2)} + D_x^{(2)})^{-1/2} (D_x^{(1)}T_x^{(1)} + T_x^{(1)}D_x^{(1)}) (T_x^{(2)} + D_x^{(2)})^{-1/2} \right\|_2 \\
&\quad \leq \left\| (T_x^{(2)} + D_x^{(2)})^{-1/2} D_x^{(1)}T_x^{(1)} (T_x^{(2)} + D_x^{(2)})^{-1/2} \right\|_2 \\
&\qquad + \left\| (T_x^{(2)} + D_x^{(2)})^{-1/2} T_x^{(1)}D_x^{(1)} (T_x^{(2)} + D_x^{(2)})^{-1/2} \right\|_2 \\
&\quad \leq \left\| (T_x^{(2)} + D_x^{(2)})^{-1/2} (T_x^{(2)} + d_x^{(2)}I)^{1/2} \right\|_2 \cdot \left\| (T_x^{(2)} + d_x^{(2)}I)^{-1/2} \right\|_2 \\
&\qquad \cdot \left\| D_x^{(1)} \right\|_2 \cdot \left\| (T_x^{(2)} + d_x^{(2)}I)^{1/2} (T_x^{(2)} + D_x^{(2)})^{-1/2} \right\|_2 \\
&\qquad \cdot \left[\left\| T_x^{(1)} (T_x^{(2)} + d_x^{(2)}I)^{-1/2} \right\|_2 + \left\| (T_x^{(2)} + d_x^{(2)}I)^{-1/2} T_x^{(1)} \right\|_2 \right] \\
&\quad \leq \left\| (T_x^{(2)} + d_x^{(2)}I)^{-1/2} \right\|_2 \cdot \left\| D_x^{(1)} \right\|_2 \\
&\qquad \cdot \left[\left\| T_x^{(1)} (T_x^{(2)} + d_x^{(2)}I)^{-1/2} \right\|_2 + \left\| (T_x^{(2)} + d_x^{(2)}I)^{-1/2} T_x^{(1)} \right\|_2 \right]. \qquad (3.11)
\end{aligned}$$

Here, we have applied the fact

$$\begin{aligned}
&\left\| (T_x^{(2)} + D_x^{(2)})^{-1/2} (T_x^{(2)} + d_x^{(2)}I)^{1/2} \right\|_2^2 \\
&\quad = \left\| (T_x^{(2)} + d_x^{(2)}I)^{1/2} (T_x^{(2)} + D_x^{(2)})^{-1/2} \right\|_2^2 \\
&\quad = \lambda_{\max} \left((T_x^{(2)} + D_x^{(2)})^{-1/2} (T_x^{(2)} + d_x^{(2)}I) (T_x^{(2)} + D_x^{(2)})^{-1/2} \right) \\
&\quad = \max_{w \neq 0} \left\{ \frac{w^*(T_x^{(2)} + D_x^{(2)})^{-1/2} (T_x^{(2)} + d_x^{(2)}I) (T_x^{(2)} + D_x^{(2)})^{-1/2} w}{w^*w} \right\} \\
&\quad = \max_{w \neq 0} \left\{ \frac{w^*(T_x^{(2)} + d_x^{(2)}I)w}{w^*(T_x^{(2)} + D_x^{(2)})w} \right\} \\
&\quad \leq \max \left\{ 1, \quad d_x^{(2)} \cdot \max_{w \neq 0} \frac{w^*w}{w^*D_x^{(2)}w} \right\} \\
&\quad \leq \ 1.
\end{aligned}$$

Now, we estimate the terms $\|(T_x^{(2)} + d_x^{(2)}I)^{-1/2}\|_2$, $\|T_x^{(1)}(T_x^{(2)} + d_x^{(2)}I)^{-1/2}\|_2$ and $\|(T_x^{(2)} + d_x^{(2)}I)^{-1/2}T_x^{(1)}\|_2$ involved in (3.11).

As the generating function of the Toeplitz matrix $T_x^{(2)}$ is θ^2, we have

$$\begin{aligned}\left\|(T_x^{(2)}+d_x^{(2)}I)^{-1/2}\right\|_2^2 &= \lambda_{\max}\left((T_x^{(2)}+d_x^{(2)}I)^{-1}\right)\\ &= \max_{w\neq 0}\left\{\frac{w^*(T_x^{(2)}+d_x^{(2)}I)^{-1}w}{w^*w}\right\}\\ &< \max_{-\pi\le\theta\le\pi}\left\{\frac{1}{\theta^2+d_x^{(2)}}\right\}\\ &= \frac{1}{d_x^{(2)}}.\end{aligned}$$

Therefore,

$$\left\|(T_x^{(2)}+d_x^{(2)}I)^{-1/2}\right\|_2 < \frac{1}{\sqrt{d_x^{(2)}}}.$$

In addition, we have

$$\begin{aligned}&\left\|T_x^{(1)}(T_x^{(2)}+d_x^{(2)}I)^{-1/2}\right\|_2^2\\ &= \left\|(T_x^{(2)}+d_x^{(2)}I)^{-1/2}T_x^{(1)}\right\|_2^2\\ &= \lambda_{\max}\left((T_x^{(2)}+d_x^{(2)}I)^{-1/2}T_x^{(1)}(T_x^{(1)})^*(T_x^{(2)}+d_x^{(2)}I)^{-1/2}\right)\\ &= \max_{w\neq 0}\left\{\frac{w^*(T_x^{(2)}+d_x^{(2)}I)^{-1/2}T_x^{(1)}(T_x^{(1)})^*(T_x^{(2)}+d_x^{(2)}I)^{-1/2}w}{w^*w}\right\}\\ &= \max_{w\neq 0}\left\{\frac{w^*T_x^{(1)}(T_x^{(1)})^*w}{w^*(T_x^{(2)}+d_x^{(2)}I)w}\right\}.\end{aligned}$$

By making use of Theorems 3.1 and 3.3 in [11], we know that for any $\epsilon>0$ there exist a positive semidefinite matrix R_x of fixed rank and a matrix E_x of small norm such that $\|E_x\|<d_x^{(2)}\epsilon$, and

$$T_x^{(1)}(T_x^{(1)})^*+R_x+E_x=T_x^{(3)},$$

where $T_x^{(3)}$ is the Toeplitz matrix generated by the positive function θ^2. We remark that the generating function of $T_x^{(1)}$ is $\imath\theta$ and $T_x^{(1)}$ is a skew-symmetric matrix. Because

$$\frac{w^*R_xw}{w^*(T_x^{(2)}+d_x^{(2)}I)w}\ge 0 \quad\text{and}\quad \left|\frac{w^*E_xw}{w^*(T_x^{(2)}+d_x^{(2)}I)w}\right|\le\epsilon \quad \forall w\neq 0,$$

it follows from the above matrix decomposition that

$$\max_{w\neq 0}\left\{\frac{w^*T_x^{(1)}(T_x^{(1)})^*w}{w^*(T_x^{(2)}+d_x^{(2)}I)w}\right\} < \max_{w\neq 0}\left\{\frac{w^*T_x^{(3)}w}{w^*(T_x^{(2)}+d_x^{(2)}I)w}\right\}+\epsilon.$$

Since ϵ can be arbitrarily small, it holds that

$$\begin{aligned}\max_{w\neq 0}\left\{\frac{w^*T_x^{(1)}(T_x^{(1)})^*w}{w^*(T_x^{(2)}+d_x^{(2)}I)w}\right\} &\le \max_{w\neq 0}\left\{\frac{w^*T_x^{(3)}w}{w^*(T_x^{(2)}+d_x^{(2)}I)w}\right\}\\ &< \max_{-\pi\le\theta\le\pi}\left\{\frac{\theta^2}{\theta^2+d_x^{(2)}}\right\}\\ &= \frac{\pi^2}{\pi^2+d_x^{(2)}}. \end{aligned}\tag{3.12}$$

Therefore,

$$\left\|T_x^{(1)}(T_x^{(2)}+d_x^{(2)}I)^{-1/2}\right\|_2 = \left\|(T_x^{(2)}+d_x^{(2)}I)^{-1/2}T_x^{(1)}\right\|_2 < \frac{\pi}{\sqrt{\pi^2+d_x^{(2)}}}.$$

Based upon (3.11) we immediately have

$$\begin{aligned}\left|\frac{v^*[(D_x^{(1)}T_x^{(1)}+T_x^{(1)}D_x^{(1)})\otimes Q_t]v}{v^*[(T_x^{(2)}+D_x^{(2)})\otimes Q_t]v}\right| &\le \frac{1}{\sqrt{d_x^{(2)}}}\cdot d_x^{(1)}\cdot\frac{2\pi}{\sqrt{\pi^2+d_x^{(2)}}}\\ &= \frac{2\pi d_x^{(1)}}{\sqrt{(\pi^2+d_x^{(2)})d_x^{(2)}}}\\ &\le \mu^{(1)}.\end{aligned}$$

This shows that (3.5) holds true.

Analogously to (3.11), for all $v\neq 0$ we can obtain

$$\begin{aligned}&\left|\frac{v^*[(D_x^{(1)}B_x^{(1)}+B_x^{(1)}D_x^{(1)})\otimes Q_t]v}{v^*[(B_x^{(2)}+D_x^{(2)})\otimes Q_t]v}\right|\\ &\quad\le \left\|(B_x^{(2)}+d_x^{(2)}I)^{-1/2}\right\|_2\cdot\left\|D_x^{(1)}\right\|_2\\ &\quad\quad\cdot\left[\left\|B_x^{(1)}(B_x^{(2)}+d_x^{(2)}I)^{-1/2}\right\|_2+\left\|(B_x^{(2)}+d_x^{(2)}I)^{-1/2}B_x^{(1)}\right\|_2\right].\end{aligned}\tag{3.13}$$

Here, we have applied the fact

$$\left\|(B_x^{(2)}+D_x^{(2)})^{-1/2}(B_x^{(2)}+d_x^{(2)}I)^{1/2}\right\|_2^2 = \left\|(B_x^{(2)}+d_x^{(2)}I)^{1/2}(B_x^{(2)}+D_x^{(2)})^{-1/2}\right\|_2^2 \le 1.$$

Now, we estimate the terms $\|(B_x^{(2)}+d_x^{(2)}I)^{-1/2}\|_2$, $\|B_x^{(1)}(B_x^{(2)}+d_x^{(2)}I)^{-1/2}\|_2$, and $\|(B_x^{(2)}+d_x^{(2)}I)^{-1/2}B_x^{(1)}\|_2$ involved in (3.13).

As the generating function of the Toeplitz matrix $B_x^{(2)}$ is $2-2\cos\theta$, we have

$$\begin{aligned}\left\|(B_x^{(2)}+d_x^{(2)}I)^{-1/2}\right\|_2^2 &= \lambda_{\max}\left((B_x^{(2)}+d_x^{(2)}I)^{-1}\right)\\ &= \max_{w\neq 0}\left\{\frac{w^*(B_x^{(2)}+d_x^{(2)}I)^{-1}w}{w^*w}\right\}\\ &< \max_{-\pi\le\theta\le\pi}\left\{\frac{1}{2(1-\cos\theta)+d_x^{(2)}}\right\}\\ &= \frac{1}{d_x^{(2)}}.\end{aligned}$$

Therefore,

$$\left\|(B_x^{(2)} + d_x^{(2)} I)^{-1/2}\right\|_2 < \frac{1}{\sqrt{d_x^{(2)}}}.$$

By making use of Theorem 3.1 in [11] again, we know that there exists a positive semidefinite matrix R_x of fixed rank such that

$$B_x^{(1)}(B_x^{(1)})^* + R_x = B_x^{(3)},$$

where $B_x^{(3)}$ is the Toeplitz matrix generated by the positive function $\sin^2\theta$. We remark that the generating function of $B_x^{(1)}$ is $\imath \sin\theta$ and $B_x^{(1)}$ is a skew-symmetric matrix. Because

$$\frac{w^* R_x w}{w^*(B_x^{(2)} + d_x^{(2)} I)w} \geq 0 \quad \forall w \neq 0,$$

it follows from the above matrix decomposition that

$$\begin{aligned}
&\left\|B_x^{(1)}(B_x^{(2)} + d_x^{(2)} I)^{-1/2}\right\|_2^2 \\
&\quad = \left\|(B_x^{(2)} + d_x^{(2)} I)^{-1/2} B_x^{(1)}\right\|_2^2 \\
&\quad = \lambda_{\max}\left((B_x^{(2)} + d_x^{(2)} I)^{-1/2} B_x^{(1)}(B_x^{(1)})^*(B_x^{(2)} + d_x^{(2)} I)^{-1/2}\right) \\
&\quad = \max_{w\neq 0}\left\{\frac{w^*(B_x^{(2)} + d_x^{(2)} I)^{-1/2} B_x^{(1)}(B_x^{(1)})^*(B_x^{(2)} + d_x^{(2)} I)^{-1/2} w}{w^* w}\right\} \\
&\quad = \max_{w\neq 0}\left\{\frac{w^* B_x^{(1)}(B_x^{(1)})^* w}{w^*(B_x^{(2)} + d_x^{(2)} I)w}\right\} \\
&\quad \leq \max_{w\neq 0}\left\{\frac{w^* B_x^{(3)} w}{w^*(B_x^{(2)} + d_x^{(2)} I)w}\right\} \\
&\quad < \max_{-\pi\leq\theta\leq\pi}\left\{\frac{\sin^2\theta}{2(1-\cos\theta) + d_x^{(2)}}\right\} \\
&\quad = 1 + \frac{d_x^{(2)}}{2} - \frac{1}{2}\sqrt{(d_x^{(2)} + 4) d_x^{(2)}} \\
&\quad = \frac{1}{4}\left(\sqrt{d_x^{(2)} + 4} - \sqrt{d_x^{(2)}}\right)^2. \qquad (3.14)
\end{aligned}$$

Therefore,

$$\left\|B_x^{(1)}(B_x^{(2)} + d_x^{(2)} I)^{-1/2}\right\|_2 = \left\|(B_x^{(2)} + d_x^{(2)} I)^{-1/2} B_x^{(1)}\right\|_2 < \frac{1}{2}\left(\sqrt{d_x^{(2)} + 4} - \sqrt{d_x^{(2)}}\right).$$

Based upon (3.13) we immediately have

$$\left|\frac{v^*[(D_x^{(1)}B_x^{(1)}+B_x^{(1)}D_x^{(1)})\otimes Q_t]v}{v^*[(B_x^{(2)}+D_x^{(2)})\otimes Q_t]v}\right| \le \frac{1}{\sqrt{d_x^{(2)}}}\cdot d_x^{(1)}\cdot\left(\sqrt{d_x^{(2)}+4}-\sqrt{d_x^{(2)}}\right)$$
$$= d_x^{(1)}\cdot\left(\sqrt{1+\frac{4}{d_x^{(2)}}}-1\right)$$
$$\le \mu^{(2)}.$$

This shows the validity of (3.6).

We remark that the last bound in (3.14) is obtained by using the maximum value of the one-variable function

$$\varphi(\theta)=\frac{\sin^2\theta}{2(1-\cos\theta)+d_x^{(2)}}$$

attained at the stationary point

$$\theta_*=\arccos\left(\frac{2+d_x^{(2)}-\sqrt{(d_x^{(2)})^2+4d_x^{(2)}}}{2}\right).$$

Now, we are going to prove (3.7)–(3.10). Analogously, for all $v\neq 0$, by straightforward computations we obtain

$$\left|\frac{v^*[Q_x\otimes(D_t^{(3)}T_t^{(1)}+T_t^{(1)}D_t^{(3)})]v}{v^*[Q_x\otimes D_t^{(4)}]v}\right|$$
$$\le \max_{v\neq 0}\left\{\left|\frac{\frac{1}{i}v^*[Q_x\otimes(D_t^{(3)}T_t^{(1)}+T_t^{(1)}D_t^{(3)})]v}{v^*[Q_x\otimes D_t^{(4)}]v}\right|\right\}$$
$$\le \left|\lambda_{\max}\left(\frac{1}{i}(D_t^{(4)})^{-\frac12}(D_t^{(3)}T_t^{(1)}+T_t^{(1)}D_t^{(3)})(D_t^{(4)})^{-\frac12}\right)\right|$$
$$\le \left\|(D_t^{(4)})^{-\frac12}(D_t^{(3)}T_t^{(1)}+T_t^{(1)}D_t^{(3)})(D_t^{(4)})^{-\frac12}\right\|_2$$
$$\le \left\|(D_t^{(4)})^{-\frac12}(d_t^{(2)}I)^{\frac12}\right\|_2\left\|(d_t^{(2)}I)^{-\frac12}\right\|_2\left\|D_t^{(3)}\right\|_2\left\|(d_t^{(2)}I)^{\frac12}(D_t^{(4)})^{-\frac12}\right\|_2$$
$$\cdot\left[\left\|T_t^{(1)}(d_t^{(2)}I)^{-\frac12}\right\|_2+\left\|(d_t^{(2)}I)^{-\frac12}T_t^{(1)}\right\|_2\right]. \tag{3.15}$$

It is straightforward that

$$\left\|(D_t^{(4)})^{-\frac12}(d_t^{(2)}I)^{\frac12}\right\|_2\le 1 \quad\text{and}\quad \left\|(d_t^{(2)}I)^{\frac12}(D_t^{(4)})^{-\frac12}\right\|_2\le 1 \tag{3.16}$$

hold true, and it follows from

$$\begin{aligned}\left\|T_t^{(1)}(d_t^{(2)}I)^{-\frac{1}{2}}\right\|_2^2 &= \left\|(d_t^{(2)}I)^{-\frac{1}{2}}T_t^{(1)}\right\|_2^2\\ &\le \max_{v\neq 0}\left\{\frac{v^*T_t^{(1)}(T_t^{(1)})^*v}{v^*(d_t^{(2)}I)v}\right\}\\ &\le \max_{-\pi\le\theta\le\pi}\left\{\frac{\theta^2}{d_t^{(2)}}\right\}\\ &= \frac{\pi^2}{d_t^{(2)}}\end{aligned}$$

that

$$\left\|T_t^{(1)}(d_t^{(2)}I)^{-\frac{1}{2}}\right\|_2 = \left\|(d_t^{(2)}I)^{-\frac{1}{2}}T_t^{(1)}\right\|_2 < \frac{\pi}{\sqrt{d_t^{(2)}}};$$

see (3.12). By substituting these bounds into (3.15) we know that (3.7) holds true.

Similar arguments can lead to the estimate

$$\left|\frac{v^*[(D_x^{(3)}T_x^{(1)}+T_x^{(1)}D_x^{(3)})\otimes(Q_t\Omega)]v}{v^*[D_x^{(4)}\otimes(Q_t\Omega)]v}\right| \le \frac{2\pi d_x^{(1)}}{d_x^{(2)}} \le \mu^{(1)},$$

which is exactly (3.9).

Moreover, it follows from (3.14) and (3.16) that

$$\begin{aligned}&\left|\frac{v^*[Q_x\otimes(D_t^{(3)}B_t^{(1)}+B_t^{(1)}D_t^{(3)})]v}{v^*[Q_x\otimes D_t^{(4)}]v}\right|\\ &\le \left\|(D_t^{(4)})^{-\frac{1}{2}}(d_t^{(2)}I)^{\frac{1}{2}}\right\|_2\left\|(d_t^{(2)}I)^{-\frac{1}{2}}\right\|_2\left\|D_t^{(3)}\right\|_2\left\|(d_t^{(2)}I)^{\frac{1}{2}}(D_t^{(4)})^{-\frac{1}{2}}\right\|_2\\ &\quad\cdot\left[\left\|B_t^{(1)}(d_t^{(2)}I)^{-\frac{1}{2}}\right\|_2+\left\|(d_t^{(2)}I)^{-\frac{1}{2}}B_t^{(1)}\right\|_2\right]\\ &\le \frac{1}{\sqrt{d_t^{(2)}}}\cdot d_t^{(1)}\cdot 2\cdot\max_{-\pi\le\theta\le\pi}\left\{\frac{|\sin\theta|}{\sqrt{d_t^{(2)}}}\right\}\\ &\le \frac{2d_t^{(1)}}{d_t^{(2)}}\\ &\le \mu^{(2)}\end{aligned}$$

and

$$\left|\frac{v^*[(D_x^{(3)}B_x^{(1)}+B_x^{(1)}D_x^{(3)})\otimes(Q_t\Omega)]v}{v^*[D_x^{(4)}\otimes(Q_t\Omega)]v}\right| \le \frac{2d_x^{(1)}}{d_x^{(2)}} \le \mu^{(2)}$$

hold true, which are exactly (3.8) and (3.10), respectively. □

Lemma 3.4 immediately implies the following bounds about the functions $f_A(v)$ and $f_M(v)$ defined in Theorem 3.2.

THEOREM 3.5. $D_z^{(i)}(i = 2, 4 \quad z \in \{x, t\}), Q_z(z \in \{x, t\})$ Ω $d_z^{(i)}(i = 1, 2$ $z \in \{x, t\})$ 3.4 $f_A(v)$ $f_M(v)$ 3.2

$$|f_A(v)| \le \mu^{(1)} \qquad |f_M(v)| \le \mu^{(2)}.$$

. Based on Lemmas 3.3 and 3.4, by straightforward computations we have

$$|f_A(v)| = \left|\frac{v^*\mathcal{S}(A)v}{v^*\mathcal{H}(A)v}\right| =$$

$$\left|\frac{v^*[\varepsilon(D_x^{(1)}T_x^{(1)} + T_x^{(1)}D_x^{(1)}) \otimes Q_t + Q_x \otimes (D_t^{(3)}T_t^{(1)} + T_t^{(1)}D_t^{(3)}) + (D_x^{(3)}T_x^{(1)} + T_x^{(1)}D_x^{(3)}) \otimes (Q_t\Omega)]v}{v^*[\varepsilon(T_x^{(2)} + D_x^{(2)}) \otimes Q_t + Q_x \otimes D_t^{(4)} + D_x^{(4)} \otimes (Q_t\Omega)]v}\right|$$

$$\le \max_{v\neq 0}\left\{\left|\frac{v^*[\varepsilon(D_x^{(1)}T_x^{(1)} + T_x^{(1)}D_x^{(1)}) \otimes Q_t]v}{v^*[\varepsilon(T_x^{(2)} + D_x^{(2)}) \otimes Q_t]v}\right|, \left|\frac{v^*[Q_x \otimes (D_t^{(3)}T_t^{(1)} + T_t^{(1)}D_t^{(3)})]v}{v^*(Q_x \otimes D_t^{(4)})v}\right|,\right.$$

$$\left.\left|\frac{v^*[(D_x^{(3)}T_x^{(1)} + T_x^{(1)}D_x^{(3)}) \otimes (Q_t\Omega)]v}{v^*(D_x^{(4)} \otimes (Q_t\Omega))v}\right|\right\}$$

$$\le \mu^{(1)}$$

and, similarly,

$$|f_M(v)| = \left|\frac{v^*\mathcal{S}(M)v}{v^*\mathcal{H}(M)v}\right| \le \mu^{(2)}.$$

Here, we have used the inequality

$$\frac{\alpha_1 + \alpha_2 + \alpha_3}{\beta_1 + \beta_2 + \beta_3} \le \max_{1\le i\le 3}\left\{\frac{\alpha_i}{\beta_i}\right\},$$

with α_i and β_i, $i = 1, 2, 3$, being positive reals. □

By using Theorems 3.2 and 3.5, and Lemmas 3.3 and 3.4, we can straightforwardly obtain the main theorem of this paper.

THEOREM 3.6. $D_z^{(i)}(i = 2, 4 \quad z \in \{x, t\})$ $Q_z(z \in \{x, t\})$ Ω

$$\frac{1 - \mu^{(1)}\mu^{(2)}}{1 + (\mu^{(2)})^2} \le \mathrm{Re}(\lambda(M^{-1}A)) \le \frac{\pi^2(1 + \mu^{(1)}\mu^{(2)})}{4} \quad \mu^{(1)}\mu^{(2)} < 1,$$

$$-\frac{\pi^2(\mu^{(1)} + \mu^{(2)})}{4} \le \mathrm{Im}(\lambda(M^{-1}A)) \le \frac{\pi^2(\mu^{(1)} + \mu^{(2)})}{4}.$$

Based on Theorem 3.6, we can immediately obtain a theoretical estimate about the asymptotic convergence rate of the preconditioned GMRES method with the preconditioner M in (3.4) for solving the system of linear equations (3.2). Here, we should suitably scale the Burgers equation (1.3) and appropriately choose the weighting functions $\omega_x(x)$ and $\omega_t(t)$ and the conformal mappings $\phi_x(x)$ and $\phi_t(t)$ such that $\mu^{(1)}\mu^{(2)} < 1$. For details, we refer to [20, 7].

4. Numerical examples. In this section, we illustrate the effectiveness of the proposed preconditioner when it is used to precondition the linear subsystems involved in each step of the Newton or the fixed-point iteration for solving the system of nonlinear equations (1.1).

In our experiments, the initial guess is taken to be zero, and each iteration process is terminated when the current residual $r^{(j)}$ satisfies

$$\frac{\|r^{(j)}\|_2}{\|r^{(0)}\|_2} \leq 10^{-6}.$$

The GMRES method is applied to solve the preconditioned linear subsystems of the form

$$M^{-1}A\mathbf{z} = M^{-1}\mathbf{r},$$

which forms the inner iteration process for solving the linear subsystems involved in each step of the Newton or the fixed-point method. Here, the stopping criterion for the preconditioned GMRES is that the relative reduction on the norm of the residual is less than 10^{-6}. Besides, all codes are written in MATLAB 7.01, and all experiments are tested on a PC with 0.99G memory.

The following two examples of the Burgers equation (1.3) are used to examine the numerical performance of our new preconditioner M defined in (3.4):

(i) The Burgers equation

$$\begin{cases} \dfrac{\partial u}{\partial t}(x,t) + u(x,t)\dfrac{\partial u}{\partial x}(x,t) - \epsilon\dfrac{\partial^2 u}{\partial x^2}(x,t) & \\ \quad = e^{-\pi^2 t}\sin(\pi x) & \\ \qquad \cdot(1-\pi^2 t + \pi t^2 e^{-\pi^2 t}\cos(\pi x) + \epsilon\pi^2 t), & 0 < x < 1 \ \text{ and } \ t \geq 0, \\ u(0,t) = 0 \quad \text{and} \quad u(1,t) = 0, & t \geq 0, \\ u(x,0) = 0, & 0 \leq x \leq 1. \end{cases}$$

(ii) The Burgers equation

$$\begin{cases} \dfrac{\partial u}{\partial t}(x,t) + u(x,t)\dfrac{\partial u}{\partial x}(x,t) - \epsilon\dfrac{\partial^2 u}{\partial x^2}(x,t) & \\ \quad = e^{x-t}(1-t+t^2e^{x-t}-\epsilon t) & \\ \qquad -e^{-t}(1-t)(1-x+ex) & \\ \qquad -t^2e^{-2t}(e-1)(1-x+xe), & 0 < x < 1 \ \text{ and } \ t \geq 0, \\ u(0,t) = te^{-t} \quad \text{and} \quad u(1,t) = te^{1-t}, & t \geq 0, \\ u(x,0) = 0, & 0 \leq x \leq 1. \end{cases}$$

The conformal mappings are chosen as $\phi(z) = \ln(\frac{z}{1-z})$ and $\psi(z) = \ln(\sinh(z))$ so that their restrictions onto the real intervals $(0,1)$ and $(0,+\infty)$ are $\phi_x(x) := \phi(x) = \ln(\frac{x}{1-x})$ and $\phi_t(t) := \psi(t) = \ln(\sinh(t))$, which are used for the discretizations of the x and t variables, respectively. The weighting functions are chosen to be $\omega_x(x) = 1/\phi_x^{'}(x)$ and $\omega_t(t) = 1/\phi_t^{'}(t)$.

In the numerical tables, the symbol "I" means that no preconditioner is used when solving the linear subsystems involved in the nonlinear iterations, while "M" represents that the preconditioner M defined in (3.4) is used. We use "N" to denote the number of the Newton iteration steps, "F" the number of the fixed-point iteration steps, "G" the average number of GMRES iteration steps in each Newton or

fixed-point iteration, "CPU" the total computing times, "Se" the maximum absolute discretization error at the Sinc grid points, and "Ue" the maximum absolute discretization error on the corresponding uniform grid points, while we use "average Se" and "average Ue" to represent the average absolute errors at all the Sinc grid points and at all the uniform grid points, respectively. In addition, the symbol "$*$" is used to denote that the iteration does not satisfy the terminating criterion within 50 steps of the Newton or the fixed-point iteration, while "+" means that the inner iteration does not satisfy the GMRES terminating criterion within 1000 iteration steps.

With respect to different ϵ, Tables 4.1–4.6 list the numbers of iteration steps and the CPU times required for the convergence of the Newton iteration, and Tables 4.7–4.12 list those required for the convergence of the fixed-point iteration, respectively, when they are applied to solve the system of nonlinear equations (1.1) resulting from the Sinc–Galerkin discretization of example (i). Some errors for reflecting the accuracy of the computed solutions are also listed in these tables. From these tables, we can see that our new preconditioner can considerably improve the convergence properties of both iteration methods and greatly reduce the running times. Moreover, with increasing of the problem size m, the number of the Newton or the fixed-point iteration steps almost remains the same if the inner iteration solver GMRES is preconditioned by the new preconditioner. The GMRES cannot achieve the prescribed tolerance within 1000 iteration steps and, therefore, the Newton or the fixed-point iteration cannot achieve the prescribed tolerance within 50 iteration steps, if the inner iteration solver GMRES is employed without using a preconditioner. Therefore, the new preconditioning technique substantially improves the convergence behaviors of both Newton and fixed-point iterations and, consequently, leads to fast convergent methods for solving the Sinc–Galerkin nonlinear systems of the Burgers equations.

TABLE 4.1
Results for example (i): $h_x = h_t = \pi/\sqrt{3m}$, $\epsilon = 1.0$, *and the Newton method is used.*

	I			M						
m	N	G	CPU	N	G	Se	Average Se	Ue	Average Ue	CPU
4	2	76	0.25	2	13	1.3×10^{-3}	3.8×10^{-4}	2.2×10^{-3}	2.4×10^{-4}	0.14
8	2	268	2.78	2	20	1.1×10^{-3}	1.7×10^{-4}	5.1×10^{-4}	2.5×10^{-5}	0.28
16	2	924	90.9	2	33	3.9×10^{-4}	3.7×10^{-5}	1.6×10^{-4}	4.5×10^{-6}	1.36
32	*	+	—	2	55	3.6×10^{-5}	2.5×10^{-6}	1.4×10^{-5}	3.0×10^{-7}	10.8

TABLE 4.2
Results for example (i): $h_x = h_t = \pi/\sqrt{3m}$, $\epsilon = 0.1$, *and the Newton method is used.*

	I			M						
m	N	G	CPU	N	G	Se	Average Se	Ue	Average Ue	CPU
4	2	77	0.30	2	20	9.3×10^{-3}	2.1×10^{-3}	9.4×10^{-3}	6.8×10^{-4}	0.16
8	2	265	2.63	2	32	2.2×10^{-3}	3.7×10^{-4}	1.8×10^{-3}	1.5×10^{-4}	0.41
16	3	932	119.9	3	47	2.5×10^{-4}	3.3×10^{-5}	2.0×10^{-4}	1.7×10^{-5}	2.83
32	*	+	—	3	79	3.6×10^{-5}	3.3×10^{-6}	3.1×10^{-5}	2.4×10^{-6}	21.3

TABLE 4.3
Results for example (i): $h_x = h_t = \pi/\sqrt{3m}$, $\epsilon = 0.01$, *and the Newton method is used.*

	I			M						
m	N	G	CPU	N	G	Se	Average Se	Ue	Average Ue	CPU
4	3	78	0.28	3	25	7.1×10^{-3}	1.8×10^{-3}	6.7×10^{-3}	1.2×10^{-3}	0.22
8	3	271	3.78	3	40	5.1×10^{-4}	1.0×10^{-4}	3.6×10^{-4}	8.2×10^{-5}	0.66
16	3	942	124.4	3	59	5.6×10^{-5}	7.8×10^{-6}	4.3×10^{-5}	4.61×10^{-6}	3.53
32	*	+	—	3	103	3.1×10^{-6}	3.4×10^{-7}	2.4×10^{-6}	2.3×10^{-7}	27.0

TABLE 4.4
Results for example (i): $h_x = h_t = \pi/\sqrt{3m}$, $\epsilon = 0.001$, *and the Newton method is used.*

	I			M						
m	N	G	CPU	N	G	Se	Average Se	Ue	Average Ue	CPU
4	3	75	0.39	3	34	2.6×10^{-3}	7.2×10^{-4}	2.7×10^{-3}	5.7×10^{-4}	0.25
8	3	275	4.06	3	57	4.2×10^{-4}	8.6×10^{-5}	2.7×10^{-4}	6.1×10^{-5}	0.94
16	3	939	124.1	2	80	5.6×10^{-5}	6.4×10^{-6}	2.4×10^{-5}	3.2×10^{-6}	3.14
32	*	+	—	2	132	3.1×10^{-6}	3.0×10^{-7}	1.4×10^{-6}	1.5×10^{-7}	23.0

TABLE 4.5
Results for example (i): $h_x = h_t = \pi/\sqrt{3m}$, $\epsilon = 0.0001$, *and the Newton method is used.*

	I			M						
m	N	G	CPU	N	G	Se	Average Se	Ue	Average Ue	CPU
4	3	78	0.53	3	34	2.4×10^{-3}	8.1×10^{-4}	2.7×10^{-3}	5.6×10^{-4}	0.27
8	2	275	2.77	2	56	4.6×10^{-4}	6.9×10^{-5}	2.1×10^{-4}	5.6×10^{-5}	0.59
16	2	916	79.9	2	63	5.6×10^{-5}	4.8×10^{-6}	1.0×10^{-5}	2.9×10^{-6}	3.13
32	*	+	—	2	146	3.1×10^{-6}	2.3×10^{-7}	4.8×10^{-6}	1.2×10^{-7}	25.4

TABLE 4.6
Results for example (i): $h_x = h_t = \pi/\sqrt{3m}$, $\epsilon = 0.00001$, *and the Newton method is used.*

	I			M						
m	N	G	CPU	N	G	Se	Average Se	Ue	Average Ue	CPU
4	3	77	0.39	3	28	2.4×10^{-3}	7.6×10^{-4}	2.6×10^{-3}	5.5×10^{-4}	0.22
8	3	281	6.50	3	64	8.0×10^{-4}	1.0×10^{-4}	3.9×10^{-4}	6.0×10^{-5}	1.02
16	2	944	85.3	2	91	7.0×10^{-5}	4.1×10^{-6}	1.2×10^{-5}	2.8×10^{-6}	3.67
32	*	+	—	2	151	3.1×10^{-6}	1.9×10^{-7}	2.3×10^{-7}	1.2×10^{-7}	26.6

TABLE 4.7
Results for example (i): $h_x = h_t = \pi/\sqrt{3m}$, $\epsilon = 1.0$, *and the fixed-point method is used.*

	I			M						
m	F	G	CPU	F	G	Se	Average Se	Ue	Average Ue	CPU
4	3	51	0.16	3	8	1.3×10^{-3}	3.8×10^{-4}	2.2×10^{-3}	2.4×10^{-4}	0.05
8	3	178	2.31	2	10	1.1×10^{-3}	1.7×10^{-4}	5.1×10^{-4}	2.5×10^{-5}	0.11
16	2	444	35.3	2	16	4.0×10^{-4}	3.7×10^{-5}	1.6×10^{-4}	4.5×10^{-6}	0.61
32	*	+	—	2	27	3.6×10^{-5}	2.5×10^{-6}	1.4×10^{-5}	3.0×10^{-7}	4.31

TABLE 4.8
Results for example (i): $h_x = h_t = \pi/\sqrt{3m}$, $\epsilon = 0.1$, *and the fixed-point method is used.*

	I			M						
m	F	G	CPU	F	G	Se	Average Se	Ue	Average Ue	CPU
4	4	58	0.23	3	13	9.3×10^{-3}	2.1×10^{-3}	9.4×10^{-3}	6.8×10^{-4}	0.05
8	3	114	2.28	3	21	2.2×10^{-3}	3.7×10^{-4}	1.8×10^{-3}	1.5×10^{-4}	0.31
16	3	577	66.9	3	32	2.5×10^{-4}	3.3×10^{-5}	2.0×10^{-4}	1.7×10^{-5}	1.83
32	*	+	—	3	55	3.6×10^{-5}	3.3×10^{-6}	3.1×10^{-5}	2.4×10^{-6}	12.8

Figures 4.1 and 4.3 depict the spectral distributions of the preconditioned matrix $M^{-1}A$ for examples (i) and (ii), respectively, corresponding to the Newton iteration, while Figure 4.5 depicts those for example (ii) corresponding to the fixed-point iteration, when $\epsilon = 1.0$. These figures clearly show that the matrices without preconditioning are very ill conditioned and, therefore, the corresponding GMRES method

TABLE 4.9
Results for example (i): $h_x = h_t = \pi/\sqrt{3m}$, $\epsilon = 0.01$, *and the fixed-point method is used.*

	I			M						
m	F	G	CPU	F	G	Se	Average Se	Ue	Average Ue	CPU
4	4	59	0.20	4	19	7.1×10^{-3}	1.8×10^{-3}	6.7×10^{-3}	1.2×10^{-3}	0.13
8	4	211	5.19	4	32	5.1×10^{-4}	1.0×10^{-4}	3.6×10^{-4}	8.3×10^{-5}	0.64
16	3	653	102.9	3	40	5.6×10^{-5}	7.8×10^{-6}	4.3×10^{-5}	4.6×10^{-6}	2.28
32	*	+	—	3	70	3.1×10^{-6}	3.4×10^{-7}	2.4×10^{-6}	2.3×10^{-7}	16.6

TABLE 4.10
Results for example (i): $h_x = h_t = \pi/\sqrt{3m}$, $\epsilon = 0.001$, *and the fixed-point method is used.*

	I			M						
m	F	G	CPU	F	G	Se	Average Se	Ue	Average Ue	CPU
4	4	58	0.39	4	26	2.6×10^{-3}	7.2×10^{-4}	2.7×10^{-3}	5.7×10^{-4}	0.20
8	3	185	2.55	3	37	4.2×10^{-4}	8.6×10^{-5}	2.7×10^{-4}	6.1×10^{-5}	0.53
16	3	633	86.1	2	41	5.6×10^{-5}	6.4×10^{-6}	2.4×10^{-5}	3.2×10^{-6}	1.55
32	*	+	—	2	68	3.1×10^{-6}	3.0×10^{-7}	1.4×10^{-6}	1.5×10^{-7}	11.0

TABLE 4.11
Results for example (i): $h_x = h_t = \pi/\sqrt{3m}$, $\epsilon = 0.0001$, *and the fixed-point method is used.*

	I			M						
m	F	G	CPU	F	G	Se	Average Se	Ue	Average Ue	CPU
4	4	58	0.58	4	26	2.4×10^{-3}	8.1×10^{-4}	2.7×10^{-3}	5.6×10^{-4}	0.17
8	3	189	2.69	3	39	4.6×10^{-4}	6.9×10^{-5}	2.1×10^{-4}	5.6×10^{-5}	0.64
16	3	625	80.0	2	43	5.6×10^{-5}	4.8×10^{-6}	1.0×10^{-5}	2.9×10^{-6}	1.55
32	*	+	—	2	76	3.1×10^{-6}	2.3×10^{-7}	4.8×10^{-7}	1.2×10^{-7}	12.4

TABLE 4.12
Results for example (i): $h_x = h_t = \pi/\sqrt{3m}$, $\epsilon = 0.00001$, *and the fixed-point method is used.*

	I			M						
m	F	G	CPU	F	G	Se	Average Se	Ue	Average Ue	CPU
4	4	58	0.41	4	20	2.4×10^{-3}	7.6×10^{-4}	2.6×10^{-3}	5.5×10^{-4}	0.14
8	4	215	4.30	3	43	7.9×10^{-4}	1.0×10^{-4}	3.8×10^{-4}	6.0×10^{-5}	0.66
16	3	647	91.3	2	47	7.0×10^{-5}	4.1×10^{-6}	1.2×10^{-5}	2.8×10^{-6}	1.70
32	*	+	—	2	78	3.1×10^{-6}	1.9×10^{-7}	2.3×10^{-7}	1.2×10^{-7}	13.1

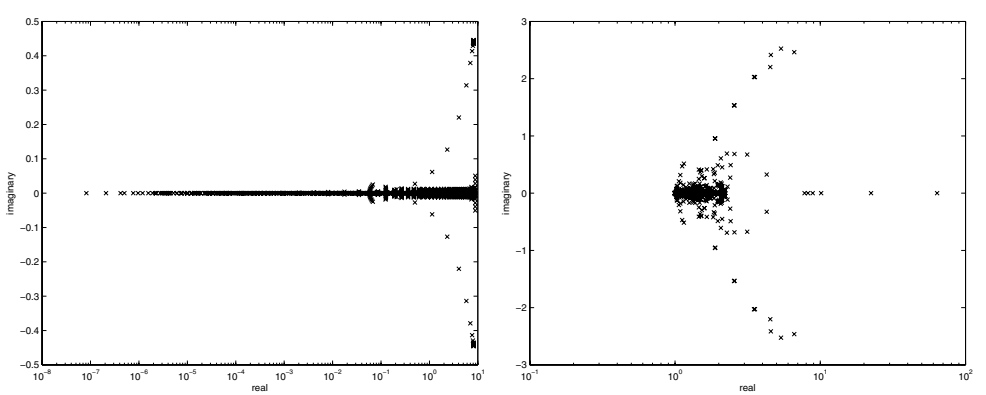


FIG. 4.1. *Spectra of the no-preconditioned (left) and preconditioned (right) matrices for example* (i), *with* $h_x = h_t = \pi/\sqrt{3m}$ *and* $m \equiv m_x = m_t = n_x = n_t = 16$, *when the Newton method is adopted.*

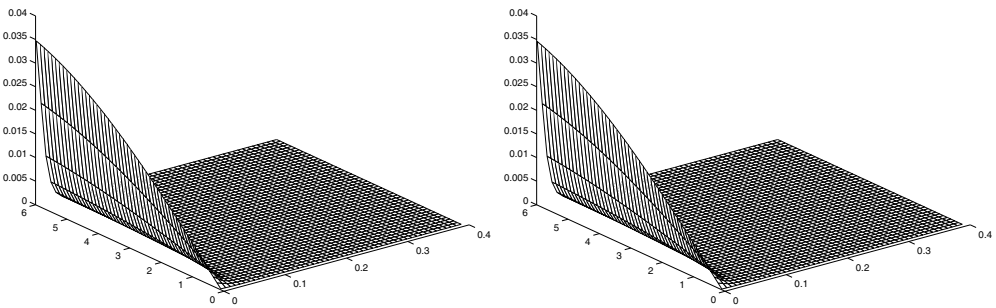


FIG. 4.2. *The exact (left) and computed (right) solutions of example* (i)*, with* $h_x = h_t = \pi/\sqrt{3m}$*,* $m \equiv m_x = m_t = n_x = n_t = 16$*, when the Newton method is adopted to obtain the computed solution.*

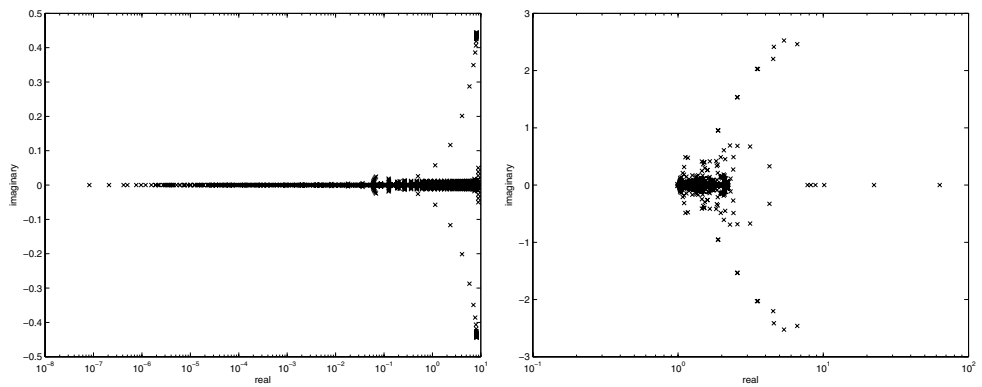


FIG. 4.3. *Spectra of the no-preconditioned (left) and preconditioned (right) matrices for example* (ii)*, with* $h_x = h_t = \pi/\sqrt{3m}$ *and* $m \equiv m_x = m_t = n_x = n_t = 16$*, when the Newton method is adopted.*

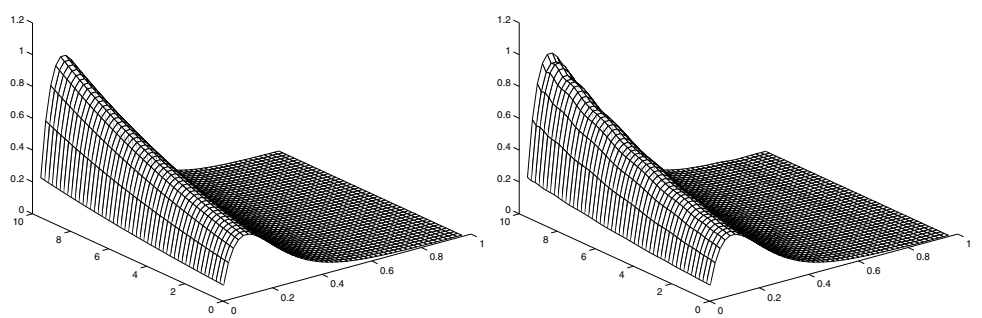


FIG. 4.4. *The exact (left) and computed (right) solutions of example* (ii)*, with* $h_x = h_t = \pi/\sqrt{3m}$*,* $m \equiv m_x = m_t = n_x = n_t = 16$*, when the Newton method is adopted to obtain the computed solution.*

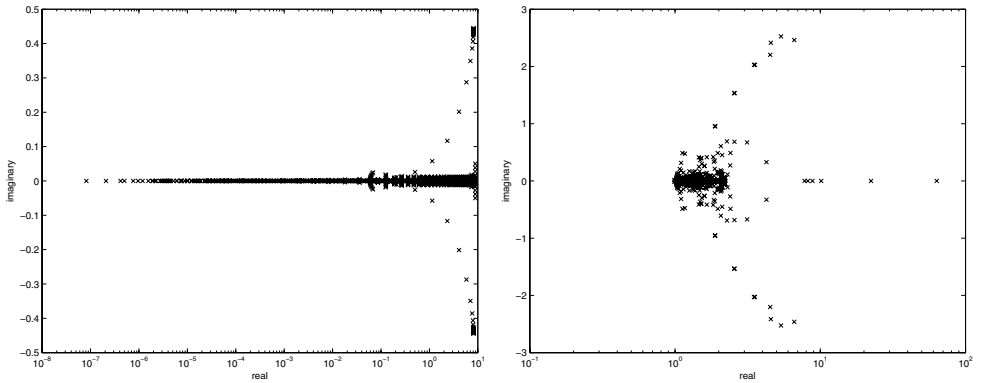


FIG. 4.5. *Spectra of the no-preconditioned (left) and preconditioned (right) matrices for example* (ii)*, with* $h_x = h_t = \pi/\sqrt{3m}$ *and* $m \equiv m_x = m_t = n_x = n_t = 16$*, when the fixed-point method is adopted.*

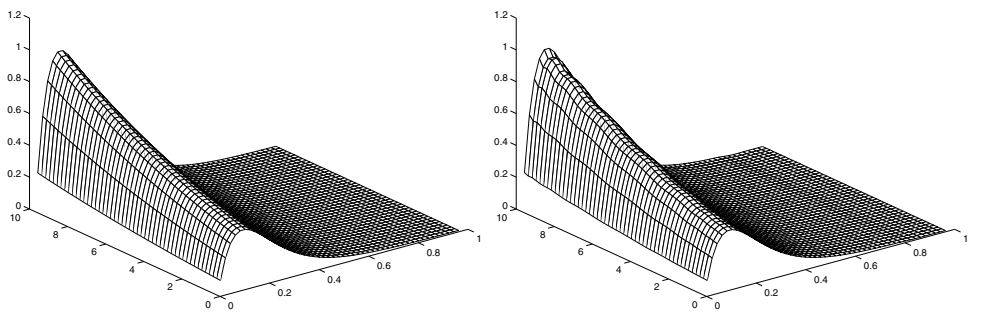


FIG. 4.6. *The exact (left) and computed (right) solutions of example* (i)*, with* $h_x = h_t = \pi/\sqrt{3m}$*,* $m \equiv m_x = m_t = n_x = n_t = 16$*, when the fixed-point method is adopted to obtain the computed solution.*

may be very slowly convergent or even divergent, while the matrices with preconditioning have tightly clustered eigenvalues and, thus, are well conditioned; as a result, the corresponding preconditioned GMRES method may converge quickly to the exact solutions of the subsystems of linear equations and, consequently, may lead to a fast convergent Newton or fixed-point method for solving the Sinc–Galerkin nonlinear systems of the Burgers equations.

In Figures 4.2, 4.4, and 4.6, we plot the exact and computed solutions of examples (i) and (ii) corresponding to the cases shown in the Figures 4.1, 4.3, and 4.5, respectively, where the computed solution is obtained by using the Newton or the fixed-point iteration method. Figures 4.2, 4.4, and 4.6 further show that our new methods can compute reasonably accurate results.

5. Concluding remarks. We have constructed a structured preconditioner that can efficiently improve the convergence property of the GMRES iteration employed to inexactly solve the subsystem of linear equations involved in each Newton or fixed-point iteration for solving the system of nonlinear equations resulting from the Sinc–Galerkin discretization of the Burgers equation. The bounds of the eigenvalues of the preconditioned matrix have been precisely estimated by making use of the generalized Bendixson theorem. Numerical experiments have shown the effectiveness of this new preconditioner.

REFERENCES

[1] Z.-Z. Bai, *Parallel multisplitting AOR method for solving a class of system of nonlinear algebraic equations*, Appl. Math. Mech., 16 (1995), pp. 675–682.
[2] Z.-Z. Bai, *Parallel nonlinear AOR method and its convergence*, Comput. Math. Appl., 31 (1996), pp. 21–31.
[3] Z.-Z. Bai, *A class of two-stage iterative methods for systems of weakly nonlinear equations,* Numer. Algorithms, 14 (1997), pp. 295–319.
[4] Z.-Z. Bai, *Parallel multisplitting two-stage iterative methods for large sparse systems of weakly nonlinear equations*, Numer. Algorithms, 15 (1997), pp. 347–372.
[5] Z.-Z. Bai, G.H. Golub, L.-Z. Lu, and J.-F. Yin, *Block triangular and skew-Hermitian splitting methods for positive-definite linear systems*, SIAM J. Sci. Comput., 26 (2005), pp. 844–863.
[6] Z.-Z. Bai, G.H. Golub, and M.K. Ng, *Hermitian and skew-Hermitian splitting methods for non-Hermitian positive definite linear systems*, SIAM J. Matrix Anal. Appl., 24 (2003), pp. 603–626.
[7] Z.-Z. Bai and M.K. Ng, *Preconditioners for nonsymmetric block Toeplitz-like-plus-diagonal linear systems*, Numer. Math., 96 (2003), pp. 197–220.
[8] Z.-Z. Bai, J.-C. Sun, and D.-R. Wang, *A unified framework for the construction of various matrix multisplitting iterative methods for large sparse system of linear equations*, Comput. Math. Appl., 32 (1996), pp. 51–76.
[9] Z.-Z. Bai and D.-R. Wang, *Asynchronous multisplitting nonlinear Gauss-Seidel type method*, Appl. Math. J. Chinese Univ. Ser. B, 9 (1994), pp. 189–194.
[10] Z.-Z. Bai and D.-R. Wang, *Asynchronous parallel multisplitting nonlinear Gauss-Seidel iteration*, Appl. Math. J. Chinese Univ. Ser. B, 12 (1997), pp. 179–194.
[11] F. Di Benedetto, *Solution of Toeplitz normal equations by sine transform based preconditioning*, Linear Algebra Appl., 285 (1998), pp. 229–255.
[12] R.H. Chan and M.K. Ng, *Conjugate gradient methods for Toeplitz systems*, SIAM Rev., 38 (1996), pp. 427–482.
[13] T. Kailath and A.H. Sayed, *Displacement structure: Theory and applications*, SIAM Rev., 37 (1995), pp. 297–386.
[14] N. Levinson, *The Wiener RMS (root mean square) error criterion in filter design and prediction*, J. Math. Phys. Mass. Inst. Tech., 25 (1947), pp. 261–278.
[15] J. Lund and K.L. Bowers, *Sinc Methods for Quadrature and Differential Equations*, SIAM, Philadelphia, 1992.
[16] M.K. Ng, *Fast iterative methods for symmetric sinc-Galerkin systems*, IMA J. Numer. Anal., 19 (1999), pp. 357–373.
[17] M.K. Ng and Z.-Z. Bai, *A hybrid preconditioner of banded matrix approximation and alternating direction implicit iteration for symmetric Sinc-Galerkin linear systems*, Linear Algebra Appl., 366 (2003), pp. 317–335.
[18] M.K. Ng and D. Potts, *Fast iterative methods for sinc systems*, SIAM J. Matrix Anal. Appl., 24 (2002), pp. 581–598.
[19] J.M. Ortega and W.C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York and London, 1970.
[20] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.

[21] D.-R. Wang, Z.-Z. Bai, and D.J. Evans, *Asynchronous multisplitting relaxed iterations for weakly nonlinear systems*, Int. J. Comput. Math., 54 (1994), pp. 57–76.

[22] D.-R. Wang, Z.-Z. Bai, and D.J. Evans, *On the monotone convergence of multisplitting method for a class of system of weakly nonlinear equations*, Int. J. Comput. Math., 60 (1996), pp. 229–242.

SIAM J. SCI. COMPUT.
Vol. 29, No. 1, pp. 440–458
© 2007 Society for Industrial and Applied Mathematics

EFFICIENT HIERARCHICAL APPROXIMATION OF HIGH-DIMENSIONAL OPTION PRICING PROBLEMS*

CHRISTOPH REISINGER† AND GABRIEL WITTUM‡

Abstract. A major challenge in computational finance is the pricing of options that depend on a large number of risk factors. Prominent examples are basket or index options where dozens or even hundreds of stocks constitute the underlying asset and determine the dimensionality of the corresponding degenerate parabolic equation. The objective of this article is to show how an efficient discretization can be achieved by hierarchical approximation as well as asymptotic expansions of the underlying continuous problem. The relation to a number of state-of-the-art methods is highlighted.

Key words. sparse grids, multigrid methods, option pricing, asymptotic expansions, dimension reduction

AMS subject classifications. 65N06, 65N12, 65N15, 65N40

DOI. 10.1137/060649616

1. Introduction.

Financial options are an essential element of today's risk management. They give the holder the right (but not the obligation) to perform a specific transaction on a risky asset X (an equity, for instance) or a number of assets $\mathbf{X} = (X_1, \dots, X_n)^T$ at some point(s) or time span in the future for a price that is agreed upon now, the [illegible] price K.

This work focuses on the cases where either the asset can only be exercised at the expiry T ([illegible] option), which means the holder may then buy ([illegible]) or sell ([illegible]) the asset for the strike K, or the asset can be exercised at any time up to T ([illegible] option).

For a put on a claim g the price u of the option at expiry (the [illegible]) is clearly

$$u(\mathbf{X}, T) = (K - g(\mathbf{X}))_+ = \max(K - g(\mathbf{X}), 0),$$

and analogously for the call. At any point up to then the value of the option depends decisively on the stochastic nature of the process.

1.1. Model framework.

The most common models for the underlying assets fall into the class of [illegible]

$$\mathrm{d}X_i = \beta_i^X(\mathbf{X}, \mathbf{Y})\,\mathrm{d}t + \alpha_i^X(\mathbf{X}, \mathbf{Y})\,\mathrm{d}W_i^X, \tag{1}$$

with correlated [illegible] W_i^X, where $\mathbf{Y} = (Y_1, \dots, Y_m)^T$ is a set of parameters. Traditionally market parameters have been assumed constant, but time series data in many markets reveal that some characteristics can be captured appropriately only if they are themselves modeled as stochastic differential equations (SDEs):

$$\mathrm{d}Y_i = \beta_i^Y(\mathbf{X}, \mathbf{Y})\,\mathrm{d}t + \alpha_i^Y(\mathbf{X}, \mathbf{Y})\,\mathrm{d}W_i^Y.$$

*Received by the editors January 11, 2006; accepted for publication (in revised form) September 13, 2006; published electronically February 26, 2007.
http://www.siam.org/journals/sisc/29-1/64961.html

†Mathematical Institute, Oxford University, 24–29 St. Giles, Oxford, OX1 3LB, UK (christoph.reisinger@maths.ox.ac.uk).

‡Simulation in Technology Center, Heidelberg University, Im Neuenheimer Feld 368, 69120 Heidelberg, Germany (wittum@uni-hd.de).

Let us write $\mathbf{Z} = (\mathbf{X}, \mathbf{Y})$, $\boldsymbol{\alpha} = (\boldsymbol{\alpha}^X, \boldsymbol{\alpha}^Y)$, $\boldsymbol{\beta} = (\boldsymbol{\beta}^X, \boldsymbol{\beta}^Y)$, $\mathbf{W} = (\mathbf{W}^X, \mathbf{W}^Y)$ and denote by ρ_{ij}, $1 \le i, j \le d := n + m$, the correlation of these Wiener processes.

In many cases assets and parameters are bounded by positivity constraints (equities, exchange rates, etc.), and this can be guaranteed by conditions of the form

$$\boldsymbol{\beta} \cdot \mathbf{n} \le 0, \qquad \boldsymbol{\alpha} \cdot \mathbf{n} = 0 \tag{2}$$

on the boundary Γ of the positive quadrant $\mathbb{R}^d_+$ with outward normal $\mathbf{n}$ and sufficiently smooth coefficients ($\boldsymbol{\alpha} \cdot \mathbf{n}$ must be of order at least $1/2$ at the boundary).

If $m = 0$, that is, parameters are deterministic, the pay-off can be replicated by a dynamic trading strategy with the assets X_i and a money market account with risk-free instantaneous interest rate r (assumed constant or more generally modeled by one of the parameter processes). This has the consequence that the price of an option can be seen as the expectation of the pay-off under a unique so-called [illegible] measure, which turns the stock price process (1), discounted by the risk-free interest rate r, into a martingale. For details of this, see, for example, [7] or [16]. The corresponding Feynman–Kac PDE for the option price reads

$$\frac{\partial u}{\partial t} + \sum_{i,j=1}^{d} a_{ij}(\mathbf{Z}, t) \frac{\partial^2 u}{\partial Z_i \partial Z_j} + \sum_{i=1}^{d} b_i(\mathbf{Z}, t) \frac{\partial u}{\partial Z_i} + c(\mathbf{Z}, t) u = 0, \tag{3}$$

which is a linear advection-diffusion-reaction equation with coefficients

$$\begin{aligned} a_{ij}(\mathbf{Z}, t) &= \rho_{ij} \alpha_i(\mathbf{Z}, t) \alpha_j(\mathbf{Z}, t), \quad 1 \le i, j \le d, \\ c(\mathbf{Z}, t) &= -r. \end{aligned}$$

The advective term $\mathbf{b}(\mathbf{Z}, t)$ depends on the product-specific hedge that creates a risk-free investment (the change of measure). Examples will be given in the next section. Condition (2) on the boundary translates to

$$\mathbf{a}(\mathbf{Z}, t) \cdot \mathbf{n}(\mathbf{Z}) = 0 \quad \text{and} \quad \mathbf{b}(\mathbf{Z}, t) \cdot \mathbf{n}(\mathbf{Z}) \ge 0 \quad \text{for } \mathbf{Z} \in \Gamma \tag{4}$$

and guarantees uniqueness of the solution without boundary conditions [24]. Equation (3) is an antidiffusion equation that is solved backwards in time and is thus parabolic and well-posed.

Equations similar to (3) can be derived in the presence of stochastic parameters ($m \ne 0$), which are often not traded in the market themselves (like volatilities) and sometimes cannot be directly observed. To hedge the additional risk introduced by the stochastic components of $\mathbf{Y}$, options with different expiration dates have to be traded dynamically, and an ad hoc assumption on the market price of risk has to be made from practical considerations.

It has to be remarked that in practice it is often not feasible to trade options dynamically due to transaction costs (for pricing under transaction costs, see, e.g., [14] and the references therein). In the case of static positions in the option a perfect hedge is no longer possible, but instead the expected utility can be optimized and the option price is adjusted such that the buyer/seller is indifferent to such a deal. This gives rise to (nonlinear) Hamilton–Jacobi–Bellmann equations [6]. In the scope of this article we will restrict ourselves to the linear setting.

1.2. Examples.

1.2.1. FX option. We consider an option on the exchange rate X between a domestic and a foreign currency, which is driven by the short rates r_d ( ) and r_f ( ) in both markets according to

$$\mathrm{d}X = (r_d - r_f)X\,\mathrm{d}t + \sigma_X X\,\mathrm{d}W$$

with volatility σ_X. In practice typically a time- and state-dependent volatility surface is calibrated to quoted option prices with different maturities and strikes. For short maturities (typically under 3 years) interest rates may be assumed constant in good approximation and the following equation for u is obtained by standard arguments:

$$\frac{\partial u}{\partial t} + \frac{1}{2}\sigma_X^2 X^2 \frac{\partial^2 u}{\partial X^2} + (r_d - r_f)X\frac{\partial u}{\partial X} - r_d u = 0. \tag{5}$$

In the more interesting case of longer time horizons the spot rate dynamics are relevant. Commonly used models are of the type

$$\mathrm{d}r_i = \mu_i(r_i, t)\,\mathrm{d}t + \nu_i(r_i, t)\,\mathrm{d}W_i,$$

where $(i = d, f)$

$$\mu_i(r_i, t) = \kappa_i(\theta_i(t) - r_i), \tag{6}$$

$$\nu_i(r_i, t) = r_i^{\beta_i}\sigma_i(t) \tag{7}$$

with positive parameters κ_i, β_i and a time-dependent reversion level θ_i, which can be calibrated to the term structure of interest rates. The corresponding PDE reads $(\mathbf{Z} = (X, r_d, r_f))$

$$\frac{\partial u}{\partial t} + \frac{1}{2}\sum_{i,j=1}^{3}\nu_i\nu_j\rho_{ij}\frac{\partial^2 u}{\partial Z_i \partial Z_j} + \sum_{i=1}^{3}\mu_i\frac{\partial u}{\partial Z_i} + \rho_{31}\nu_3\nu_1\frac{\partial u}{\partial Z_3} - Z_2 u = 0, \tag{8}$$

where $\nu_1 = \sigma_X Z_1$ and ρ_{ij}, $1 \le i, j \le 3$, are again the correlations of the Wiener processes.

For $\beta_i \ge 1/2$ the interest rates stay nonnegative and the PDE is solved backwards on $\mathbb{R}^3_+$ from the terminal condition (put)

$$u(\mathbf{Z}, T) = (K - Z_1)_+ .$$

1.2.2. Equity basket. A classical high-dimensional application is a basket (linear combination) of d stocks, assumed to follow a geometric Brownian motion

$$\mathrm{d}S_i = \mu_i S_i\,\mathrm{d}t + \sigma_i S_i\,\mathrm{d}W_i,$$

as in [1], with Brownian motions W_i with correlation $\langle \mathrm{d}W_i, \mathrm{d}W_j\rangle = \rho_{ij}$. Denote by r again the constant risk-free interest rate. The price (the price that does not allow any instantaneous risk-free return) of a European option follows the

$$\frac{\partial u}{\partial t} = \mathcal{L}_{BS}u := -\frac{1}{2}\sum_{i,j=1}^{d}\sigma_i\sigma_j\rho_{ij}S_iS_j\frac{\partial^2 u}{\partial S_i \partial S_j} - r\sum_{i=1}^{d}S_i\frac{\partial u}{\partial S_i} + ru. \tag{9}$$

A put on the basket $\sum_i \mu_i S_i$ with positive weights μ_i has terminal value

$$u(\mathbf{S}, T) = g(\mathbf{S}) := \Big(K - \sum_{i=1}^{d} \mu_i S_i\Big)_+, \qquad \mathbf{S} \in \mathbb{R}^d_+.$$

American options, which can be exercised at any time up to T, solve the linear complementarity problem

$$\frac{\partial u}{\partial t} + \mathcal{L}_{BS} u \le 0, \tag{10}$$

$$u \ge g, \tag{11}$$

$$\left(\frac{\partial u}{\partial t} + \mathcal{L}_{BS} u\right) \cdot (u - g) = 0 \tag{12}$$

(see, e.g., [22]). A free boundary separates the exercise region and the continuation region of the option.

1.3. Curse of dimensionality, Monte Carlo, and sparse grids. The dimension of (3) can be very large, depending on the number d of stochastic components considered. The complexity of a standard grid-based approach to compute the option price with a desired accuracy ε is

$$W(\varepsilon, d) = \mathcal{O}(\varepsilon^{-d/p})$$

for a discretization of convergence order p and grows exponentially with the dimension d of the problem. This is referred to as the [illegible].

As a result of this the vast majority of problems in financial practice—and virtually all with dimension larger than three—are solved by Monte Carlo or related approaches [10]. The practical drawbacks of these methods, in addition to the somewhat disturbing fact that error bounds can only be specified in a probabilistic sense, are the following:

1. The convergence of the method, although it depends only very weakly on the dimension of the problem, is very slow. Typically if N is the number of simulated paths,

   $$\varepsilon = \mathcal{O}(N^{-\frac{1}{2}}), \tag{13}$$

   where ε has to be read as the standard deviation of the result.
2. Standard Monte Carlo methods have difficulties yielding accurate sensitivities of the solution with respect to the underlying stocks, which are important trading parameters (and directly given by the partial derivatives of the PDE solution). This is particularly the case if the option exhibits [illegible] characteristics, that is, the option ceases to exist if the stock crosses a certain [illegible]. Note that such features fit very naturally in the PDE context where only the boundary conditions need to be changed.
3. Early exercise features like those in the American case do not fit into the "forward" simulation paradigm, and the determination of the free boundary by regression of the value function [13] is very costly, whereas the modification of the PDE to a linear complementarity problem is usually straightforward.

In contrast, quasi-Monte Carlo methods are based on (deterministic) low-discrepancy point sequences and thus restricted to integration problems. They exhibit the same disadvantages 2 and 3 above, but improve the asymptotic complexity (13) to

$$\varepsilon = \mathcal{O}\left((\log N)^{d-1} N^{-1}\right)$$

with N points of the sequence, requiring higher smoothness (bounded variation) of the integrand [15].

Lattice rules exploit even higher regularity and yield error bounds

$$\varepsilon = \mathcal{O}\left((\log N)^{d-1} N^{-r}\right)$$

for integrands with r derivatives [15].

A further progression in this direction is the Smolyak construction [20] of quadrature rules in spaces with bounded mixed derivatives, which was more recently studied numerically in the context of financial derivatives [8] and extended to a dimension adaptive procedure in [9].

The approximation property of sparse grids was first employed for the solution of PDEs by finite elements in [23] and was subsequently studied extensively for model problems in [3, 4], where convergence of the form

$$\varepsilon = \mathcal{O}(|\log h|^{d-1} h^p) \tag{14}$$

for grid size h is shown for elements of order p. Recently sparse wavelet bases have been analyzed in the context of parabolic problems with nonsmooth initial data in [21].

A very attractive variant from a practical viewpoint, the combination technique [12] requires only solution of the original equation on conventional subspaces defined on Cartesian grids and a subsequent extrapolation step, but still retains the convergence order (14). This approach, analyzed in [18], is the method of choice here, and details are given in the following section.

2. Discretization on sparse grids.

2.1. The combination technique. Consider now the d-dimensional unit cube and the family of (anisotropic) grids with grid sizes $h_k = 2^{-i_k}$ in direction k, $i_k \in \mathbb{N}_0$. We write the vector of grid sizes as $\mathbf{h} = 2^{-\mathbf{i}}$ with $\mathbf{i} = (i_1, \ldots, i_d) \in \mathbb{N}_0^d$ and denote numerical solutions on these grids by $u_{\mathbf{h}}$. In the present context we will consider piecewise linear interpolants of finite difference solutions. This defines a hierarchy of grid solutions $U = (u_{2^{-\mathbf{i}}})_{\mathbf{i}\in\mathbb{N}_0^d}$, depicted in Figure 1 for the first levels in the two-dimensional case.

The sparse grid solution at level n is then defined as

$$u_n = \sum_{l=n}^{n+d-1} a_{l-n} \sum_{i_1+\cdots+i_d=l} U(\mathbf{i}) \tag{15}$$

with

$$a_i := (-1)^{d-1-i} \begin{pmatrix} d-1 \\ i \end{pmatrix}, \quad 0 \le i \le d-1. \tag{16}$$

The grid solutions involved in the inner sum of (15) all have $i_1 + \cdots + i_d = l$ and correspond to columns of grids in Figure 1. The number of elements in each of these grids is 2^l, regardless of the dimension, and the number of grid solutions in this sum is $\binom{l+d-1}{d-1}$ and grows like l^{d-1}. Therefore the dimension of the sparse grid space on level n is

$$N = \mathcal{O}\left(2^n n^{d-1}\right) = \mathcal{O}\left(h^{-1} |\log h|^{d-1}\right) \tag{17}$$

if $h = 2^{-n}$ is the finest grid size. This compares to $N = \mathcal{O}(2^{dn}) = \mathcal{O}(h^{-d})$ for the full grid.

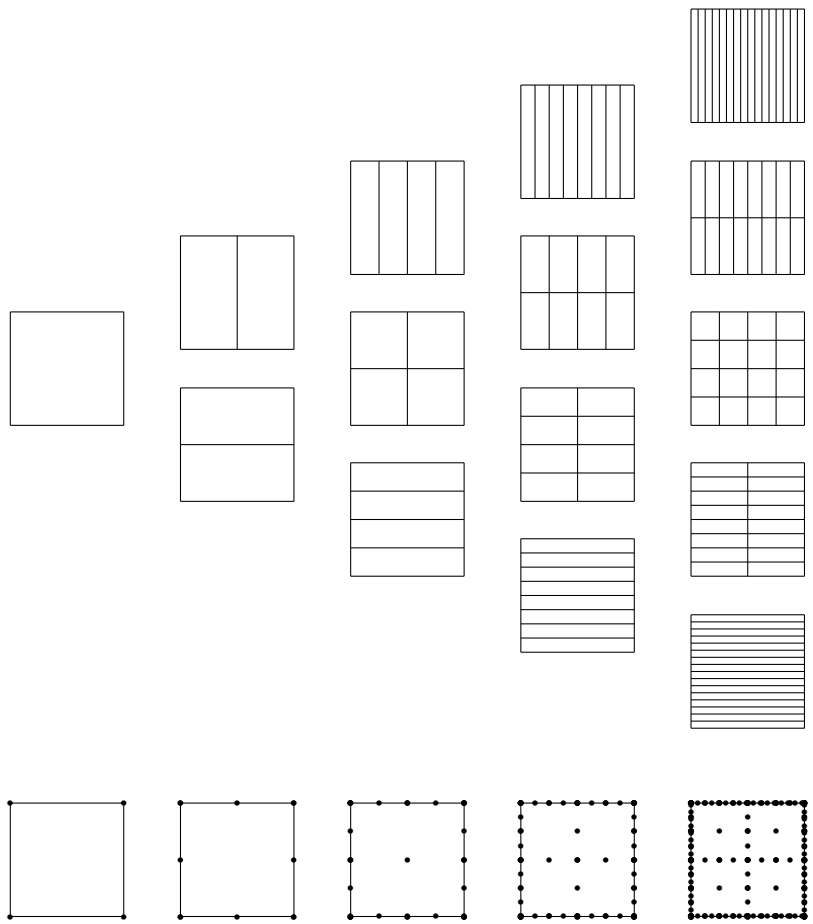


FIG. 1. *Two-dimensional sparse grid hierarchy for levels $n = 0, 1, \ldots, 4$. The grids on level n are obtained by bisection from level $n-1$ and all have 2^n elements. The number of grids on this level is $n+1$. In the bottom the degrees of freedom are shown for a vertex centered scheme.*

This approach has two crucial advantages over a conventional grid-based method: First, the asymptotic complexity of the discretized problem is reduced substantially to the complexity of a one-dimensional problem (up to polynomial factors). Thus the curse of dimensionality can be dampened to some extent. Second, the discrete system of equations decouples into much smaller conventional problems on tensor product grids, which can be solved efficiently in parallel and then superimposed using (15).

2.2. Discretization error. It remains to analyze how the reduction of the approximation space affects the convergence of the approximations. The extrapolation character of the combination technique necessitates a detailed analysis of the discretization error on the grids involved. It was already understood in [12] that for the numerical approximation $u_{\mathbf{h}}$ from a second order method on structured grid with mesh sizes $h_1, \ldots, h_d$, expansions of the form

$$u - u_{\mathbf{h}} = \sum_{m=1}^{d} \sum_{\substack{\{j_1, \ldots, j_m\} \\ \subset \{1, \ldots, d\}}} h_{j_1}^2 \cdot \ldots \cdot h_{j_m}^2 \cdot \gamma_{j_1, \ldots, j_m}(.; h_{j_1}, \ldots, h_{j_m}) \tag{18}$$

are crucial for second order convergence on the sparse grid. Such a representation with pointwise bounds $|\gamma_{j_1,\ldots,j_m}| \leq K$ is found for the two-dimensional Poisson equation in [5] and more generally for elliptic and parabolic equations in higher dimensions in [18].

The essential principle of the extrapolation is that all lower order terms cancel out in the combination formula (15) and only the highest order terms $h_1^2 \cdot \ldots \cdot h_d^2 = 4^{-n}$ remain. Taking advantage of this cancellation mechanism, Griebel, Schneider, and Zenger derive in [12] from (18) a pointwise error estimate for the three-dimensional case

$$|u - u_n| \leq K 4^{-n} \left(1 + \frac{65}{32}n + \frac{25}{32}n^2\right).$$

In [18] the following result for the general case is shown:

$$|u - u_n| \leq \frac{2K}{(d-1)!}\left(\frac{5}{2}\right)^{d-1}(n + 2(d-1))^{d-1}4^{-n}; \tag{19}$$

in fact the asymptotic behavior is

$$u - u_n = \gamma_0 \left(\frac{3}{4}\right)^{d-1} \frac{n^{d-1}}{(d-1)!}4^{-n} + \mathcal{O}(n^{d-2}4^{-n}). \tag{20}$$

For parabolic problems, it is shown in [18] that $\gamma_0 := \gamma_{1,\ldots,d}(.; 0, \ldots, 0) = 0$ depends on certain mixed derivatives of the solution and the magnitude of the error therefore depends on their behavior for high dimensions.

In particular, the case $\gamma_0 = 0$ indicates a lower superposition dimension of the solution—we shall return to this point later.

2.3. Numerical examples. We now investigate in detail the performance of sparse grids for important practical examples that arose out of the project "Analytical Methods and Efficient Numerical Algorithms for Financial Derivatives" with the Dresdner Bank AG.[1]

The examples involve equity and exchange rate derivatives and appear to be diffusion dominant with grid Péclet numbers smaller than 2 on all Cartesian grids such that central finite differences can be used. Sufficiently many time steps of the fractional-step-θ scheme are performed such that the time discretization error is negligible. This scheme, which was first introduced in [11] and for each time step δt consists of three substeps of the standard θ-scheme with step sizes $\tau_1 = (\sqrt{2}-1)\delta t$, $\tau_2 = \tau_3 = (1-\sqrt{2}/2)\delta t$ and parameters $\theta_1 = \sqrt{2}-1$, $\theta_2 = \theta_3 = 2-\sqrt{2}$, is second order accurate like the Crank–Nicolson scheme, but has more favorable stability properties (is strongly A-stable). The linear systems in each time step are solved with sufficient accuracy by multigrid techniques as outlined in section 4.1.

2.3.1. FX option. We specify model (8) further by choosing the Hull–White model for the interest rate dynamics ($\beta_i = 0$ in (7)), and for simplicity we assume that the processes are uncorrelated. The equation becomes

$$\frac{\partial u}{\partial t} + \frac{1}{2}\sigma_X^2 X^2 \frac{\partial^2 u}{\partial X^2} + \frac{1}{2}\sum_{i=d,f}\sigma_{r_i}^2 \frac{\partial^2 u}{\partial {r_i}^2} + (r_d - r_f)X\frac{\partial u}{\partial X} + \sum_{i=d,f}\kappa_i(\theta_i - r_i)\frac{\partial u}{\partial r_i} - r_d u = 0.$$

[1]The authors would like to thank Jürgen Linde for many discussions on appropriate models and for providing the data used here.

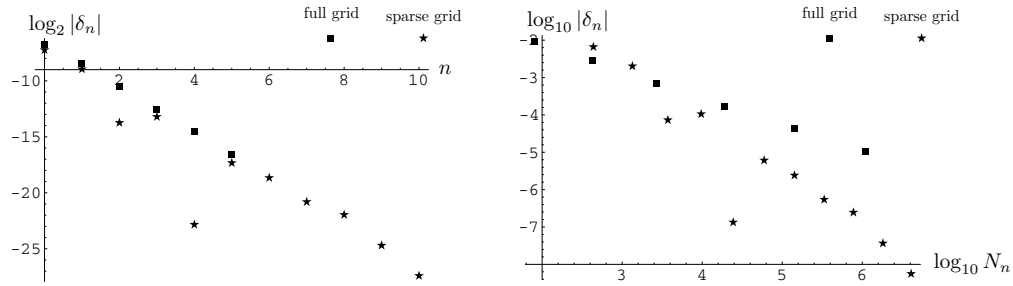


FIG. 2. *Left: Pointwise difference δ_n between levels n and $n+1$ for full and sparse grids. Regression of the curve $\log_2 |\delta_n| \approx -pn + q \log_2 n$ yields $p = 1$, $q = -3.8$, instead of the theoretically expected $p = 2$, $q = d - 1 = 2$. However, if the logarithmic term (which is difficult to estimate) is neglected, one obtains $p = 1.81$, and even $p = 1.99$ if only the last 6 values are used for the regression. Right: The same data, now shown versus the number of degrees of freedom N_n. Here the full grid data have slope $-2/3$ due to the second order convergence in three dimensions, whereas the sparse grid data still show slope -2. Hence the sparse grid method behaves like a sixth order method on the full grid.*

The other parameters are set to $\theta_i = 0.045$, $\kappa_i = 0.5$, $\sigma_X = \sigma_d = \sigma_f = 0.15$. The interest rates fluctuate around the constant level θ_i (Vasicec model); they are not bounded and become negative with (small, but) positive probability in this commonly used model. The domain is truncated to $[0, 4K] \times [0, 2\theta_d] \times [0, 2\theta_f]$, where $K = 0.25$ is the strike price. For $X = 0$ natural boundary conditions hold (note that the relevant coefficients vanish) for $X \to \infty$ $u \to 0$, and thus we set $u = 0$ for $X = 4K$, which is almost 20 standard deviations away from the strike. For the other boundaries homogeneous Neumann conditions are set.

We look at the pointwise convergence of the price of a European put with maturity $T = 1$ year, evaluated at the spot rates $X(0) = 0.9$, $r_d(0) = r_f(0) = 0.05$ (e.g., EUR/USD), which is the practically interesting measure of accuracy. Two refinement strategies, a full refinement with bisection in all directions and a sparse grid hierarchy, are compared. We start from a coarse grid that has four elements in the X direction, such that the piecewise linear initial condition with a kink at the strike is captured exactly on all grids. The difference δ_n between the solution on subsequent refinement levels of the sparse and the full grid is plotted in Figure 2. On the same levels n the errors almost coincide asymptotically, but the sparse grid can be refined much further in practice due to the reduced number of nodes. Hence if the error is plotted versus the number of degrees of freedom (right), the superior complexity of the sparse grid becomes apparent.

Figure 3 (left) shows the nodes of the sparse grid on an intermediate level. The density of nodes of a full grid on the same level can be seen from the nodes at the boundary edges.

We also solve the 1-factor model (5) where the interest rates are assumed constant at $r_d(0) = 0.05$, $r_f(0) = 0.05$ over the life span of the option, and compare in Figure 3 (right) the solution to the one on the full 3-factor model evaluated at the line $(x, r_d(0) = 0.05, r_f(0) = 0.05)$, $x \in [0, 1]$ (the coordinates have been scaled to the unit cube; the strike lies at 0.25). The deviation in the interesting range is significant and stresses the importance of including the interest rate dynamics.

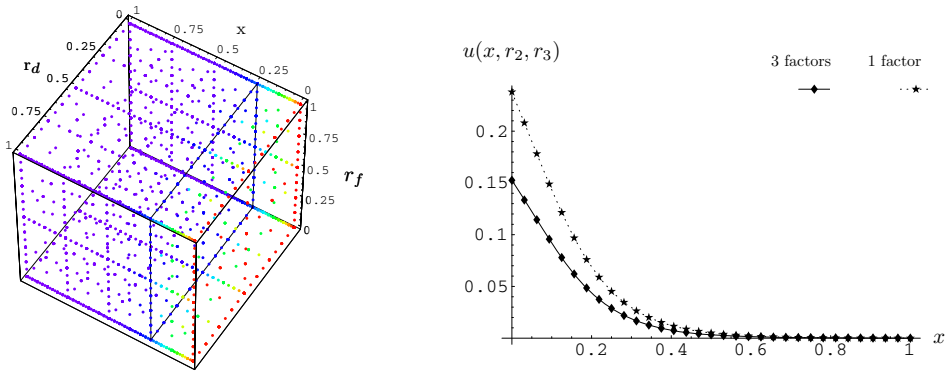


FIG. 3. *Left: Sparse grid for the* 3*-factor model with* 3726 *nodes (colored with the solution). Right: Comparison of the solution for fixed* r_i *with the* 1*-factor model.*

2.3.2. Basket option. In the rest of the paper we will be concerned with higher-dimensional applications, where no reference solutions on full grids exist. We focus on an option on a basket consisting of five assets, which is specified in Table 1.

TABLE 1
Basket of five equities with volatilities σ_i *and correlations* ρ_{ij}*, which have been estimated from historical time series. The weights* μ_i *are chosen as in the DAX.*

Equity	i	μ_i	σ_i	ρ_{ij}				
Deutsche Bank	1	38.1	0.518	1.00	0.79	0.82	0.91	0.84
Hypo-Vereinsbank	2	6.5	0.648	0.79	1.00	0.73	0.80	0.76
Commerzbank	3	5.7	0.623	0.82	0.73	1.00	0.77	0.72
Allianz	4	27.0	0.570	0.91	0.80	0.77	1.00	0.90
Münchner Rück	5	22.7	0.530	0.84	0.76	0.72	0.90	1.00

We next observe that the Black–Scholes equation for the put

$$\frac{\partial u}{\partial t} + \frac{1}{2}\sum_{i,j=1}^{d} \sigma_i\sigma_j\rho_{ij}S_iS_j\frac{\partial^2 u}{\partial S_i\partial S_j} + r\sum_{i=1}^{d} S_i\frac{\partial u}{\partial S_i} - ru = 0, \quad (\mathbf{S},t) \in \mathbb{R}^d_+ \times (0,T),$$

$$u(\mathbf{S},T) = \Big(K - \sum_{i=1}^{d}\mu_i S_i\Big)_+, \quad \mathbf{S} \in \mathbb{R}^d_+,$$

admits a transformation to the heat equation, which is composed of a change to logarithmic coordinates $\log \mathbf{S}$, a rotation with the eigenvectors $\mathbf{Q} = (\mathbf{q}_1, \ldots, \mathbf{q}_d)$ of the covariance matrix, a translation to eliminate the drift,

$$\mathbf{x} = \mathbf{Q}\ln\mathbf{S} - \mathbf{b}t,$$

where $b_i = \sum_{j=1}^{d} q_{ij}(r - 1/2\sigma_j^2)$, and a reversion of the time axis $t \to T - t$. This results in

$$\frac{\partial u}{\partial t} - \frac{1}{2}\sum_{i=1}^{d}\lambda_i\frac{\partial^2 u}{\partial {x_i}^2} + ru = 0, \quad (\mathbf{x},t) \in \mathbb{R}^d \times (0,T), \tag{21}$$

$$u(\mathbf{x},0) = \Big(K - \sum_{i=1}^{d}\mu_i e^{\sum_{j=1}^{d} q_{ji}x_j}\Big)_+, \quad \mathbf{x} \in \mathbb{R}^d, \tag{22}$$

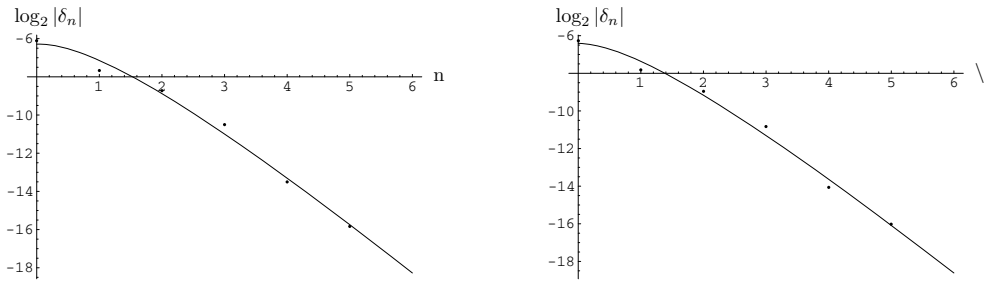


FIG. 4. *Left: Pointwise difference δ_n between the spot value on refinement levels n for the European 5 asset basket option. The asymptotic curve $-pn + q \log n$ was estimated by regression, where $p = 2.085$, $q = 3.115$ correspond well with the theoretical $p = 2$, $q = d - 1 = 4$. Right: The same analysis for the American put shows almost identical results with $p = 2.066$, $q = 2.915$. In both cases the option was evaluated at-the-money, that is, with the spot initially at the exercise boundary, with maturity $T = 1$ year.*

with the spectrum $\boldsymbol{\lambda}$ of the covariance matrix. The linear reaction term can be eliminated by introducing a discounting factor e^{-rt}. The domain in the new coordinates has to be localized, which is usually done by introducing upper bounds for the coordinates and setting asymptotic values. Instead we use a transformation

$$x_i \to \frac{1}{\pi} \arctan L x_i + \frac{1}{2}$$

of $\mathbb{R}^d$ to the unit cube ($L > 0$). Due to the nature of the transformation, the diffusion vanishes over the boundaries and no artificial conditions need to be set (see section 1.1), which would introduce additional errors that need to be controlled. Along the boundary faces the equation reduces naturally to a lower-dimensional one.

The convergence for the sparse grid combination technique, again with central differences, is shown in Figure 4. The number of unknowns on the finest grid is 372 909 780, which have been computed in parallel on 200 processors (see section 4.2).

The same numerical convergence analysis was performed for the American option, where the corresponding linear complementarity problem was solved by the techniques described in section 4.1. Despite the reduced regularity due to the presence of the free boundary, no deterioration of the convergence can be observed practically.

2.3.3. Considerations about higher dimensions. As seen in previous sections, the theoretically predicted convergence was recovered well even in cases lacking the required smoothness. In view of the asymptotic formula (17) it appears that given a certain refinement level n, the number of degrees of freedom only grows very weakly with the dimensionality. At the same time, in (19) the error only seems to deteriorate very weakly with increasing d, which is confirmed by the numerical results. Equation (20) even suggests that the prefactor goes down with increasing d. However, this asymptotic expansion is only a good approximation for $n \gg d$, and from Stirling's formula one sees that the prefactor actually grows like $(5e)^d/\sqrt{d}$.

Moreover, nothing has been said so far about the d-dependence of the "constant" K in (19), which depends crucially on higher mixed derivatives as derived in [18]. We return to this question to some extent in the context of dimension reduction techniques later.

Regarding the complexity of the method, although the total number of elements only increases asymptotically by a factor of two from one refinement level to the

next (regardless of the dimension!), this is again an asymptotic statement and the prefactors in (17) blow up with d as well.

To estimate the highest dimensions that can be handled, we consider the coarsest possible sparse grids at dimension d. In the combination technique d levels of grids are combined and it is only on level $d+1$ that the first inner point arises, yielding a grid of 3^d nodes. In terms of nodes this is in fact the smallest among the ν_d grids on that level, the largest grid having $m_d = 2^{d-1} \cdot \left(2^d + 1\right)$ nodes. A rough estimation shows that already in eight dimensions the coarsest reasonable sparse grid yields about 200 million unknowns, and in fact this was the highest dimension where this coarsest sparse grid fit into the main memory of the HELICS cluster, where the computations were performed (see section 4.2). It may be remarked that for the computations not all grids need be kept in memory simultaneously; however, it is clear that the computation time also increases proportionally. Further, the number of nonzero matrix entries in the discretized systems increases with d as well.

The above consideration also implies for Dirichlet problems that on levels lower than d the solution is interpolated only from the boundary data. Nevertheless in several cases good approximation may be achieved on much coarser levels and [21] reports relative errors below 5% in up to 20 dimensions for parabolic problems with sparse wavelets of only level $n = 5$. It has to be kept in mind that such results are not evidence for asymptotic convergence, but rather indicate an approximate lower-dimensional structure of the solution. In the following section we investigate such behavior theoretically and derive from a principal component analysis of typical financial data an approximation of the problem on a much smaller subspace of the sparse grid space.

3. Asymptotic expansions. To attribute the error components to the different dimensions we reconsider the error expansion

$$u - u_{\mathbf{h}} = \sum_{m=1}^{d} \sum_{\substack{\{j_1,\ldots,j_m\} \\ \subset \{1,\ldots,d\}}} h_{j_1}^2 \cdot \ldots \cdot h_{j_m}^2 \cdot \gamma_{j_1,\ldots,j_m}(.; h_{j_1}, \ldots, h_{j_m}). \tag{23}$$

Not only does (23) express the smoothness of the solution in terms of a multivariate Taylor expansion, but at the same time it decomposes the error into terms corresponding to lower-dimensional subspaces.

An obvious consequence is that if the solution is captured exactly in direction k, all terms containing h_k drop out and the expression reduces to a lower-dimensional one. In particular it is straightforward to reduce the problem if u has a representation of the form

$$u(\mathbf{x}, t) = u^{(m)}(x_{i_1}, \ldots, x_{i_m}, t), \tag{24}$$

and in that case u is said to have [illegible] m. Likewise, if u has a representation of the form

$$u(\mathbf{x}, t) = \sum_{\substack{\{i_1,\ldots,i_m\} \\ \subset \{1,\ldots,d\}}} u^{(i_1,\ldots,i_m)}(x_{i_1}, \ldots, x_{i_m}, t), \tag{25}$$

i.e., u has [illegible] m, again all terms containing more than m factors drop out of (23) and only m-dimensional problems need to be computed. We will now show how a representation of this sort can be obtained approximately for diffusion problems.

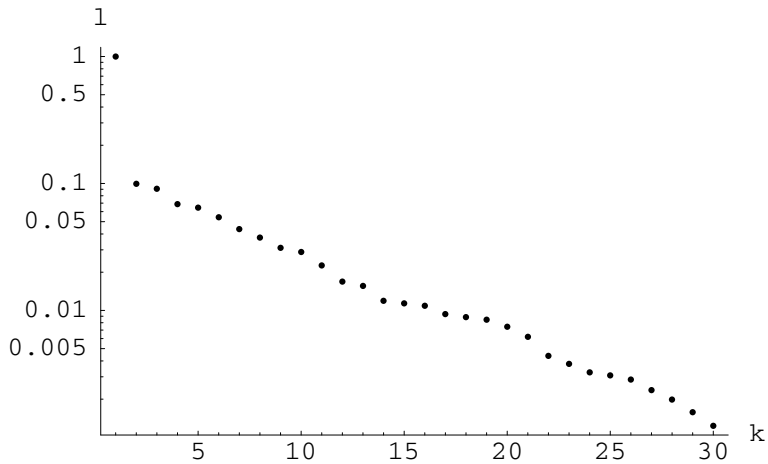


FIG. 5. *Ordered spectrum* $\{\lambda_k\}$ *of the covariance matrix of the* 30 *DAX assets from January* 16, 2003. *The volatilities* $\sigma_t = \sqrt{\frac{1}{T}\sum_{k=0}^{T-1} \varepsilon_{t-k}^2}$ *and correlations are estimated from the daily log-returns* ε_t *over a period of* $T = 250$ *days. The largest eigenvalue was scaled to* 1.

3.1. Data analysis. The geometric Brownian motion for the stocks

$$\mathrm{d}S_i/S_i = \mu_i \,\mathrm{d}t + \sigma_i \,\mathrm{d}W_i$$

is characterized by the covariance matrix of the Wiener process. The following discussion, illustrated by data from the German stock index DAX, is representative for many index and basket products and can be adapted for cases involving LIBOR rates or other correlated assets. In practice the high-dimensional driving process is often approximated by a small number of factors, for instance, in the case of equity indices by a single geometric Brownian motion, describing the movement of the index as a whole with an effective [illegible]. Often, particularly if the number of factors is moderate and they have nonnegligible idiosyncratic components, the deviation in the price is significant and such a procedure is inconsistent with the pricing of derivatives on single assets.

To allow for corrections (and to estimate the error) we study the spectrum $\boldsymbol{\lambda}$ of the covariance matrix and express the dependence of the solution on the vector $\boldsymbol{\lambda}$ of eigenvalues explicitly by $u(\boldsymbol{\lambda}, \mathbf{x}, t)$. For the DAX the spectrum is shown in Figure 5. A gap is observed after the first dominating eigenvalue that corresponds to the movement of the "market" itself. Note that in this example all correlations are positive so from Perron and Frobenius all entries of the first eigenvector are also positive.

The numerical approximations to the DAX options will be compared with results from Monte Carlo simulations.[2] The price found from sufficiently many ($N = 40000000$) Monte Carlo runs (see Table 2) for a European put option on the DAX with $T = 1$ (year), interest rate $r = 0.05$, and $K = 1$ [illegible] is

$$u(\mathbf{S}_0, 0) \approx 0.135323$$

with four significant digits.

[2]Thanks to Jürgen Schumacher from the Institute for Computer Science, Bonn University, for performing the simulations.

TABLE 2
Convergence of the Monte Carlo result u_N for a put on the DAX with N simulations, requiring CPU time t. The standard deviation of the result is $\sigma(u_N) \sim 1/\sqrt{N}$.

N	10^5	10^6	10^7	$2 \cdot 10^7$	$4 \cdot 10^7$
u_N	0.135083	0.135251	0.135343	0.135372	0.135323
$\sigma(u_N)$	5.36 e-4	1.69 e-4	5.36 e-5	3.79 e-5	2.68 e-5
t [sec]	7.7	76.9	768	1537	—

To assess the approximation quality in more detail we also study the five-factor basket already introduced in Table 1. This example was chosen because it is computationally fully tractable, and very accurate sparse grid solutions can be given. At the same time it shows the same characteristic features as higher-dimensional examples. There the spectrum is

$$\{1.409, 0.113, 0.101, 0.0388, 0.0213\},$$

and the eigenvector corresponding to the largest eigenvalue is

$$(0.41185, 0.49201, 0.46551, 0.45490, 0.40580)$$

and has (again positive) entries of similar magnitude. The price of a put on this subbasket (with the same other data) is

$$u(\mathbf{S}_0, 0) \approx 0.175866.$$

3.2. Principal component analysis. In (21) the Black–Scholes problem was expressed in terms of the eigensystem of the covariance matrix. This equation is now approximated by lower-dimensional ones by truncating the parameter vector $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_d)$ to $\boldsymbol{\lambda}^{(n)}$ with

$$\lambda_i^{(n)} := \begin{cases} \lambda_i, & 1 \le i \le n, \\ 0 & \text{else.} \end{cases} \tag{26}$$

The corresponding solution

$$u^{(n)}(\mathbf{x}, t) := u(\mathbf{x}, t, \boldsymbol{\lambda}^{(n)}) \tag{27}$$

solves

$$\frac{\partial u}{\partial t} = \frac{1}{2} \sum_{i=1}^{n} \lambda_i \frac{\partial^2 u}{\partial {x_i}^2} - ru, \quad (\mathbf{x}, t) \in \mathbb{R}^n \times (0, T), \tag{28}$$

$$u(x_1, \dots, x_n, 0) = \left(K - \sum_{i=1}^{d} \tilde{\mu}_i \mathrm{e}^{\sum_{j \le n} q_{ji} x_j} \right)_+, \quad \mathbf{x} \in \mathbb{R}^n. \tag{29}$$

Here $\tilde{\mu}_i = \mu_i \exp(\sum_{n<j\le d} q_{ji} x_j)$.

Note that if the solution is required only at a specific point $\mathbf{x}_0$ (in our context the option price is meaningful only at the current stock price $\mathbf{S}_0$), $x_j = x_j(\mathbf{S}_0, t)$ are kept fixed for $j > n$. It is sufficient to solve the PDE at an n-dimensional plane through $\mathbf{x}_0$. Table 3 shows the corresponding results for the five-factor basket which have been obtained with sufficient accuracy on sparse grids.

Note that approximations (27) do not indicate any truncation dimension in the sense of (24), because the solution is parameterized by the point $\mathbf{z}_0$. However, if we are interested in the solution at $\mathbf{z}_0$ alone, the consequences for the computation are the same.

TABLE 3
Approximations of dimension $n = 1, \dots, 4$ to the five-factor basket as in (27).

n	1	2	3	4	5
$u(\mathbf{S}_0, T)$	0.1806	0.1796	0.1777	0.1764	0.1758
rel. error	2.7 %	2.2 %	1.1 %	0.38 %	—

3.3. Asymptotic expansion. The results of the previous section show that additional components do not significantly improve the result, because the decay in the spectrum is rather slow. For problems of higher dimension this will not result in a reasonable approach, for many components will be required for reasonable accuracy and a sparse grid approximation would be practically impossible. We conclude that the smaller contributions cannot be entirely neglected, but possibly can be approximated linearly as

$$u(\mathbf{x}, t, \boldsymbol{\lambda}) = u^{(1)}(\mathbf{x}, t) + \sum_{j=2}^{d} \lambda_j \left.\frac{\partial u}{\partial \lambda_j}(\mathbf{x}, t, \boldsymbol{\lambda})\right|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}^{(1)}} + \mathcal{O}\left(\|\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(1)}\|^2\right) \tag{30}$$

with $\boldsymbol{\lambda}^{(1)}$ as in (26). This requires differentiability of the solution with respect to λ_j for $j > 1$ in $\boldsymbol{\lambda} = \boldsymbol{\lambda}^{(1)}$, which can be shown for sufficiently regular pay-offs. In the case of the basket option this means that the kink in the pay-off is not aligned with the principal component [17]. In fact in the Black–Scholes case the directional derivatives $\frac{\partial u}{\partial \lambda_j}$ can be explicitly expressed via Green's function of the heat equation.

Here we follow a more general approach and substitute a finite difference

$$\left.\frac{\partial u}{\partial \lambda_j}(\mathbf{x}, t, \boldsymbol{\lambda})\right|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}^{(j)}} = \frac{u^{(1,j)}(\mathbf{x}, t) - u^{(1)}(\mathbf{x}, t)}{\lambda_j} + \mathcal{O}(\lambda_j^2),$$

where

$$u^{(1,j)}(\mathbf{x}, t) = u(\mathbf{x}, t, \boldsymbol{\lambda}^{(1,j)}) \quad \text{with}$$
$$\lambda_i^{(1,j)} = \begin{cases} \lambda_i, & i = 1 \vee i = j, \\ 0 & \text{else} \end{cases}$$

can be computed by solving a two-dimensional problem that results from a perturbation of the one-dimensional equation for $u^{(1)}$ by λ_j. Substitution in (30) yields

$$u(\mathbf{x}, t) = u^{(1)}(\mathbf{x}, t) + \sum_{j=2}^{d} \left(u^{(1,j)}(\mathbf{x}, t) - u^{(1)}(\mathbf{x}, t)\right) + \mathcal{O}\left(\|\boldsymbol{\lambda} - \boldsymbol{\lambda}^{(1)}\|^2\right). \tag{31}$$

The crucial point is that the value $u(\mathbf{x}_0, t)$ that is the solution of the full d-dimensional equation can be expressed up to second order errors in λ_j $(j > 1)$ by the solution of one- and two-dimensional equations.

Application to the five-dimensional basket gives the results in Table 4, where the corrections to the one-dimensional approximation are compiled. The resulting approximation for $u(\mathbf{x}_0, t)$ is 0.175972 with an absolute error of 0.000106, i.e., 0.06%. For the full DAX the result is 0.135393 with comparable error 0.000073, i.e., 0.06%.

Note that even for the thirty-dimensional DAX only one one-dimensional and twenty-nine two-dimensional equations need to be solved. This number increases

TABLE 4
Corrections to the one-dimensional approximation $u^{(1)}$ according to (31).

k	2	3	4	5
$u^{(1,k)}$	0.179264	0.180095	0.178896	0.179344
$u^{(1,k)} - u^{(1)}$	-1.3810^{-3}	-5.5210^{-4}	-1.7510^{-3}	-1.0110^{-3}

linearly in d. The computational cost, however, increases superlinearly due to the evaluation of the transformation which is hidden in the initial condition and from the coordinate rotation yields an order d^2. Hence the total complexity is asymptotically $\mathcal{O}(d^3)$, which is not visible for low dimensions where the cost is dominated by the actual solution algorithm (solution of the linear systems in each time step) and not the transformation. Consequently even problems with hundreds of dimensions can be solved very quickly and accurately.

Finally it can be observed that the representation (31) is a special case of a hierarchical grid-based approach where the solution is assumed constant in all but two directions and is effectively represented by a single element in these directions. This can be the basis for a [illegible] procedure, which automatically detects lower-dimensional structures of this type by estimation of the hierarchical surplus.

4. Implementation issues. In connection with the reduction of the computational complexity it seems worthwhile pointing out a few key ingredients for the fast and robust solution of the implicit equations and an efficient implementation of the presented schemes. To fully benefit from the optimal approximation results and inherent parallelism, it is essential that the discrete systems are solved robustly in linear complexity and that a reasonable load balancing algorithm can be devised.

4.1. Multigrid. The main characteristics that arise in the discrete systems are related to the anisotropy of the grids and equations. In addition to the notion of grid size independent convergence rates over a specific refinement path, it is crucial here that these rates are robust with respect to the anisotropic refinements. In the transformed system (21) additional anisotropies of similar type enter through the diffusion coefficients and both effects can be dealt with by semicoarsening. In most practical cases where no transformation to constant coefficients exists, and especially if the diffusion operator comes from a bounded process or stretched grids and transformations to finite domains are employed, local degeneracy and anisotropies are inherently present (see, e.g., (9)). In [19] smoothers have been developed that generalize the concepts of line and plane smoothing to higher dimensions and are robust with respect to the effects just outlined. For the linear complementarity problems of types (10) to (12) (American options), projected smoothers and transfer operators that are adapted at the free boundary give results comparable to the European case.

As an example we consider the three-dimensional Black–Scholes equation for uncorrelated assets with $\sigma_i = 0.4$, $r = 0.05$, $T = 1$ (one time-step). Plane smoothing in the $V(1,1)$ cycle, where the plane systems are solved approximately by a single multigrid step with line smoothing, was used. The solution time is recorded in Table 5 for various grids with $N = \left(2^{l_1}+1\right) \cdot \left(2^{l_2}+1\right) \cdot \left(2^{l_3}+1\right)$ points, where $l_1 + l_2 + l_3 = 15$. These are the type of grids required for the combination technique. Ideally T/N would be constant over all grids, which is achieved within a factor of about 3. The consequences for load balancing are explained in section 4.2.

TABLE 5

CPU time T (on one node of the HELICS cluster; see below) for a relative error reduction by 10^{-12} for multigrid with the smoother explained above. On refinement levels (l_1, l_2, l_3) the $2^{l_i} + 1$ points in direction i give different totals of N, although the number of elements is equal for all grids. The sum of levels l_i is constant at 15 as in a sparse grid of that level.

l_1	l_2	l_3	N	T	T/N
0	0	15	131076	106.88	0.000815405
0	3	12	73746	29.71	0.000402869
0	6	9	66690	21.46	0.000321787
1	2	12	61455	27.89	0.000453828
1	5	9	50787	21.7	0.000427275
2	2	11	51225	25.54	0.000498585
2	5	8	42405	20.12	0.000474472
3	4	8	39321	19.91	0.000506345
4	4	7	37281	33.61	0.000901532
5	5	5	35937	17.58	0.000489189

TABLE 6

Total degrees of freedom M_n, maximum number of points m_n on a single grid, and the number ν_n of grids in the inner sum of (15) *in five dimensions.*

n	M_n	m_n	ν_n	n	M_n	m_n	ν_n	n	N_n	m_n	ν_n
1	32	32	1	6	42363	528	126	11	6042330	16400	1001
2	240	48	5	7	122125	1040	210	12	15185610	32784	1365
3	1120	80	15	8	337755	2064	330	13	37600980	65552	1820
4	4200	144	35	9	904745	4112	495	14	91913985	131088	2380
5	13890	272	70	10	2362620	8208	715	15	222166875	262160	3060

4.2. Parallelization. The combination technique has an inherent coarse-grain parallelism: the PDE is solved independently on a family of grids, which need to be combined linearly in the end only. In the present case, where we are interested in the evaluation of the solution at a single point, the numerical solution on each grid is interpolated at this point and then added up with the appropriate weights. The communication between processors is restricted to a single concentration step for a scalar value.

Table 6 collects data for the (largest) five-dimensional sparse grid used in the simulation of Figure 4. Evidently the large number of nodes is distributed over a large number of grids that can be solved fully in parallel, and the remaining problem is efficient load balancing. In the implementation that produced the data of this section the grids were numbered level by level in their canonical order and then distributed in as many loops as required over the processors. That way each processor obtained roughly an equal number of unknowns, where the deviation was of the order of magnitude of the largest single grid. A more sophisticated a priori distribution which minimized the memory imbalance did not necessarily lead to shorter CPU times.

The parallel computations were performed on the Heidelberg Linux Cluster System (HELICS, http://helics.iwr.uni-heidelberg.de).

From the distribution of solution times of the single processors in a 200 processor simulation (Figure 6) we see that the difference to the ideal case (in which all processors would finish at the same time) is about one-third. This comes from two reasons. First, a perfectly even distribution of the unknowns is not possible, but bounded by the number of unknowns on the finest grids in the combination technique. In this case

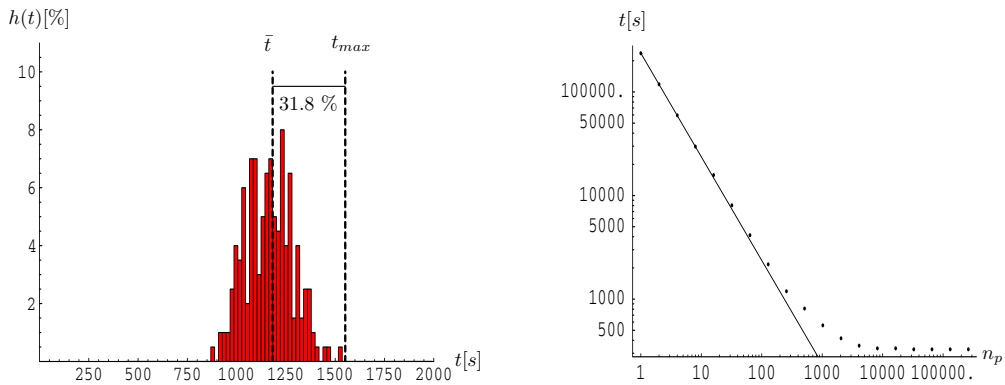


FIG. 6. *Histogram $h(t)$ of CPU time t of each of* 200 *processors and mean $\bar{t}$ (left) and scaling with increasing number n_p of processors (right; see also Table* 7*).*

TABLE 7
Scaling of CPU time t with number of processors n_p.

$\log_2 n_p$	1	2	3	4	5	6	7	
	8	9	10	11	12	13	14,15	≥ 16
$t[s]$	2.37 e+5	11.8 e+5	5.94 e+4	2.98 e+4	1.58 e+4	8.04 e+3	4.14 e+3	
	2.17 e+3	1.19 e+3	8.11 e+2	5.59 e+2	4.18 e+2	3.55 e+2	3.34 e+2	3.27 e+2

the number of grids per processor, which corresponded to the finest level, was either six or seven. This can be expected to account for about half of the observed difference. Second, and more importantly, due to the anisotropic structure of the discrete problems, even the presented robust and asymptotically linear solver is of course not exactly linear for finitely many unknowns.

Since the communication between processors is negligible, it is possible to predict the efficiency of a parallelization strategy for a certain number of processors, even if such a large number is actually not available, given the solution time on all the grids involved in the combination technique. Table 7 was compiled through this procedure; the data are also plotted in Figure 6. It shows that the speedup is almost optimal as long as the number of grids to distribute is large compared to the number of processors. In the given example very good results were observed for up to 100 processors. Clearly if there are more processors than grids, there is no improvement and the curve flattens out. In this case the problem is just too small for the computer. From a more reasonable viewpoint one would then increase the problem size linearly with the number of nodes and then the speedup remains constant at the given level.

Finally an a posteriori analysis which identified a CPU-time optimal distribution after the computation showed that no significant improvement to the simple original scheme can be achieved in most cases.

5. Discussion and outlook. We have seen that there are two seemingly complementary aspects to the sparse grid combination technique. On the one hand, if the problem can be well approximated by a sum of lower-dimensional functions, the hierarchical representation is well suited to decompose the solution in this way and the complexity is reduced accordingly. On the other hand, for a fully high-dimensional structure, sparse grids provide an asymptotically optimal approximation (see [4]), for

which the complexity grows only very weakly with the dimension of the problem. This can be combined in an adaptive algorithm which identifies refinement directions from already computed hierarchical surpluses. Such a strategy has been applied to quadrature in [9]; a robust strategy that automatically reduces to the cases presented in this article (in the respective setting) is currently under investigation by the authors.

In financial practice a central question is the calibration of the models, and often time-dependent and local parameters are fitted such that the model hits quoted option prices for different strikes and maturities. The same ideas illustrated here can be carried forward to the solution of the resulting PDEs with nonconstant coefficients, and in fact the FX-option model was such an example. Principal component analysis has to be performed locally in such a setting. A further open question is then the interplay between calibration and hierarchical approximation.

As an example of products with a more complicated cash-flow structure, interest rate derivatives have been studied in [2]. Swaptions are options on interest rate swaps between fixed and floating interest rates at predefined dates, often with Bermudan features such that the option may also be exercised at these tenor dates. This results in update conditions at these dates, between which a parabolic equation is satisfied. Dimension reduction for such a correlation structure between forward rates with similar time horizons is currently being investigated.

Along similar lines, extensions of the results for basket options to basket credit derivatives are the subject of ongoing research.

REFERENCES

[1] F. Black and M. Scholes, *The pricing of options and corporate liabilities*, J. Pol. Econ., 81 (1973), pp. 637–659.

[2] J. Blackham, *Sparse Grid Solutions to the LIBOR Market Model*, Master's thesis, Oxford University, Oxford, UK, 2004.

[3] H.-J. Bungartz, *Dünne Gitter und deren Anwendung bei der adaptiven Lösung der dreidimensionalen Poisson–Gleichung*, Ph.D. thesis, Technische Universität München, Munich, Germany, 1992.

[4] H.-J. Bungartz, *Finite Elements of Higher Order on Sparse Grids*, Habilitationsschrift, Technische Universität München, Munich, Germany, 1998.

[5] H.-J. Bungartz, M. Griebel, D. Röschke, and C. Zenger, *Pointwise convergence of the combination technique for the Laplace equation*, East-West J. Numer. Math., 2 (1994), pp. 21–45.

[6] R. Carmona, ed., *Utility Indifference Pricing*, Princeton University Press, Princeton, NJ, 2006.

[7] A. Etheridge, *A Course in Financial Calculus*, Cambridge University Press, Cambridge, UK, 2002.

[8] T. Gerstner and M. Griebel, *Numerical integration using sparse grids*, Numer. Algorithms, 18 (1998), pp. 209–232.

[9] T. Gerstner and M. Griebel, *Dimension-adaptive tensor-product quadrature*, Computing, 71 (2003), pp. 65–87.

[10] P. Glasserman, *Monte Carlo Methods in Financial Engineering*, Springer-Verlag, New York, 2004.

[11] R. Glowinski, *Numerical methods for the Navier–Stokes equations*, Comput. Phys. Rep., 6 (1987), pp. 73–187.

[12] M. Griebel, M. Schneider, and C. Zenger, *A combination technique for the solution of sparse grid problems*, in Iterative Methods in Linear Algebra, P. de Groen and R. Beauwens, eds., IMACS, Elsevier, North–Holland, Amsterdam, 1992, pp. 263–281.

[13] F. A. Longstaff and E. S. Schwartz, *Valuing American options by simulation: A simple least-squares approach*, Rev. Financial Stud., 14 (2001), pp. 113–148.

[14] M. Monoyios, *Efficient option pricing with transaction costs*, J. Comput. Fin., 7 (2003), pp. 107–126.

[15] H. Niederreiter, *Random Number Generation and Quasi-Monte Carlo Methods*, SIAM, Philadelphia, 1992.
[16] B. Øksendal, *Stochastic Differential Equations*, 5th ed., Springer-Verlag, Berlin, 1998.
[17] C. Reisinger, *Numerische Methoden für hochdimensionale parabolische Gleichungen am Beispiel von Optionspreisaufgaben*, Ph.D. thesis, Universität Heidelberg, 2004.
[18] C. Reisinger, *Analysis of Linear Difference Schemes in the Sparse Grid Combination Technique*, preprint, http://eprints.maths.ox.ac.uk, 2006.
[19] C. Reisinger and G. Wittum, *On multigrid for anisotropic equations and variational inequalities*, Comput. Vis. Sci., 7 (2004), pp. 189–197.
[20] S. A. Smolyak, *Quadrature and interpolation formulas for tensor products of certain classes of functions.*, Dokl. Akad. Nauk SSSR, 148 (1963), pp. 1042–1043 (in Russian); English translation in Soviet Math. Dokl. 4 (1963), pp. 240–243.
[21] T. von Petersdorff and C. Schwab, *Numerical Solution of Parabolic Equations in High Dimensions*, Tech. report, Isaac Newton Institute, Cambridge, UK, 2002.
[22] P. Wilmott, S. D. Howison, and J. N. Dewynne, *The Mathematics of Financial Derivatives*, Cambridge University Press, Cambridge, UK, 1995.
[23] C. Zenger, *Sparse grids*, in Parallel Algorithms for Partial Differential Equations, Proceedings of the 6th GAMM-Seminar, W. Hackbusch, ed., Notes Numer. Fluid Mech. 31, Vieweg Verlag, 1990, pp. 241–251.
[24] Y.-L. Zhu and J. Li, *Multi-factor financial derivatives on finite domains*, Commun. Math. Sci., 1 (2003), pp. 343–359.

SIAM J. SCI. COMPUT.
Vol. 29, No. 2, pp. 459–480
© 2007 Society for Industrial and Applied Mathematics

STRING GRADIENT WEIGHTED MOVING FINITE ELEMENTS IN MULTIPLE DIMENSIONS WITH APPLICATIONS IN TWO DIMENSIONS*

A. WACHER† AND I. SOBEY‡

Abstract. We formulate the string gradient weighted moving finite element method (SGWMFE) for systems of PDEs in multiple dimensions. Then we illustrate implementation issues for the method using two dimensions. The method is applied successfully to solve the highly nonlinear unsteady porous medium equation in two dimensions and the results are compared to exact solutions. We proceed to present results for the method when applied to the Gray Scott chemical reaction diffusion model in two dimensions. Finally, we apply the method to the shallow water equations in two dimensions and compare the meshes produced with those produced using gradient weighted finite element methods (GWMFE). Our conclusion is that SGWMFE is an easily applied member of the group of moving finite element methods.

Key words. moving finite elements, mesh adaptation

AMS subject classifications. 74S05, 65M60, 65M50

DOI. 10.1137/040619557

1. Introduction.

This paper has two objectives. The first is to present the general formulation for a string gradient weighted moving finite element method (SGWMFE) in multiple space dimensions. The second is to demonstrate the implementation of SGWMFE in several nonlinear systems of PDEs, the porous medium equation, the Gray Scott equations, and the shallow water equations. Solutions of each problem can be found using other methods, and we are not suggesting or attempting to demonstrate that SGWMFE is better or worse than other moving (or nonmoving) finite element or discretization methods; rather, it is an implementation of gradient weighted moving finite elements (GWMFE) with an elegant general form that is straightforward to use.

Moving finite element methods (MFE), originally introduced in [14], are well established to provide reliable solutions to unsteady systems of PDEs, often with less computational cost than fixed mesh methods. There is a wide literature dealing with moving mesh methods, both for finite elements and for finite differences. The text of [1] gives an extensive view of MFE and [5, 11, 15, 16] describe recent applications of moving mesh methods. GWMFE were developed in detail in [8] and [9], and the string version, suggested in [12], was implemented in [17]. For other background on moving mesh methods, see [17], where there is an extended reference list on moving mesh methods including MFE, GWMFE, and moving finite difference type methods. Here our interest is however in developing details of SGWMFE for systems of PDEs in two or more dimensions. Using a projection matrix, a very general form of the method can be developed and details are illustrated for a system of two PDEs. There

*Received by the editors November 24, 2004; accepted for publication (in revised form) August 30, 2006; published electronically March 15, 2007.
http://www.siam.org/journals/sisc/29-2/61955.html

†Faculty of Aerospace Engineering, Technion City, Haifa 32000, Israel (abigail.wacher@comlab.ox.ac.uk). This author was supported in part by the National Science and Engineering Research Council of Canada in the form of the NSERC graduate scholarship and by Corpus Christi College, Oxford.

‡Oxford University Computing Laboratory, Parks Road, Oxford OX1 3PQ, UK (ian.sobey@comlab.ox.ac.uk).

are a number of implementation issues for MFE and the details of implementation of SGWMFE are examined for a number of commonly occurring terms in first and second order PDEs.

In the remainder of this section we set out the three test problems that are considered in this paper. In the second section we develop a mathematical formulation for SGWMFE that is elegant and easily generalized to arbitrary systems of evolutionary PDEs in multiple space dimensions. The third section considers implementation issues for a system two independent variables in two space dimensions, focusing the theory from section two for this case and showing details of how different terms are treated. The fourth section gives solutions for the three test problems using SGWMFE.

1.1. Porous medium equation. The first test problem we consider is the application to the porous medium equation in two space dimensions. While this is a single PDE rather than a system, it admits an exact similarity solution under some initial conditions so that the results of applying SGWMFE can be compared to the exact solution.

The porous medium equation is well known (see [6] or [4]) but in computational terms represents a hard problem because under some parameter values, the solution develops a singularity in the slope of an advancing front. The porous medium equation in two dimensions is given by

$$u_t = \nabla \cdot (u^m \nabla u), \tag{1.1}$$

and we consider the problem with $u = 0$ imposed at the boundary.

This is a problem with a known set of similarity solutions when the initial data has circular symmetry. The symmetric self-similar solutions used are of the same form as in [2] and originally from [4]. Earlier studies on self-similar numerical solutions of the porous medium equation can be found in [6] and [7]. The following form represents the solutions for the one- ($d = 1$), two- ($d = 2$), and three- ($d = 3$) dimensional problems for an initial distribution centered at the origin and starting at time t_0:

$$u(r,t) = \left\{ \begin{array}{ll} \dfrac{1}{\lambda^d(t)}\left(1 - \left(\dfrac{r}{r_0\lambda(t)}\right)^2\right)^{1/m}, & r \le r_0\lambda(t) \\ 0, & r > r_0\lambda(t) \end{array} \right\}, \tag{1.2}$$

where r is distance from the origin, m is an integer greater than 0, and

$$\lambda(t) = \left(\frac{t}{t_0}\right)^{\frac{1}{2+md}} \quad \text{with} \quad t_0 = \frac{r_0^2 m}{2(2+md)} \ . \tag{1.3}$$

The computations described below are for times $T = t - t_0 > 0$. The computational difficulty with this problem is that the slope of the solution at the moving front, $r = r_0\lambda(t)$, is singular when $m > 1$.

1.2. Gray Scott equations. The Gray Scott equations are a model for chemical species undergoing reaction and diffusion. Reaction and diffusion of chemical species can lead to a variety of patterns which have beautiful and interesting solutions. For a description of the physics of the model studied here, see [10]. The reader is also referred to [20] for a comparison of results for the following problem using a different class of moving mesh method.

The chemical reaction we consider is that given by

$$\begin{aligned} U + 2V &\to 3V, \\ V &\to P, \end{aligned}$$

where U, V, and P are chemical species. The PDEs that model this process are

$$\begin{aligned} u_t &= r_u \nabla^2 u - uv^2 + f(1-u), \\ v_t &= r_v \nabla^2 v + uv^2 - (f+k)v, \end{aligned} \tag{1.4}$$

where u and v represent their concentrations, r_u and r_v are their diffusion rates, k represents the rate of conversion of V to P, and f represents the rate of the process that feeds U and drains U, V, and P.

The initial conditions considered are block functions for each of the chemical concentrations over the domain $[0,1] \times [0,1]$:

$$u(x,y,t) = \left\{ \begin{array}{ll} 0.5, & 0.3 \le (x,y) \le 0.7 \\ 1 & \text{elsewhere} \end{array} \right\}, \tag{1.5}$$

$$v(x,y,t) = \left\{ \begin{array}{ll} 0.25, & 0.3 \le (x,y) \le 0.7 \\ 0 & \text{elsewhere} \end{array} \right\}. \tag{1.6}$$

The coefficients chosen for this problem are $r_u = 8(10^{-5})$, $r_v = 4(10^{-5})$, $f = 0.024$, and $k = 0.06$ with Dirichlet boundary conditions prescribed on all four boundaries of the domain.

1.3. Shallow water equations. The nonlinear shallow water equations provide a further model problem to test SGWMFE. It is also possible to compare the results with those obtained from the GWMFE code used in [8]. In our shallow water problem, a "hump" of stationary water is released at time zero in the center of a square box. As the hump subsides under gravity, a wave propagates away from the center, collides with the reflective boundaries, and propagates back to the center of the box. As they travel, a wave front steepens, forming a shock in the absence of viscosity. In this problem a simulation should capture the shape of the wave front, the wave height, and the front speed.

The nonlinear shallow water equations studied here are, in nondimensional terms, given by

$$\begin{aligned} u_t &= -\nabla \cdot \mathbf{f} + \epsilon \nabla^2 u, \\ v_t &= -\nabla \cdot \mathbf{g} + \epsilon \nabla^2 v, \\ w_t &= -\nabla \cdot \mathbf{h} + \epsilon \nabla^2 w. \end{aligned} \tag{1.7}$$

The physical domain is given by $0 \le x \le 5$, $0 \le y \le 5$, time $t \ge 0$, u is the height of the fluid from the flat bottom, v is the fluid momentum in the x direction, and w is the fluid momentum in the y direction.

The vector functions in (1.7) are

$$\mathbf{f} = (v, w), \quad \mathbf{g} = \left(\frac{v^2}{u} + \frac{u^2}{2}, \frac{vw}{u} \right), \quad \mathbf{h} = \left(\frac{vw}{u}, \frac{w^2}{u} + \frac{u^2}{2} \right). \tag{1.8}$$

The initial conditions assigned are

$$u = 0.2 + e^{-((x-2.5)^2 + (y-2.5)^2)}, \tag{1.9}$$

$$v = 0, \tag{1.10}$$

$$w = 0. \tag{1.11}$$

On the south and north of the domain, the boundary conditions are that $u_y = 0$, $v_y = 0$, and $w = 0$. On the east and west of the domain, the boundary conditions are that $u_x = 0$, $w_x = 0$, and $v = 0$.

2. Formulation of SGWMFE in multiple dimensions. We consider a system of n PDEs in k dimensions with independent variables $x = (x_1, \dots, x_k)$ and dependent variables $u = (u_1, \dots, u_n)$ satisfying

$$\frac{\partial u_l}{\partial t} = L_l(u_1, \dots, u_n), \qquad l = 1, \dots n, \tag{2.1}$$

where the differential operators L_l are first or second order nonlinear differential operators in space.

The solution u can be considered as an n-dimensional manifold in $(n + k)$-dimensional space; that is,

$$\mathbf{s} = (x_1, \dots, x_k, u_1, \dots, u_n)^{\mathrm{T}}. \tag{2.2}$$

Consider a parameterization of the manifold $\xi = (\xi_1, \dots, \xi_k)$ so that the solution manifold is described by

$$\mathbf{s}(\xi, t) = (x_m(\xi, t), m = 1 \dots k, u_l(x_j(\xi, t), j = 1 \dots k, t), l = 1 \dots n)^{\mathrm{T}}. \tag{2.3}$$

The velocity of the manifold, $\dot{\mathbf{s}}$, is not computed directly; rather, the normal component of the velocity, $\dot{\mathbf{s}}_n$ (not to be confused with the rate of change of the normal to the manifold, so that $\dot{\mathbf{s}}_n \equiv [\dot{\mathbf{s}}]_n$), is calculated, and normal movement of the manifold is deduced from that velocity. Tangential movement of the manifold is not considered since the location of the nodes is not affected.

In order to compute the normal velocity it is necessary to be able to evaluate the normal component of vectors on the manifold. This can be done by defining k tangent vectors to the manifold $\mathbf{X}_m = (0 \dots 1 \dots 0, \frac{\partial u_l}{\partial x_m}, l = 1 \dots n)^{\mathrm{T}}$, $m = 1, \dots, k$, where the unit value occurs in the mth row. These tangent vectors are always nonparallel and nonzero. Define matrices

$$X = [\mathbf{X}_1, \dots, \mathbf{X}_k], \quad V = (\mathbf{X}_l^{\mathrm{T}} \mathbf{X}_m) = X^{\mathrm{T}} X,$$

and the properties of the component vectors (nonzero and nonparallel) will guarantee that V^{-1} will exist. Define D by

$$D = \det(V). \tag{2.4}$$

LEMMA 2.1. [illegible] $\mathbf{s}_n$ [illegible] $\mathbf{s}$ [illegible]

$$\mathbf{s}_n = P\mathbf{s}, \tag{2.5}$$

[illegible] P [illegible]

$$P = I - XV^{-1}X^{\mathrm{T}}. \tag{2.6}$$

[illegible]. If $\mathbf{s}_n = \mathbf{s} - \sum_{l=1}^{k} a_l \mathbf{X}_l = \mathbf{s} - X\mathbf{a}$ then choosing $\mathbf{a}$ so that $\mathbf{s}_n \cdot \mathbf{X}_m = 0$ for $m = 1, \dots, k$ gives $\mathbf{a} = V^{-1}X^{\mathrm{T}}\mathbf{s}$ and hence the equation for P. □

Note that $P\mathbf{X}_m = 0$ for $m = 1, \dots, k$ trivially as the tangent vectors have no normal component.

Having identified the normal component of a vector on the manifold, the normal velocity can be determined. To do this, use

$$\dot{x}_m = \frac{\partial x_m(\xi, t)}{\partial t}, \quad m = 1, \dots, k, \tag{2.7}$$

and

$$\dot{u}_l = \frac{\partial u_l}{\partial t} + \sum_{j=1}^{k} \frac{\partial u_l}{\partial x_j}\dot{x}_j, \quad l = 1, \ldots, n, \tag{2.8}$$

and rearrangement gives that, if

$$R = \begin{pmatrix} 0_{k\times k} & 0_{k\times n} \\ [-u_{l,m}]_{n\times k} & I_{n\times n} \end{pmatrix}, \tag{2.9}$$

where $0_{k\times n}$ is a k-row, n-column matrix of zeros, $I_{n\times n}$ is an n by n identity matrix, and $[-u_{l,m}]$ is an n-row, k-column matrix with entries $-\frac{\partial u_l}{\partial x_m}$, then

$$R\dot{\mathbf{s}} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \frac{\partial u_1}{\partial t} \\ \vdots \\ \frac{\partial u_n}{\partial t} \end{pmatrix}. \tag{2.10}$$

LEMMA 2.2. [illegible] P [illegible] R [illegible]

$$PR = P. \tag{2.11}$$

[illegible]. The matrix R is rewritten as $R = I - [\mathbf{X}_1 \cdots \mathbf{X}_k \mathbf{0} \cdots \mathbf{0}]$, and then we use $P\mathbf{X}_m = 0$. □

Thus we obtain

$$\dot{\mathbf{s}}_n = P\dot{\mathbf{s}} = PR\dot{\mathbf{s}} = P\begin{pmatrix} 0 \\ \vdots \\ 0 \\ \frac{\partial u_1}{\partial t} \\ \vdots \\ \frac{\partial u_n}{\partial t} \end{pmatrix} = P\begin{pmatrix} 0 \\ \vdots \\ 0 \\ L_1 \\ \vdots \\ L_n \end{pmatrix} = P\mathbf{L}, \tag{2.12}$$

with $\mathbf{L} = (0, \ldots, 0, L_1, \ldots, L_n)^{\mathrm{T}}$. Thus the normal motion of the manifold is determined by

$$P\big[\dot{\mathbf{s}} - \mathbf{L}\big] = \mathbf{0}. \tag{2.13}$$

This system is central to the solution method since the original n PDEs have been replaced by $n + k$ equations for the manifold velocities $(\dot{x}_1, \ldots, \dot{x}_k, \dot{u}_1, \ldots, \dot{u}_n)$ and upon space discretization, these will become $n + k$ nonlinear ODEs in time.

3. Implementation of SGWMFE in two dimensions. Having considered the very general formulation, in this section we look at the details of implementing the method for a system of two PDEs in two dimensions. For simplicity, denote the independent variables x and y and the dependent variables u and v, and let them be governed by

$$u_t = L_1(u, v), \qquad v_t = L_2(u, v). \tag{3.1}$$

The solution (u, v) can be considered as a two-dimensional manifold in four-dimensional space; that is,

$$\mathbf{s} = (x, y, u, v)^{\mathrm{T}}. \tag{3.2}$$

Two tangent vectors to the manifold will be $\mathbf{X}^{\mathrm{T}} = (1, 0, u_x, v_x)$ and $\mathbf{Y}^{\mathrm{T}} = (0, 1, u_y, v_y)$, and the determinant D is

$$D = |\mathbf{X}|^2|\mathbf{Y}|^2 - (\mathbf{X} \cdot \mathbf{Y})^2 = (|\mathbf{X}||\mathbf{Y}||\sin(\theta)|)^2, \tag{3.3}$$

where θ is the angle between $\mathbf{X}$ and $\mathbf{Y}$.

The normal $\mathbf{s}_n$ to the manifold $\mathbf{s}$ is given by

$$\mathbf{s}_n = P\mathbf{s} \tag{3.4}$$

(see (2.5)), where the matrix P from (2.6) is given by

$$P = I - \frac{1}{D}[|\mathbf{Y}|^2(\mathbf{X}\mathbf{X}^T) - (\mathbf{X} \cdot \mathbf{Y})(\mathbf{X}\mathbf{Y}^T) + |\mathbf{X}|^2(\mathbf{Y}\mathbf{Y}^T) - (\mathbf{X} \cdot \mathbf{Y})(\mathbf{Y}\mathbf{X}^T)]. \tag{3.5}$$

The matrix R is

$$R = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -u_x & -u_y & 1 & 0 \\ -v_x & -v_y & 0 & 1 \end{pmatrix} \tag{3.6}$$

so that

$$R\dot{\mathbf{s}} = R \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{u} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ u_t \\ v_t \end{pmatrix}. \tag{3.7}$$

As in the general case, obtain

$$\dot{\mathbf{s}}_n = P\dot{\mathbf{s}} = PR\dot{\mathbf{s}} = P \begin{pmatrix} 0 \\ 0 \\ u_t \\ v_t \end{pmatrix} = P \begin{pmatrix} 0 \\ 0 \\ L_1 \\ L_2 \end{pmatrix} = P\mathbf{L}, \tag{3.8}$$

with $\mathbf{L} = (0, 0, L_1, L_2)^{\mathrm{T}}$. Thus, as before, the normal motion of the manifold is determined by

$$P\big[\dot{\mathbf{s}} - \mathbf{L}\big] = \mathbf{0}. \tag{3.9}$$

Discretization of this system can be cast in a finite element framework by using appropriate test functions—in this case the piecewise linear basis functions which are used also to discretize the solutions for (u, v) at time varying (x, y) nodal locations. If the basis functions are denoted α^i then the discrete equations for the normal motion of the manifold are

$$\int_{\text{manifold}} P(\dot{\mathbf{s}} - \mathbf{L})\alpha^i \mathrm{d}S = \mathbf{0}, \tag{3.10}$$

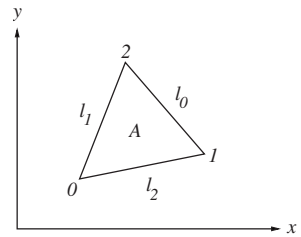


FIGURE 3.1. *Triangle T, showing the orientation of nodes 0, 1, and 2, the projected $x-y$ area A, and the edge lengths ℓ_0, ℓ_1, and ℓ_2.*

where $\mathrm{d}S$ is an element of the manifold and the superscript i ranges over the basis functions. In two dimensions, a triangular cell T can be used, as in Figure 3.1, and then, after piecewise linear discretization, u_x, v_x, u_y, v_y, and hence also P are constant on each cell. The parameterization used for the implementation of SGWMFE is $\xi_1 = x$ and $\xi_2 = y$, so that two infinitesimal vectors $\mathbf{X}\mathrm{d}x$ and $\mathbf{Y}\mathrm{d}y$ span a parallelogram in $4-D$ of area

$$\mathrm{d}S = |\mathbf{X}\mathrm{d}x||\mathbf{Y}\mathrm{d}y||\sin(\theta)| = \sqrt{D}\mathrm{d}x\mathrm{d}y. \tag{3.11}$$

It can also be noted that this SGWMFE formulation is equivalent to a variational formulation which chooses to minimize the functional

$$\psi = \int ||P(\dot{\mathbf{s}} - \mathbf{L})||^2 \mathrm{d}S, \tag{3.12}$$

with respect to the nodal velocities $\dot{\mathbf{s}}_i = (\dot{x}_i, \dot{y}_i, \dot{u}_i, \dot{v}_i)$.

In order to solve the discrete system of ODEs, an algorithm such as the well-known backward differentiation formula 2 (BDF2) found in [3] can be used. The BDF2 integrator is a robust stiff integrator which is necessary for the stiff systems arising in SGWMFE; however, any robust stiff integrator may be used. The BDF2 integration code developed and made available by Carlson and Miller was used for the results in this paper. A detailed outline of the BDF2 integrator routine is in [8].

3.1. Implementation. The implementation of SGWMFE requires some notation for the average value of a scalar or vector valued function f on a cell or an edge. An element of area on the manifold, $\mathrm{d}S$, is related to the projected $x-y$ area by (3.11), and since D is constant on each element, the projected area A of an element with area S is related by $S = \sqrt{D}A$. Hence the average of a function f over a triangular cell T is given by

$$[f]_T = \frac{1}{A}\int_T f\mathrm{d}x\mathrm{d}y = \frac{1}{S}\int_T f\mathrm{d}S. \tag{3.13}$$

We will also assume that the three nodes of the triangle T are labeled $0, 1, 2$, as in Figure 3.1.

The average of f over an edge E of length ℓ is given by

$$[f]_E = \frac{1}{\ell}\int_E f\mathrm{d}s, \tag{3.14}$$

where the integrand is with respect to the infinitesimal length along the edge: $\mathrm{d}s = \sqrt{\mathrm{d}x^2 + \mathrm{d}y^2}$.

3.2. Time derivative terms. As P is constant on an element, the components of the system for time derivatives on a triangular cell T are given by

$$\int_T P\dot{\mathbf{s}}\alpha^0 \mathrm{d}S = P\int_T \dot{\mathbf{s}}\alpha^0 \mathrm{d}S. \tag{3.15}$$

Similar contributions will occur for nodes 1 and 2 of the cell. Since the $\dot{\mathbf{s}}$ and α^0 are linear functions of the two-dimensional parameter ξ, the integrand in (3.15) is quadratic in the components of ξ and hence can be evaluated by the "edge-midpoint rule" which is exact for quadratic functions.

3.3. Flux terms. Since many systems of interest contain a flux term, consider the case where $L_1 = \nabla\cdot\mathbf{f}$. For the given triangular cell T, the contributions from this flux term from the cell onto its 0 node are

$$\int_T P\begin{pmatrix}0\\0\\-\nabla\cdot\mathbf{f}\\0\end{pmatrix}\alpha^0\mathrm{d}S = \sqrt{D}P\begin{pmatrix}0\\0\\1\\0\end{pmatrix}\int_T(-\nabla\cdot\mathbf{f})\alpha^0\mathrm{d}x\mathrm{d}y, \tag{3.16}$$

where, again, the constancy of P and D on each cell enables these to be taken out of the integral. The scalar integral expression may be integrated by parts, as in [9]:

$$\int_T -\nabla\cdot\mathbf{f}\alpha^0\mathrm{d}x\mathrm{d}y = \int_T \mathbf{f}\cdot\nabla\alpha^0\mathrm{d}x\mathrm{d}y - \int_{E_1}\alpha^0\mathbf{f}\cdot\widehat{\nu}_1\mathrm{d}s - \int_{E_2}\alpha^0\mathbf{f}\cdot\widehat{\nu}_2\mathrm{d}s, \tag{3.17}$$

where E_1 and E_2 are edges adjacent to node 0 and $\widehat{\nu}_1$, $\widehat{\nu}_2$ are outward normals to those edges (α^0 is zero on the third edge).

This component can be simplified by using average values. Defining ℓ_i as the $x-y$ projected lengths of the corresponding edge E_i, (3.17) is rewritten in terms of average values as

$$\int_T -\nabla\cdot\mathbf{f}\alpha^0\mathrm{d}x\mathrm{d}y = A\nabla\alpha^0\cdot[\mathbf{f}]_T - l_1\widehat{\nu}_1\cdot[\alpha^0\mathbf{f}]_{E_1} - l_2\widehat{\nu}_2\cdot[\alpha^0\mathbf{f}]_{E_2}. \tag{3.18}$$

3.4. Constant coefficient diffusion terms. Another common component of the differential operators L_1 and L_2 is a diffusion term. We consider first the case of constant diffusion; the second case of nonconstant or nonlinear diffusion is considered later. In order to evaluate the integrals of the Laplacian terms, mollification is used. For piecewise linear approximations, the Laplacian is singular at an edge. In order to overcome this, consider the integrals in neighborhoods of each edge and assume that the piecewise linear approximation is the limit of an approximation which can be differentiated twice; that is, the piecewise linear representations of the solution variables need to be mollified so that the first derivatives vary smoothly from cell to cell in a neighborhood whose thickness will be allowed to vanish. The principle here is the same as in the two-dimensional paper of [9]. To do this, some of the same labels as in that paper are used to define cells, edges, endpoints, and unit normal vectors.

Without loss of generality, assume that the 0 to 1 edge E is aligned in the y direction as shown in Figure 3.2. This is done so that the unit outward normal vector to E is $\widehat{\nu} = \widehat{x}$, and the unit tangent vector to E is $\widehat{\tau} = \widehat{y}$. Then on crossing the edge, the values u_y and v_y stay constant, but u_x and v_x are functions of x, which increase by $2a$ and $2b$, respectively, with mean values m_u and m_v. Now assume all three are

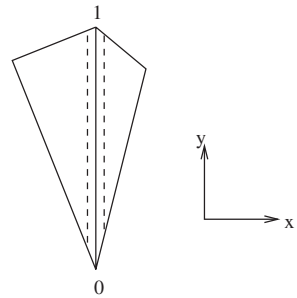


FIGURE 3.2. *Mollification of edge of two triangles.*

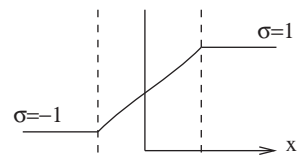


FIGURE 3.3. $\sigma(x)$.

mollified equally; that is, the "**X**" tangent vector to the two-dimensional manifold in four dimensions is

$$\mathbf{X}(x) = \mathbf{M} + \sigma(x)\mathbf{A} = \begin{pmatrix} 1 \\ 0 \\ u_x \\ v_x \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ m_u \\ m_v \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ a \\ b \end{pmatrix} \sigma(x), \tag{3.19}$$

where $\sigma(x)$ varies from -1 to 1 in the neighborhood of the edge as shown in Figure 3.3. Thus the Laplacian contribution is

$$\mathbf{L} \equiv \begin{pmatrix} 0 \\ 0 \\ u_{xx} + u_{yy} \\ v_{xx} + v_{yy} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ a \\ b \end{pmatrix} \sigma'(x). \tag{3.20}$$

The "**Y**" tangent vector to the two-dimensional manifold is the constant vector

$$\mathbf{Y} = \begin{pmatrix} 0 \\ 1 \\ u_y \\ v_y \end{pmatrix}. \tag{3.21}$$

Note that the integrals are considered only in a small neighborhood *nbd* of the edge—that is, the neighborhood of the $x - y$ projection of edge E. This is because u_{xx} and v_{xx} are identically zero away from the edge neighborhood where the rapid variation occurs. First use the same approach as in the section on constant coefficient diffusion terms—that of assuming the edge E is aligned with the y axis and then rotating the results accordingly. We note that α^0 is equivalent to a function of y alone in an infinitesimal neighborhood of the edge E. Now take the integral $\int\int P\mathbf{L}\alpha^0 \mathrm{d}S$ over the neighborhood *nbd* to get

(3.22)
$$\begin{aligned}\int\int_{nbd} P\begin{pmatrix}0\\0\\u_{xx}\\v_{xx}\end{pmatrix}\alpha^0(y)\mathrm{d}S &= \int\int_{nbd}\sqrt{D}P\begin{pmatrix}0\\0\\a\\b\end{pmatrix}\sigma'(x)\alpha^0(y)\mathrm{d}x\mathrm{d}y\\ &= \int_{y=0}^{y=\ell_{01}}\left(\int_{x_{nbd}}\sqrt{D}P\begin{pmatrix}0\\0\\a\\b\end{pmatrix}\sigma'(x)\mathrm{d}x\right)\alpha^0(y)\mathrm{d}y\\ &= \frac{\ell_{01}}{2}\int_{\sigma=-1}^{\sigma=1}\sqrt{D(\sigma)}P(\sigma)\mathbf{A}\mathrm{d}\sigma,\end{aligned}$$

where ℓ_{01} is the length of the edge joining node 0 and node 1, as in Figure 3.2. Further, once the matrix $P(\sigma)$ has been postmultiplied through by the vector $\mathbf{A}$ (defined in (3.19)) and by $\sqrt{D(\sigma)}$, then

$$\sqrt{D(\sigma)}P(\sigma)\mathbf{A} = \frac{1}{\sqrt{D(\sigma)}}[D(\sigma)\mathbf{A} - D_1(\sigma)\mathbf{X}(\sigma) - D_2(\sigma)\mathbf{Y}], \tag{3.23}$$

where the vector $\mathbf{X}(\sigma)$ and variables $D(\sigma), D_1(\sigma)$, and $D_2(\sigma)$ are functions of σ:

$$\begin{aligned}\mathbf{X}(\sigma) &= \mathbf{M} + \mathbf{A}\sigma,\\ D(\sigma) &= |\mathbf{X}(\sigma)|^2|\mathbf{Y}|^2 - (\mathbf{X}(\sigma)\cdot\mathbf{Y})^2,\\ D_1(\sigma) &= (\mathbf{X}(\sigma)\cdot\mathbf{A})|\mathbf{Y}|^2 - (\mathbf{Y}\cdot\mathbf{A})(\mathbf{Y}\cdot\mathbf{X}(\sigma)),\\ D_2(\sigma) &= (\mathbf{Y}\cdot\mathbf{A})|\mathbf{X}(\sigma)|^2 - (\mathbf{X}\cdot\mathbf{A})(\mathbf{X}(\sigma)\cdot\mathbf{Y}).\end{aligned} \tag{3.24}$$

Simplifying, the expression $D(\sigma)\mathbf{A} - D_1(\sigma)\mathbf{X}(\sigma) - D_2(\sigma)\mathbf{Y}$ is linear in σ as quadratic terms cancel and may be written as $\mathbf{G} + \mathbf{H}\sigma$, where $\mathbf{G}$ and $\mathbf{H}$ are constant vectors in σ. Also notice that $D(\sigma)$ is quadratic in σ and may be written as $c_1 + c_2\sigma + c_3\sigma^2$, where c_1, c_2, and c_3 are constants in σ. Using this information, the integral $\int_{-1}^{1}\sqrt{D(\sigma)}P(\sigma)\mathbf{A}\mathrm{d}\sigma$ may thus be written as

$$\mathbf{G}\int_{-1}^{1}\frac{1}{\sqrt{c_1 + c_2\sigma + c_3\sigma^2}}\mathrm{d}\sigma + \mathbf{H}\int_{-1}^{1}\frac{\sigma}{\sqrt{c_1 + c_2\sigma + c_3\sigma^2}}\mathrm{d}\sigma. \tag{3.25}$$

In the one-dimensional theory of SGWMFE both the integrals multiplied by $\mathbf{G}$ and $\mathbf{H}$, respectively, were expressed analytically in [19] and [17], since they were necessary for the calculation of the contributions from the Laplacian terms in one dimension; however, sixteen point Gauss quadrature is used for the integrals for the studies carried out for this paper. Once the terms of the vector have been calculated, it is necessary to rotate the x and y components to the correct orientation, since the calculations were made based on the assumption that the $x-y$ projection of the edge E was aligned with the y axis.

This rotation is done by replacing the first two components of the vector (corresponding to its x and y components) and multiplying by a rotation matrix:

$$\begin{pmatrix}\dfrac{\mathrm{d}y}{\ell_i} & \dfrac{\mathrm{d}x}{\ell_i}\\ \dfrac{-\mathrm{d}x}{\ell_i} & \dfrac{\mathrm{d}y}{\ell_i}\end{pmatrix}. \tag{3.26}$$

3.5. Source terms. The contributions over a cell from a source term, say, $S_1(u,v)$ in L_1 onto its 0th node, are obtained as follows:

$$\int_T P \begin{pmatrix} 0 \\ 0 \\ S_1(u,v) \\ 0 \end{pmatrix} \alpha^0 \mathrm{d}S = \sqrt{D} P \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \int_T S_1(u,v) \alpha^0 \mathrm{d}x\mathrm{d}y, \tag{3.27}$$

resulting in the equivalent compact form

$$\int_T S_1(u,v) \alpha^0 \mathrm{d}x\mathrm{d}y = A[\alpha^0 S_1(u,v)]_T. \tag{3.28}$$

3.6. Nonuniform diffusion terms in conservative form. Consider a nonconstant or nonlinear diffusion term of the form $L_1 = \nabla\cdot(a\nabla u)$, where $a = a(x,y,u,v)$. The contributions from this term in the cell onto its 0th node are obtained as follows:

$$\int_T P \begin{pmatrix} 0 \\ 0 \\ \nabla\cdot(a\nabla u) \\ 0 \end{pmatrix} \alpha^0 \mathrm{d}S = \int_T P \begin{pmatrix} 0 \\ 0 \\ a\nabla^2 u \\ 0 \end{pmatrix} \alpha^0 \mathrm{d}S + \int_T P \begin{pmatrix} 0 \\ 0 \\ \nabla a \cdot \nabla u \\ 0 \end{pmatrix} \alpha^0 \mathrm{d}S. \tag{3.29}$$

Consider the first integral on the right-hand side, noting that the term $\nabla^2 u$ is zero on each cell not including the edges. So the contributions from this first integral need only be considered in the δ-neighborhood of the edges in the support of α^0, as is the case for the constant coefficient Laplacian, as discussed in the section about constant coefficient diffusion terms. It will suffice to express the integral here for one edge E. Again we use the same approach as in the section on constant coefficient diffusion terms—that of assuming the edge E is aligned with the y axis and then rotating the results accordingly. As in [9] we note that, like α^0, $a(x,y,u,v)$ is equivalent to a function of y alone in an infinitesimal neighborhood of the edge E. Thus

$$\int_T P \begin{pmatrix} 0 \\ 0 \\ a\nabla^2 u \\ 0 \end{pmatrix} \alpha^0 \mathrm{d}S = \int_{y=0}^{y=\ell_{01}} \left(\int_{x_{nbd}} \sqrt{D} P \begin{pmatrix} 0 \\ 0 \\ \nabla^2 u \\ 0 \end{pmatrix} \mathrm{d}x \right) a(y) \alpha^0(y) \mathrm{d}y, \tag{3.30}$$

and the integral thus becomes (using the notation from (3.14))

$$\ell_{01}[a\alpha^0]_E \left(\int_{x_{nbd}} \sqrt{D} P \begin{pmatrix} 0 \\ 0 \\ \nabla^2 u \\ 0 \end{pmatrix} \mathrm{d}x \right). \tag{3.31}$$

The integral over the x_{nbd} is identical to the one derived for the Laplacian in (3.22), and thus the details will not be repeated here. The second integral in (3.29) can be treated as a source term:

$$\int P \begin{pmatrix} 0 \\ 0 \\ \nabla a \cdot \nabla u \\ 0 \end{pmatrix} \alpha^0 \mathrm{d}S. \tag{3.32}$$

Since P is constant over each cell and the term $\nabla a \cdot \nabla u$ is assumed bounded on each cell (u is constant on each cell and a is assumed continuous and bounded), then it is only necessary to consider the integral over each cell. For each cell T, use the theory derived for source terms with $S_1(u, v) = \nabla a \cdot \nabla u$, and similarly if there are equivalent terms in the second PDE $S_2(u, v) = \nabla a \cdot \nabla v$.

3.7. Regularization. As in the one-dimensional SGWMFE approach it may be necessary to add regularization terms in order to remove any degeneracies which may occur in the discrete system of ODEs. The regularization terms used here are similar to the terms introduced as pressure regularization in [8], as opposed to the internodal viscosity regularization terms more commonly known. As in the one-dimensional case, the need for regularization may arise when the solution has two consecutive cells with the same slope or when two nodes run into each other or overlap causing a degeneracy in the system of equations (see [17] for one-dimensional analysis). In order to remove the possibility of this happening, in [12] and [8] it was shown that adding regularization terms which are proportional to nodal velocities (described by Miller as internodal viscosities) is successful in two dimensions. These internodal viscosities are not needed for the porous medium equation; however, in some instances when the solution has an infinite slope it is necessary to use the pressure regularization terms as suggested by [13], added to keep nodes from overlapping.

The pressure terms used are derived from a quality functional. That is, the terms added are a constant times the negative gradient of a mesh quality functional Q, where the gradient is with respect to the discrete nodal velocities $[\dot{\mathrm{x}}_i, \dot{\mathrm{y}}_i, \dot{\mathrm{u}}_i, \dot{\mathrm{v}}_i]$. The quality functional used here is

$$Q = \frac{P^2}{\Delta S}, \tag{3.33}$$

on each cell T, where P is the perimeter of the triangular cell and ΔS is the area of the triangle in the four-dimensional space as introduced in (3.11). Thus for a four-dimensional set of ODEs resulting from the SGWMFE discretization, the pressure terms added to the equations at each node is

$$-C^2 \begin{pmatrix} \dfrac{\mathrm{d}Q}{\mathrm{d}x_i} \\ \dfrac{\mathrm{d}Q}{\mathrm{d}y_i} \\ \dfrac{\mathrm{d}Q}{\mathrm{d}u_i} \\ \dfrac{\mathrm{d}Q}{\mathrm{d}v_i} \end{pmatrix}. \tag{3.34}$$

The solutions are not very sensitive to the value of the pressure regularization parameter C^2, so long as C^2 is much smaller than the error tolerance, that is, so that the regularization terms added are well below the order of accuracy required. For the porous medium equation when the pressure terms are not needed, the addition of these terms makes no difference to the solution within the error tolerance. These terms are not needed for the case when $m = 1$ in the porous medium equation. However, as explained later, there are some cases when these terms are needed and the C^2 is set to $(tol/10)^2$, where tol is the local truncation error tolerance of the time integrator.

4. Results.

4.1. Porous medium equation. The parameter m in (1.1) determines the strength of the nonlinearity in the diffusion. In this paper we consider two cases, $m = 1$ and $m = 3$. For both cases the problem is started with an initial condition of $r_0 = 1/2$ and the initial profile is obtained from the similarity solution using this information. It is worthwhile noting that the choice of m is not limited; while not all possible integer values for $m > 0$ were tested, some integer values larger than 3 were tested, such as the problem with $m = 5$, which produced results in agreement with (1.2), as in the cases presented in this paper. The key point that makes the cases $m = 1$ and $m > 1$ different is that for the first case the slopes of the solutions are finite over the domain for this problem, and for cases $m > 1$ the solutions have infinite slopes at the moving boundary. In all cases, SGWMFE performs well.

The results illustrated in Figures 4.1 and 4.2 show the differences between the cases $m = 1$, which clearly diffuses faster, and $m = 3$, which has infinite slope near the moving boundary. The solutions were computed in the first quadrant using a two-dimensional SGWMFE code and the graphs show the solutions reflected onto the three remaining quadrants. This is because the problem is axially symmetric. The mesh plots are on the actual computational domain (see Figures 4.3 and 4.4), where it is seen that the SGWMFE code does indeed produce axially symmetric results. The speed of the spreading of the solutions can also be seen more clearly from the mesh plots. For both problems ($m = 1$ and $m = 3$) the initial conditions begin with an arced domain of radius 1/2. However, in the case $m = 1$, the solution boundary reaches a radius of almost 1.5, and in the case $m = 3$, the radius reaches a value of around 0.8 in the same amount of time. It is interesting that, regardless of the initial mesh chosen, for the porous medium equation, the meshes converge almost instantly, so that at any particular later time, the meshes will be the same; i.e., the mesh acts as an attractor. This can be seen in Figures 4.5 and 4.6.

It is worth mentioning that the case of $m = 3$ required use of pressure regularization terms. This is because the slopes are infinite and in the two-dimensional case (as in the one-dimensional case found in [17, 18]) the nodes tend to bunch up

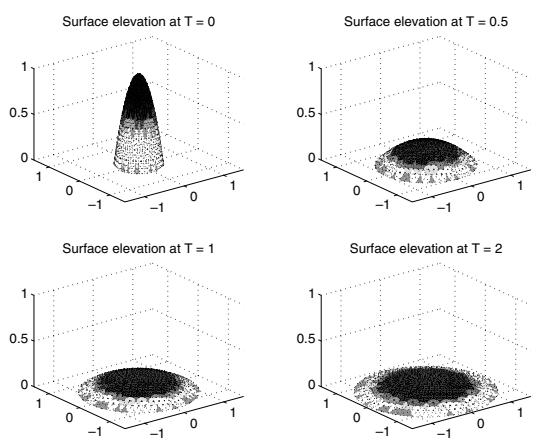


FIGURE 4.1. *SGWMFE solutions for the porous medium equation with* $m = 1$. *Solutions were obtained with* 136 *nodes initially distributed uniformly. Viscous* (A^2) *regularization terms and pressure* (C^2) *regularization terms were not needed for this problem.*

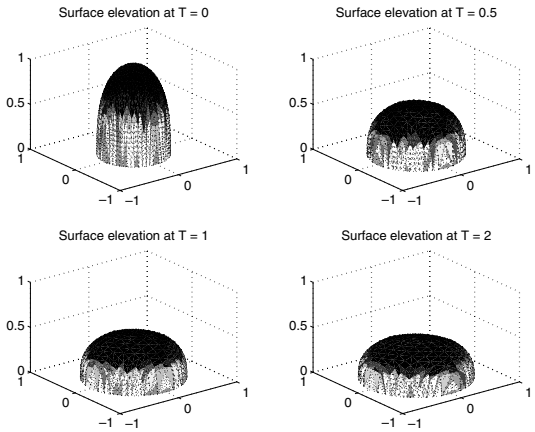


FIGURE 4.2. *SGWMFE solutions for the porous medium equation with* $m = 3$. *Solutions were obtained with* 136 *nodes initially distributed uniformly. Pressure regularization terms of* $C^2 = (10)^{-10}$ *were used for this problem. Viscous* (A^2) *regularization terms were not needed for this problem.*

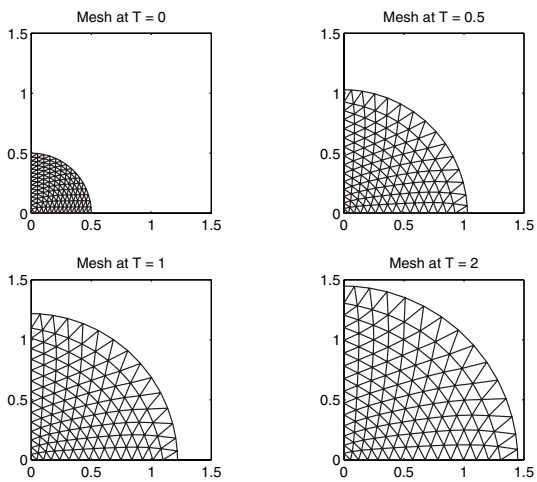


FIGURE 4.3. *SGWMFE meshes corresponding to solutions for the porous medium equation with* $m = 1$. *Solutions were obtained with* 136 *nodes initially distributed uniformly. Viscous* (A^2) *regularization terms and pressure* (C^2) *regularization terms were not needed for this problem.*

toward the infinite slope. In this case, it is so severe that the regularization pressure terms are needed in order to keep the nodes from overlapping and thus maintaining a well-defined mesh. The regularization pressure terms used for this problem are very small—much smaller than the time integrator tolerance used for this problem, which was chosen at 10^{-4}. Also this is the only problem presented in this paper where regularization terms were not needed, regardless of the value of m and regardless of the dimension in which the problem was solved. This is because the problem is diffusive and has its own natural diffusivity in the nonlinear terms, thus keeping the solution smooth. This is different from the shallow water equations because artificial viscosity was added for the purpose of maintaining a smooth solution in the numerical calculation.

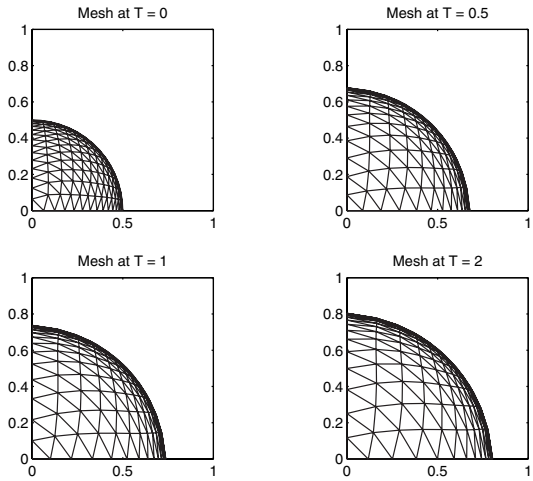


FIGURE 4.4. *SGWMFE meshes corresponding to solutions for the porous medium equation with* $m = 3$. *Solutions were obtained with* 136 *nodes initially distributed uniformly. Pressure regularization terms of* $C^2 = (10)^{-10}$ *were used for this problem. Viscous* (A^2) *regularization terms were not needed for this problem.*

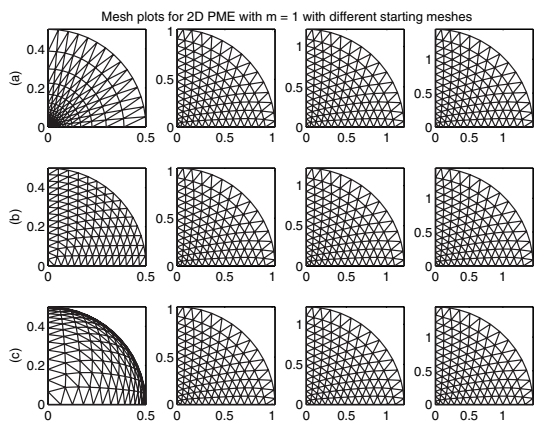


FIGURE 4.5. *SGWMFE meshes corresponding to solutions for the porous medium equation with* $m = 1$. *Solutions were obtained with* 136 *nodes at times* $T = 0, 0.5, 1, 2$ *shown from left to right. In* (a) *the initial mesh is one with nodes clustered toward the origin,* (b) *shows an initial mesh which is uniform, and* (c) *shows nodes initially clustered toward the moving boundary. Viscous* (A^2) *regularization terms and pressure* (C^2) *regularization terms were not needed for this problem.*

Convergence results for SGWMFE applied to the porous medium equation in two dimensions are shown. In the two-dimensional cases (regardless of the value of m) the slopes of the convergence lines for all the times shown are close to 1. Since the relation between N and the average mesh parameter h is $h = \sqrt{area(\Omega)/N}$, or $N \sim h^{-2}$ (where Ω is the area of the computational domain), this result corresponds to an h^2 rate of convergence, which is optimal for linear finite elements.

The L_∞ norm is chosen here for simplicity and to avoid introducing error from the numerical calculation of the L_1 norm over the nonuniform moving triangular grid. The L_∞ error norms are shown in Figures 4.7 and 4.8. As expected from the results shown in the one-dimensional cases in [17, 18], the convergence rates are the same

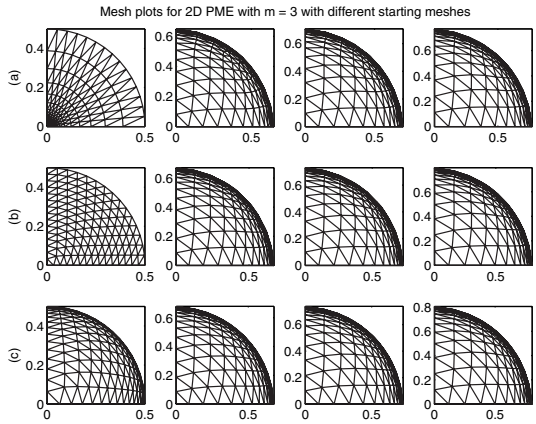


FIGURE 4.6. *SGWMFE meshes corresponding to solutions for the porous medium equation with* $m = 3$. *Solutions were obtained with* 136 *nodes at times* $T = 0, 0.5, 1, 2$ *shown from left to right. In* (a) *the initial mesh is one with nodes clustered toward the origin,* (b) *shows an initial mesh which is uniform, and* (c) *shows nodes initially clustered toward the moving boundary. Viscous* (A^2) *regularization terms were not needed for this problem. Pressure regularization terms of* $C^2 = (10)^{-10}$ *were used for this problem.*

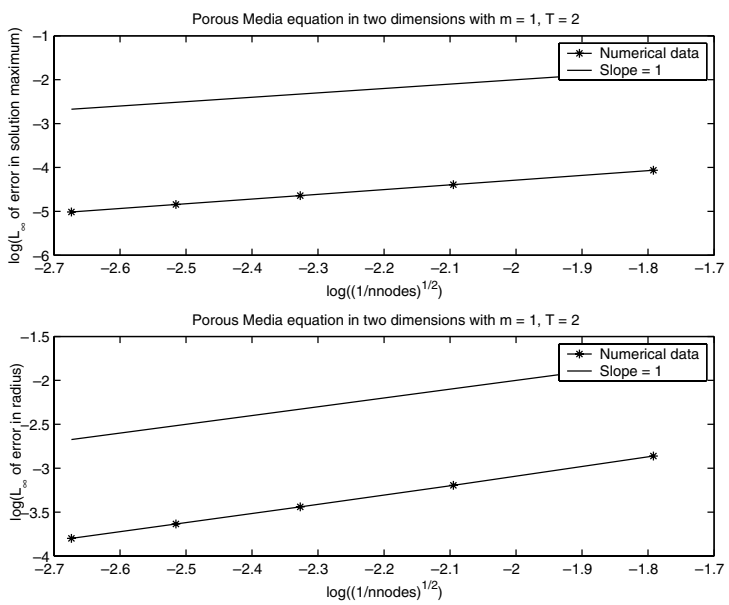


FIGURE 4.7. *Errors in SGWMFE solutions for the two-dimensional porous medium equation with parameter* $m = 1$. *The solutions were obtained using meshes with a varying number of nodes at time* $T = 2$. *Viscous* (A^2) *regularization terms and pressure* (C^2) *regularization terms were not needed for this problem. nnodes = number of nodes.*

regardless of the time to which the problem is solved. Thus only the convergence results for solutions calculated at time $T = 2$ are presented here.

The results in Figure 4.8 show more data results, from solving the problem using more nodes, than used to produce Figure 4.7 since originally it was not clear if the order of convergence of the radius was the same as the error over the entire solution

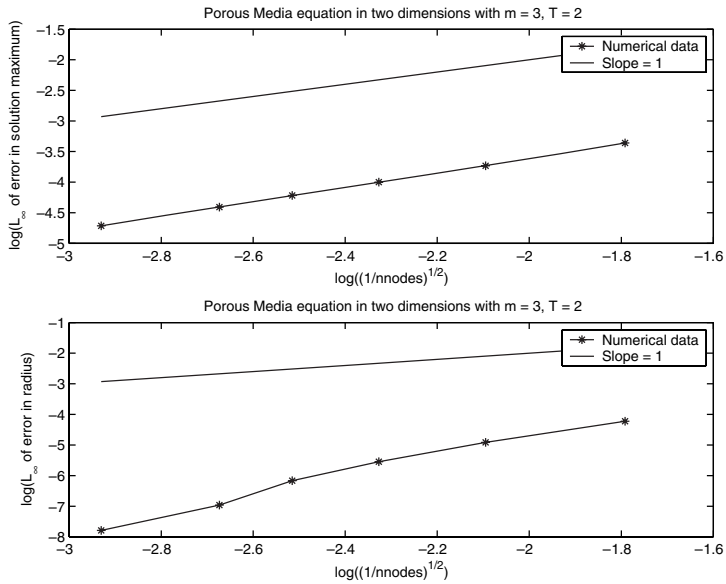


FIGURE 4.8. *Errors in SGWMFE solutions for the two-dimensional porous medium equation with parameter* $m = 3$. *The solutions were obtained using meshes with a varying number of nodes at time* $T = 2$. *Viscous* (A^2) *regularization terms were not needed for this problem and pressure regularization terms of* $C^2 = (10)^{-10}$ *were used for this problem.* $nnodes$ = *number of nodes.*

domain, or if it was better for the case when $m = 3$. The extra data was needed since from the graph of the convergence rate, Figure 4.8, shows a higher order convergence rate than the case when $m = 1$ as the number of nodes are increased. However, since the error tolerance used for this problem is 10^{-4}, using more nodes than used to produce the convergence results will not be more conclusive, since the results shown are already of the same order as the error tolerance. If all the data were to be reproduced with a lower truncation error tolerance, this would lead to more accurate solutions, though the results would not necessarily show convergence rates that are significantly different.

4.2. Gray Scott equations. First we examine the convergence of the method on the Gray Scott equations. Since an analytic solution is not available, we construct a reference solution (to be regarded as the "exact solution") by using an extremely fine discretization—much finer than that used for the actual computation. Solutions with variable nodes are compared pointwise to solutions on a fine reference mesh with 55 by 55 nodes. We then define the error measure in the component u as

$$E = \| u_{ref} - u \| / \| u_{ref} \|, \tag{4.1}$$

where u is the actual SGWMFE solution, u_{ref} is the reference solution, and $\| \cdot \|$ is the discrete L_2 norm. Note that the values used for calculating the norms in (4.1) are mapped onto a fixed uniform fine mesh (using the linear shape functions) in order to enable objective comparison of the various meshes. Figure 4.9 shows the convergence curves of $E(N)$ on a log-log scale, where N is the total number of nodal points, showing that the solutions with a variable number of nodes converge to the solution on the finest mesh. Similar results have been obtained when applying this procedure to the shallow water equations.

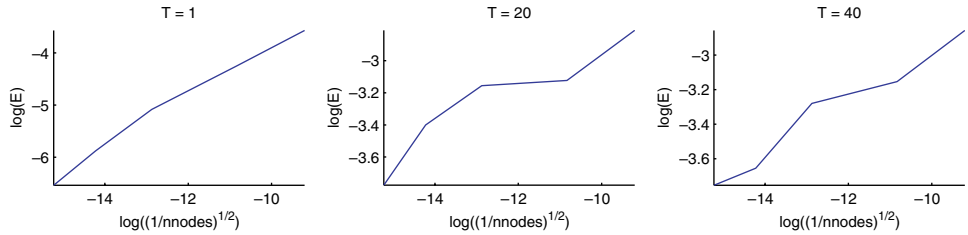


FIGURE 4.9. *SGWMFE convergence in u for the Gray Scott equations at times (from left to right): $T = 1$, 20, and 40. The error tolerance used is 10^{-4}. The corresponding viscous regularization terms of $A^2 = 5(10)^{-8}$ and pressure regularization terms of $C^2 = (10)^{-8}$ were used for this problem. nnodes = number of nodes.*

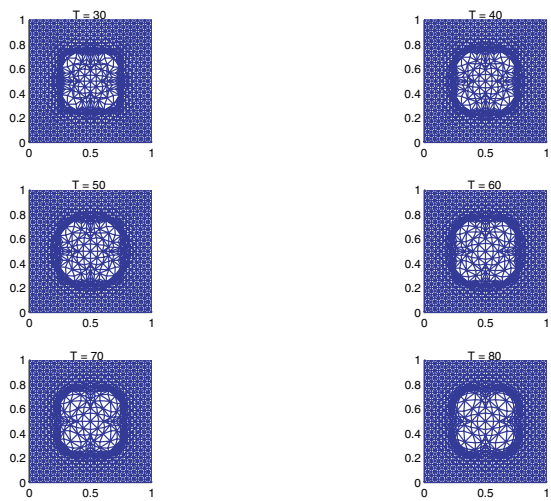


FIGURE 4.10. *SGWMFE meshes produced when solving the Gray Scott equations with 35 by 35 nodes. The error tolerance used is 10^{-4}. The corresponding viscous regularization terms of $A^2 = 5(10)^{-8}$ and pressure regularization terms of $C^2 = (10)^{-8}$ were used for this problem.*

Next we show mesh plots at several times for SGWMFE solutions of the Gray Scott equations using 35 by 35 nodes, much fewer nodes than for the solution plots following, for the sake of clarity of the figure. It is clear in Figure 4.10 that the mesh is attracted to the reaction fronts. See Figures 4.11 and 4.12 for contours of the solutions of the component u at corresponding times. The patterns observed in the solution figures are similar to those found in [20]. However, the meshes obtained for SGWMFE as compared to those shown in [20] are quite different, noting that for SGWMFE there is a much richer concentration of the nodes at the reaction front in comparison to the meshes produced by the method in [20].

4.3. Shallow water equations. Figure 4.13 shows SGWMFE results for the shallow water equations in two dimensions. The mesh plots are shown on the computational domain in order to make clearer comparisons between the mesh produced by SGWMFE and those produced by GWMFE. All of the numerical results for this problem conserve "mass" (the integral of the variable u over the domain) within the error tolerance. SGWMFE solutions are visually the same in comparison to the results obtained using GWMFE, though the meshes produced by each method differ.

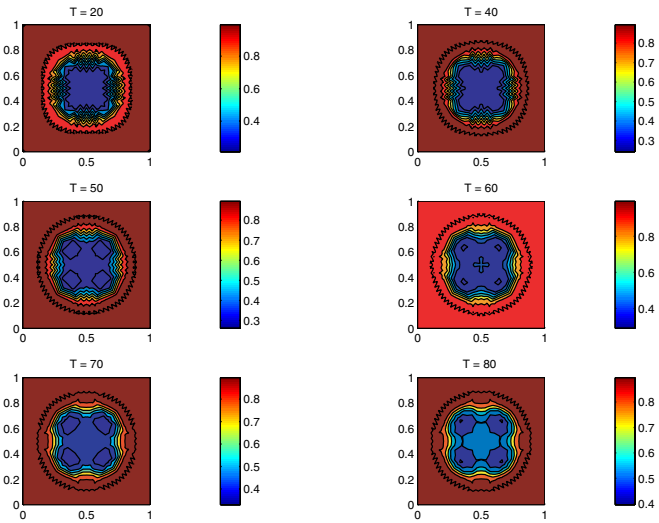


FIGURE 4.11. *SGWMFE solutions produced when solving the Gray Scott equations with* 45 *by* 45 *nodes. The error tolerance used is* 10^{-4}. *The corresponding viscous regularization terms of* $A^2 = 5(10)^{-8}$ *and pressure regularization terms of* $C^2 = (10)^{-8}$ *were used for this problem.*

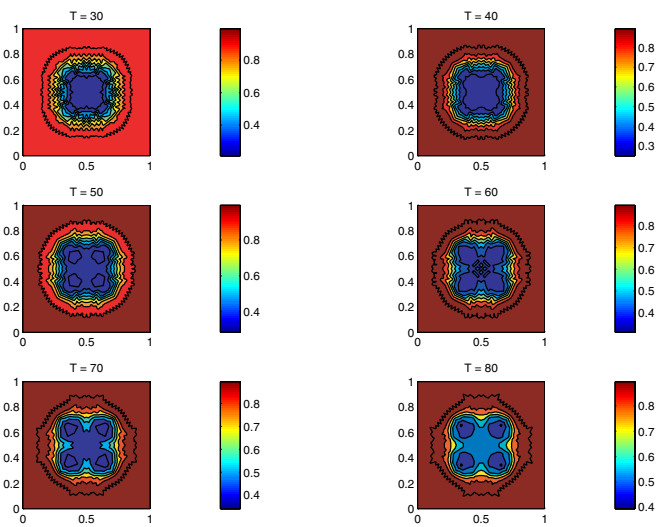


FIGURE 4.12. *SGWMFE solutions produced when solving the Gray Scott equations with* 55 *by* 55 *nodes. The error tolerance used is* 10^{-4}. *The corresponding viscous regularization terms of* $A^2 = 5(10)^{-8}$ *and pressure regularization terms of* $C^2 = (10)^{-8}$ *were used for this problem.*

The results shown here are calculated to the user-chosen tolerance, which was chosen to be 10^{-3} in this case. The solutions were in agreement with those obtained with the GWMFE code used in [9]. Further noting that the solutions are in agreement with GWMFE, it is worth discussing the uniqueness of the meshes produced. The meshes obtained for the shallow water equations are not "unique." In Figures 4.14 and 4.15, it is shown that with different initial meshes for solving the two-dimensional shallow water equations (with the same artificial viscosity), at various times, different

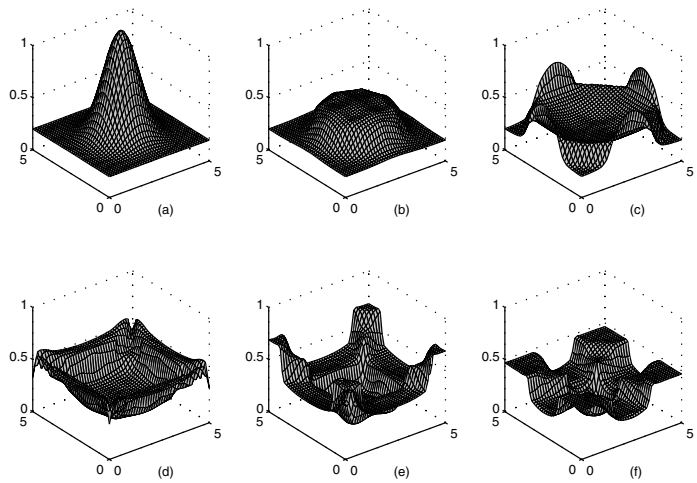


FIGURE 4.13. *SGWMFE solutions for the shallow water equations with artificial viscosity* $\epsilon = (10)^{-2}$ *at times* $T = 0, 1, 2, 3, 4, 5$. *Solutions were obtained with* 289 *nodes using the viscous regularization terms* $A^2 = 5(10)^{-5}$. *Regularization pressure terms were not needed for this problem.*

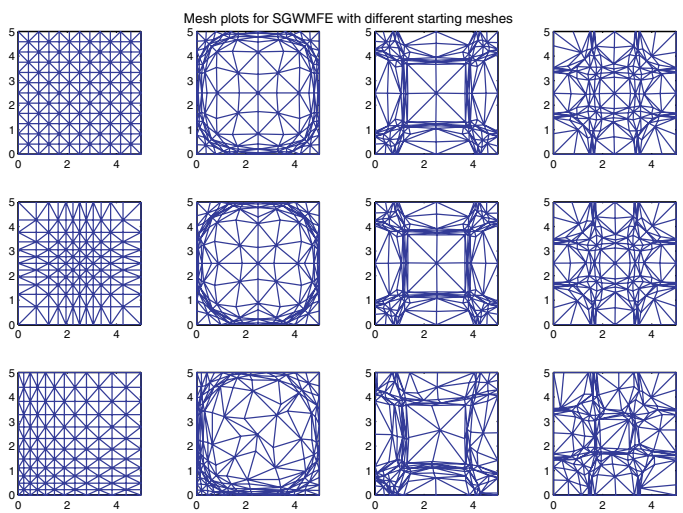


FIGURE 4.14. *SGWMFE meshes corresponding to solutions for the shallow water equations with artificial viscosity* $\epsilon = (10)^{-2}$ *at times* $T = 0, 2, 4, 5$. *Solutions were obtained with* 289 *nodes using the viscous regularization terms* $A^2 = 5(10)^{-5}$. *Pressure regularization terms were not needed for this problem.*

meshes are obtained (although the methods converge to solutions within the same accuracy, regardless of the initial mesh). It can also be seen in Figures 4.14 and 4.15, as expected, that the GWMFE and SGWMFE produce different meshes. This is true for both the one-dimensional and two-dimensional formulations of the shallow water equations. This could have been shown using results obtained for the one-dimensional shallow water equations; however, the results in two dimensions, presented here, are more demonstrative of the differences.

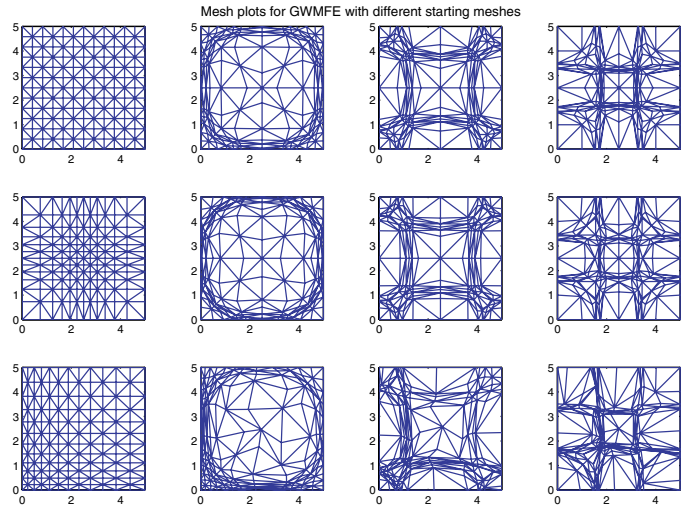


FIGURE 4.15. *GWMFE meshes corresponding to solutions for the shallow water equations with artificial viscosity* $\epsilon = (10)^{-2}$ *at times* $T = 0, 2, 4, 5$. *Solutions were obtained with* 289 *nodes using the viscous regularization terms* $A^2 = 5(10)^{-5}$. *Pressure regularization terms were not needed for this problem.*

5. Conclusions. The equations for SGWMFE have been presented using a projection matrix, making the underlying formulation easy to understand. The resulting equations resulting for SGWMFE are an elegant extension to existing gradient weighted methods. A convergence study of the method was carried out for the porous medium equation in two dimensions. The results show accurate solutions that correspond to an h^2 rate of convergence, which is optimal for linear finite elements. Further results were shown for the Gray Scott equations agreeing with results of a different moving mesh method and solutions of the nonlinear shallow water equations for both SGWMFE and GWMFE, obtaining equally accurate results on the different meshes produced by each corresponding method.

In [17] further calculations are given for the shallow water equations in both one and two dimensions, as well as for reaction diffusion equations in one dimension. In each case SGWMFE was straightforward to implement and gave reliable solutions for relatively little computational effort.

In conclusion, derivation and implementation of SGWMFE is straightforward and the formulation is easily generalized for arbitrary systems of PDEs in multiple dimensions. The method has proved to solve the problems presented here with steep moving fronts and a chemical reaction diffusion problem with pattern formation, both of which are difficult numerical problems. For such problems, SGWMFE can produce accurate results efficiently with respect to the number of nodes used as well as the calculation time.

6. Acknowledgments. This work has been motivated and to some extent, guided, by Keith Miller, whose pioneering work on gradient weighted methods contains the origins of SGWMFE, and we are immensely grateful to him for his help and advice. We are also very grateful to Andy Wathen and Mike Baines for discussions about moving finite element methods and to Mike Baines for suggesting a study of the porous medium equation.

REFERENCES

[1] M. Baines, *Moving Finite Elements*, Oxford University Press, New York, 1994.

[2] M. Baines, M. Hubbard, and P. Jimack, *A moving mesh finite element algorithm for the adaptive solution of time-dependent partial differential equations with moving boundaries*, Appl. Numer. Math., 54 (2005), pp. 450–469.

[3] R. E. Bank, W. C. Coughran, Jr., W. Fichtner, E. Grosse, D. Rose, and R. Smith, *Transient simulation of silicon devices and circuits*, IEEE Trans. CAD, 4 (1985), pp. 436–451.

[4] G. I. Barenblatt, *Scaling*, Cambridge Texts In Applied Mathematics, Cambridge University Press, Cambridge, UK, 2003.

[5] C. Budd, R. Carretero-Gonzlez, and R. Russell, *Precise computations of chemotactic collapse using moving mesh methods*, J. Comput. Phys., 202 (2005), pp. 463–487.

[6] C. J. Budd, G. Collins, W. Huang, and R. Russell, *Self-similar numerical solutions of the porous-medium equation using moving mesh methods*, Phil. Trans. Roy. Soc. London Ser. A, 357 (1999), pp. 1047–1077.

[7] C. J. Budd and M. Piggott, *The geometrix integration of scale-invariant ordinary and partial differential equations*, J. Comput. Appl. Math., 128 (2001), pp. 399–422.

[8] N. N. Carlson and K. Miller, *Design and application of a gradient-weighted moving finite element code* I: *In one dimension*, SIAM J. Sci. Comput., 19 (1998), pp. 728–765.

[9] N. N. Carlson and K. Miller, *Design and application of a gradient-weighted moving finite element code* II: *In two dimensions*, SIAM J. Sci. Comput., 19 (1998), pp. 766–798.

[10] J. L. Kyoung, W. McCormick, Q. Ouyang, and H. L. Swinney, *Pattern formation by interacting chemical fronts*, Science, 261 (1993), pp. 192–194.

[11] R. Li, W. Liu, and H. Ma, *Moving mesh method with error-estimator-based monitor and its applications to static obstacle problem*, J. Sci. Comput., 21 (2004), pp. 31–55.

[12] K. Miller, *A geometrical-mechanical interpretation of gradient-weighted moving finite elements*, SIAM J. Numer. Anal., 34 (1997), pp. 67–90.

[13] K. Miller, *private communication*, 2002.

[14] K. Miller and R. N. Miller, *Moving finite elements.* I, SIAM J. Numer. Anal., 18 (1981), pp. 1019–1032.

[15] Z. Tan, Z. Zhang, Y. Huang, and T. Tang, *Moving mesh methods with locally varying time steps*, J. Comput. Phys., 200 (2004), pp. 347–367.

[16] T. Tang, *Moving Mesh Methods for Computational Fluid Dynamics*, CSCAMM-05-04, Center for Scientific Computation and Mathematical Modeling, University of Maryland, College Park, MD, 2005.

[17] A. Wacher, *String Gradient Weighted Moving Finite Elements For Systems of Partial Differential Equations*, Ph.D. thesis, Oxford University, Oxford, UK, 2004.

[18] A. Wacher, *Mmpde vs. Sgwmfe, Experiments in One Dimension*, Tech. report, Oxford Computing Laboratory, Oxford, UK, 2005.

[19] A. Wacher, I. Sobey, and K. Miller, *String Gradient Weighted Moving Finite Elements for Systems of Partial Differential Equations*, Tech. report, Oxford Computing Laboratory, Oxford, UK, 2003.

[20] P. A. Zegeling and H. P. Kok, *Adaptive moving mesh computations for reaction–diffusion systems*, J. Comput. Appl. Math., 168 (2004), pp. 519–528.

SIAM J. SCI. COMPUT.
Vol. 29, No. 2, pp. 481–514
© 2007 Society for Industrial and Applied Mathematics

NEARLY OPTIMAL PRECONDITIONED METHODS FOR HERMITIAN EIGENPROBLEMS UNDER LIMITED MEMORY. PART I: SEEKING ONE EIGENVALUE*

ANDREAS STATHOPOULOS†

Abstract. Large, sparse, Hermitian eigenvalue problems are still some of the most computationally challenging tasks. Despite the need for a robust, nearly optimal preconditioned iterative method that can operate under severe memory limitations, no such method has surfaced as a clear winner. In this research we approach the eigenproblem from the nonlinear perspective, which helps us develop two nearly optimal methods. The first extends the recent Jacobi–Davidson conjugate gradient (JDCG) method to JDQMR, improving robustness and efficiency. The second method, generalized-Davidson+1 (GD+1), utilizes the locally optimal conjugate gradient recurrence as a restarting technique to achieve almost optimal convergence. We describe both methods within a unifying framework and provide theoretical justification for their near optimality. A choice between the most efficient of the two can be made at runtime. Our extensive experiments confirm the robustness, the near optimality, and the efficiency of our multimethod over other state-of-the-art methods.

Key words. Jacobi–Davidson, eigenvalues, Newton, conjugate gradient, optimality, preconditioning

AMS subject classifications. 65F15, 65H17

DOI. 10.1137/050631574

1. Introduction.

The numerical solution of large, sparse, Hermitian, and real symmetric eigenvalue problems is central to many applications in science and engineering. It is also one of the most time-consuming tasks. Recently, electronic structure calculations, with eigenproblems at their core, have displaced quantum chromodynamics as the top supercomputer cycle user. The symmetric eigenvalue problem seems deceptively simple to solve, given well-conditioned eigenvalues and a wealth of theoretical knowledge. However, these advantages have enabled researchers to push modeling accuracy to unprecedented levels, routinely solving for a few extreme eigenvalues of matrices of size more than a million, while an order of a billion has also been tried [48].

The sheer size of these problems can only be addressed by iterative methods. At the same time, the size imposes limits on the memory available to these methods. Moreover, preconditioning becomes imperative for reducing the total number of iterations. While many eigenvalue iterative methods have been proposed, there is little consensus on which method is the best and in what situations. The unrestarted Lanczos method is known to be optimal for solving Hermitian eigenvalue problems, but, unlike the conjugate gradient (CG) method for linear systems, it requires unlimited storage of its iteration vectors. With preconditioning, or under limited storage, the question of optimality remains open. Furthermore, there is a noticeable scarcity of high quality, general purpose software for preconditioned eigensolvers. In this research we seek an optimal, or nearly optimal, method that can utilize preconditioning and that can be implemented in robust software that is also flexible.

*Received by the editors May 13, 2005; accepted for publication (in revised form) September 26, 2006; published electronically March 15, 2007. This work was supported by National Science Foundation grants ITR/DMR 0325218 and ITR/AP-0112727.
http://www.siam.org/journals/sisc/29-2/63157.html
†Department of Computer Science, College of William and Mary, Williamsburg, VA 23187-8795 (andreas@cs.wm.edu).

In the particular case of seeking one eigenpair, if the eigenvalue were known, solving a linear system using CG to obtain the corresponding eigenvector would yield the optimal Lanczos convergence. In practice both the eigenvalue and the eigenvector are unknown, so the appropriate way to approach this problem is through the resulting slight nonlinearity, which then helps us to identify practical and near optimal methods. Currently, the best methods follow this approach, adopting either the Newton point of view or the nonlinear conjugate gradients (NLCG) point of view.

In the first part of this paper we argue that regardless of the approach (Newton or NLCG), the underlying iterative method should be the generalized Davidson (GD) method. In the second part, we examine the optimality from the Newton point of view, and identify a truncated Newton version followed by the Jacobi–Davidson conjugate gradient (JDCG) method [46] as within a small factor of optimal. In fact, we argue that this factor is less than three and usually much less than two. We then extend JDCG to JDQMR by adapting the quasi-minimal residual (QMR) instead of CG, which allows for better stopping criteria of the inner iteration, for indefinite preconditioners, interior eigenvalues, and improved robustness. In the third part of the paper, we study the NLCG point of view and, in particular, the related locally optimal CG recurrence. The approach can be used for short recurrences (as in LOBPCG [32]) or for restarting the subspace accelerated GD+1 [44, 68]. We show why local optimality of the Ritz value extends also to the Ritz vector, and we provide new intuition on the performance of GD+1 by relating it to the limited memory quasi-Newton methods (such as L-BFGS) [45]. In the fourth part, we present one of the most extensive sets of experiments in the literature, comparing our methods with state-of-the-art methods.

One conclusion to be drawn from our experiments is that both GD+1 and JDQMR always solve a given problem, regardless of the difficulty, and one of the two is always the fastest method. Over all the methods tested, GD+1 typically yields the smallest number of iterations, often matching the optimal convergence. The cheaper inner iteration of JDQMR benefits cases where the preconditioner and the matrix-vector operations are inexpensive. As this expense can be measured at runtime, an implementation based on the common GD driver can switch dynamically between GD+1 and JDQMR to make the most effective use of these two nearly optimal methods.

We note that this is not a review paper, although a substantial amount of review material is used to identify, build, and provide new intuition on the various components of the desired method. To keep the paper tractable both in size and readability, we focus on finding only one eigenpair. The additional problems of deflation, locking, correction equation projectors, blocking, and subspace acceleration which are relevant when a large number of eigenvalues are required will be addressed in a separate paper.

2. The GD as an outer iteration model. There is a plethora of iterative methods for large, sparse, symmetric eigenvalue problems. Invariably, these are Krylov or Krylov-like subspace methods that approximate the required eigenvectors from a subspace that usually grows in size as it is iteratively updated. The methods host a variety of differentiating features, of which two are the defining features for our research: We need methods that can utilize arbitrary preconditioners and that can operate on a limited memory space.

Shift-and-invert is probably the most well-studied and powerful preconditioning approach [52], which when coupled with a Lanczos-type method can produce robust, industrial strength methods [27]. In this research we assume that an exact factorization for shift-and-invert is not possible because of the size of the problem. In such

cases, approximations to $(A-\eta I)^{-1}$ are often used as preconditioners to linear solvers for the inverse iteration (INVIT), Rayleigh quotient iteration (RQI), or the (Jacobi–) Davidson methods [52, 71, 43, 60]. Although the accuracy of the term "eigenvalue preconditioning" for this approach has received some scrutiny [15, 31], the resulting methods improve convergence similarly to the corresponding preconditioned linear system solvers. Moreover, there are many established and tunable ways to produce such preconditioners [56]. Some applications may use different types of preconditioners, including polynomial transformations, frequency filtering or smoothing, or other techniques that do not correspond to approximately solving a linear system [55, 67].

The GD method was one of the first methods to cater to such an arbitrary selection of preconditioners [43]. Extending the original Davidson method [11], at every iteration GD applies a user-defined function on the residual of the sought eigenpair to "precondition" it—or to enrich it in the required directions. The vector is then used to enlarge the search space from which a new approximation is obtained through the Rayleigh–Ritz procedure [52]. When the search space exhausts the available memory, the method restarts with the best approximations of the required eigenvectors and possibly with additional information to improve convergence. Algorithm 2.1 depicts a version of the basic GD algorithm for finding one, smallest eigenpair $(\lambda_1, \mathbf{x}_1)$ of a real symmetric matrix, A.

Despite many efforts over two decades, the theory behind the convergence of the GD algorithm is still not well understood [43, 10, 69, 60]. For some recent, promising results see [49, 50, 34, 47]. One of the early unsettling issues with the GD algorithm had been the stagnation of the method if an accurate enough approximation to $(A-\theta^{(m)}I)^{-1}$, where $\theta^{(m)}$ is the Ritz value, is used as a preconditioner. An elegant resolution to this problem has been given by Sleijpen and van der Vorst with the Jacobi–Davidson (JD) method [60]. Instead of inverting $(A-\eta I)$, they showed that the appropriate form for this preconditioning should be an approximate solution to the correction equation:

$$(I-\mathbf{u}^{(m)}\mathbf{u}^{(m)T})(A-\eta I)(I-\mathbf{u}^{(m)}\mathbf{u}^{(m)T})\mathbf{t}^{(m)} = -\mathbf{r}^{(m)} = \theta^{(m)}\mathbf{u}^{(m)} - A\mathbf{u}^{(m)}, \tag{2.1}$$

where η is a shift close to the wanted eigenvalue. Thus, by working orthogonally to the Ritz vector $\mathbf{u}^{(m)}$, JD avoids stagnation. Among many known properties for this correction equation, one that is relevant to this paper is that if (2.1) is solved accurately, the new vector $\mathbf{u}^{(m)}+\mathbf{t}^{(m)}$ is collinear to the iterate produced by INVIT [41, 60, 58, 15, 74]. In addition, if $\eta=\theta^{(m)}$, the method is equivalent to RQI with subspace acceleration of all previous iterates. The true flexibility of GD and JD, however, is that they converge even when (2.1) is solved approximately through an iterative method such as CG. A preconditioner K can also be used as long as it is inverted orthogonally to the space of $Q=\mathbf{u}^{(m)}$. To this effect, the pseudoinverse of such a preconditioner can be written as

$$\left((I-QQ^T)K(I-QQ^T)\right)^+ = (I-K^{-1}Q(Q^TK^{-1}Q)^{-1}Q^T)K^{-1}(I-QQ^T) \tag{2.2}$$

$$= K^{-1}(I-Q(Q^TK^{-1}Q)^{-1}Q^TK^{-1})(I-QQ^T). \tag{2.3}$$

Moreover, CG can be implemented efficiently with only one projection with Q per iteration. See [17, 60, 3] for details.

Algorithm 2.1 is identical to the JD algorithm as provided in [3], except for step (19), which is handled specially in JD. In this sense, we can view JD as a subcase of GD. Practitioners, however, usually refer to GD as a [illegible] application of the available

ALGORITHM 2.1.
(1) [illegible] $\mathbf{v}_0$ [illegible]
(2) $\mathbf{t}^{(0)} = \mathbf{v}_0,\ m = 0,\ nmv = 0$
(3) **while** $nmv < max_num_matvecs$
(5) [illegible] $\mathbf{t}^{(m)}$ [illegible] $\mathbf{v}_i, i = 1, \ldots, m$
(6) $m = m + 1$; $nmv = nmv + 1$; $\mathbf{v}_m = \mathbf{t}^{(m-1)}$; $\mathbf{w}_m = A\mathbf{v}_m$
(7) $H_{i,m} = \mathbf{v}_i^T \mathbf{w}_m$ [illegible] $i = 1, \ldots, m$
(8) [illegible] $H = S\Theta S^T$ [illegible] $\theta_1 \leq \theta_2 \leq \cdots \leq \theta_m$
(9) $\mathbf{u}^{(m)} = Vs_1$; $\theta^{(m)} = \theta_1$; $\mathbf{w}^{(m)} = Ws_1$
(10) $\mathbf{r}^{(m)} = \mathbf{w}^{(m)} - \theta^{(m)}\mathbf{u}^{(m)}$
(11) **if** $\|\mathbf{r}^{(m)}\| \leq tol$ **return** $\theta^{(m)}, \mathbf{u}^{(m)}$
(12) **if** $m \geq m_{max}$ [illegible]
(13) $H = 0$
(14) **for** $i = 2, \ldots, m_{min}$
(15) $\mathbf{v}_i = Vs_i$; $\mathbf{w}_i = Ws_i$; $H_{ii} = \theta_i$
(16) **end for**
(17) $\mathbf{v}_1 = \mathbf{u}^{(m)}$; $\mathbf{w}_1 = \mathbf{w}^{(m)}$; $H_{11} = \theta^{(m)}$; $m = m_{min}$
(18) **end if**
(19) [illegible] $\mathbf{t}^{(m)} = Prec(\mathbf{r}^{(m)})$
(20) **end while**

preconditioner K^{-1} to the residual. In that case, preconditioning orthogonally to $\mathbf{u}^{(m)}$, using (2.2)–(2.3), is usually called Olsen's method [48] (or JD with no inner iterations), and it involves two operations with K^{-1} which can be very expensive. In practice, K does not often approximate $(A - \theta^{(m)}I)$, but A, and even when it does it is rarely accurate enough to require or even benefit from the JD projections, and thus GD is widely used in many applications [74, 2]. Exceptions exist for "non-standard" cases and when there is an inner iterative method as discussed in [61] and more extensively in [47]. In the remainder of this paper, we will use GD to refer to the method that preconditions the residual without Olsen's method.

Generality is an additional reason for considering GD as our basic iterative framework. Algorithm 2.1 is strikingly similar to an Arnoldi method with a flexible preconditioner, making GD a chameleon method. Without preconditioning, GD is equivalent to Arnoldi, albeit with an expensive implementation, that facilitates not only the testing of software correctness, but also a common platform for comparison between methods, abstracting from the implementation details. For example, the restarted, unpreconditioned GD method is mathematically equivalent to implicitly restarted Lanczos (IRL) and thick restarted Lanczos [63, 38, 42, 70, 73]. A block GD implementation is also possible [39, 65, 23], yielding the equivalents of block Lanczos [52] and subspace iteration [9] (without preconditioning), or preconditioned subspace iteration [3]. In addition, combining a block GD with our earlier GD+k restarting scheme [68], which we study closer in section 4.3, allows an equivalent and stable implementation of the LOBPCG method [32]. Needless to say, with an appropriate solution to the correction equation, the algorithm is identical to many variants of RQI, INVIT, JD, and the more recent JDCG of Notay that we study closely in the following section [46]. Beyond emulating the above methods, GD can enhance them with block, subspace acceleration, while working under given memory constraints.

Often the choice of algorithm and its parameters depends on the particular prob-

lem solved. The above discussion supports the argument that if there is to be one general purpose method that allows for arbitrary preconditioning and converges near optimally for a given problem, that algorithm must follow the GD template. The rest of the paper presents arguments why the tuning of the GD parameters, a task dreaded by practitioners, can be almost fully automated in the symmetric case.

3. The Newton view. Central to our discussion is the slight nonlinearity of the eigenvalue problem, i.e., both eigenvectors and eigenvalues are unknown. If the required eigenvalue λ_1 were known, the problem would degenerate to a symmetric linear system of equations $(A - \lambda_1 I)x_1 = 0$. This singular system can be solved optimally with CG or some other three-term recurrence method, provided that our initial guess x^0 is not defective in the direction of x_1, i.e., $x^0 = x_1 + y$ with $x_1 \perp y \in \text{Range}(A - \lambda_1 I)$. A Krylov solver approximates the correction y from the subspace $\mathcal{K}((A - \lambda_1 I), (A - \lambda_1 I)y)$. Because $\mathcal{K}((A - \lambda_1 I), (A - \lambda_1 I)y) \subseteq \text{Range}(A - \lambda_1 I)$, the solver will converge to x_1 with rate determined by the condition number of the deflated system: λ_{max}/λ_2. The optimality of CG implies that this is the ultimate convergence rate that any Krylov eigensolver can reach for finding the smallest eigenvalue [31, 33]. Note that unrestarted Lanczos achieves this rate, because the space that includes y yields also the eigenvalue λ_1. Equivalently, but more stably, we could solve (2.1) for $t = y$ with $\eta = \lambda_1$, obtaining the same optimal rate and converging in one outer JD step. Because we use symmetric QMR, we call this benchmark QMRopt. In general, before reaching that rate, an eigensolver would also have to find λ_1.

This nonlinearity can be resolved by applying the Newton method on the Grassmann manifold (to enforce normalization of the eigenvectors), which is equivalent both to RQI and to the JD method with no subspace acceleration and with accurate solution of the correction equation [59, 74, 15, 57]. Convergence of the outer iterations is known to be ultimately cubic when the initial guess is sufficiently close to x_1 [52]. In the absence of a good initial guess, truncated Newton methods or a subspace method such as Lanczos or GD must be used to ensure global convergence.

It has long been noticed that using iterative methods to solve the linear equation in RQI and INVIT beyond some level of accuracy does not decrease the number of outer iterations, while increasing the overall number of matrix-vector multiplications which typically correlates to the computational cost of the method. Interestingly, solving the linear equation approximately through a constant number of inner iterations has been shown to be equivalent to truncated (inexact) Newton methods [58]. However, the equivalence does not hold if the linear system in RQI is solved inexactly to a certain accuracy. Analysis of the stopping criteria for these inner-outer methods has been the focus of considerable research [53, 36, 21, 37, 62, 26, 25]. Because the solution vector for RQI and INVIT grows in norm the closer it is to the actual eigenvector, all proposed techniques test the convergence of the linear system relative to the norm of the solution vector. This norm can be monitored during the inner iteration.

The JD method is a better representative of inexact Newton minimization methods, because these apply the pseudoinverse of the Hessian to the gradient of the function. This is equivalent to solving (2.1) through a certain number of inner iterations or to a specified accuracy. Hence various stopping criteria for truncated Newton methods have been used in JD [60, 13]. Still, the objective is the convergence of the eigenvector, not of the linear system. Recently, an analysis for the interplay of inner-outer iterations was given by Notay [46]. Notay provided theoretical and experimental arguments on why his method, JDCG, provides nearly optimal convergence.

3.1. The JDCG approach. To simplify the notation we denote the projected operators as

$$A_{\eta,u^{(m)}} = (I - \mathbf{u}^{(m)}\mathbf{u}^{(m)T})(A - \eta I)(I - \mathbf{u}^{(m)}\mathbf{u}^{(m)T}), \tag{3.1}$$

$$K_{u^{(m)}} = (I - \mathbf{u}^{(m)}\mathbf{u}^{(m)T})K(I - \mathbf{u}^{(m)}\mathbf{u}^{(m)T}). \tag{3.2}$$

Assume that a Krylov iterative method (e.g., CG) is used to solve (2.1) and that no Rayleigh–Ritz is performed in the outer JD step, so the method is similar to the unaccelerated Newton method [58]. The latter assumption facilitates an inexpensive way to monitor the eigenvalue convergence inside the linear solver, while being a worst-case scenario for the subspace accelerated JD. At the kth inner iteration, the linear system residual is

$$\mathbf{g}_k = \mathbf{r}^{(m)} + A_{\eta,u^{(m)}}\mathbf{t}_k, \tag{3.3}$$

where $\mathbf{t}_k$ is the current approximate solution to the correction equation. If the linear solver were stopped at the kth step, the corrected eigenvector, its Rayleigh quotient, and its residual would be

$$\mathbf{u}_k^{(m+1)} = (\mathbf{u}^{(m)} + \mathbf{t}_k)/\|\mathbf{u}^{(m)} + \mathbf{t}_k\|, \tag{3.4}$$

$$\theta_k^{(m+1)} = \mathbf{u}_k^{(m+1)T} A \mathbf{u}_k^{(m+1)}, \tag{3.5}$$

$$\mathbf{r}_k^{(m+1)} = A\mathbf{u}_k^{(m+1)} - \theta_k^{(m+1)}\mathbf{u}_k^{(m+1)}. \tag{3.6}$$

Note that although $\mathbf{g}_k \to 0$, as $k \to \infty$, the eigenresidual $\mathbf{r}_k^{(m+1)}$ converges not to zero but to the eigenvalue residual of the next INVIT iterate: $\mathbf{r}_\infty^{(m+1)}$. Notay showed that $\|\mathbf{g}_k\|$ and $\|\mathbf{r}_k^{(m+1)}\|$ converge at similar rates up to the point where $\|\mathbf{r}_k^{(m+1)}\|$ approaches $\|\mathbf{r}_\infty^{(m+1)}\|$. Beyond this point, there is no benefit in continuing the inner linear solver, and this is the main idea behind JDCG. Notay also developed an inexpensive way to express the norm of the eigenresidual $\|\mathbf{r}_k^{(m+1)}\|$ as a function of the linear system residual $\|\mathbf{g}_k\|$ when CG is the inner solver. When the linear system residual starts to converge at a faster rate than the corresponding eigenvalue residual, the inner iteration is stopped.

Notay showed that for extremal eigenvalues JDCG demonstrates the same convergence as a periodically restarted CG method. Because the number of restarts is small, corresponding to the number of Newton steps rather than to the size of the search space, JDCG achieves near optimality, converging within a small factor of optimal. In section 5.2 we provide intuition on how to quantify this small factor. Notay also showed that if the Ritz value $\theta^{(m)}$ is closer to λ_1 than to λ_2, then (2.1) with $\eta = \theta^{(m)}$ is positive definite and CG can be used safely. See [40] for a similar result.

However, positive definiteness of the correction equation can be a limiting factor for a general purpose eigenvalue solver based on JDCG. There are several reasons. First, when good initial guesses are not available, $\theta^{(m)}$ can be well inside the spectrum causing (2.1) to be indefinite. Second, eigensolvers of the GD/JD type can be easily modified to find interior eigenvalues closer to a given shift η. Regardless of the quality of the initial guess, JDCG cannot be used in this case, and we have to rely on methods such as MINRES or SYMMLQ [51]. Third, and perhaps most importantly, the preconditioner K cannot be guaranteed to be positive definite. Practitioners solving linear systems would rarely choose an indefinite preconditioner, but for eigenvalue problems good approximations to the shifted matrix $(A - \theta^{(m)}I)$ arise naturally in

many applications. Examples include strongly diagonal dominant matrices, for which the Davidson method was originally developed [11], and discretizations of PDEs with spectral or Fourier bases (see experiments in section 5.5). Unfortunately, neither CG nor MINRES can utilize an indefinite preconditioner.

In the following section, we extend the JDCG ideas to the symmetric QMR of Freund and Nachtigal [18, 19] which allows for both indefinite operator and preconditioner. In addition, the monotonic convergence of the linear system quasi-residual allows for more intuitive and efficient stopping criteria than those used in JDCG.

3.2. The JDQMR method. In case of indefinite correction equation or preconditioner, GMRES or BiCGSTAB have been suggested as robust inner solvers [3]. GMRES, however, requires either storage of its whole search space or frequent restarts that degrade its convergence. BiCGSTAB, on the other hand, does have a short-term recurrence, capable of generating a large Krylov space implicitly, but it is oblivious to the symmetry of the matrix and thus twice as expensive per step as CG. Although convergence may be also twice as fast as that of BiCG [72] in the general case, for symmetric matrices this benefit may be smaller as the squared polynomial of BiCGSTAB is clearly suboptimal to the same degree polynomial obtained through an [illegible] method based on the Lanczos recurrence.

3.2.1. Symmetric QMR. The QMR method for nonsymmetric matrices uses the BiCG [4] short-term recurrence but obtains the approximations by a quasi-minimization of the residual over the available terms. Although there is no global optimality property, and the actual residual norm is not minimized, convergence is much smoother than with BiCG. Freund and Nachtigal noticed in [18] that when the matrix is symmetric, the simplification of the Lanczos process gives rise to a symmetric version of BiCG that can be used with any symmetric, possibly indefinite preconditioner. Based on this symmetric BiCG, they proposed the symmetric QMR (sQMR) algorithm. sQMR improves the functionality of MINRES, while keeping the same computational requirements. sQMR and MINRES are equivalent without preconditioning.

Assume the correction equation with operator $A_{\eta,u^{(m)}}$, preconditioner $K_{u^{(m)}}$, and the right-hand side $-\mathbf{r}^{(m)}$. As is typical in the JD method, assume that the initial guess is zero. Algorithm 3.1 shows the sQMR algorithm for solving the above correction equation with right preconditioning.

Similarly to CG, the $\mathbf{d}_k$ are the $A_{\eta,u^{(m)}}$ conjugate vectors that BiCG produces, and $\mathbf{r}_k$ are the BiCG residuals of the linear system at the kth step. Hence (see [4])

$$d_i^T A_{\eta,u^{(m)}} d_k = 0 \quad \forall i = 0, \ldots, k-1, \tag{3.7}$$

$$d_i^T r_k = 0 \quad \forall i = 0, \ldots, k-1. \tag{3.8}$$

Note that the sQMR residual, $\mathbf{g}_k$, and its norm $g_k = \|\mathbf{g}_k\| = \|\mathbf{r}^{(m)} + A_{\eta,u^{(m)}}\mathbf{t}_k\|$ are [illegible] computed during the algorithm, but only the norm from the quasi-minimization $\hat{g}_k$. The actual norm g_k, although typically larger than $\hat{g}_k$, is bounded by

$$g_k \leq \sqrt{k+1}\ \hat{g}_k. \tag{3.9}$$

Without preconditioning, $g_k = \hat{g}_k$, and depending on the preconditioner the above bound can be sharp. The actual norm g_k could be computed at the expense of two additional vector updates and one inner product, and by storing an extra vector for $A_{\eta,u^{(m)}}\boldsymbol{\delta}_k$. However, as we show in section 3.4, this expense is not justified. In the following section, we consider only the actual residual norm g_k.

ALGORITHM 3.1.
$A_{\eta,u^{(m)}}, K_{u^{(m)}}, -\mathbf{r}^{(m)}, maxiter$
t_k
(1) $\mathbf{t}_0^{(m)} = 0, \boldsymbol{\delta}_0 = 0, \mathbf{r}_0 = -\mathbf{r}^{(m)}, \mathbf{d}_0 = K_{u^{(m)}}^{-1}\mathbf{r}_0$
(2) $\hat{g}_0 = \|\mathbf{r}_0\|, \Theta_0 = 0, \rho_0 = \mathbf{r}_0^T\mathbf{d}_0$
(3) **if** $(maxiter = 0)$ $\mathbf{t}_0 = \mathbf{d}_0$ **return**
(4) **for** $k = 1, \ldots, maxiter$
(5) $\mathbf{w} = A_{\eta,u^{(m)}}\mathbf{d}_{k-1}$
(6) $\sigma_{k-1} = \mathbf{d}_{k-1}^T\mathbf{w}$ **if** $(\sigma_{k-1} = 0)$ **return**
(7) $\alpha_{k-1} = \frac{\rho_{k-1}}{\sigma_{k-1}}$
(8) $\mathbf{r}_k = \mathbf{r}_{k-1} - \alpha_{k-1}\mathbf{w}$
(9) $\Theta_k = \frac{\|\mathbf{r}_k\|}{\hat{g}_{k-1}}, c_k = \frac{1}{\sqrt{1+\Theta_k^2}}, \hat{g}_k = \hat{g}_{k-1}\Theta_k c_k$
(10) $\boldsymbol{\delta}_k = (c_k^2\Theta_{k-1}^2)\boldsymbol{\delta}_{k-1} + (c_k^2\alpha_{k-1})\mathbf{d}_{k-1}$
(11) $\mathbf{t}_k = \mathbf{t}_{k-1} + \boldsymbol{\delta}_k$
(12) **if** $\hat{g}_k$ [illegible] $\rho_{k-1} = 0)$ **return**
(13) $\mathbf{w} = K_{u^{(m)}}^{-1}\mathbf{r}_k, \rho_k = \mathbf{r}_k^T\mathbf{w}, \beta_k = \frac{\rho_k}{\rho_{k-1}}$
(14) $\mathbf{d}_k = \mathbf{w} + \beta_k\mathbf{d}_{k-1}$
(15) **end for**

Finally, note that we have used right preconditioning so that the norm g_k corresponds to the original correction equation, not the preconditioned one (see [18]). This is desirable, as we want to relate the eigenvalue residual to the monotonically decreasing linear system residual.

3.2.2. Monitoring the eigenresidual in sQMR. Replacing CG with sQMR results in a method which we call JDQMR. As with JDCG, we must provide expressions for $\theta_k^{(m+1)}$ and $\|\mathbf{r}_k^{(m+1)}\|$ of (3.4)–(3.6) that can be calculated through short recurrences in sQMR and without additional matrix-vector, preconditioning, or vector operations. The difference between the expressions for JDCG and JDQMR is strictly due to the projection process. Similarly to [46], we consider a simplified JD process where the Ritz vector is updated through (3.4) and not through a Rayleigh–Ritz procedure in the whole search space. A subspace accelerated JDQMR would improve the Ritz value, $\theta_k^{(m+1)}$, but not necessarily the residual norm, $\|\mathbf{r}_k^{(m+1)}\|$. This simplification, therefore, constitutes a worst-case scenario for the complete JDQMR, and it also keeps the expressions inexpensive and tractable.

THEOREM 3.1. [illegible] 3.1 [illegible] (2.1) [illegible] (3.1)–(3.6) [illegible]

$$\mathrm{B}_k = \mathbf{t}_k^T(A - \eta I)\mathbf{u}^{(m)}, \tag{3.10}$$

$$\Gamma_k = \mathbf{t}_k^T(A - \eta I)\mathbf{t}_k = \mathbf{t}_k^T A_{\eta,u^{(m)}}\mathbf{t}_k, \tag{3.11}$$

$$\Delta_k = \boldsymbol{\delta}_k^T\mathbf{r}^{(m)} = -\boldsymbol{\delta}_k^T\mathbf{r}_0, \tag{3.12}$$

$$\Phi_k = \boldsymbol{\delta}_k^T(A - \eta I)\boldsymbol{\delta}_k = \boldsymbol{\delta}_k^T A_{\eta,u^{(m)}}\boldsymbol{\delta}_k, \tag{3.13}$$

$$\Psi_k = \mathbf{t}_{k-1}^T(A - \eta I)\boldsymbol{\delta}_k = \mathbf{t}_{k-1}^T A_{\eta,u^{(m)}}\boldsymbol{\delta}_k, \tag{3.14}$$

$$\gamma_k = c_k^2\Theta_{k-1}^2, \ \xi_k = c_k^2\alpha_{k-1}. \tag{3.15}$$

$$\theta_k^{(m+1)} = \eta + \frac{\left(\theta^{(m)} - \eta + 2\mathrm{B}_k + \Gamma_k\right)}{1 + \|\mathbf{t}_k\|^2}, \tag{3.16}$$

$$\|\mathbf{r}_k^{(m+1)}\|^2 = \frac{\|\mathbf{g}_k\|^2}{1 + \|\mathbf{t}_k\|^2} + \frac{(\theta^{(m)} - \eta + \mathrm{B}_k)^2}{1 + \|\mathbf{t}_k\|^2} - (\theta_k^{(m+1)} - \eta)^2, \tag{3.17}$$

B_k Γ_k

$$\Delta_k = \gamma_k \Delta_{k-1} - \xi_k \rho_{k-1}, \tag{3.18}$$

$$\mathrm{B}_k = \mathrm{B}_{k-1} + \Delta_k, \tag{3.19}$$

$$\Gamma_k = \Gamma_{k-1} + 2\Psi_k + \Phi_k, \tag{3.20}$$

$$\Phi_k = \gamma_k^2 \Phi_{k-1} + \xi_k^2 \sigma_{k-1}, \tag{3.21}$$

$$\Psi_k = \gamma_k \Psi_{k-1} + \gamma_k \Phi_{k-1}, \tag{3.22}$$

$\mathrm{B}_0 = \Delta_0 = \Gamma_0 = \Phi_0 = \Psi_0 = 0$

From (3.4)–(3.5), $\theta^{(m)} = \mathbf{u}^{(m)T} A \mathbf{u}^{(m)}$, $\|\mathbf{u}^{(m)}\| = 1$, the symmetry of A, and the fact that $\mathbf{t}_k^T \mathbf{u}^{(m)} = 0$, we obtain $\|\mathbf{u}^{(m)} + \mathbf{t}_k\|^2 = 1 + \|\mathbf{t}_k\|^2$ and (3.16):

$$\begin{aligned}\theta_k^{(m+1)} &= (\mathbf{u}^{(m)} + \mathbf{t}_k)^T A (\mathbf{u}^{(m)} + \mathbf{t}_k)/(1 + \|\mathbf{t}_k\|^2) \\ &= \eta + (\theta^{(m)} - \eta + 2\mathbf{t}_k^T (A - \eta I)\mathbf{u}^{(m)} + \mathbf{t}_k^T A_{\eta, u^{(m)}} \mathbf{t}_k)/(1 + \|\mathbf{t}_k\|^2).\end{aligned}$$

Because $\mathbf{r}_k^{(m+1)} \perp \mathbf{u}_k^{(m+1)}$ we derive (3.17) as in (23)–(25) from [46]:

$$\begin{aligned}\|\mathbf{r}_k^{(m+1)}\|^2 &= \|(A - \eta I)(\mathbf{u}^{(m)} + \mathbf{t}_k)\|^2/(1 + \|\mathbf{t}_k\|^2) - (\theta_k^{(m+1)} - \eta)^2 \\ &= \Big(\|(I - \mathbf{u}^{(m)}\mathbf{u}^{(m)T})(A - \eta I)(\mathbf{u}^{(m)} + \mathbf{t}_k)\|^2 \\ &\quad + \|\mathbf{u}^{(m)}\mathbf{u}^{(m)T}(A - \eta I)(\mathbf{u}^{(m)} + \mathbf{t}_k)\|^2\Big)/(1 + \|\mathbf{t}_k\|^2) - (\theta_k^{(m+1)} - \eta)^2 \\ &= (\|\mathbf{g}_k\|^2 + (\theta^{(m)} - \eta + \mathrm{B}_k)^2)/(1 + \|\mathbf{t}_k\|^2) - (\theta_k^{(m+1)} - \eta)^2.\end{aligned}$$

The recurrences are obtained through the vector update formulas in sQMR.

$$\begin{aligned}\mathrm{B}_k &= \mathbf{t}_k^T (A - \eta I)\mathbf{u}^{(m)} = \mathbf{t}_k^T (A - \theta^{(m)} I)\mathbf{u}^{(m)} = \mathbf{t}_k^T \mathbf{r}^{(m)} \\ &= (\mathbf{t}_{k-1} + \boldsymbol{\delta}_k)^T \mathbf{r}^{(m)} = \mathrm{B}_{k-1} + \Delta_k, \\ \Delta_k &= \boldsymbol{\delta}_k^T \mathbf{r}^{(m)} = (\gamma_k \boldsymbol{\delta}_{k-1} + \xi_k \mathbf{d}_{k-1})^T \mathbf{r}^{(m)} = \gamma_k \Delta_{k-1} + \xi_k (\mathbf{d}_{k-1}^T \mathbf{r}^{(m)}).\end{aligned}$$

Notice that $\mathbf{r}^{(m)} = -\mathbf{r}_0 = -\mathbf{r}_{k-1} - A_{\eta, u^{(m)}} \mathbf{t}_{k-1}^{bcg}$, where $\mathbf{r}_{k-1}, \mathbf{t}_{k-1}^{bcg}$ are the residual and the solution that the BiCG process produces at the $k-1$ inner step. We know that the BiCG solution $\mathbf{t}_{k-1}^{bcg}$ is a linear combination of the first $\mathbf{d}_{i, i=0:k-2}$ BiCG vectors, and from (3.7) we have $\mathbf{d}_{k-1}^T A_{\eta, u^{(m)}} \mathbf{t}_{k-1}^{bcg} = 0$. Thus, using the update formula of $\mathbf{d}_{k-1}$ in sQMR and (3.8), we obtain (3.18):

$$\begin{aligned}\Delta_k &= \gamma_k \Delta_{k-1} - \xi_k \mathbf{d}_{k-1}^T \mathbf{r}_{k-1} = \gamma_k \Delta_{k-1} - \xi_k (\mathbf{r}_{k-1}^T K_{u^{(m)}}^{-1} \mathbf{r}_{k-1} + \beta_{k-1} \mathbf{d}_{k-2}^T \mathbf{r}_{k-1}) \\ &= \gamma_k \Delta_{k-1} - \xi_k \mathbf{r}_{k-1}^T K_{u^{(m)}}^{-1} \mathbf{r}_{k-1} = \gamma_k \Delta_{k-1} - \xi_k \rho_{k-1}.\end{aligned}$$

The recurrence for Γ_k is decomposed as follows:

$$\Gamma_k = (\mathbf{t}_{k-1} + \boldsymbol{\delta}_k)^T A_{\eta, u^{(m)}} (\mathbf{t}_{k-1} + \boldsymbol{\delta}_k) = \Gamma_{k-1} + 2\Psi_k + \Phi_k.$$

We first consider Φ_k. Notice that from the sQMR algorithm that $\boldsymbol{\delta}_k$ is a linear combination of the $\mathbf{d}_{i,i=0:k-1}$. Therefore $\boldsymbol{\delta}_i$ are also conjugate to $\mathbf{d}_i$, i.e.,

$$\boldsymbol{\delta}_{k-1}^T A_{\eta,u^{(m)}} \mathbf{d}_{k-1} = 0. \tag{3.23}$$

Then, we can derive (3.21) as follows:

$$\begin{aligned}\Phi_k &= (\gamma_k \boldsymbol{\delta}_{k-1} + \xi_k \mathbf{d}_{k-1})^T A_{\eta,u^{(m)}} (\gamma_k \boldsymbol{\delta}_{k-1} + \xi_k \mathbf{d}_{k-1}) \\ &= \gamma_k^2 \Phi_{k-1} + \xi_k^2 \mathbf{d}_{k-1}^T A_{\eta,u^{(m)}} \mathbf{d}_{k-1} = \gamma_k^2 \Phi_{k-1} + \xi_k^2 \sigma_{k-1}.\end{aligned}$$

Now, we consider Ψ_k. We observe that the sQMR solution vector $\mathbf{t}_k$ is a linear combination of the $\boldsymbol{\delta}_{i,i=0:k}$ vectors. Using the conjugacy of $\boldsymbol{\delta}_i$ with $\mathbf{d}_i$ in (3.23), we have $\mathbf{t}_k^T A_{\eta,u^{(m)}} \mathbf{d}_k = 0$. Then (3.22) is obtained as

$$\begin{aligned}\Psi_k &= \mathbf{t}_{k-1}^T A_{\eta,u^{(m)}} (\gamma_k \boldsymbol{\delta}_{k-1} + \xi_k \mathbf{d}_{k-1}) = \gamma_k \mathbf{t}_{k-1}^T A_{\eta,u^{(m)}} \boldsymbol{\delta}_{k-1} \\ &= \gamma_k \mathbf{t}_{k-2}^T A_{\eta,u^{(m)}} \boldsymbol{\delta}_{k-1} + \gamma_k \boldsymbol{\delta}_{k-1}^T A_{\eta,u^{(m)}} \boldsymbol{\delta}_{k-1} = \gamma_k (\Psi_{k-1} + \Phi_{k-1}).\end{aligned}$$

Finally, it can be simply verified that for $k = 0$ the above scalar quantities have the value of zero. ∎

3.3. Stopping criteria. The above theorem provides the means to dynamically and inexpensively monitor the eigenvector convergence inside the linear solver. It also characterizes the convergence behavior of both the linear system and the eigenvalue residual. We first consider the unpreconditioned case, where the sQMR is equivalent to MINRES, and $g_k = \hat{g}_k$. To simplify the discussion we introduce the following notation where nonboldface letters are used to denote the norms of the vectors with corresponding boldface letters. Specifically,

$$r_k^{(m+1)} = \|\mathbf{r}_k^{(m+1)}\|, \tag{3.24}$$

$$r_\infty^{(m+1)} = \|\mathbf{r}_\infty\| = \|\lim_{k\to\infty} \mathbf{r}_k^{(m+1)}\|, \tag{3.25}$$

$$g_k = \|\mathbf{g}_k\|, \tag{3.26}$$

$$t_k = \|\mathbf{t}_k\|, \tag{3.27}$$

$$p_k = \frac{(\theta^{(m)} - \eta + \mathrm{B}_k)^2}{1 + \|\mathbf{t}_k\|^2} - (\theta_k^{(m+1)} - \eta)^2. \tag{3.28}$$

Equation (3.17) suggests that as the linear system is solved, $g_k \to 0$, $r_k^{(m+1)} \to r_\infty^{(m+1)}$, and $p_k \to (r_\infty^{(m+1)})^2$. Note that p_k could be negative initially, but converges to a positive limit with a rate similar to that of g_k. In addition, Notay showed for a similar equation for the CG that p_k is much smaller than $(r_k^{(m+1)})^2$. Therefore, $r_k^{(m+1)}$ stays close to g_k until the proximity of the limit. Notay proposed stopping the inner solver when convergence rates start to deviate: $g_k/g_{k-1} < (r_k^{(m+1)}/r_{k-1}^{(m+1)})^{\alpha'}$ for some user-defined α', say 0.9. The lack of monotonic convergence of g_k in CG complicates the above stopping criterion, and the practical JDCG algorithm introduces several different tests and user-defined parameters. The smooth convergence of the JDQMR residuals allows for a much more accurate use of convergence rates. Moreover, we show how to provide a set of almost parameter-free, adaptive stopping criteria.

First rewrite (3.17) as

$$r_k^{(m+1)2} = \frac{g_k^2}{1 + t_k^2} + p_k. \tag{3.29}$$

We want to quantify how accurate an approximation of the INVIT iterate we should obtain, or, equivalently, to identify an α, such that we stop when

$$r_k^{(m+1)} \leq \alpha \; r_\infty^{(m+1)}, \text{ with } \alpha > 1. \tag{3.30}$$

Based on our discussion above, the results in [46], and the fact that in the domain of Newton convergence (for the outer iteration), $r_\infty^{(m+1)}$ is much smaller than $r_0^{(m+1)}$, we can assume that $p_k \approx r_\infty^{(m+1)2}$ even for relatively small k, and in particular for k close to the required stopping point. Thus, we can turn condition (3.30) into a testable condition involving only g_k and $r_k^{(m+1)}$:

$$\begin{aligned} r_k^{(m+1)2} \leq \alpha^2 \; p_k &= \alpha^2 r_k^{(m+1)2} - \alpha^2 g_k^2/(1+t_k^2) \\ &\Leftrightarrow g_k \leq r_k^{(m+1)} \sqrt{\frac{\alpha^2-1}{\alpha^2}} \sqrt{1+t_k^2}. \end{aligned} \tag{3.31}$$

Because the norm of the correction is typically $t_k \ll 1$, especially close to convergence, the above condition is determined by the factor based on α. A first approach to obtaining this factor is to observe that when the two terms $g_k^2/(1+t_k^2)$ and p_k in (3.29) become equal, $r_k^{(m+1)}$ is approximately within a factor of $\sqrt{2}$ from its limit. Thus, we could stop if the linear system residual is smaller than $0.707\sqrt{1+t_k^2}$ of the eigenresidual. Our experiments, however, have shown this not to be an effective factor. Indeed, a factor close to $0.99\sqrt{1+t_k^2}$ has yielded consistently better results.

To identify an appropriate factor α, we should first understand its effects on the global convergence of the method. On exit from the inner iteration, the eigenvalue errors satisfy the Bauer–Fike bounds:

$$|\theta_k^{(m+1)} - \lambda_1| \leq r_k^{(m+1)}, \tag{3.32}$$

$$|\theta_\infty^{(m+1)} - \lambda_1| \leq r_\infty^{(m+1)}. \tag{3.33}$$

We avoid using the tighter bounds of [52] because λ_1 may not be sufficiently separated from the spectrum relative to $r_k^{(m+1)}$. After completing the following outer step, exact INVIT will reduce the eigenvalue error by a factor f, i.e., $f r_\infty^{(m+1)}$. Therefore, a small enough value of α—obviously smaller than $1/f$—is needed for the inexact INVIT to guarantee convergence. An analysis of the bounds that such a value has to satisfy and some heuristics are presented in [62] (see also [6]). Similarly, exact RQI will yield an eigenvalue error $f|\theta_\infty^{(m+1)} - \lambda_1|^2$ for some factor f. As Smit and Paardekooper derive in [62], quadratic convergence in inexact RQI is maintained even for relatively large values of α, especially during the later stages of the iteration, as long as sufficiently small ones are used in the first few iterations.

In the first few outer iterations, the JDQMR may not have a good estimate of the eigenvalue, suggesting the contrary, i.e., that the correction equation should not be solved accurately at all. This presents no problem, because JDQMR is more flexible than INVIT/RQI, in that even if no accuracy is required in the solution of the correction equation (zero inner iterations), the subspace accelerated algorithm still converges as fast as Arnoldi or GD (with preconditioning). The outer step, however, can be rather expensive compared with the recurrence-based inner solver. For this reason, it is beneficial to postpone exiting from the QMR as much as possible to achieve an outer convergence similar to that of the exact INVIT/RQI. This implies that $r_k^{(m+1)}$ and $r_\infty^{(m+1)}$ should be of the same order of magnitude, e.g., for INVIT

$\alpha < \min(1/f, 10)$. We have chosen $\alpha = 7$, which yields the parameter 0.99 of the previous paragraph. The choice of α that gives an optimal total number of inner iterations, i.e., number of matrix-vector operations, is beyond the scope of this paper and is the focus of current research. We note, however, that preliminary results with an RQI model based on (3.29) and (3.31) have confirmed that when a constant α is to be used its value should be 8, which is close to our experimental choice.

The criterion (3.31) can be complemented for the case where the convergence of the linear system solver is very slow or experiences plateaus. In that case, we want $r_k^{(m+1)}$ to be even closer to g_k relative to the convergence rate of g_k and not only by a constant factor. We demand, therefore, that the average convergence rate of the eigenresidual $(r_k^{(m+1)}/r_0)^{1/k} = (r_k^{(m+1)}/g_0)^{1/k}$ is no more (i.e., not slower) than the geometric average of the last two average rates of the linear system, i.e., stop when

$$\sqrt{\frac{g_k}{g_0}\frac{g_{k-1}}{g_0}} < \frac{r_k^{(m+1)}}{g_0} \Leftrightarrow$$

$$g_k < r_k^{(m+1)}\sqrt{g_k/g_{k-1}}. \tag{3.34}$$

Putting together inequalities (3.31) and (3.34) we obtain our first stopping criterion:

$$g_k \le r_k^{(m+1)} \max\left(0.99\sqrt{1+t_k^2}, \sqrt{g_k/g_{k-1}}\right). \tag{3.35}$$

As we discuss in section 5.2, despite the optimality of the above criteria, sQMR may experience convergence plateaus in more than one outer JD iteration. During these plateaus it attempts to resolve certain eigencomponents that prevent convergence for the specific correction equation. Although this plateau would be built even by our QMRopt benchmark, repeating it every outer iteration would be wasteful—wasteful in a different sense than what we have studied till now. The g_k and $r_k^{(m+1)}$ may still be close and the inner iteration may still offer improvements to the correction, but to do so it needs components that it keeps rebuilding for different correction equations at every outer step. Although not severe, this is sometimes observed in problems with slow convergence. In such cases, we have found it beneficial to stop when r_0 is reduced by a certain factor of, say,

$$r_k^{(m+1)} < 0.1\ r_0. \tag{3.36}$$

Note that this is very different from the traditional $g_k < 0.1\ g_0$ and that it should be used in conjunction with the criteria that guarantee closeness of g_k and $r_k^{(m+1)}$.

For extreme eigenvalues, the variational principle guarantees monotonic convergence of the Ritz value at the outer step, but not of the approximation $\theta_k^{(m+1)}$ during the execution of QMR. We have found it beneficial, however, to stop QMR when

$$\theta_k^{(m+1)} > \theta_{k-1}^{(m+1)}. \tag{3.37}$$

If $\theta_k^{(m+1)}$ increases, an eigenvalue was missed and the solver is trying to retarget convergence to the missed one. It is thus beneficial to exit the inner iteration and let the outer step improve on the new Ritz value through the Rayleigh–Ritz procedure, providing a better correction equation during the next inner solution.

Finally, we should stop the inner iteration when the required tolerance has been achieved (or the maximum number of iterations reached). Although $g_k < r_k^{(m+1)}$,

except possibly for a few initial steps, it is safer to check both g_k and $r_k^{(m+1)} < \epsilon_{inn}$. The threshold for the inner iterations, ϵ_{inn}, could be the same as the user-defined threshold, ϵ, for the eigenresiduals. The $r_k^{(m+1)}$, however, is the residual norm of the corrected Ritz vector, and not from the outer Rayleigh–Ritz method on the basis V that includes the correction. Thus the residual norm computed by the outer method could be slightly larger than ϵ, causing the inner method to be called again to provide a very small correction. This restarting could force QMR to build again certain vector components, wasting some iterations. Therefore, we suggest $\epsilon_{inn} = \epsilon/2$, while making sure that this is achievable within the machine's precision, $\epsilon_{\text{machine}}$. Estimating the norm of A by the largest Ritz value seen thus far, we stop QMR if

$$g_k < \epsilon_{inn} \text{ OR } r_k^{(m+1)} < \epsilon_{inn}, \tag{3.38}$$

$$\text{with } \epsilon_{inn} = \max(\epsilon/2, \epsilon_{\text{machine}}\|A\|). \tag{3.39}$$

3.4. The effect of quasi-minimization. With a preconditioner K, the BiCG residuals $\mathbf{r}_k$ in Algorithm 3.1 are K^{-1}-orthogonal, but unlike MINRES, which implicitly considers the orthogonal basis $K^{-1/2}\mathbf{r}_k$, sQMR uses the $\mathbf{r}_k$, and thus the actual norm g_k of the sQMR residual is bounded as shown in inequality (3.9). A natural question is whether the above stopping criteria would be adversely affected by using the readily computed $\hat{g}_k$ instead of g_k.

The only quantity in Theorem 3.1 that is affected by replacing g_k with $\hat{g}_k$ is the eigenresidual norm $r_k^{(m+1)}$. Neither the eigenvalue estimate $\theta_k^{(m+1)}$ nor the p_k term of (3.29) depends on the choice of g_k. Let $g_k = c_k\hat{g}_k$, with $1 \le c_k \le \sqrt{k+1}$. We do not consider the possibility $0 \le c_k < 1$, because in the rare case that this occurs, c_k would still be very close to 1. Denote by $\hat{r}_k^{(m+1)}$ the eigenresidual norm approximation computed through (3.29) using $\hat{g}_k$. We can express $r_k^{(m+1)}$ as a function of the approximate $\hat{r}_k^{(m+1)2}$ as follows:

$$\begin{aligned} r_k^{(m+1)2} &= \frac{g_k^2}{1+t_k^2} + p_k = c_k^2 \frac{\hat{g}_k^2}{1+t_k^2} + p_k = c_k^2(\hat{r}_k^{(m+1)2} - p_k) + p_k \\ &= c_k^2 \hat{r}_k^{(m+1)2} - (c_k^2 - 1)p_k. \end{aligned} \tag{3.40}$$

Ideally, criterion (3.35) should check the ratio of the exact residuals $g_k/r_k^{(m+1)}$. Denoting $R = \max(0.99\sqrt{1+t_k^2}, \sqrt{g_k/g_{k-1}})$ and using (3.40) we should check the following:

$$\frac{g_k^2}{r_k^{(m+1)2}} = \frac{c_k^2\hat{g}_k^2}{c_k^2\hat{r}_k^{(m+1)2} - (c_k^2-1)p_k} = \frac{\hat{g}_k^2}{\hat{r}_k^{(m+1)2}} \frac{1}{1 - \frac{(c_k^2-1)}{c_k^2}\frac{p_k}{\hat{r}_k^{(m+1)2}}} < R^2.$$

Equivalently, using our computed residual approximations we should check

$$\hat{g}_k < \hat{r}_k^{(m+1)}\ R\sqrt{1 - \frac{(c_k^2-1)}{c_k^2}\frac{p_k}{\hat{r}_k^{(m+1)2}}}. \tag{3.41}$$

In practice, because c_k is not known, we can only check

$$\hat{g}_k < \hat{r}_k^{(m+1)}\ R. \tag{3.42}$$

We have used the same R in both ideal and practical tests, because $\hat{g}_k/\hat{g}_{k-1} = (c_{k-1}/c_k)g_k/g_{k-1} \approx g_k/g_{k-1}$ as c_k can change only slowly between successive steps.

When $\hat{r}_k^{(m+1)}$ is far from convergence, $p_k \ll \hat{r}_k^{(m+1)2}$, and since $(c_k^2-1)/c_k 2 < 1$, the factor multiplying R in (3.41) is almost one, and the approximate criterion (3.42) is practically equivalent to (3.41). Near convergence, $\hat{r}_k^{(m+1)2} \to r_\infty^{(m+1)2} \approx p_k$, so a large c_k could yield a substantially small factor, thus invalidating the approximate criterion. Two issues prevent this.

First, criterion (3.35) was derived from the condition that $r_k^{(m+1)2}$ should not come closer than α^2 to p_k. This gave rise to (3.30)–(3.31). Working backwards from R we have $\alpha^2 = 1/(1-R^2)$, so at any time before criterion (3.35) exits, it holds that

$$\frac{p_k}{r_k^{(m+1)}} < 1 - R^2. \tag{3.43}$$

Substituting $r_k^{(m+1)2}$ for $\hat{r}_k^{(m+1)2}$ from (3.40) and using (3.43), we can bound $\frac{p_k}{\hat{r}_k^{(m+1)}}$:

$$\frac{p_k}{\hat{r}_k^{(m+1)}} = \frac{p_k c_k^2}{r_k^{(m+1)} + p_k(c_k^2-1)} < \frac{c_k^2(1-R^2)}{1+(1-R^2)(c_k^2-1)}. \tag{3.44}$$

Then the factor multiplying R in the ideal bound (3.41) is bounded from below:

$$\sqrt{1 - \frac{(c_k^2-1)}{c_k^2}\frac{p_k}{\hat{r}_k^{(m+1)2}}} > \sqrt{\frac{1}{1+(1-R^2)(c_k^2-1)}}. \tag{3.45}$$

The above factor is always very close to 1, so checking (3.42) instead of (3.41) is a good approximation. For example, given that $R \geq 0.99$, it requires an excess of 1000 sQMR iterations, and a c_{1000} close to the bound (3.9) to require a stopping criterion of 0.95 instead of the used 0.99.

The second reason preventing a small factor (3.45) is that the term of $1 - R^2 = 1 - g_k/g_{k-1}$ becomes increasingly small with slow linear system convergence. Thus, it compensates for a possible increase in the term $c_k^2 - 1$ that could result from a large number of sQMR iterations, keeping the overall term $1 + (1-R^2)(c_k^2-1) \approx 1$.

Finally, even if the factor (3.45) does become slightly less than one, the inner sQMR is stopped a few iterations earlier than if the correct residual norms were used. This, in fact, may be beneficial because no time is wasted overconverging the inner iteration when more outer iterations are needed. On the contrary, using the upper bound $\sqrt{k+1}\,\hat{g}_k$ as the norm estimate requires more iterations to satisfy the stopping criterion and has consistently underperformed the $\hat{g}_k$ choice in our experiments.

The use of the approximate $\hat{g}_k$ and $\hat{r}_k^{(m+1)}$ may also affect criterion (3.39). Convergence may be signaled a few iterations earlier than it is actually achieved, which could cause an additional correction equation to be solved through a few additional sQMR iterations. However, this does not affect the overall convergence behavior of the algorithm. Moreover, as without preconditioning, the inner tolerance ϵ_{inn} can be chosen heuristically to reduce the number of unnecessary iterations. For example, the norm ratio $\hat{c}_k = \hat{r}_k^{(m)}/r^{(m)}$ from the previous outer iteration can be used to estimate c_k and converge to $\epsilon_{inn} = \hat{c}_k \epsilon$. In our experiments the simple criterion from (3.39) was used and performed consistently well.

3.5. The JDQMR algorithm. Our stopping criteria and the recurrences of Theorem 3.1 can be implemented trivially into the sQMR Algorithm 3.1, which in turn is called at step (19) of Algorithm 2.1. Algorithm 3.2 provides the additional lines required in Algorithm 3.1. Except for the initializations, only line (12) is replaced.

ALGORITHM 3.2.
(2.1) $\mathrm{B}_0 = \Delta_0 = \Gamma_0 = \Phi_0 = \Psi_0 = 0$

(12.0) $\gamma_k = c_k^2 \Theta_{k-1}^2,\ \xi_k = c_k^2 \alpha_{k-1}, f = 1 + \|\mathbf{t}_k\|^2$
(12.1) $\Psi_k = \gamma_k(\Psi_{k-1} + \Phi_{k-1})$
(12.2) $\Phi_k = \gamma_k^2 \Phi_{k-1} + \xi_k^2 \sigma_{k-1}$
(12.3) $\Gamma_k = \Gamma_{k-1} + 2\Psi_k + \Phi_k$
(12.4) $\Delta_k = \gamma_k \Delta_{k-1} - \xi_k \rho_{k-1}$
(12.5) $\mathrm{B}_k = \mathrm{B}_{k-1} + \Delta_k$
(12.6) $p = \left(\theta^{(m)} - \eta + 2\mathrm{B}_k + \Gamma_k\right)/f$
(12.7) $\theta_k^{(m+1)} = \eta + p$
(12.8) $p_k = (\theta^{(m)} - \eta + \mathrm{B}_k)^2/f - p^2$
(12.9) $r_k^{(m+1)} = \sqrt{g_k^2/f + p_k}$
(12.10) **if** ρ_{k-1} **return**
(12.11) **if** $r_k^{(m+1)}$ $r_k^{(m+1)} = \sqrt{g_k^2/f}$
(12.12) **if** $g_k \le r_k^{(m+1)} \max\left(0.99\sqrt{f}, \sqrt{g_k/g_{k-1}}\right)$
$\theta_k^{(m+1)} > \theta_{k-1}^{(m+1)}$ $r_k^{(m+1)} < 0.1 r_0$
$g_k < \epsilon_{inn}$ $r_k^{(m+1)} < \epsilon_{inn}$
then return $\mathbf{t}_k$

The GD outer algorithm can be implemented with storage for $2m_{max}$ long vectors of size N. If JDQMR is used, there is an additional requirement of 5 long vectors.

Excluding matvec and preconditioning operations, the expense of the outer GD Algorithm 2.1 can be calculated as a function of flops. Following classic literature, each of the steps takes: reorthogonalization $O(8Nm + 2N)$ (step (5)), updating of H $O(2mN)$ (step (7)), the small eigenproblem $O(4/3m^3)$ (step (8)), Ritz vector computation $O(2Nm)$ (step (9)), residual computation $O(2Nm + 4N)$ (steps (9)–(10)), norm computation $O(2N)$ (step (11)), and the restarting cost is $O(4Nm_{max}m_{min})$ (steps (12)–(19)). Averaging over $m = m_{min} \ldots m_{max}$ and setting $\mu = m_{min}/m_{max}$, we have the average cost per step:

$$\mathrm{GDcost} = O\left(\frac{7 + 4\mu - 7\mu^2}{1 - \mu} N m_{max} + \frac{13 + \mu}{1 - \mu} N + 1/3 m_{max}^3\right).$$

Typically, $m_{min} = m_{max}/3$, in which case $\mathrm{GDouter} = 11.3Nm_{max} + 20N + m_{max}^3/3$. These are mostly BLAS level 2 and level 1 operations. Similarly, we can obtain the cost for each QMR step, including the JD projectors: $\mathrm{QMRcost} = 24N$. However, these are strictly level 1 BLAS operations. The QMR part of JDQMR is about 15% more expensive than the CG part of JDCG, but this is justified by the increased robustness and efficiency.

4. The conjugate gradients view. An alternative to Newton is the use of the nonlinear conjugate gradient (NLCG) method to minimize the quadratic form of the Rayleigh quotient on the unit ball. As with the Newton approach, the NLCG must be applied on the Grassmann (or Stiefel) manifold for this constrained minimization problem. The appropriate forms of the constrained version for the NLCG and for the case of many required eigenvalues are given in [15]. Researchers have been using similar types of recurrences quite successfully for many decades; see Bradbury and Fletcher's seminal work [8], the long list of references in [15], as well as work in [20, 5].

NLCG methods build iterates within a Krylov space as they only utilize gradient information. Therefore they cannot converge faster than unrestarted Lanczos or the optimal QMR benchmark. NLCG, however, uses a three-term recurrence to compute both eigenvalue and eigenvector without storing all the intermediate iterates. Therefore, NLCG can often find one eigenpair in less time than the Lanczos method despite a larger number of iterations.

Although the relations between NLCG, Newton, and quasi-Newton methods are well documented [45], the question whether truncated Newton methods are more efficient than NLCG (or quasi-Newton) is problem dependent. In the extreme case of a linear unconstrained problem the NLCG is equivalent to CG, and since only one Newton step is required in that case, NLCG and truncated Newton are equivalent. For eigenvalue problems, the nonlinearity is not severe and the above NLCG variants perform quite well. Given that JDQMR is a truncated Newton method with a number of matrix-vector products within a small factor of optimal, we would like to explore its relation to NLCG variants.

4.1. The locally optimal recurrence. Most NLCG approaches consider minimization of the functional along a search direction which is usually conjugate to the previous search direction with respect to some variation of the Hessian. Although not usually expressed like this, the approximation $\mathbf{u}^{(m+1)}$ at the $(m+1)$th step belongs in the space $L = \left\{\mathbf{u}^{(m-1)}, \mathbf{u}^{(m)}, \mathbf{r}^{(m)}\right\}$, where $\mathbf{r}^{(m)} = A\mathbf{u}^{(m)} - \theta^{(m)}\mathbf{u}^{(m)}$ is the residual of $\mathbf{u}^{(m)}$. It is natural to consider a method that minimizes the Rayleigh quotient on the whole space L instead of only along one search direction. The method

$$\mathbf{u}^{(m+1)} = \text{Rayleigh–Ritz}\left(\{\mathbf{u}^{(m-1)}, \mathbf{u}^{(m)}, \mathbf{r}^{(m)}\}\right), \quad m > 1, \tag{4.1}$$

was proposed by D'yakonov in 1983 [14] and was studied further under the name locally optimal conjugate gradient (LOCG) in [30]. It is captivating in its simplicity, especially because it avoids the question of how to pick the search directions. Most importantly, the method seems to consistently outperform other NLCG-type methods.

Yet, for many years since its inception, LOCG was plagued by numerical instability problems. Obviously, a converging vector $\mathbf{u}^{(m)}$ implies an increasingly linearly dependent basis for the Rayleigh–Ritz process. Orthogonalization of the basis was not considered, mainly to keep recurrence costs low, but also because orthogonalizing two almost identical vectors would result primarily in a noise vector. In that case, the effectiveness of the method can be no worse than steepest descent. However, these problems do not arise until a good level of convergence has already been achieved. An extension to this method for many eigenvalues, and with a more stable recurrence (using $\mathbf{u}^{(m)} - \tau^{(m)}\mathbf{u}^{(m-1)}$ for some weight $\tau^{(m)}$ instead of $\mathbf{u}^{(m-1)}$) is the more recently proposed locally optimal block preconditioned CG (LOBPCG) method [32].

4.2. The locally optimal restarting. Restarting Krylov and Krylov-like methods every k iterations can have detrimental effects on their convergence, as the minimization of the Rayleigh quotient occurs only over the last k basis vectors. In the extreme case of $k = 1$ the method is simply the steepest descent. In linear systems of equations, CG has the remarkable property of implicitly remembering all the visited directions. Although the Lanczos method has the same memory, the Lanczos vectors must be revisited to compute the eigenvector. For symmetric problems, a larger subspace acceleration of steepest descent (larger k) yields better convergence, although beyond a certain basis size orthogonalization and restarting costs start to dominate.

Thick restarting [70] is a technique that improves convergence of restarted iterative methods by keeping more than the required Ritz vectors at restart ($m_{min} > nev$). The technique harnesses the superlinear convergence of Lanczos as nearby Ritz vectors are improved, and therefore are gradually deflated. It is theoretically equivalent to implicit restarting [63], and it has proved particularly effective when more than one eigenvalue is required. Still, by itself, thick restarting cannot capture the memory of directions that CG captures so well.

In [44], Murray, Racine, and Davidson proposed restarting the Davidson method with not only the required Ritz vector at the current step ($\mathbf{u}^{(m_{max})}$), but also with the Ritz vector at the previous step ($\mathbf{u}^{(m_{max}-1)}$). The motivation was to maintain the same three-term space L that CG uses to get its optimal approximation when solving a linear system of equations. However, it was unclear what this linear system is, and without the obvious connection to NLCG the method went relatively unnoticed for some years. Also, the suggested implementation was orthogonalization-heavy, requiring even extra matrix-vector products. On the other hand, the convergence improvements for Davidson, with or without preconditioning, were impressive.

4.3. The GD+k method. In [68], we showed a theoretical justification of the connection of the above restarting scheme, which we called GD+1, to CG. For completeness we mention the following two pertinent results. Proofs and additional discussion are given in [68, 64].

THEOREM 4.1. $\mathbf{u}^{(0)}$ $\|\mathbf{u}^{(0)}\| = 1$ $\theta^{(0)} = \mathbf{u}^{(0)T} A \mathbf{u}^{(0)}$ $\eta \in \Re$ $(\theta^{(j)}, \mathbf{u}^{(j)})$ j $\mathbf{u}^{(0)}$ $\mathbf{u}_j^{(1)} = \mathbf{u}^{(0)} + \mathbf{t}_j$ $\mathbf{t}_j$ j (2.1) $(m = 0)$

$$\mathbf{u}_j^{(1)} = \mathbf{u}^{(j)} \Leftrightarrow \eta = \theta^{(j)}.$$

The theorem says that if we knew the Ritz value $\theta^{(j)}$ at any Lanczos step, we could construct a (CG) three-term recurrence to yield the corresponding Ritz vector. Intermediate vector iterates would differ in general. This is similar to the relation between the iterates obtained by the optimal QMR process ($\eta = \lambda$) and the Ritz vectors of the unrestarted Lanczos. The effectiveness of the GD+1 restarting technique is attributed to the fact that the Lanczos Ritz vectors approximate the CG iterates as $\theta^{(j)}$ converges. For the smallest eigenvalue λ_1 we have the following.

LEMMA 4.2. $\gamma = \lambda_2 - \lambda_1$ $|\theta^{(0)} - \lambda_1| < \delta$ $\delta < \gamma/2$ (2.1)

$$\frac{\|\mathbf{u}^{(j)} - \mathbf{u}_j^{(1)}\|}{\|\mathbf{u}_j^{(1)}\|} \leq \frac{|\theta^{(j)} - \eta|}{\gamma - 2\delta}.$$

If we were to restart GD at the m_{max} step, the Ritz value at the $(m_{max} + 1)$ step would be minimum over only $\mathbf{u}^{(m_{max})}$ and $\mathbf{r}^{(m_{max})}$. If we also kept $\mathbf{u}^{(m_{max}-1)}$, the three-vector subspace would be similar to the three-term CG recurrence subspace that yields the exact $\mathbf{u}^{(m_{max}+1)}$. $GD+1$ $(m_{max}+1)$ $O(|\theta^{(m_{max}-1)} - \theta^{(m_{max}+1)}|)$ This justifies a stronger local optimality of the GD+1, not only with respect to the Rayleigh quotient as in LOCG, but also to the Ritz vectors.

ALGORITHM 4.1. [illegible] 2.1 [illegible]

(7.1) $s_i^{old} = s_i,\ i = 1, \ldots, m$

(12.1) [illegible] $s_i^{old},\ i = 1, \ldots, k$ [illegible]
[illegible] $s_i,\ i = 1, \ldots, m_{min}$

(12.2) [illegible] $H_{sub} = s^{oldT} H s^{old}$

(12.3) [illegible] $s = [s_1, \ldots, s_{m_{min}}, s_1^{old}, \ldots, s_k^{old}]$

(14) **for** $i = 2, \ldots, m_{min} + k$

(17.1) $H(m_{min} + 1 : m_{min} + k, m_{min} + 1 : m_{min} + k) = H_{sub}$

(17.2) $m = m_{min} + k$

In [68], we extended GD+1 to GD+k and combined it with thick restarting. By GD(m_{min}, m_{max})+k we denote the GD method with basis size m_{max}, where at restart we retain m_{min} smallest Ritz vectors from step m_{max} and in addition k smallest Ritz vectors from step $m_{max} - 1$. The special case GD(1, 3)+1 is mathematically equivalent to LOCG (and in general a block GD($k, 3k$)+k with block size k is equivalent to LOBPCG). We also provided an implementation that requires no additional orthogonalization of long vectors and no extra matrix-vector products by working exclusively with the Rayleigh–Ritz coefficients. The modifications required to the GD Algorithm 2.1 are shown in Algorithm 4.1. The expense in terms of flops of steps (12.1)–(12.3) is $O(m_{max}k^2)$ and thus insignificant for large problems. The only additional expense over GD is that we have to restart with a basis size of $m_{min} + k$ (step (14)). Because convergence improves significantly, even with $k = 1$, a much smaller m_{min} can be used than regular GD, and thus GD+1 can be less expensive per step than GD.

We emphasize that the GD+k stability stems from the orthogonality of the basis V. Typically, the small coefficient vectors s_1 and s_1^{old} will be very close. Orthogonalizing s_1^{old} against s_1 will produce a vector that may not lie exactly in the span of s_1^{old}, but it will be orthogonal to s_1. This orthogonality is bequeathed to Vs (as $V^TV = I$), which becomes a stable basis for the LOCG recurrence. In contrast, the LOBPCG method considers the vector $V(s_1 - s_1^{old})$ (with $V^TV \neq I$), which is not exactly orthogonal to Vs_1, and so the Rayleigh–Ritz procedure gives rise to a generalized eigenvalue problem, with inferior stability properties. The dangers of a nonorthogonal basis in LOBPCG have been pointed out in [28].

4.4. Global optimality and subspace acceleration. Despite the local optimality of GD+1, from both the Ritz value and Ritz vector viewpoints, there is no theoretical result showing that GD(1, 3)+1 (or equivalently LOCG) is within a small factor of optimal QMR. Extensive numerical experiments in [68, 64, 32] and in this paper suggest that this is true for the majority of cases. The above theory can provide some intuition for this. Because the eigenvalue error in symmetric problems converges as the square of the residual norm, the Ritz value quickly approximates the eigenvalue, and therefore the LOCG recurrence approximates well the optimal CG method (i.e., solving (2.1) with $\eta = \lambda_1$). Interestingly, even when the Ritz value nearly stagnates over many steps, Lemma 4.2 suggests that the LOCG recurrence can effectively build a space similar to the unrestarted Lanczos. When the algorithm realizes that it is targeting the wrong eigenvalue, the global optimality of the algorithm is lost. This is similar to terminating the inner iteration of the JDQMR to update the shift in the correction equation. Unlike JDQMR, however, each step of GD+1 involves the most current Ritz value, which may explain its faster observed convergence.

Yet, for some spectrum distributions or adversely chosen initial guesses, the Ritz

value may vary substantially, often locked temporarily on various interior eigenvalues. The result is several breaks in the global optimality of LOCG, similarly to many outer steps of a JDQMR (or RQI) algorithm. For such cases subspace acceleration and thicker restarting can improve convergence dramatically. The GD(m_{min}, m_{max})+1 method takes advantage of the global optimality when this is present, and when not, it maintains the subspace convergence of the restarted GD/Lanczos for required and nearby eigenvalues. This in turn makes the Ritz value converge faster, thus expediting the return to benefiting from the LOCG recurrence. See also [50] for the effects of subspace acceleration on classic GD. Finally, although GD(m_{min}, m_{max})+1 is a stable implementation, convergence of the coefficient vectors s_1 and s_1^{old} in Algorithm 4.1 yields a basis V which, although orthogonal, may not fully correspond to the ideal three-term space of the LOCG recurrence. Without subspace acceleration, GD(1, 3)+1 or LOBPCG revert back to steepest descent in these cases.

There is an interesting analogue from the nonlinear viewpoint. NLCG is close to optimal when the nonlinearity of the problem is not severe. When it is, e.g., when the minimization function gets trapped in various saddle points (interior eigenvalues), quasi-Newton methods almost always converge faster than NLCG [24]. These methods use the vector iterates to construct incrementally an approximation to the Hessian. A less expensive—but equally effective in terms of convergence—alternative is to use only the last few vector iterates as in the popular L-BFGS method, which is a limited memory version of the Broyden–Fletcher–Goldfarb–Shanno method [45]. In the context of eigenvalue problems, implementing L-BFGS is similar to the subspace acceleration of the restarted GD+1 method, the only difference being the minimization occurring over lines instead of the whole subspace in GD. At one extreme, L-BFGS with memory 1 (in addition to the current iterate) is equivalent to NLCG and similar to GD(1, 3)+1, while at the other extreme unlimited memory BFGS is similar to unrestarted GD (or Arnoldi). In the case of a strictly convex, quadratic objective function the equivalence extends between certain forms of L-BFGS, NLCG, and the usual CG [35]. Similar connections between quasi-Newton methods and iterative methods for linear systems have been studied. For example, the Broyden method for linear systems gives rise to the Eirola–Nevanlinna method, which is equivalent to GMRES [16]. In conclusion, we expect the subspace accelerated GD(m_{min}, m_{max})+1 to be more robust and more effective than simple recurrences (such as LOBPCG).

5. Experiments. Our first goal in the experiments is to verify that JDQMR and GD+1 are indeed within a small factor of the optimal method QMRopt. QMRopt refers to the JDQMR method where the exact eigenvalue is given as shift. Thus, one JDQMR outer step, with QMR solving (2.1), resolves the smallest eigenpair. Our second goal is to show the robustness and consistency of the JDQMR and GD+1 algorithms in a variety of situations, including also interior eigenvalue problems. Third, we provide extensive comparisons with JDCG, JDQR, ARPACK, LOBPCG, JDBSYM, and various versions of shift-invert Lanczos, with and without preconditioning. Fourth, as JDQMR and GD+1 arise as the methods of choice, we assess their relative merits. This allows for a dynamic choice between the two methods, as they both share the same implementation, leading to an overall nearly optimal method.

5.1. Tests and environment. Most of the algorithms used in our experiments are implemented in MATLAB. Our JDQMR/GD+1 code implements Algorithms 2.1–3.2 relatively closely by modifying the JDCG code of [46], which in turn is a modification of the code in [17]. To compare with ARPACK we use the MATLAB built-in function `eigs()`, and for LOBPCG we use Revision 4.0, Beta 4, with the modification

TABLE 5.1

The matrices used in the experiments. Most matrices come from Matrix Market (MM) [7], the University of Florida matrix repository (UF) maintained by Tim Davis [12], and the FEAP collection by Mark Adams [1]. SPARSKIT (SKIT) [54] was used to generate a five-point finite difference Laplacian on the unit three-dimensional cube with Dirichlet conditions. nd3kf and or56f are difficult eigenvalue problems from Yang et al. [75].

No. Matrix	N	NNZ	Source	No. Matrix	N	NNZ	Source
1. 1138BUS	1138	4054	MM	9. nd3kf	9000	3279690	Yang
2. BCSSTK09	1083	18437	MM	10. or56f	9000	2890150	Yang
3. LUNDA	147	2449	MM	11. Fillet13K_A	13572	632146	FEAP
4. 494BUS	494	1666	MM	12. Cone_A	22032	1433068	FEAP
5. 685BUS	685	3249	MM	13. Plate33K_A0	39366	914116	FEAP
6. BCSSTK16	4884	290378	MM	14. Wing22K_A	22266	923922	FEAP
7. cfd1	70656	1825580	UF	15. Laplacian3d	27000	189000	SKIT
8. finan512	74752	596992	UF	16. NASASRB	54870	2677324	MM

that `eig(A,B)` is used instead of `eig(A,B,'chol')` for numerical stability. We also provide a set of experiments with three packages written in C: our PRIMME multimethod package that implements also GD+1 and JDQMR;[1] BLOPEX, which is a LOBPCG implementation [29]; and JDBSYM, a block JD implementation [22].

In all experiments we seek only one, smallest algebraic eigenpair. For GD-based methods (GD+1, JDCG, JDQMR, JDQR) we use $m_{min} = 6$ and $m_{max} = 18$. With `eigs()` we select a basis size of 36 so that it requires the same storage as GD+1. LOBPCG requires storage for 6 vectors. The SI-eigs implements the shift-invert Lanczos by calling QMR to perform the inversions at each step of `eigs()`. A shift very close to the eigenvalue ensures that only a few outer eigs steps are needed. Because for LOBPCG we have found it far more stable to scale the matrix first, we always use $A = A/\|A\|_F$, where $\|A\|_F$ is the Frobenius norm of the matrix. Then we iterate all methods until the residual norm falls below 10^{-15}. The tolerance for `eigs()` is also set to 10^{-15}. All methods start with the same random initial guess. When there is a preconditioner, this is always the MATLAB function `cholinc(A+sI,1e-3)`, except for NASASRB, for which the threshold is `1e-5`. The shift `s` is chosen for each matrix so that the incomplete Choleski factorization can be carried out stably. Experiments are run in MATLAB 7 (R14SP3) on an Apple G5 with 1 GB of memory and two 2GHz processors, each with 512 MB L2 cache. For the C experiments we do not scale the matrix, but converge until the residual norm falls below $\|A\|_F 10^{-15}$. Experiments are run on the same Apple G5, using the gcc-4.0.0 compiler with -O flags, but without optimized BLAS/LAPACK libraries.

One of our goals is to provide experiments that can be confirmed independently by other researchers, but are also representative of various classes of problems. Thus, we have selected thirteen different matrix problems available in the following repositories: Matrix Market, University of Florida, and the FEAP collection. Two of these matrices (LUNDA and NASASRB) stem from eigenvalue computations, while the rest have spectra that present various levels of difficulty to iterative methods and the preconditioner. In addition, we have selected three matrices from eigenvalue applications, two from vibrational analysis of molecular structures [75], and the usual five-point Laplacian operator on the unit cube. A larger five-point Laplacian is used for the C experiments. Our test problems cover ranges from easy to difficult, from small to relatively large, and from sparse to relatively dense. Table 5.1 shows the name, sizes, and source for each matrix.

[1] Available at http://www.cs.wm.edu/∼andreas/software.

5.2. Nearly optimal convergence. First, we present results from matrices 1138BUS and NASASRB, as they are quite representative and demonstrate both the near optimality and robustness of GD+1 and JDQMR. Figures 5.1 and 5.2 show, for 1138BUS and NASASRB, respectively, the convergence history of the residual norm for several unpreconditioned methods as a function of matvec operations (top graphs) and as a function of time (bottom graphs). As expected, the QMRopt is the fastest method both in matvecs and in actual time, since it basically solves one linear system with QMR. Nevertheless, both JDQMR and GD+1 are very close to QMRopt.

For matrix 1138BUS, QMRopt displays superlinear convergence after about 800 matvecs, which is expected despite the highly clustered eigenvalues as the matrix is of small dimension. At the other extreme, LOBPCG achieves a linear convergence similar to the initial phase of QMRopt, but it never switches to superlinear convergence. Despite the limited basis size of 18, GD+1 manages to capture enough spectral characteristics and converges much faster than LOBPCG and similar to JDCG. The importance of the +1 restarting scheme is notable: a thick restarted, larger basis `eigs()` takes 38376 matvecs (7 times more than GD+1) to converge. Note also that the time for GD+1 is only slightly faster than that for LOBPCG because the matrix is extremely sparse and minimizing the number of matvecs does not substantially outweigh the more expensive iteration step. Similarly, JDQMR convergence is faster than JDCG, but their times are similar. Both JDQMR and JDCG convergence curves are very close to the QMRopt one.

For matrix NASASRB, LOBPCG does not converge fast enough, and `eigs()` stagnates after a large number of steps. Yet, the convergence of GD+1, JDCG, and JDQMR is almost identical to the one of QMRopt! There is one exception: JDCG has been converging to the second lowest eigenvalue and only realized it near convergence. At that point, JDCG retargets, and convergence is delayed accordingly. Besides slower convergence, this behavior may cause misidentification of eigenvalues for larger tolerances. We have noticed this JDCG behavior in a few of the other matrices we have studied. Our stopping criteria are apparently more robust in this direction, making JDQMR the fastest method (and closest to QMRopt) for this matrix.

Figure 5.3 shows a summary of our results from applying several methods on all 16 matrices without preconditioning. For each matrix, the bars show the ratios of matvecs and time (top and lower graphs, respectively) taken by each method over the QMRopt method. For completeness, we also provide the actual matvecs taken by QMRopt, GD+1, and eigs in Table 5.2. The results are surprisingly consistent, and they reflect also our experience with a larger number of different test problems. The main conclusion is that truncated Newton methods (JDCG/JDQMR) are much more efficient in finding one eigenvalue than `eigs()` (and much more so than LOBPCG) [illegible] This goes against the common wisdom that one should use the Lanczos method if no preconditioner is available. For most cases, JDQMR requires slightly fewer matvecs than JDCG, but JDCG is slightly faster because of the 15% more expensive iteration of QMR over CG. However, in three cases JDQMR is much faster than JDCG, justifying it as the more robust, general purpose method. In the absence of a preconditioner, the matvec operation in sparse matrices is rarely dense enough to justify the more expensive step of GD+1. Yet, the matvecs in GD+1 are usually the closest (and often equal) to those of QMRopt. Therefore, a runtime check in the program can assess the expense of the matvec operation, and if it is relatively high, as in matrices nd3kf and or56f, we choose GD+1 over JDQMR.

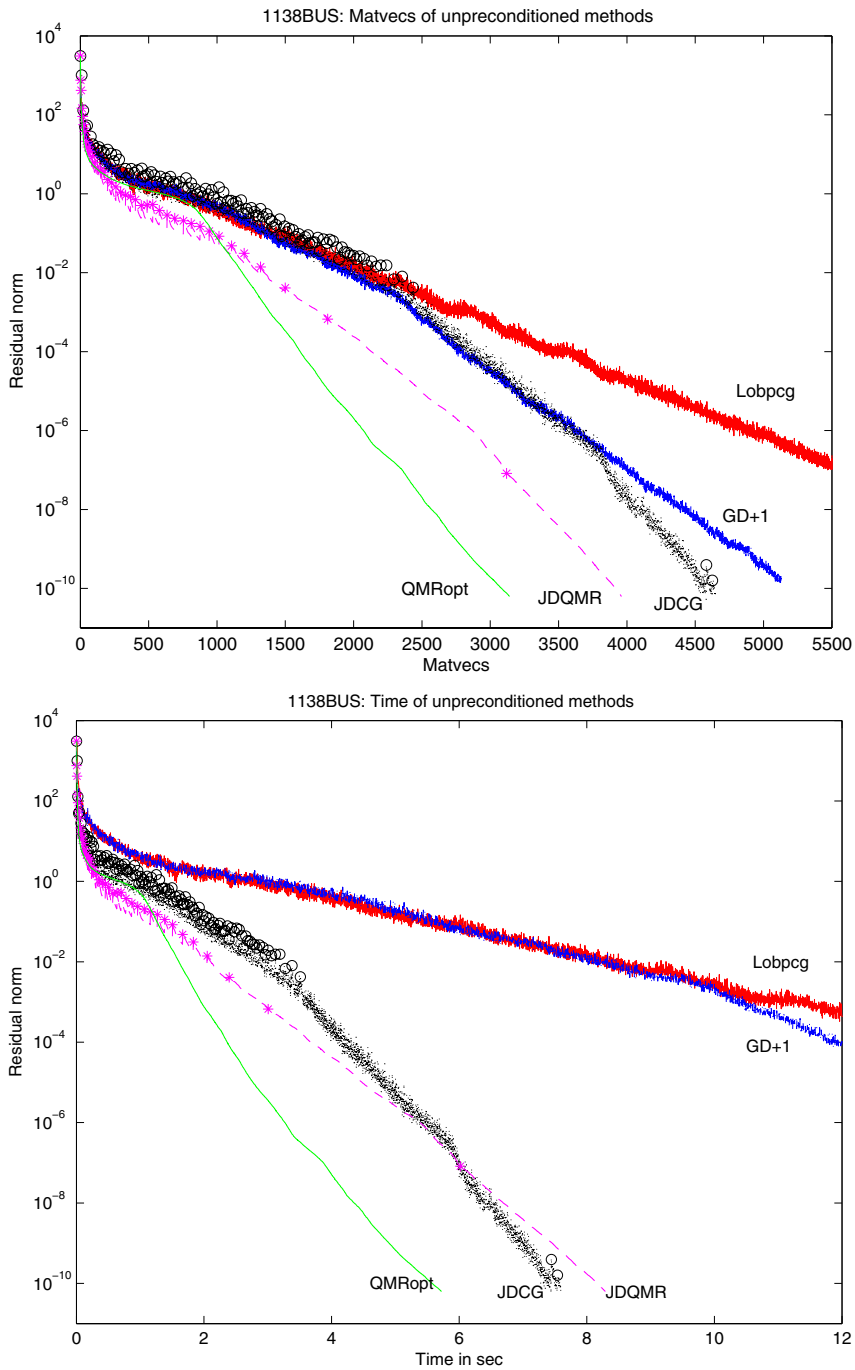


FIG. 5.1. *Residual convergence history for various methods without preconditioning for matrix* 1138*BUS* (*see Table* 5.1). (∗) *and* (○) *designate the residual norms at outer steps of JDQMR and JDCG, respectively. JDQMR improves convergence over JDCG, and GD*+1*, while being within a factor of* 1.26 *of the optimal QMR method. Because of the very inexpensive matrix-vector operation for* 1138*BUS, the convergence advantages of JDQMR are amortized by a slightly more expensive inner iteration over JDCG. This observation is amplified for the GD*+1*, which improves only on LOBPCG. eigs and SI-eigs require about* 40000 *matvecs and* 18 *seconds each.*

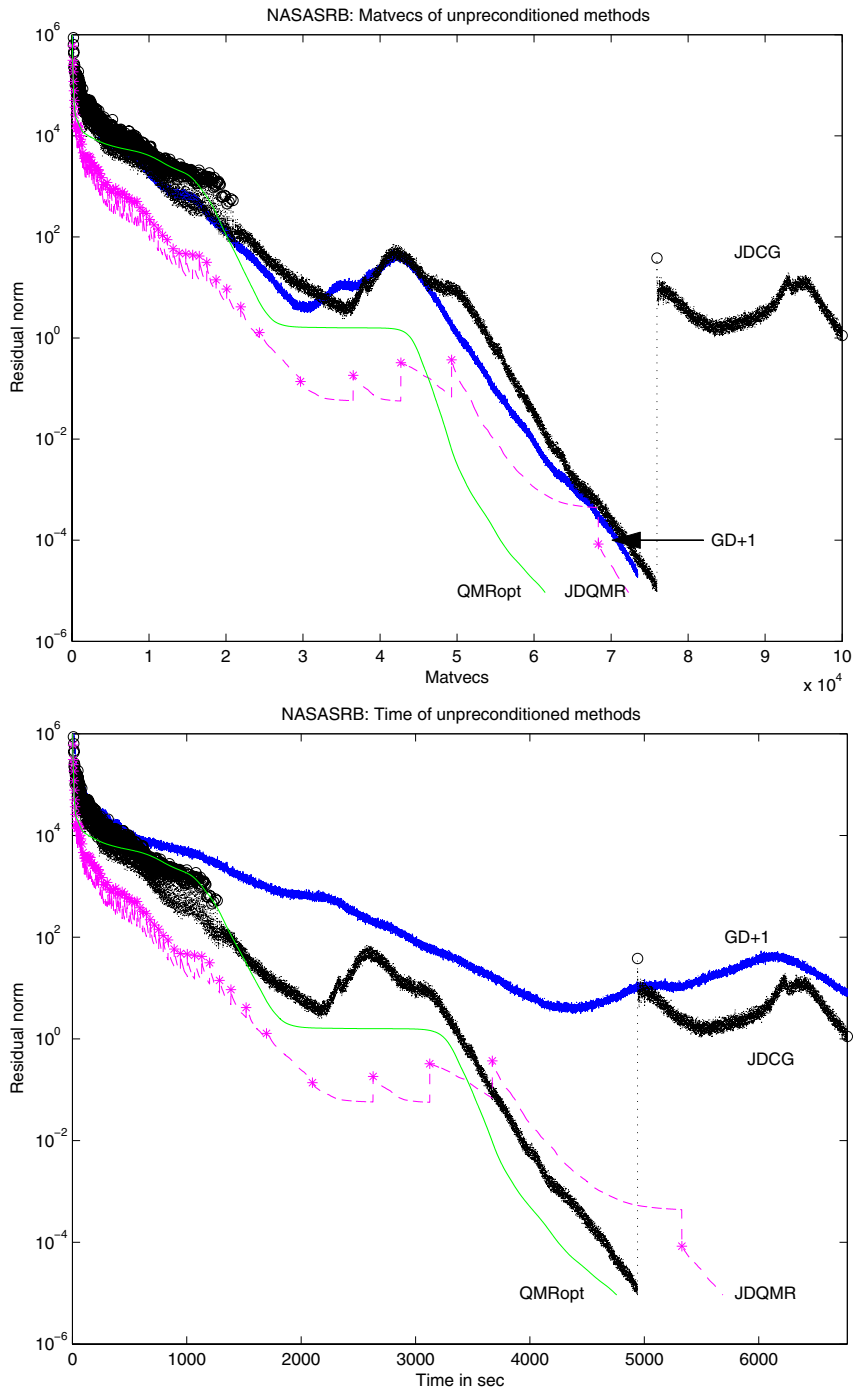


FIG. 5.2. *Residual convergence history for various methods without preconditioning for matrix NASASRB. The smallest eigenvalues are extremely hard to find. eigs seems to stagnate after* 300000 *matvecs. JDQMR is within* 1.17 *of the optimal QMR, and so is GD+1. JDCG converges similarly to GD+1 up to* 76000 *matvecs, at which point the outer step realizes that the correction vector did not correspond to the lowest eigenvalue. We have noticed this behavior also on other matrices. A relatively inexpensive matrix-vector operation plagues the time of GD+1. Convergence for LOBPCG was far slower and is not shown.*

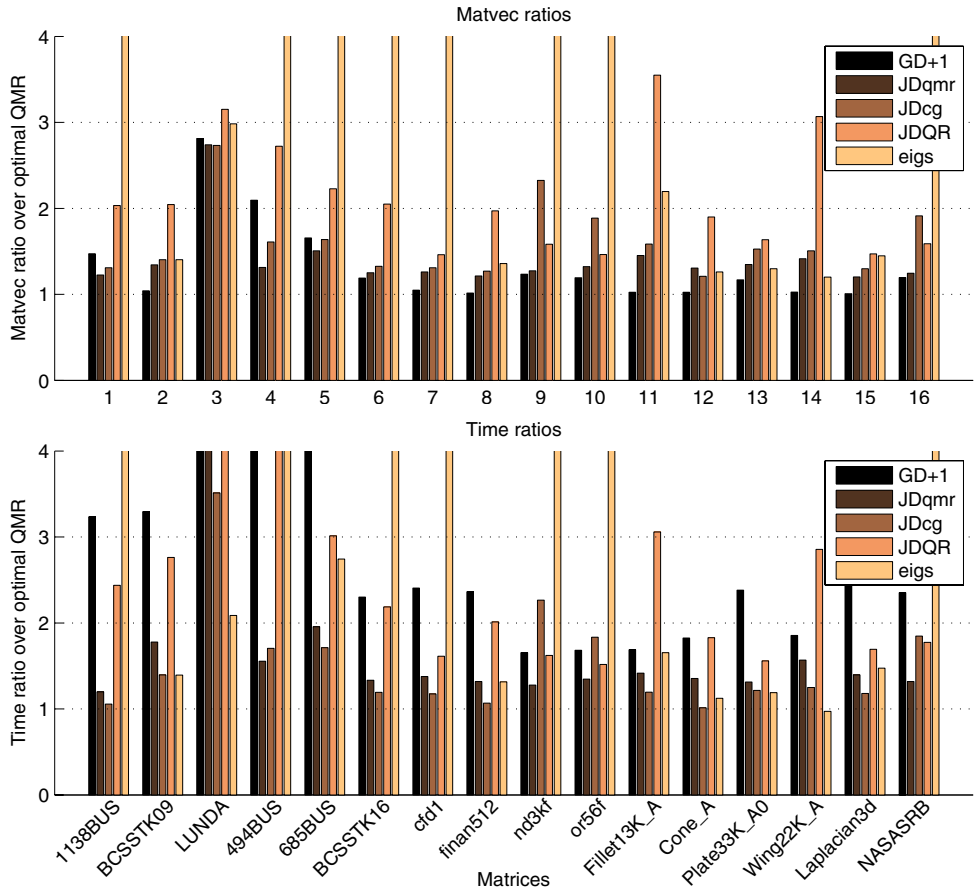


FIG. 5.3. *Ratios of matrix-vector operations (top graph) and time (bottom graph) taken by each of the five unpreconditioned methods over the the corresponding matvecs and time taken by the QMRopt. We show GD+1, JDQMR, JDCG, JDQR with* 10 *inner steps of QMR, and* `eigs()`. *LOBPCG was not competitive in these tests and is not shown. Matvecs for GD+1 are very close to QMRopt, followed closely by JDQMR. JDQMR is best timewise, because of a less expensive inner step and sparse matrix vector products.*

TABLE 5.2

Number of matrix-vector operations performed for each test matrix by three unpreconditioned methods: the optimal QMRopt, GD+1, and ARPACK (eigs).

Matrix No.	1	2	3	4	5	6	7	8
Opt QMR	3468	359	362	1798	733	439	2437	318
GD+1	5105	374	1018	3767	1214	522	2558	323
`eigs()`	38376	504	1080	17032	3293	2142	9864	432

Matrix No.	9	10	11	12	13	14	15	16
Opt QMR	8518	4765	82	200	610	75	858	61448
GD+1	10506	5687	84	205	713	77	864	73474
`eigs()`	80910	30636	180	252	792	90	1242	–

Preconditioners, when available, tend to dominate the expense at each step. In addition, as fewer iterations are needed with preconditioning, the effects of restarting are less dramatic. In such cases, we expect the GD+1 method to have an advantage.

Figure 5.4 shows the convergence history of various preconditioned methods on NASASRB. All methods perform within a factor of two of optimal, but GD+1 convergence is identical to optimal! Moreover, their relative convergence behaviors extend also to execution time, as the application of the `cholinc` preconditioner for this matrix dominates the costs. Based on the same principles, JDQMR and JDCG are similarly effective, with JDQMR slightly better.

Figure 5.5 shows a summary of results from applying five methods on our 16 matrices using preconditioning. We also provide the actual matvecs taken by QMRopt, GD+1, and SI-eigs in Table 5.3. Similarly to NASASRB, GD+1 is the clear winner in all cases, both iteration- and timewise. Moreover, for all but two cases (nd3kf and or56f) its convergence is almost identical to optimal. Even for nd3kf and or56f, convergence is well within a factor of 2 from optimal. Clearly, inner-outer SI-eigs is far from competitive. Even adaptive inner-outer schemes for RQI and INVIT as in [62, 6] cannot be any better than the subspace accelerated JDQMR.

Valuable intuition on the JDQMR convergence can be obtained through a closer look at its convergence curve in Figure 5.4. Recall that the CG/QMR convergence is usually described by three phases; an initial linear, a plateau, and a final superlinear. At the beginning of the JDQMR curve, while the easiest components of the spectrum are being identified, stopping the inner QMR process frequently does not have any negative effects on the convergence. Later, the inner QMR builds a plateau from step 100 until it is stopped at step 160. This stopping causes the inner QMR to rebuild this plateau from step 180 to step 240. In Figure 5.4, this second plateau accounts primarily for the slowdown over the optimal method. Fortunately, for the inner QMR to start to plateau, the outer JD must have reached the neighborhood of an eigenvalue. After that point, its convergence is cubic, similarly to RQI, and thus two or three outer steps are sufficient. A plateau is usually not expected during the third outer step, which is the one yielding the full accuracy. Therefore, we conjecture that JDQMR-type methods cannot be more than three times slower than the optimal method, and usually they are significantly less than two times slower. All our experiments support this conjecture. Avoiding this repetition with truncated Newton methods (such as JDQMR) is an open problem that methods of quasi-Newton type (such as GD+1) do not seem to have. Our criterion (3.36) alleviates this problem.

5.3. The effects of subspace acceleration. We provide a set of experiments that shows the effects of increasing the basis size in the GD(m_{min}, m_{max})+1 method. In Table 5.4 we report the number of matvecs and running time for GD+1 with various basis sizes, and for LOBPCG on five matrices without preconditioning. First, we observe that GD(1, 3)+1 and LOBPCG converge similarly, and often identically, supporting the theoretical equivalence of the two methods. The differences in timings are due to their different implementations and cache behaviors. Second, convergence improves drastically with a small subspace acceleration ($m_{max} = 6, 8$), but increasing m_{max} further only offers rapidly diminishing additional improvements. Consequently, the best time is achieved usually with $m_{max} = 6, 8$, unless the problem is too difficult and the matrix vector operator too expensive (see the nd3kf case).

Similar results are reported in Table 5.5 using the same preconditioner as in section 5.2. Again even a small subspace acceleration on LOCG achieves a nearly optimal convergence. Note that the orthogonal basis of GD(1, 3)+1 may improve numerical

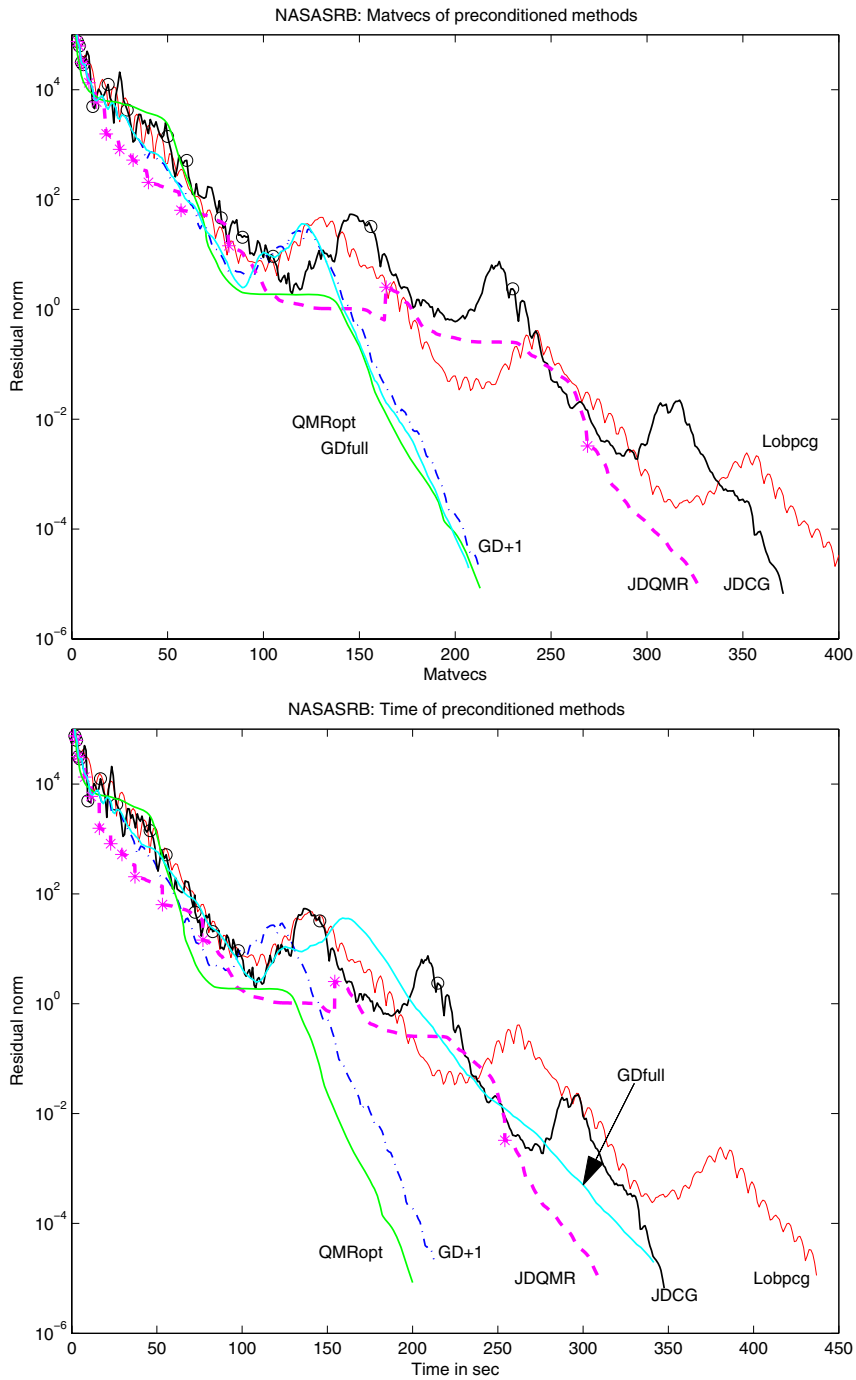


FIG. 5.4. *Residual convergence history for various preconditioned methods for matrix NASASRB. GD+1 converges identically to QMRopt and to the unrestarted GD. This shows the remarkable effectiveness of the +1 restarting scheme when coupled with subspace acceleration. LOBPCG is more than twice as slow. JDQMR and JDCG are also effective but not as much as GD+1. The inexpensive step of JDQMR/JDCG cannot outweigh the fewer applications of an expensive matrix-vector and preconditioning operators.*

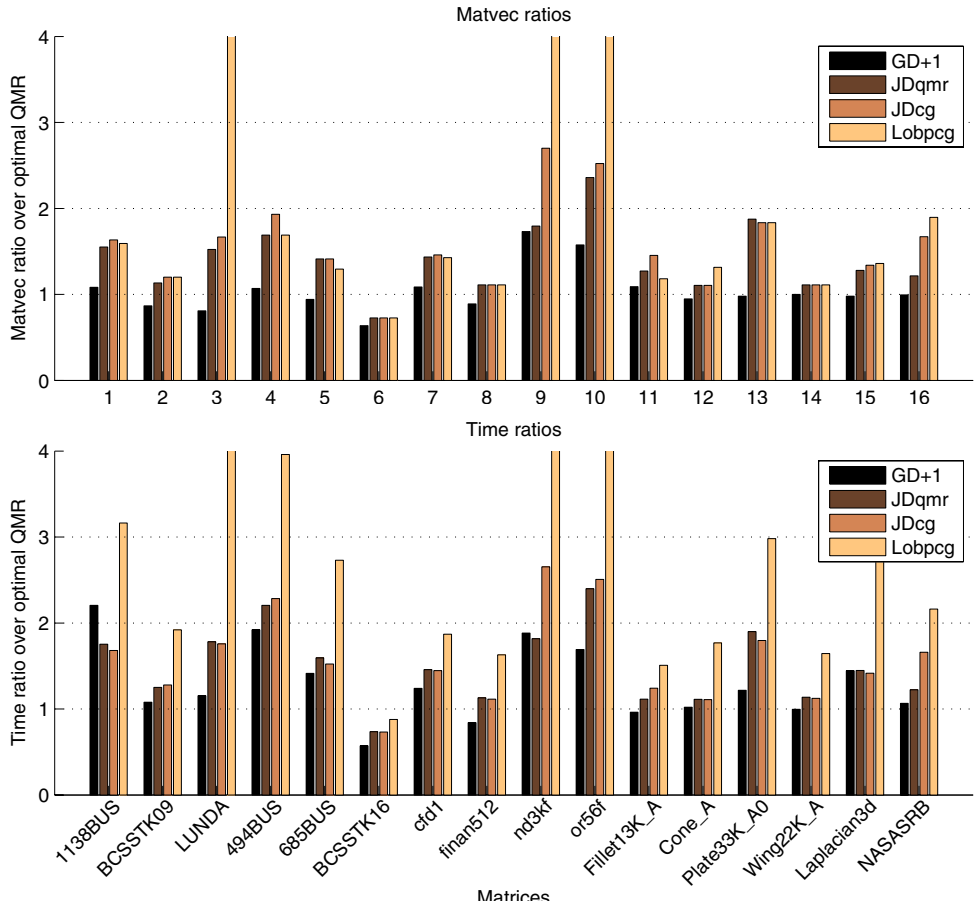


FIG. 5.5. *Ratios of matrix-vector operations (top graph) and time (bottom graph) taken by each of the four preconditioned methods over the the corresponding matvecs and time taken by the QMRopt. We show GD+1, JDQMR, JDCG, and LOBPCG. JDQR with* 10 *inner steps of QMR was not competitive in these examples and is not shown. Matvecs for GD+1 are impressively close to QMRopt and often lower, always improving on LOBPCG. JDQMR and JDCG are also very effective, but usually slower than GD+1.*

stability in some problems (see the next section), but not necessarily convergence (first two cases in Table 5.5). Finally, we note that the timings for GD$(6, 18)$+1 in the previous section could improve if a smaller m_{max} of 6 or 8 were to be used.

5.4. Experiments with PRIMME. In Table 5.6 we provide comparisons between three state-of-the-art eigenvalue packages for symmetric eigenvalue problems. Our goal in developing PRIMME (PReconditioned Iterative MultiMethod Eigensolver) was to provide a robust and nearly optimal multimethod software for which users are not required to set any parameters. The adaptivity of the JDQMR and the relatively robust choices of parameters for GD+1 make these two methods excellent candidates for default eigensolvers. We compare against BLOPEX and JDBSYM. The first three matrices are preconditioned with ILUT [54] with its parameters chosen to provide a stable factorization. The last two matrices are unpreconditioned.

First, we observe that BLOPEX (LOPBCG), although it requires no parameter

TABLE 5.3

Number of matrix-vector (equivalently preconditioning) operations performed for each test matrix by three methods: the optimal QMRopt, the GD+1, and the SI-eigs using preconditioned QMR to solve for $(A-\eta I)^{-1}$ at every step. The top table shows those matrices that required at least one restart for GD+1. The performance of GD+1 is very close to optimal. When restarting is not needed (lower table) the full subspace optimization of GD+1 even improves on the QMR recurrence. Every step of the SI-eigs requires about as many matvecs as the optimal QMR method. Thus, even if only 2 steps were required, SI-eigs would not have been competitive with GD+1. Similar results are expected with inner-outer implementations of INVIT or RQI.

Matrix No.	1	4	7	9	10	13	15	16
Opt QMR	49	29	218	1045	92	48	50	213
GD+1	53	31	237	1809	145	47	49	212
SI-eigs	458	247	2028	27683	1498	942	8553	2369

Matrix No.	2	3	5	6	8	11	12	14
Opt QMR	15	21	17	11	9	11	19	9
GD+1	13	17	16	7	8	12	18	9
SI-eigs	83	145	128	221	220	182	322	83

TABLE 5.4

The effects of subspace acceleration on GD(m_{min}, m_{max})+1 without preconditioning. LOBPCG and GD$(1,3)$+1 are theoretically equivalent.

	1138BUS		494BUS		BCSSTK16		Laplacian3d		nd3kf	
	MV	sec	MV	sec	MV	sec	MV	sec	MV	sec
LOBPCG	7495	29.72	6919	20.00	584	14.33	1546	166.91	30275	1894.17
GD$(1,3)$+1	8242	30.43	6919	16.93	610	5.84	1752	54.71	30000	1401.02
GD$(3,6)$+1	4884	16.91	4158	11.13	503	5.48	943	36.64	15732	645.18
GD$(5,8)$+1	4904	19.36	4200	12.00	507	6.49	864	41.22	10860	470.21
GD$(8,15)$+1	5196	22.98	4604	14.19	486	7.34	864	51.24	9460	447.82
GD$(15,30)$+1	4967	30.50	3386	12.87	477	10.10	863	78.84	9060	539.91

TABLE 5.5

The effects of subspace acceleration on GD(m_{min}, m_{max})+1 with preconditioning.

	Plate33K_A0		cfd1		or56f		Laplacian3d		NASASRB	
	MV	sec	MV	sec	MV	sec	MV	sec	MV	sec
LOBPCG	88	23.23	311	244.44	>1001	>211.65	64	10.01	404	435.43
GD$(1,3)$+1	127	21.47	500	309.51	>1000	>166.10	64	5.40	403	379.51
GD$(3,6)$+1	54	9.77	289	182.35	195	33.57	51	4.64	224	215.42
GD$(5,8)$+1	50	9.31	251	166.09	161	28.01	49	4.95	213	207.71
GD$(8,15)$+1	47	9.52	231	163.41	144	25.14	49	5.42	212	213.19
GD$(15,30)$+1	46	11.15	230	179.51	109	19.58	48	6.35	212	227.52

setting, is not competitive. In addition, in the cases where it failed to converge, it had reached a residual norm of $\|A\|_F 10^{-10}$ but then encountered numerical problems. The JDBSYM was used with a symmetric QMR as inner solver, so the primary difference from JDQMR is the stopping criteria. The experiments confirm that when the JDBSYM criteria [13] capture the Newton convergence well, JDBSYM is close to JDQMR and sometimes competitive (unpreconditioned cfd1). The results in parentheses, however, show cases where JDBSYM could not converge, until the user provided a shift for the correction equation that was very close to the desired eigenvalue. GD+1 and JDQMR always converge, in the least time, and with no a priori information.

TABLE 5.6
Comparison of three state-of-the-art preconditioned eigensolver codes. BLOPEX implements LOBPCG, JDBSYM implements JD with sQMR as inner solver, and our PRIMME software includes both JDQMR and GD+1, which provide almost parameter-free near optimality.

	cfd1		or56f		Plate33K_A		cfd1		Cone_A	
	MV	sec	MV	sec	MV	sec	MV	sec	MV	sec
BLOPEX	669	114.14	332	21.89	–	–	6426	186.63	–	–
GD+1	270	49.92	174	11.56	272	39.89	2858	113.86	214	3.49
JDQMR	294	44.82	190	11.88	381	51.35	2370	49.45	281	3.19
JDBSYM	373	60.42	221	14.34	(747)	(102.6)	2412	48.95	(708)	(7.70)

5.5. Interior eigenproblem with indefinite preconditioner. We borrow a model problem from [illegible] calculations that are common in many fields such as computational chemistry and materials science. In these applications, the Schrödinger operator is a sum of the Laplacian and certain local and nonlocal potential functions. We consider the eigenvalue problem stemming from the simplified operator

$$(-\nabla^2 + V)\psi = \epsilon\psi, \tag{5.1}$$

where V is a local potential. In some cases, a few lowest eigenvalues are needed corresponding to the ground (most stable) states of the system. In many cases, however, we need a few eigenvalues around the Fermi energy level, a small energy gap that separates occupied from unoccupied states. Obtaining all eigenvalues lower than the Fermi level becomes extremely expensive for systems with a large number of particles, suggesting that an interior eigenmethod should be preferable.

In real space discretizations V is a diagonal matrix. In planewave (Fourier) discretizations, however, ∇^2 becomes a diagonal matrix (and thus easily invertible) and V a dense matrix. Many planewave codes exploit this property, performing preconditioning with $(\nabla^2)^{-1}$ in Fourier space, while operating with V in real space.

Consider (5.1) in the two-dimensional unit square, with Neumann boundary conditions, and with a potential inversely proportional to the distance from z_0: $V(x) = 1/\|x - z_0\|$, with $z_0, x \in \Re^2$. We discretize the operator with a uniform finite difference, five-point stencil, yielding the usual five-diagonal Laplacian matrix with V added on the diagonal. We use $n = 110$ points in each direction, for a matrix dimension of 12100, and place the largest potential in the middle of the grid-square closest to 0: $z_0 = [0.5/n,\ 0.5/n]^T$. We look (arbitrarily) for the eigenvalue closest to 0.86, which is beyond the 800th lowest eigenvalue of our matrix. All computations are performed in real space, except preconditioning. To precondition a vector $\mathbf{r}$, we transform it through a fast sine transform to Fourier space, where the preconditioner

$$K^{-1} = (\nabla^2 - \eta_K I)^{-1} \tag{5.2}$$

is diagonal. An inverse sine transform returns the preconditioned vector to real space.

Naturally, JDCG cannot be used if the shift η in the correction (2.1) is inside the spectrum. Moreover, JD with MINRES as inner solver cannot be used either if η_K makes the preconditioner (5.2) indefinite. The question is whether treating the problem as an interior with indefinite preconditioner is beneficial. Table 5.7 shows the performance as the number of preconditioning operations (equivalently matrix vector products) for various methods. Because our MATLAB implementation of the preconditioner is very expensive we do not report times.

GD+1 is the closest to the QMRopt method that uses the exact eigenvalue both in (2.1) and in (5.2). Note that the threefold factor in slowdown does not reflect a

TABLE 5.7

The benefits of JDQMR for finding one interior eigenvalue closest to 0.86. *The lowest eigenvalue of A is greater than* -0.05. *In the upper table an interior problem is solved, where both preconditioner and correction equation can be indefinite. A positive definite preconditioner* ($\eta_K = -0.05$), *as required by MINRES, does not converge. JDCG does not converge either when solving the correction equation for the extreme eigenvalue. The lower table shows that the transformation into an extreme eigenproblem for* $(A - 0.86I)^2$ *is not a good alternative, even though JDQMR is far better than JDCG. ARPACK* (*eigs*) *without preconditioning cannot converge.*

Seek eigenvalue closest to $\sigma = 0.86$. ($\lambda_{exact} = 0.859768$), $-0.05 < \lambda_i < 7.99$				
Interior approach Correction equation with operator $(A - \eta I)$ and preconditioner $(\nabla^2 - \eta_K I)$				
Method	η	η_K	Matvecs	Equivalent method
QMRopt	λ_{exact}	λ_{exact}	261	Optimal correction/no dynamic stopping
GD+1	$\theta^{(m)}$	$\theta^{(m)}$	790	GD with recurrence restarting
JDQMR	$\theta^{(m)}$	$\theta^{(m)}$	1330	Adaptive accelerated RQI
JDQMR	0.86	$\theta^{(m)}$	1418	
JDQMR	0.86	0.86	1669	Adaptive accelerated INVIT
SI-eigs	0.86	0.86	12294	Shift-invert ARPACK, QMR inner solver
JDQMR	0.86	-0.05	–	Adaptive JDMINRES
JDCG	-0.05	-0.05	–	
Exterior approach Smallest eigenvalue of $(A - 0.86I)^2$ with preconditioner $(\nabla^2 - 0.86I)^2$				
Method			Matvecs	
GD+1			2642	
JDQMR			4490	
JDCG			12736	
eigs			$\gg 30000$	(no preconditioning)

decrease in convergence rate but rather the fact that the solver spends most of the time trying to identify the proper eigenvalue to target. Once the eigenvalue is located, the convergence rate is similar to the optimal method. We have not investigated the use of harmonic Ritz values that could improve this identification. JDQMR converges in roughly 70% more iterations than GD+1. However, JDQMR too spends most of its time identifying the eigenpair, in fact exiting after one inner iteration. The astute reader may have noticed that in such cases the JDQMR is simply GD+1, only with two preconditioning operations per step. A more economical implementation is possible, but beyond our current discussion. As expected, fixing one or both of the shifts η, η_K makes convergence slightly worse, as demonstrated by various adaptive versions of INVIT. Without inner-outer adaptivity, the SI-eigs is an order of magnitude slower than JDQMR. For this example, JDMINRES with a positive definite preconditioner, or JDCG on a definite correction equation do not converge.

A common practice for interior eigenproblems is to solve for the smallest eigenvalue of $(A - \sigma I)^2$. This approach is attractively simple, but rarely competitive because of worse conditioning. The lower half of Table 5.7 shows the performance of four methods on the squared matrix with a similarly squared preconditioner. The ratio between GD+1 and JDQMR iterations is the same as in the interior method, while the JDCG converges but significantly slower. As it cannot exploit preconditioning, `eigs()` cannot converge in tractable time.

6. Conclusions. Our goal is to develop a method, or multimethod, that converges near optimally and robustly for large, difficult eigenvalue problems. The

method must operate under limited memory and be capable of using various preconditioners. We showed that such a method should approach the eigenvalue problem from the nonlinear perspective, either through the truncated Newton or through the limited memory quasi-Newton techniques. Regardless of the technique, however, the underlying outer method must follow the GD outer scheme.

We described a truncated Newton technique, JDQMR, which extends the previously proposed JDCG method. JDQMR can work with indefinite correction equation and preconditioners, and thus can be used for finding interior eigenvalues, while providing better stopping criteria that improve robustness. In addition, we argued why JDQMR converges within a small factor (2 or 3) of optimal.

We also described methods that use the locally optimal CG either as a recurrence (LOBPCG) or as a restarting (GD+1) and unified them under the framework of limited memory quasi-Newton. This framework, together with our previous theoretical results, provided new intuition on both the global and local convergence of these methods, for instance, explaining why GD+1 improves on unaccelerated recurrence methods such as LOBPCG.

A rather unexpected conclusion from our experiments is that limited memory BFGS-type methods (GD+1) outperform truncated Newton methods (such as JDCG or JDQMR) in terms of convergence, even with a small acceleration basis. Equally impressive is the fact that GD+1 usually matches the convergence, both asymptotic and global, of the optimal QMRopt benchmark.

As the acceleration basis shrinks, the convergence of JDQMR, which is less sensitive to acceleration, matches or surpasses the convergence of GD+1, which, in turn, is always better than the extreme of using no acceleration basis (LOBPCG). Furthermore, when the preconditioner and matrix-vector operations are inexpensive, the cheaper inner iteration of JDQMR requires less overall time than GD+1. The advantage of our approach is that the GD driver implements both methods, switching at runtime when the actual expense of the user-provided operators is measured.

Finally, we have provided an extensive set of experiments. We considered a variety of problems and methods, with and without preconditioning, and problems that involved finding an interior eigenpair using indefinite preconditioning. Interestingly, both the theory and our experiments suggest that even without preconditioning, our multimethod should be preferred over ARPACK. Our results and conclusions extend also to finding a small number of eigenpairs. When many eigenvalues are required, the challenges and results are different; these topics are addressed in a companion paper [66].

Acknowledgments. The author would like to thank the two anonymous referees for their significant contribution to the clarity of the presentation and for prompting the analysis in section 3.4. The presentation also benefited from comments from the editor, Dr. T. Kolda, and the author's colleagues, Profs. V. Torczon and P. Stockmeyer.

REFERENCES

[1] M. F. Adams, *Evaluation of three unstructured multigrid methods on* 3*d finite element problems in solid mechanics*, Internat. J. Numer. Methods Engrg., 55 (2002), pp. 519–534.

[2] P. Arbenz, U. L. Hetmaniuk, R. B. Lehoucq, and R. S. Tuminaro, *A comparison of eigensolvers for large-scale* 3*D modal analysis using AMG-preconditioned iterative methods*, Internat. J. Numer. Methods Engrg., 64 (2005), pp. 204–236.

[3] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, eds., *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, SIAM, Philadelphia, 2000.

[4] R. Barrett, M. W. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, 1994.

[5] L. Bergamaschi, G. Gambolati, and G. Pini, *Asymptotic convergence of conjugate gradient methods for the partial symmetric eigenproblem*, Numer. Linear Algebra Appl., 4 (1997), pp. 69–84.

[6] J. Berns-Mueller, *Inexact Inverse Iteration Using Galerkin Krylov Solvers*, Ph.D. thesis, University of Bath, Bath, UK, 2003.

[7] R. F. Boisvert, R. Pozo, K. Remington, R. Barrett, and J. Dongarra, *The Matrix Market: A web resource for test matrix collections*, in Quality of Numerical Software, Assessment and Enhancement, R. F. Boisvert, ed., Chapman & Hall, London, 1997, pp. 125–137.

[8] W. W. Bradbury and R. Fletcher, *New iterative methods for solution of the eigenproblem*, Numer. Math., 9 (1966), pp. 259–267.

[9] M. Clint and A. Jennings, *The evaluation of eigenvalues and eigenvectors of a real symmetric matrix by simultaneous iteration*, Comput. J., 13 (1970), pp. 76–80.

[10] M. Crouzeix, B. Philippe, and M. Sadkane, *The Davidson method*, SIAM J. Sci. Comput., 15 (1994), pp. 62–76.

[11] E. R. Davidson, *The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices*, J. Comput. Phys., 17 (1975), pp. 87–94.

[12] T. Davis, *University of Florida Sparse Matrix Collection*, Technical report, University of Florida. NA Digest, vol. 92, no. 42, October 16, 1994, NA Digest, vol. 96, no. 28, July 23, 1996, and NA Digest, vol. 97, no. 23, June 7, 1997.

[13] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *Inexact Newton methods*, SIAM J. Numer. Anal., 19 (1982), pp. 400–408.

[14] E. G. D'yakonov, *Iteration methods in eigenvalue problems*, Math. Notes, 34 (1983), pp. 945–953.

[15] A. Edelman, T. A. Arias, and S. T. Smith, *The geometry of algorithms with orthogonality constraints*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 303–353.

[16] T. Eirola and O. Nevanlinna, *Accelerating with rank-one updates*, Linear Algebra Appl., 121 (1989), pp. 511–520.

[17] D. R. Fokkema, G. L. G. Sleijpen, and H. A. van der Vorst, *Jacobi–Davidson style QR and QZ algorithms for the reduction of matrix pencils*, SIAM J. Sci. Comput., 20 (1998), pp. 94–125.

[18] R. W. Freund and N. M. Nachtigal, *A new Krylov-Subspace Method for Symmetric Indefinite Linear Systems*, Technical report, AT&T Bell Laboratories, Murray Hill, NJ, 1994.

[19] R. W. Freund and N. M. Nachtigal, *Software for simplified Lanczos and QMR algorithms*, Appl. Numer. Math., 19 (1995), pp. 319–341.

[20] G. Gambolati, F. Sartoretto, and P. Florian, *An orthogonal accelerated deflation technique for large symmetric eigenproblems*, Comput. Methods Appl. Mech. Engrg., 94 (1992), pp. 13–23.

[21] M. Genseberger and G. L. G. Sleijpen, *Alternative correction equations in the Jacobi-Davidson method*, Numer. Linear Algebra Appl., 6 (1999), pp. 235–253.

[22] R. Geus, *JDBSYM*, http://people.web.psi.ch/geus/software.html.

[23] R. Geus, *The Jacobi-Davidson Algorithm for Solving Large Sparse Symmetric Eigenvalue Problems with Application to the Design of Accelerator Cavities*, Ph.D. thesis 14734, ETH, Zurich, Switzerland, 2002.

[24] P. H. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, New York, 1986.

[25] G. H. Golub and Q. Ye, *Inexact inverse iteration for generalized eigenvalue problems*, BIT, 40 (2000), pp. 671–684.

[26] G. H. Golub, Z. Zhang, and H. Zha, *Large sparse symmetric eigenvalue problems with homogeneous linear constraints: The Lanczos process with inner-outer iterations*, Linear Algebra Appl., 309 (2000), pp. 289–306.

[27] R. G. Grimes, J. G. Lewis, and H. D. Simon, *A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 228–272.

[28] U. Hetmaniuk and R. B. Lehoucq, *Basis selection in LOBPCG*, J. Comput. Phys., 28 (2006), pp. 324–332.

[29] A. V. Knyazev, *BLOPEX*, http://www-math.cudenver.edu/~aknyazev/software/BLOPEX.

[30] A. V. Knyazev, *Convergence rate estimates for iterative methods for mesh symmetric eigenvalue problem*, Soviet J. Numer. Anal. Math. Modeling, 2 (1987), pp. 371–396.

[31] A. V. Knyazev, *Preconditioned eigensolvers—an oxymoron?*, Electron. Trans. Numer. Anal., 7 (1998), pp. 104–123.
[32] A. V. Knyazev, *Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method*, SIAM J. Sci. Comput., 23 (2001), pp. 517–541.
[33] A. V. Knyazev and K. Neymeyr, *Efficient solution of symmetric eigenvalue problems using multigrid preconditioners in the locally optimal block conjugate gradient method*, Electron. Trans. Numer. Anal., 15 (2003), pp. 38–55.
[34] A. V. Knyazev and K. Neymeyr, *A geometric theory for preconditioned inverse iteration.* III: *A short and sharp convergence estimate for generalized eigenvalue problems*, Linear Algebra Appl., 358 (2003), pp. 95–114.
[35] T. G. Kolda, D. P. O'Leary, and L. Nazareth, *BFGS with update skipping and varying memory*, SIAM J. Optim., 8 (1998), pp. 1060–1083.
[36] Y.-L. Lai, K.-Y. Lin, and W.-W. Lin, *An inexact inverse iteration for large sparse eigenvalue problems*, Numer. Linear Algebra. Appl., 4 (1997), pp. 425–437.
[37] R. B. Lehoucq and K. Meerbergen, *Using generalized Cayley transformations within an inexact rational Krylov sequence method*, SIAM J. Matrix Anal. Appl., 20 (1998), pp. 131–148.
[38] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, SIAM, Philadelphia, 1998.
[39] B. Liu, *Numerical Algorithms in Chemistry: Algebraic Methods*, C. Moler and I. Shavitt, eds., Technical Report LBL-8158, Lawrence Berkeley Laboratory, 1978.
[40] E. Lundström and L. Eldén, *Adaptive eigenvalue computations using Newton's method on the Grassmann manifold*, SIAM J. Matrix Anal. Appl., 23 (2002), pp. 819–839.
[41] K. Meerbergen, *Robust Methods for the Calculation of Rightmost Eigenvalues of Nonsymmetric Problems*, Ph.D. thesis, Department of Computer Science, K. U. Leuven, Heverlee, Belgium, 1996.
[42] R. B. Morgan, *On restarting the Arnoldi method for large nonsymmetric eigenvalue problems*, Math. Comput., 65 (1996), pp. 1213–1230.
[43] R. B. Morgan and D. S. Scott, *Generalizations of Davidson's method for computing eigenvalues of sparse symmetric matrices*, SIAM J. Sci. Comput., 7 (1986), pp. 817–825.
[44] C. W. Murray, S. C. Racine, and E. R. Davidson, *Improved algorithms for the lowest eigenvalues and associated eigenvectors of large matrices*, J. Comput. Phys., 103 (1992), pp. 382–389.
[45] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer-Verlag, New York, 1999.
[46] Y. Notay, *Combination of Jacobi-Davidson and conjugate gradients for the partial symmetric eigenproblem*, Numer. Linear Algebra Appl., 9 (2002), pp. 21–44.
[47] Y. Notay, *Is Jacobi–Davidson faster than Davidson?*, SIAM J. Matrix Anal. Appl., 26 (2005), pp. 522–543.
[48] J. Olsen, P. Jørgensen, and J. Simons, *Passing the one-billion limit in full configuration-interaction (FCI) calculations*, Chem. Phys. Lett., 169 (1990), pp. 463–472.
[49] E. Ovtchinnikov, *Convergence estimates for the generalized Davidson method for symmetric eigenvalue problems* I: *The preconditioning aspect*, SIAM J. Numer. Anal., 41 (2003), pp. 258–271.
[50] E. Ovtchinnikov, *Convergence estimates for the generalized Davidson method for symmetric eigenvalue problems* II: *The subspace acceleration*, SIAM J. Numer. Anal., 41 (2003) pp. 272–286.
[51] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.
[52] B. N. Parlett, *The Symmetric Eigenvalue Problem*, SIAM, Philadelphia, 1998.
[53] A. Ruhe and T. Wiberg, *The method of conjugate gradients used in inverse iteration*, BIT, 12 (1972) pp. 543–554.
[54] Y. Saad, *SPARSKIT: A Basic Toolkit for Sparse Matrix Computations*, Technical Report 90-20, Research Institute for Advanced Computer Science, NASA Ames Research Center, Moffet Field, CA, 1990. Software currently available at ftp://ftp.cs.umn.edu/dept/sparse/.
[55] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, Manchester University Press, Manchester, UK, 1992.
[56] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, 2003.
[57] A. Sameh and Z. Tong, *The trace minimization method for the symmetric generalized eigenvalue problem*, J. Comput. Appl. Math., 123 (2000), pp. 155–175.
[58] V. Simoncini and L. Eldén, *Inexact Rayleigh quotient-type methods for eigenvalue computations*, BIT, 42 (2002), pp. 159–182.

[59] G. L. G. Sleijpen, A. G. L. Booten, D. R. Fokkema, and H. A. van der Vorst, *Jacobi-Davidson type methods for generalized eigenproblems and polynomial eigenproblems*, BIT, 36 (1996), pp. 595–633.

[60] G. L. G. Sleijpen and H. A. van der Vorst, *A Jacobi–Davidson iteration method for linear eigenvalue problems*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 401–425.

[61] G. L. G. Sleijpen and Fred W. Wubs, *Effective Preconditioning Techniques for Eigenvalue Problems*, Technical Report 1117, Department of Mathematics, University of Utrecht, 1999.

[62] P. Smit and M. H. C. Paardekooper, *The effects of inexact solvers in algorithms for symmetric eigenvalue problems*, Linear Algebra Appl., 287 (1999), pp. 337–357.

[63] D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 357–385.

[64] A. Stathopoulos, *Some insights on restarting symmetric eigenvalue methods with Ritz and harmonic Ritz vectors*, in Iterative Methods in Scientific Computation IV, IMACS, D. R. Kincaid and Anne C. Elster, eds., Rutgers University, New Brunswick, NJ, 1999, pp. 297–311.

[65] A. Stathopoulos and C. F. Fischer, *A Davidson program for finding a few selected extreme eigenpairs of a large, sparse, real, symmetric matrix*, Comput. Phys. Commun., 79 (1994), pp. 268–290.

[66] A. Stathopoulos and J. R. McCombs, *Nearly Optimal Preconditioned Methods for Hermitian Eigenproblems under Limited Memory, Part* II: *Seeking Many Eigenvalues*, Technical Report WM-CS-2006-02; submitted for publication.

[67] A. Stathopoulos, S. Öğüt, Y. Saad, J. R. Chelikowsky, and H. Kim, *Parallel methods and tools for predicting material properties*, Comput. Sci. Engrg., 2 (2000), pp. 19–32.

[68] A. Stathopoulos and Y. Saad, *Restarting techniques for (Jacobi-)Davidson symmetric eigenvalue methods*, Electron. Trans. Numer. Algebra, 7 (1998), pp. 163–181.

[69] A. Stathopoulos, Y. Saad, and C. F. Fischer, *Robust preconditioning of large, sparse, symmetric eigenvalue problems*, J. Comput. Appl. Math., 64 (1995), pp. 197–215.

[70] A. Stathopoulos, Y. Saad, and K. Wu, *Dynamic thick restarting of the Davidson, and the implicitly restarted Arnoldi methods*, SIAM J. Sci. Comput., 19 (1998), pp. 227–245.

[71] D. B. Szyld, *Criteria for combining inverse and Rayleigh quotient iteration*, SIAM J. Numer. Anal., 25 (1988), pp. 1369–1375.

[72] H. A. van der Vorst, *Iterative Krylov Methods for Large Linear Systems*, Cambridge Monogr. Appl. Comput. Math. 13, Cambridge University Press, Cambridge, UK, 2003.

[73] K. Wu and H. Simon, *Thick-restart Lanczos method for large symmetric eigenvalue problems*, SIAM J. Matrix Anal. Appl., 22 (2000), pp. 602–616.

[74] K. Wu, Y. Saad, and A. Stathopoulos, *Inexact newton preconditioning techniques for eigenvalue problems*, Electron. Trans. Numer. Algebra, 7 (1998), pp. 202–214.

[75] C. Yang, B. W. Peyton, D. W. Noid, B. G. Sumpter, and R. E. Tuzun, *Large-scale normal coordinate analysis for molecular structures*, SIAM J. Sci. Comput., 23 (2001), pp. 563–582.

SIAM J. SCI. COMPUT.
Vol. 29, No. 2, pp. 515–538
© 2007 Society for Industrial and Applied Mathematics

A BLOCH DECOMPOSITION–BASED SPLIT-STEP PSEUDOSPECTRAL METHOD FOR QUANTUM DYNAMICS WITH PERIODIC POTENTIALS*

ZHONGYI HUANG†, SHI JIN‡, PETER A. MARKOWICH§, AND CHRISTOF SPARBER§

Abstract. We present a new numerical method for accurate computations of solutions to (linear) one-dimensional Schrödinger equations with periodic potentials. This is a prominent model in solid state physics where we also allow for perturbations by nonperiodic potentials describing external electric fields. Our approach is based on the classical Bloch decomposition method, which allows us to diagonalize the periodic part of the Hamiltonian operator. Hence, the dominant effects from dispersion and periodic lattice potential are computed together, while the nonperiodic potential acts only as a perturbation. Because the split-step communicator error between the periodic and nonperiodic parts is relatively small, the step size can be chosen substantially larger than for the traditional splitting of the dispersion and potential operators. Indeed it is shown by the given examples that our method is unconditionally stable and more efficient than the traditional split-step pseudospectral schemes. To this end a particular focus is on the semiclassical regime, where the new algorithm naturally incorporates the adiabatic splitting of slow and fast degrees of freedom.

Key words. Schrödinger equation, Bloch decomposition, time-splitting spectral method, semiclassical asymptotics, lattice potential

AMS subject classifications. 65M70, 74Q10, 35B27, 81Q20

DOI. 10.1137/060652026

1. Introduction.

One of the main problems in solid state physics is to describe the motion of electrons within the periodic potentials generated by the ionic cores. This problem has been studied from a physical as well as a mathematical point of view in, e.g., [1, 8, 28, 29, 33], resulting in a profound theoretical understanding of the novel dynamical features. Indeed one of the most striking effects, known as [illegible], is a modification of the dispersion relation for Schrödinger's equation, where the classical energy relation $E_{\text{free}}(k) = \frac{1}{2}|k|^2$ has to be replaced by $E_m(k)$, $m \in \mathbb{N}$, the energy corresponding to the mth [illegible] [7]. The basic idea behind this replacement is a separation of scales which is present in this context. More precisely one recognizes that experimentally imposed, and thus called external, electromagnetic fields typically vary on much [illegible] spatial scales than the periodic potential generated by the cores. Moreover, this external field can be considered [illegible] in comparison to the periodic fields of the cores [2].

To study this problem, consider the Schrödinger equation for the electrons in a

*Received by the editors February 13, 2006; accepted for publication (in revised form) September 19, 2006; published electronically March 15, 2007. This work was partially supported by the Wittgenstein Award 2000 of P. A. M., NSF grant DMS-0305080, NSFC projects 10301017 and 10228101, the National Basic Research Program of China under grant 2005CB321701, SRF for ROCS, SEM, and the Austrian-Chinese Technical-Scientific Cooperation Agreement.
http://www.siam.org/journals/sisc/29-2/65202.html

†Department of Mathematics, Tsinghua University, Beijing, 10084, China (zhuang@math.tsinghua.edu.cn).

‡Department of Mathematics, University of Wisconsin, Madison, WI 53706-1388 (jin@math.wisc.edu).

§Wolfgang Pauli Institute Vienna & Faculty of Mathematics, University of Vienna, Nordbergstrasse 15, A-1090 Vienna, Austria (peter.markowich@univie.ac.at, christof.sparber@univie.ac.at).
The fourth author was supported by the APART grant of the Austrian Academy of Science.

asymptotic scaling [11, 29, 31], i.e., in $d = 1$ dimensions,

$$\begin{cases} \mathrm{i}\varepsilon\partial_t\psi = -\dfrac{\varepsilon^2}{2}\,\partial_{xx}\psi + V_\Gamma\left(\dfrac{x}{\varepsilon}\right)\psi + U(x)\psi, \quad x \in \mathbb{R},\ t \in \mathbb{R}, \\ \psi\big|_{t=0} = \psi_{\mathrm{in}}(x), \end{cases} \tag{1.1}$$

where $0 < \varepsilon \ll 1$ denotes the small describing the microscopic/macroscopic scale ratio. The (dimensionless) equation (1.1) consequently describes the motion of the electrons on the macroscopic scales induced by the external potential $U(x) \in \mathbb{R}$. The highly oscillating $V_\Gamma(y) \in \mathbb{R}$ is assumed to be with respect to some Γ. For definiteness we shall assume that

$$V_\Gamma(y + 2\pi) = V_\Gamma(y) \quad \forall\, y \in \mathbb{R}, \tag{1.2}$$

i.e., $\Gamma = 2\pi\mathbb{Z}$. In the following we shall assume $\psi_{\mathrm{in}} \in L^2(\mathbb{R})$, such that the is $\mathrm{M}_{\mathrm{in}} \equiv \|\psi_{\mathrm{in}}\|_{L^2} = 1$, a normalization which is henceforth preserved by the evolution.

The mathematically precise asymptotic description of $\psi(t)$, solution to (1.1), as $\varepsilon \to 0$, has been intensively studied in, e.g., [6, 16, 20, 29], relying on different analytical tools. On the other hand the numerical literature on these issues is not so abundant [17, 18, 19]. Here we shall present a novel approach to the numerical treatment of (1.1) relying on the classical method, as explained in more detail below. The main idea is to treat in one step the purely dispersive part $\propto \partial_{xx}$ of the Schrödinger equation together with the periodic potential V_Γ, since this combined operator allows for some sort of "diagonalization" via the Bloch transformation. The corresponding numerics is mainly concerned with the case $\varepsilon \ll 1$, but we shall also show examples for a rather large $\varepsilon = \frac{1}{2}$. Our numerical experiments show that the new method converges with $\Delta x = \mathcal{O}(\varepsilon)$ and $\Delta t = \mathcal{O}(1)$, the latter being a huge advantage in comparison with the more standard time-splitting method used in [17, 18, 19], and which usually requires $\Delta t = \mathcal{O}(\varepsilon)$. Moreover, we find that the use of only a few Bloch bands is mostly enough to achieve very high accuracy, even in cases where $U(x)$ is no longer smooth. We note that our method is unconditionally stable and comprises spectral convergence for the space discretization as well as second order convergence in time. The only drawback of the method is that we first have to compute the energy bands for a given periodic potential, although this is needed only in a preprocessing step rather than during the time marching. On the other hand, this preprocessing also handles a possible lack of regularity in V_Γ, which consequently does not lead to numerical problems during the time-evolution. In any case the numerical cost of this preliminary step is much smaller than the costs spent in computing the time-evolution, and this holds true for whatever method we choose.

We remark that linear and nonlinear evolutionary PDEs with periodic coefficients also arise in the study of photonic crystals, laser optics, and Bose–Einstein condensates in optical lattices; cf. [9, 11, 21] and the references given therein. We expect that our algorithm can be adapted to these kinds of problems too. Also note that in the case of a so-called (see, e.g., [6, 5]), an adaptation of our code to higher dimensions is very likely. Finally, the use of the Bloch transformation in problems of has been discussed in [12, 14] and numerically studied in [13] for elliptic problems. Our algorithm might be useful in similar time-dependent numerical homogenization problems.

The paper is organized as follows: In section 2, we recall in detail the Bloch decomposition method and show how to numerically calculate the corresponding energy

bands. Then, in section 3 we present our new algorithm, as well as the usual time-splitting spectral method for Schrödinger equations. In section 4, we show several numerical experiments and compare both methods. Different examples of U and V_Γ are considered, including the nonsmooth cases. Finally we shall also study a Wentzel–Kramers–Brillouin (WKB) type semiclassical approximation in section 5 and compare its numerical solution to the solution of the full problem. This section is mainly included since it gives a more transparent description of the Bloch transformation, at least in cases where a semiclassical approximation is justified.

2. The emergence of Bloch bands. First, let us introduce some notation used throughout this paper and recall some basic definitions used when dealing with periodic Schrödinger operators [2, 6, 31, 32].

With V_Γ obeying (1.2) we have the following:

- The fundamental domain of our lattice $\Gamma = 2\pi\mathbb{Z}$ is $\mathcal{C} = (0, 2\pi)$.
- The [illegible] Γ^* can then be defined as the set of all wave numbers $k \in \mathbb{R}$, for which plane waves of the form $\exp(\mathrm{i}kx)$ have the same periodicity as the potential V_Γ. This yields $\Gamma^* = \mathbb{Z}$ in our case.
- The fundamental domain of the dual lattice, i.e., the (first) [illegible], $\mathcal{B} = \mathcal{C}^*$ is the set of all $k \in \mathbb{R}$ closer to zero than to any other dual lattice point. In our case, that is $\mathcal{B} = \left(-\frac{1}{2}, \frac{1}{2}\right)$.

2.1. Recapitulation of Bloch's decomposition method. One of our main points in all that follows is that the dynamical behavior of (1.1) is mainly governed by the periodic part of the Hamiltonian, in particular for $\varepsilon \ll 1$. Thus it will be important to study its spectral properties. To this end consider the periodic [illegible] (where for the moment we set $y = x/\varepsilon$ for simplicity)

$$H = -\frac{1}{2}\,\partial_{yy} + V_\Gamma(y)\,, \tag{2.1}$$

which we will regard here only on $L^2(\mathcal{C})$. This is possible due to the periodicity of V_Γ, which allows us then to cover all of $\mathbb{R}$ by simple translations. More precisely, for $k \in \overline{\mathcal{B}} = \left[-\frac{1}{2}, \frac{1}{2}\right]$ we equip the operator H with the following [illegible] boundary conditions:

$$\begin{cases} \psi(t, y + 2\pi) = \mathrm{e}^{2\mathrm{i}k\pi}\psi(t, y) \quad \forall\, y \in \mathbb{R},\ k \in \overline{\mathcal{B}}, \\ \partial_y\psi(t, y + 2\pi) = \mathrm{e}^{2\mathrm{i}k\pi}\partial_y\psi(t, y) \quad \forall\, y \in \mathbb{R},\ k \in \overline{\mathcal{B}}. \end{cases} \tag{2.2}$$

It is well known [32] that under very mild conditions on V_Γ, the operator H admits a complete set of eigenfunctions $\varphi_m(y, k), m \in \mathbb{N}$, providing, for each fixed $k \in \overline{\mathcal{B}}$, an orthonormal basis in $L^2(\mathcal{C})$. Correspondingly there exists a countable family of real-valued eigenvalues which can be ordered according to $E_1(k) \le E_2(k) \le \cdots \le E_m(k) \le \ldots$, $m \in \mathbb{N}$, including the respective multiplicity. The set $\{E_m(k) \,|\, k \in \mathcal{B}\} \subset \mathbb{R}$ is called the mth [illegible] of the operator H and the eigenfunctions $\varphi_m(\cdot, k)$ are usually called the [illegible]. (In the following the index $m \in \mathbb{N}$ will [illegible] denote the [illegible].) Concerning the dependence on $k \in \mathcal{B}$, it has been shown [32] that for any $m \in \mathbb{N}$ there exists a closed subset $\mathcal{A} \subset \mathcal{B}$ such that $E_m(k)$ is analytic and $\varphi_m(\cdot, k)$ can be chosen to be the real analytic function for all $k \in \overline{\mathcal{B}}\backslash\mathcal{A}$. Moreover,

$$E_{m-1} < E_m(k) < E_{m+1}(k) \quad \forall\, k \in \overline{\mathcal{B}}\backslash\mathcal{A}. \tag{2.3}$$

If this condition indeed holds for all $k \in \mathcal{B}$, then $E_m(k)$ is called an [illegible] [31]. Moreover, it is known that

$$\operatorname{meas}\mathcal{A} = \operatorname{meas}\{k \in \overline{\mathcal{B}} \mid E_n(k) = E_m(k),\ n \neq m\} = 0. \tag{2.4}$$

In this set of measure zero, one encounters so-called [illegible]. Note that due to (2.2) we can rewrite $\varphi_m(y,k)$ as

$$\varphi_m(y,k) = \mathrm{e}^{\mathrm{i}ky}\chi_m(y,k) \quad \forall\, m \in \mathbb{N} \tag{2.5}$$

for some 2π-periodic function $\chi_m(\cdot,k)$. In terms of $\chi_m(y,k)$ the [illegible] reads

$$\begin{cases} H(k)\chi_m(y,k) = E_m(k)\chi_m(y,k), \\ \chi_m(y+2\pi,k) = \chi_m(y,k) \quad \forall\, k \in \mathcal{B}, \end{cases} \tag{2.6}$$

where $H(k)$ denotes the shifted Hamiltonian

$$H(k) := \frac{1}{2}(-\mathrm{i}\partial_y + k)^2 + V_\Gamma(y). \tag{2.7}$$

Let us now introduce the so-called [illegible] $\mathcal{T}$ of some function $\psi(t,\cdot) \in L^2(\mathbb{R})$ for any fixed $t \in \mathbb{R}$, as can be found in, e.g., [29, 31]. (Some other variants of this transformation can also be found in the literature.) The Bloch transformation $\mathcal{T}$ is just the regular Fourier transform $\mathcal{F}$ on the factor $\ell^2(\Gamma)$ followed by a multiplication with $\mathrm{e}^{-\mathrm{i}yk}$, i.e.,

$$(\mathcal{T}\psi)(t,k,y) := \sum_{\gamma\in\mathbb{Z}} \psi(t, y+2\pi\gamma)\,\mathrm{e}^{-\mathrm{i}k(2\pi\gamma+y)}, \quad y \in \mathcal{C},\ k \in \mathcal{B}. \tag{2.8}$$

It is then easy to see that

$$\mathcal{T}H\mathcal{T}^{-1} = H(k), \tag{2.9}$$

which (upon rescaling) provides a link between the eigenvalue problem (2.6) and the periodic part of our Schrödinger equation acting on $\psi(t,\cdot)$.

Most importantly, though, the Bloch transformation allows us to decompose our original Hilbert space $\mathcal{H} = L^2(\mathbb{R})$ into a direct sum of so-called [illegible], i.e.,

$$L^2(\mathbb{R}) = \bigoplus_{m=1}^{\infty} \mathcal{H}_m, \quad \mathcal{H}_m := \left\{ \psi_m(t,y) = \int_{\mathcal{B}} f(t,k)\,\varphi_m(y,k)\,\mathrm{d}k,\ f(t,\cdot) \in L^2(\mathcal{B}) \right\} \tag{2.10}$$

for any fixed $t \in \mathbb{R}$. This is the well-known [illegible], which implies that

$$\forall\, \psi(t,\cdot) \in L^2(\mathbb{R}): \quad \psi(t,y) = \sum_{m\in\mathbb{N}} \psi_m(t,y), \quad \psi_m \in \mathcal{H}_m. \tag{2.11}$$

The corresponding projection of $\psi(t)$ onto the mth band space is thereby given as

$$\psi_m(t,y) \equiv (\mathbb{P}_m\psi)(t,y) = \int_{\mathcal{B}} \left(\int_{\mathbb{R}} \psi(t,\zeta)\overline{\varphi}_m(\zeta,k)\,\mathrm{d}\zeta \right) \varphi_m(y,k)\,\mathrm{d}k, \tag{2.12}$$

and we consequently denote by

$$C_m(t,k) := \int_{\mathbb{R}} \psi(t,\zeta)\overline{\varphi}_m(\zeta,k)\,\mathrm{d}\zeta \tag{2.13}$$

the coefficients of the Bloch decomposition. For a complete description and a rigorous mathematical proof of this decomposition we refer to, e.g., [30, Chapter XI]. Here it is important to note only that the Bloch transformation allows us to obtain a spectral decomposition of our periodic Hamiltonians H upon solving the eigenvalue problem (2.6). Roughly speaking $\mathcal{T}$ can be seen as some sort of Fourier transform adapted to the inclusion of periodic coefficients (potentials).

This consequently implies that if $U(x) \equiv 0$, we can indeed Bloch transform the whole evolution problem (1.1) and decompose it into the corresponding band spaces $\mathcal{H}_m$, i.e., we gain some sort of "diagonalization" for our evolution problem. In this case each $\psi_m(t,\cdot) \in \mathcal{H}_m$ then evolves according to the newly obtained PDE

$$\begin{cases} \mathrm{i}\varepsilon\partial_t\psi_m = E_m(-\mathrm{i}\partial_y)\psi_m, & y \in \mathbb{R},\ t \in \mathbb{R}, \\ \psi_m\big|_{t=0} = (\mathbb{P}_m\psi_{\mathrm{in}})(y). \end{cases} \tag{2.14}$$

Here $E_m(-\mathrm{i}\partial_y)$ denotes the pseudodifferential operator corresponding to the (Fourier) symbol $E_m(k)$ (cf. [16, 29, 31]). The above given evolution equation comprises a rigorous justification of Peierls substitution. Moreover, (2.14) is easily solved invoking the standard Fourier transformation $\mathcal{F}$ on $L^2(\mathbb{R})$, which yields

$$\psi_m(t,y) = \mathcal{F}^{-1}(\mathrm{e}^{-\mathrm{i}E_m(k)t/\varepsilon}(\mathcal{F}(\mathbb{P}_m\psi_{\mathrm{in}}))(k)). \tag{2.15}$$

Here the energy band $E_m(k)$ is understood to be periodically extended on all of $\mathbb{R}$. Note that the relation

$$C_m(t,k) = \mathrm{e}^{-\mathrm{i}E_m(k)t/\varepsilon}C_m(0,k) \tag{2.16}$$

holds, as can be shown by a lengthy but straightforward calculation.

Of course if $U(x) \neq 0$ (the nonperiodic part of the potential), the time evolution (1.1) in general [illegible] all band spaces $\mathcal{H}_m$, i.e., we can no longer hope to be able to diagonalize the whole Hamiltonian operator (which now also involves nonperiodic coefficients). On the other hand, since $U(x) = U(\varepsilon y)$ varies only slowly on the fast (periodic) scale $y = x/\varepsilon$, one might hope that even if $U(x) \neq 0$, the [illegible]

$$\begin{cases} \mathrm{i}\varepsilon\partial_t\psi_m^{\mathrm{eff}} = E_m(-\mathrm{i}\partial_y)\psi_m^{\mathrm{eff}} + U(\varepsilon y)\psi_m^{\mathrm{eff}}, & y \in \mathbb{R},\ t \in \mathbb{R}, \\ \psi_m^{\mathrm{eff}}\big|_{t=0} = (\mathbb{P}_m\psi_{\mathrm{in}})(y) \end{cases} \tag{2.17}$$

holds true, at least approximately for small $\varepsilon \ll 1$. In other words, we expect the slowly varying external potential to be almost constant on the lattice scale and thus yielding only a small perturbation of the band structure determined via (2.1). Indeed this is the case as has been rigorously proved in [11, 20, 29], using different analytical approaches (for a broader overview, see [31] and the references given therein), where it is shown that

$$\sup_{t\in I} \big\|(\mathbb{P}_m\psi)(t) - \psi_m^{\mathrm{eff}}(t)\big\|_{L^2(\mathbb{R})} \leq \mathcal{O}(\varepsilon) \tag{2.18}$$

holds true for any finite time-interval $I \subset \mathbb{R}$. Here $\psi(t)$ is the solution of the full Schrödinger equation (1.1) and $\psi_m^{\mathrm{eff}}(t)$ is the solution of the effective model (2.17) for the mth energy band. To this end one has to assume that the mth energy band is [illegible] from the rest of the spectrum though. If this is not the case, energy transfer of order $\mathcal{O}(1)$ can occur at band crossings, the so-called Landau–Zener phenomena. In all that follows, the mth band is assumed to be isolated.

2.2. Numerical computation of the Bloch bands. As a preparatory step for our algorithm we shall first calculate the Bloch energy bands $E_m(k)$ numerically as follows. Analogously to [18, 26], we consider the potential $V_\Gamma \in C^1(\mathbb{R})$ and expand it in its Fourier series, i.e.,

$$V_\Gamma(y) = \sum_{\lambda\in\mathbb{Z}} \widehat{V}(\lambda)\,\mathrm{e}^{\mathrm{i}\lambda y}, \quad \widehat{V}(\lambda) = \frac{1}{2\pi}\int_0^{2\pi} V_\Gamma(y)\,\mathrm{e}^{-\mathrm{i}\lambda y}\,\mathrm{d}y. \tag{2.19}$$

Likewise, we expand any Bloch eigenfunctions $\chi_m(\cdot,k)$ in its respective Fourier series

$$\chi_m(y,k) = \sum_{\lambda\in\mathbb{Z}} \widehat{\chi}_m(\lambda,k)\,\mathrm{e}^{\mathrm{i}\lambda y}, \quad \widehat{\chi}_m(\lambda,k) = \frac{1}{2\pi}\int_0^{2\pi} \chi_m(y,k)\,\mathrm{e}^{-\mathrm{i}\lambda y}\,\mathrm{d}y. \tag{2.20}$$

(The latter should be confused with the so-called , which are given as the Fourier transformation of φ_m with respect to $k \in \mathcal{B}$.) Clearly the Fourier approximation of V_Γ, and thus also the one of χ_m, depends on the regularity of V_Γ. If $V_\Gamma \in C^\infty(\mathbb{R})$, the corresponding Fourier coefficients $\widehat{V}(\lambda)$ decay faster than any power, as $\lambda \to \pm\infty$, and thus we only need to take into account a few coefficients in this case.

For $\lambda \in \{-\Lambda, \dots, \Lambda-1\} \subset \mathbb{Z}$, we consequently aim to approximate the Sturm–Liouville problem (2.6) by the algebraic eigenvalue problem

$$\mathbf{H}(k)\begin{pmatrix} \widehat{\chi}_m(-\Lambda) \\ \widehat{\chi}_m(1-\Lambda) \\ \vdots \\ \widehat{\chi}_m(\Lambda-1) \end{pmatrix} = E_m(k)\begin{pmatrix} \widehat{\chi}_m(-\Lambda) \\ \widehat{\chi}_m(1-\Lambda) \\ \vdots \\ \widehat{\chi}_m(\Lambda-1) \end{pmatrix}, \tag{2.21}$$

where the $2\Lambda \times 2\Lambda$ matrix $\mathbf{H}(k)$ is given by

$$\mathbf{H}(k) = \begin{pmatrix} \widehat{V}(0) + \frac{(k-\Lambda)^2}{2} & \widehat{V}(-1) & \cdots & \widehat{V}(1-2\Lambda) \\ \widehat{V}(1) & \widehat{V}(0) + \frac{(k-\Lambda+1)^2}{2} & \cdots & \widehat{V}(2-2\Lambda) \\ \vdots & \vdots & \ddots & \vdots \\ \widehat{V}(2\Lambda-1) & \widehat{V}(2\Lambda-2) & \cdots & \widehat{V}(0) + \frac{(k+\Lambda-1)^2}{2} \end{pmatrix}. \tag{2.22}$$

The above given matrix $\mathbf{H}(k)$ comprises 2Λ eigenvalues. Clearly, this number has to be large enough such that all the eigenvalues $E_m(k)$ which we need to use in our simulations below are counted, i.e., we need $m \le 2\Lambda$. The numerical cost for this algebraic problem is about $\mathcal{O}(\Lambda^3)$; cf. [22]. Note, however, that this is the most expensive case, which becomes considerably smaller if one exploits possible symmetries within the potential V_Γ; cf. Example 4.1 below (see also [9, 26, 21, 34]). In any case the number Λ is of the spatial grid, and thus the numerical costs of this eigenvalue problem are almost negligible compared to those spend in the evolutionary algorithms below. The approximate numerical computations of the Bloch bands $E_m(k)$ can be seen as a preprocessing, to be done only once and remain unchanged as time evolves.

2.1. Accurate computations of the energy bands needed in practical applications, i.e., in more than one spatial dimension and for different kinds of (composite) material, become a highly nontrivial task. Nowadays, though, there already exists a huge amount of numerical data comprising the energy band structure of the most important materials used in, e.g., the design of semiconductor devices;

cf. [15, 25, 27]. We note that some of these data is available online via the websites http://www.research.ibm.com/DAMOCLES/home.html, or http://cmt.dur.ac.uk/sjc, and also http://cms.mpi.univie.ac.at/vasp/vasp/vasp.html. In the context of photonic crystals the situation is similar [21]. Thus, relying on such data one can in principle avoid the above given eigenvalue computations (and its generalizations to more dimensions) completely. To this end, one should also note that given the energy bands $E_m(k)$, we do not need any knowledge about V_Γ in order to solve (1.1) numerically; cf. the algorithm described below.

3. Bloch decomposition–based algorithm vs. time-splitting spectral methods. For the convenience of computations, we shall consider (1.1) on a bounded domain $\mathcal{D}$, say on the interval $\mathcal{D} = [-\kappa_1, \kappa_2]$, for some large enough $\kappa_1, \kappa_2 > 0$. Moreover, we shall equip $\mathcal{D}$ with periodic boundary conditions. However, this periodic computational domain $\mathcal{D}$ should not be confused with the periodic structure induced by the lattice potential. Without loss of any generality, we assume that $\mathcal{D} = [0, 2\pi]$.

For practical reasons we shall now introduce, for any fixed $t \in \mathbb{R}$, a new unitary transformation of $\psi(t, \cdot) \in L^2(\mathbb{R})$,

$$\widetilde{\psi}(t, y, k) := \sum_{\gamma \in \mathbb{Z}} \psi(t, \varepsilon(y + 2\pi\gamma))\, \mathrm{e}^{-\mathrm{i}2\pi k\gamma}, \quad y \in \mathcal{C},\ k \in \mathcal{B}, \tag{3.1}$$

which has the properties that $\widetilde{\psi}$ is quasi-periodic with respect to $y \in \Gamma$ and periodic with respect to $k \in \Gamma^*$, i.e.,

$$\widetilde{\psi}(t, y + 2\pi, k) = \mathrm{e}^{\mathrm{i}2\pi k}\, \widetilde{\psi}(t, y, k), \quad \widetilde{\psi}(t, y, k + 1) = \widetilde{\psi}(t, y, k). \tag{3.2}$$

One should note that $\widetilde{\psi}$ is not the standard Bloch transform of ψ, i.e., $\mathcal{T}$, as defined in (2.8), but it is indeed closely related to it, for $\varepsilon = 1$, via

$$(\mathcal{T}\psi)(t, y, k) = \widetilde{\psi}(t, y, k)\mathrm{e}^{-\mathrm{i}yk}, \quad k \in \mathcal{B}. \tag{3.3}$$

Furthermore, we have the inversion formula

$$\psi(t, \varepsilon(y + 2\pi\gamma)) = \int_{\mathcal{B}} \widetilde{\psi}(t, y, k)\mathrm{e}^{\mathrm{i}2\pi k\gamma} dk, \tag{3.4}$$

which is again very similar to the one of the standard Bloch transformation [31]. The main advantage in using $\widetilde{\psi}$ instead of $\mathcal{T}\psi$ itself is that we can rely on a standard fast Fourier transform (FFT) in the numerical algorithm below. If one aims to use $\mathcal{T}\psi$ directly, one would be forced to modify a given FFT code accordingly. A straightforward computation then shows that

$$C_m(t, k) = \int_{\mathcal{C}} \widetilde{\psi}(t, \zeta, k)\overline{\varphi}_m(\zeta, k)\, \mathrm{d}\zeta, \tag{3.5}$$

where $C_m(t, k)$ is the Bloch coefficient, defined in (2.13).

In what follows, let the time step be $\Delta t = T/N$ for some $N \in \mathbb{N}$, $T > 0$. Suppose that there are $L \in \mathbb{N}$ lattice cells within the computational domain $\mathcal{D} = [0, 2\pi]$. In this domain, the wave function ψ is numerically computed at $L \times R$ grid points for some $R \in \mathbb{N}$. In other words we assume that there are R grid points in each lattice cell, which yields the discretization

$$\begin{cases} k_\ell = -\dfrac{1}{2} + \dfrac{\ell - 1}{L}, & \text{where } \ell = \{1, \dots, L\} \subset \mathbb{N}, \\ y_r = \dfrac{2\pi(r - 1)}{R}, & \text{where } r = \{1, \dots, R\} \subset \mathbb{N}, \end{cases} \tag{3.6}$$

and thus we finally evaluate $\psi^n = \psi(t_n)$ at the grid points $x = \varepsilon(2\pi\gamma + y)$, i.e.,

$$x_{\ell,r} = \varepsilon(2\pi(\ell - 1) + y_r). \tag{3.7}$$

We remark that in our numerical computations we can use $R \ll L$, whenever $\varepsilon \ll 1$, i.e., we use only a few grid points within each cell. Now we shall describe precisely the Bloch decomposition–based algorithm used to solve (1.1).

3.1. The Bloch decomposition–based algorithm (BD). Suppose that at time t_n we are given $\psi(t_n, x_{\ell,r}) \approx \psi^n_{\ell,r}$. Then $\psi^{n+1}_{\ell,r}$, i.e., the solution at the (next) time step $t_{n+1} = t_n + \Delta t$, is obtained as follows.

, , 1. First, we solve the equation

$$\mathrm{i}\varepsilon\partial_t\psi = -\frac{\varepsilon^2}{2}\,\partial_{xx}\psi + V_\Gamma\left(\frac{x}{\varepsilon}\right)\psi \tag{3.8}$$

on a fixed time-interval Δt. To this end we shall heavily use the Bloch decomposition method; see below.

, , 2. In a second step, solve the ordinary differential equation (ODE)

$$\mathrm{i}\varepsilon\partial_t\psi = U(x)\psi \tag{3.9}$$

on the same time-interval, where the solution obtained in Step 1 serves as the initial condition for Step 2. We easily obtain the exact solution for this linear ODE by

$$\psi(t, x) = \psi(0, x)\,\mathrm{e}^{-\mathrm{i}U(x)t/\varepsilon}. \tag{3.10}$$

, , , 3.1. Clearly, the algorithm given above is first order in time. But we could easily obtain a second order scheme also by the Strang splitting method, which means that we use Step 1 with time step $\triangle t/2$, then Step 2 with time step $\triangle t$, and finally integrate Step 1 again with $\triangle t/2$. Note that in both cases the scheme conserves the particle density $\rho(t, x) := |\psi(t, x)|^2$ also on the fully discrete level.

Indeed Step 1 consists of several intermediate steps which we shall present as follows.

, , 1.1. We first compute $\widetilde{\psi}$ at time t^n by

$$\widetilde{\psi}^n_{\ell,r} = \sum_{j=1}^{L} \psi^n_{j,r}\,\mathrm{e}^{-\mathrm{i}k_\ell\cdot x_{j,1}}. \tag{3.11}$$

, , 1.2. Next, we compute the mth band Bloch coefficient $C_m(t, k)$, at time t^n, via (3.5), i.e.,

$$\begin{aligned} C_m(t_n, k_\ell) \approx C^n_{m,\ell} &= \frac{2\pi}{R}\sum_{r=1}^{R} \widetilde{\psi}^n_{\ell,r}\overline{\chi_m}(y_r, k_\ell)\,\mathrm{e}^{-\mathrm{i}k_\ell y_r} \\ &\approx \frac{2\pi}{R}\sum_{r=1}^{R} \widetilde{\psi}^n_{\ell,r}\sum_{\lambda=-R/2}^{R/2-1} \overline{\widehat{\chi}_m}(\lambda, k_\ell)\,\mathrm{e}^{-\mathrm{i}(k_\ell+\lambda)y_r}, \end{aligned} \tag{3.12}$$

where for the second line we simply inserted the Fourier expansion of χ_m, given in (2.20). Note that in total we have R Fourier coefficients for χ_m. Clearly this implies that we need $\Lambda > R/2$ to hold, where Λ is the number of Fourier modes

required in the numerical approximation of Bloch's eigenvalue problem as discussed in section 2.2. Here we take only the R lowest frequency Fourier coefficients.

, , 1.3. The obtained Bloch coefficients are then evolved up to the time t^{n+1}, according to the explicit solution formula (2.15), taking into account (2.16). This yields

$$C_{m,\ell}^{n+1} = C_{m,\ell}^{n}\, \mathrm{e}^{-\mathrm{i}E_m(k_\ell)\Delta t/\varepsilon}. \tag{3.13}$$

, , 1.4. From here, we consequently compute $\widetilde{\psi}$ at the new time t^{n+1} by summing up all band contributions and using the analytical formulas (2.12) and (2.13), i.e.,

$$\widetilde{\psi}_{\ell,r}^{n+1} = \sum_{m=1}^{M} (\mathbb{P}_m \widetilde{\psi})_{\ell,r}^{n+1} \approx \sum_{m=1}^{M} C_{m,\ell}^{n+1} \sum_{\lambda=-R/2}^{R/2-1} \widehat{\chi}_m(\lambda, k_\ell)\, \mathrm{e}^{\mathrm{i}(k_\ell+\lambda)y_r}. \tag{3.14}$$

, , 1.5. Finally we numerically perform the inverse transformation to (3.1), i.e., we compute $\psi_{\ell,r}^{n+1}$ from $\widetilde{\psi}_{\ell,r}^{n+1}$. Thus from (3.4), we get

$$\psi_{\ell,r}^{n+1} = \frac{1}{L} \sum_{j=1}^{L} \widetilde{\psi}_{j,r}^{n+1}\, \mathrm{e}^{\mathrm{i}k_j x_{\ell,1}}. \tag{3.15}$$

Note that in the BD algorithm, the main numerical costs are introduced via the FFT in Steps 1.1 and 1.5. This also implies that on the same spatial grid, the numerical costs of our Bloch transform–based algorithm is of the same order as the classical time-splitting spectral method below. Moreover, we want to stress the fact that if there is no external potential, i.e., $U(x) \equiv 0$, then the above given algorithm numerically computes the , , , solution of the evolutionary problem (1.1), which can be seen analogous to a standard spectral method, adapted to periodic potentials. In particular this fact allows us to solve the Schrödinger equation (1.1) for very long time steps, even if ε is small (see the results given below). Moreover, one should note that a possible lack of regularity in V_Γ requires numerical care only when approximating (2.6) by the algebraic problem (2.21). In particular, V_Γ itself does not enter in the time evolution but only $E_m(k)$.

3.2. A simple time-splitting spectral method (TS). Ignoring for a moment the additional structure provided by the periodic potential V_Γ, one might wish to solve (1.1) by using a classical time-splitting spectral scheme. Such schemes already proved to be successful in similar circumstances; see, e.g., [3, 4, 18, 23]. For the purpose of a detailed comparison, we present this method here.

, , 1. In the first step we solve the equation

$$\mathrm{i}\varepsilon \partial_t \psi = -\frac{\varepsilon^2}{2}\, \partial_{xx}\psi \tag{3.16}$$

on a fixed time-interval Δt, relying on the pseudospectral method.

, , 2. Then, in a second step, we solve the ODE

$$\mathrm{i}\varepsilon \partial_t \psi = \left(V_\Gamma\left(\frac{x}{\varepsilon}\right) + U(x) \right) \psi \tag{3.17}$$

on the same time-interval, where the solution obtained in Step 1 serves as the initial condition for Step 2. Again it is easily seen that such a scheme conserves the particle

density. It is clear, however, that due to the inclusion of $V_\Gamma\left(\frac{x}{\varepsilon}\right)$, the exact solution of (3.17),

$$\psi(t,x) = \psi_{\rm in}(x)\,\mathrm{e}^{-\mathrm{i}(V_\Gamma(x/\varepsilon)+U(x))t/\varepsilon}, \tag{3.18}$$

involves high oscillations on [illegible] length- and time-scales as $\varepsilon \to 0$ (which one has to resolve), in contrast to (3.10), where only t/ε-oscillations are present. In our BD algorithm, we compute the dominant effects from dispersion and periodic lattice potential in one step, and treat the nonperiodic potential as a perturbation. Because the split-step communicator error between the periodic and nonperiodic parts is relatively small, the step size can be chosen considerably larger than for the SP algorithm.

[illegible] 3.2. Clearly, if there is no lattice potential, i.e., $V_\Gamma(y) \equiv 0$, the BD algorithm simplifies to the described time-splitting method TS. Moreover, a second order scheme (based on the Strang splitting algorithm) can be obtained analogously to the one described above (see Remark 3.1), and a comparison of these second order schemes gives similar results to those shown in the following.

For the BD algorithm, the complexities of Steps 1.1 and 1.5 are $\mathcal{O}(RL\log(L))$, the complexities of Steps 1.2 and 1.4 are $\mathcal{O}(MLR\log(R))$, and for Step 1.3 we have $\mathcal{O}(ML)$. Also the complexity of the eigenvalue problem (2.21) is $\mathcal{O}(\Lambda^3)$. However, since Λ (or R) is independent of ε and since we only need to solve the eigenvalue problem (2.21) once in a preparatory step, the computation costs for this problem are negligible. On the other hand, for the TS algorithm, the complexities of Steps 1 and 2 are $\mathcal{O}(RL\log(RL))$ and $\mathcal{O}(RL)$, respectively. As M and R are independent of ε, we can use $R \ll L$ and $M \ll L$ whenever $\varepsilon \ll 1$. Finally the complexities of the BD and TS algorithms in each time step are comparable.

4. Numerical experiments. In this section, we shall use several numerical examples to show the efficiency of our algorithm. We shall choose for (1.1) initial data $\psi_{\rm in} \in \mathcal{S}(\mathbb{R})$ of the form

$$\psi_{\rm in}(x) = \left(\frac{10}{\pi}\right)^{1/4} \mathrm{e}^{-5(x-\pi)^2}. \tag{4.1}$$

Let us perform a decomposition of $\psi_{\rm in}$ in terms of the Bloch bands, and take a summation of the first $m = 1, \ldots, M_0$ energy bands for some finite (cut-off) number $M_0 \in \mathbb{N}$. A picture of the corresponding band densities $\rho_m^\varepsilon := |\mathbb{P}_m^\varepsilon \psi_{\rm in}|^2$ is given in Figure 1 for $m = 1, \ldots, 4$. Since $\psi_{\rm in}$ is smooth we expect that only very few bands have to be taken into account in the Bloch decomposition. Indeed we observe that the amount of mass corresponding to $\mathbb{P}_m^\varepsilon \psi_{\rm in}$, i.e., the mass concentration in each Bloch band, decays rapidly as $m \to \infty$; see Table 1. In other words, the number M_0 is essentially determined by the regularity of $\psi_{\rm in}$ in each cell. Note that M_0 is independent of ε.

To compute the evolution of these initial data we shall take into account $M \geq M_0$ bands. Note that only in cases where $U(x) \equiv 0$ can one take M to be identical to M_0, the initial band cut-off. The reason is that if $U(x)$ is nonzero, Step 2 in the BD algorithm given above mixes all bands. In particular all the $\psi_m(t)$ are no longer orthogonal to each other. Roughly speaking, however, if ε is very small, all band spaces $\mathcal{H}_m$ remain "almost orthogonal" and thus the mass within each Bloch band, i.e., $\mathrm{M}_m^\varepsilon(t) := \left\| \mathbb{P}_m^\varepsilon \psi(t) \right\|^2_{L^2(\mathbb{R})}$, is "almost conserved." More precisely it is conserved up to errors $\mathcal{O}(\varepsilon)$ on time-scales $\mathcal{O}(1)$. Thus, by checking mass conservation after each

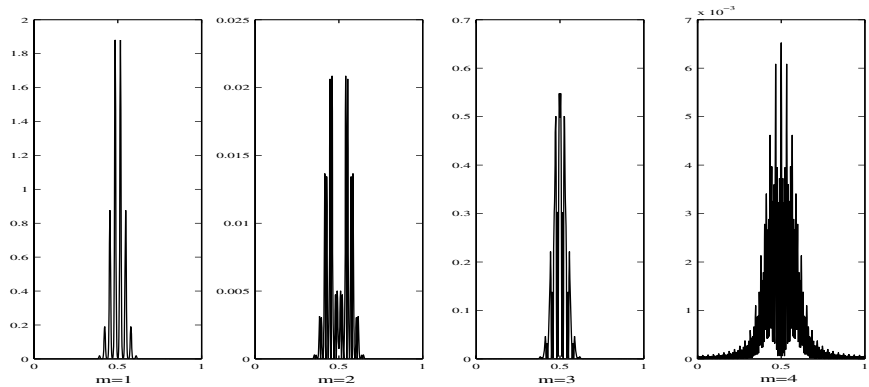


FIG. 1. $|\mathbb{P}^\varepsilon_m \psi_{\rm in}|^2$, $m = 1, \dots, 4$, *for* $\varepsilon = \frac{1}{32}$.

TABLE 1
The values of $\mathrm{M}^\varepsilon_m := \left\| \mathbb{P}^\varepsilon_m \psi_{\rm in} \right\|_{L^2(\mathbb{R})}$ *for* $\varepsilon = \frac{1}{32}$.

m	1	2	3	4
M^ε_m	7.91E − 1	1.11E − 1	5.92E − 1	8.80E − 2
m	5	6	7	8
M^ε_m	8.67E − 2	2.81E − 3	2.80E − 3	4.98E − 5

time step one gets a rather reliable measure on the amount of mixing of the bands. In other words if the mass conservation after some time steps gets worse, one has to take into account more bands to proceed.

We find numerically that the use of $M = M_0 \approx 8$ bands already yields satisfactory results for $\varepsilon = \frac{1}{32}$. In the following, though, we shall even compute $M = 32$ energy bands, which is by far sufficient for our purposes (even if $\varepsilon = \frac{1}{2}$). Note that the number of required bands M depends on the regularity properties of $U(x)$, as well as on the considered time-scales (which might be even longer than $\mathcal{O}(1)$, the case considered here). This approximation problem is more or less analogous to the one appearing in spectral schemes for PDEs with nonsmooth coefficients.

Concerning slowly varying, external potentials U, we shall choose, on the one hand, smooth functions which are either of the form

$$U(x) = \mathcal{E}x, \tag{4.2}$$

modeling a constant (electric) force field $\mathcal{E} \in \mathbb{R}$, or given by a [illegible]–type potential

$$U(x) = |x - \pi|^2. \tag{4.3}$$

On the other hand, we shall also consider the case of an external (nonsmooth) [illegible], i.e.,

$$U(x) = \begin{cases} 1, & x \in \left[\frac{\pi}{2}, \frac{3\pi}{2}\right], \\ 0 & \text{else.} \end{cases} \tag{4.4}$$

Within the setting described above, we shall focus on two particular choices for the lattice potential, namely, the following two examples.

EXAMPLE 4.1 (Mathieu model). The so-called [illegible], i.e.,

$$V_\Gamma(x) = \cos(x), \tag{4.5}$$

as already considered in [18]. (For applications in solid state physics this is rather unrealistic; however, it fits quite well with experiments on Bose–Einstein condensates in optical lattices.) In this case all Fourier coefficients $\widehat{V}(\lambda)$ appearing in (2.19) are zero, except for $\widehat{V}(\pm 1) = \frac{1}{2}$, and thus $\mathbf{H}(k)$, given in (2.22), simplifies to a tridiagonal matrix.

EXAMPLE 4.2 (Kronig–Penney model). The so-called [illegible], i.e.,

$$V_\Gamma(x) = 1 - \sum_{\gamma\in\mathbb{Z}} \mathbf{1}_{x\in\left[\frac{\pi}{2}+2\pi\gamma,\,\frac{3\pi}{2}+2\pi\gamma\right]}, \tag{4.6}$$

where $\mathbf{1}_\Omega$ denotes the characteristic function of a set $\Omega \subset \mathbb{R}$. In contrast to Mathieu's model this case comprises a nonsmooth lattice potential. The corresponding Bloch eigenvalue problem is known to be explicitly solvable (see, e.g., [18]).

In order to compare the different numerical algorithms we denote by $\psi^{\mathrm{ts}}(t,x)$ the solution gained from the time-splitting spectral method, whereas $\psi^{\mathrm{bd}}(t,x)$ denotes the solution obtained via the new method base on Bloch's decomposition. Both methods will be compared to the "exact" solution $\psi^{\mathrm{ex}}(t,x)$, which is obtained using a very fine spatial grid. We consider the errors

$$\begin{aligned} \Delta_\infty^{\mathrm{bd/ts}}(t) &:= \left\| \psi^{\mathrm{ex}}(t,\cdot) - \psi^{\mathrm{bd/ts}}(t,\cdot)\right\|_{L^\infty(\mathbb{R})}, \\ \Delta_2^{\mathrm{bd/ts}}(t) &:= \left\| \psi^{\mathrm{ex}}(t,\cdot) - \psi^{\mathrm{bd/ts}}(t,\cdot)\right\|_{L^2(\mathbb{R})} \end{aligned} \tag{4.7}$$

between the "exact solution" and the corresponding solutions obtained via the Bloch decomposition–based algorithm, respectively, the classical time splitting spectral method. The numerical experiments are now done in a series of three different settings:

- First we shall study both cases of V_Γ, imposing additionally $U(x) \equiv 0$, i.e., no external potential. The obtained results are given in Table 2, where $\varepsilon = \frac{1}{2}$, $\frac{1}{32}$, and $\frac{1}{1024}$, respectively. In the last case the oscillations are extremely spurious. As discussed before, we can use only one step in time to obtain the numerical solution, because the Bloch decomposition method indeed is "exact" in this case (independently of ε). Thus, even if we would refine the time steps in the BD algorithm, we would not get more accurate approximations. On the other hand, by using the usual time-splitting method, one has to refine the time steps (depending on ε) as well as the mesh size in order to achieve the same accuracy. More precisely we find that $\Delta t = \mathcal{O}(\varepsilon)$, $\Delta x = \mathcal{O}(\varepsilon^\alpha)$, for some $\alpha \geq 1$, is needed when using TS (see also the computations given in [18]). In particular $\alpha > 1$ is required for the case of a nonsmooth lattice potential V_Γ. (Note that if $V_\Gamma = 0$, it is well known that $\Delta x = \mathcal{O}(\varepsilon)$ is sufficient; cf. [3, 4, 23].)
- In a second series of numerical experiments we shall consider only Example 4.1 for the periodic potential but taking into account all three cases of the external potentials U, as given above. In Figures 2–7 and Table 3, we show the obtained numerical results for $\varepsilon = \frac{1}{2}$, and $\varepsilon = \frac{1}{1024}$, respectively. We observe that if $\varepsilon = \mathcal{O}(1)$, the Bloch decomposition method gives almost the same results as the time-splitting spectral method. However, if $\varepsilon \ll 1$, we can achieve quite good accuracy by using the Bloch decomposition method with

TABLE 2
The results of Example 4.1 *with* $U(x) = 0$.

Spatial discretization error test at time $t = 1.0$ for $\varepsilon = 1/2$.
For TS $\triangle t = 0.0001$ and for BD $\triangle t = 1$.

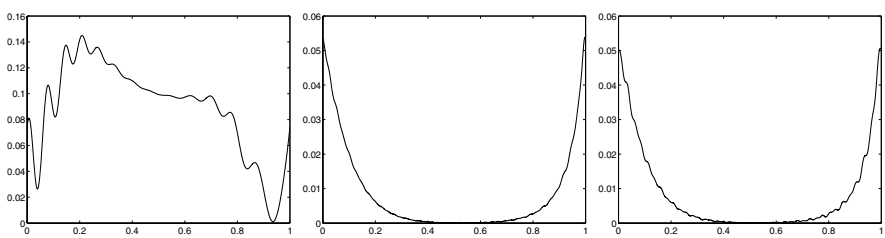
Mesh size $\triangle x/\varepsilon$	1/2	1/4	1/8	1/16
$\left\|\psi^{\rm ts}_{\triangle x,\triangle t}(t,\cdot) - \psi^{\rm ex}(t,\cdot)\right\|_{l^2}$	4.33E-1	2.53E-1	2.80E-2	6.42E-6
Convergence order		0.8	3.2	12.1
$\left\|\psi^{\rm bd}_{\triangle x,\triangle t}(t,\cdot) - \psi^{\rm ex}(t,\cdot)\right\|_{l^2}$	3.01E-1	1.95E-1	1.39E-2	1.17E-6
Convergence order		0.6	3.8	13.5

Spatial discretization error test at time $t = 0.1$ for $\varepsilon = 1/32$.
For TS $\triangle t = 0.00001$ and for BD $\triangle t = 0.1$.

Mesh size $\triangle x/\varepsilon$	1/2	1/4	1/8	1/16
$\left\|\psi^{\rm ts}_{\triangle x,\triangle t}(t,\cdot) - \psi^{\rm ex}(t,\cdot)\right\|_{l^2}$	2.88E-1	1.08E-1	9.63E-4	1.33E-7
Convergence order		1.4	6.8	12.8
$\left\|\psi^{\rm bd}_{\triangle x,\triangle t}(t,\cdot) - \psi^{\rm ex}(t,\cdot)\right\|_{l^2}$	2.53E-1	7.34E-2	8.97E-4	4.95E-10
Convergence order		1.8	6.4	20.8

Spatial discretization error test at time $t = 0.01$ for $\varepsilon = 1/1024$.
For TS $\triangle t = 0.000001$ and for BD $\triangle t = 0.01$.

Mesh size $\triangle x/\varepsilon$	1/2	1/4	1/8	1/16
$\left\|\psi^{\rm ts}_{\triangle x,\triangle t}(t,\cdot) - \psi^{\rm ex}(t,\cdot)\right\|_{l^2}$	5.14E-1	1.94E-1	1.08E-3	6.08E-8
Convergence order		1.4	7.5	14.1
$\left\|\psi^{\rm bd}_{\triangle x,\triangle t}(t,\cdot) - \psi^{\rm ex}(t,\cdot)\right\|_{l^2}$	2.64E-1	6.83E-2	2.29E-4	1.71E-10
Convergence order		2.0	8.2	20.4



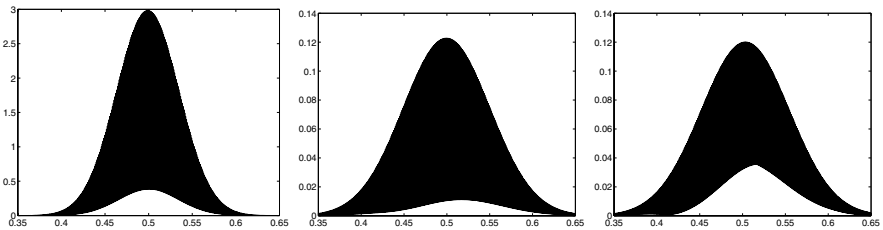
$|\psi^{\rm ex}(t,x)|^2$, $|\psi^{\rm ex}(t,x) - \psi^{\rm ts}(t,x)|$ and $|\psi^{\rm ex}(t,x) - \psi^{\rm bd}(t,x)|$ at $t = 1.0$

FIG. 2. *Numerical results for Example* 4.1 *with* $U(x)$ *given by* (4.2) *and* $\varepsilon = \frac{1}{2}$. *We use* $\triangle t = \frac{1}{100}$, $\triangle x = \frac{1}{64}$ *for the TS and BD methods, and* $\triangle t = \frac{1}{100000}$, $\triangle x = \frac{1}{8192}$ *for the "exact" solution.*

$$\Delta^{\rm ts}_{\infty}(t) = 5.39\text{E}-2,\ \Delta^{\rm bd}_{\infty}(\text{t}) = 5.07\text{E}-2,\ \Delta^{\rm ts}_{2}(\text{t}) = 1.56\text{E}-2,\ \Delta^{\rm bd}_{2}(\text{t}) = 1.51\text{E}-2.$$

$\Delta t = \mathcal{O}(1)$ and $\Delta x = \mathcal{O}(\varepsilon)$. On the other hand, using the standard TS algorithm, we again have to rely on much finer spatial grids and time steps to achieve the same accuracy.

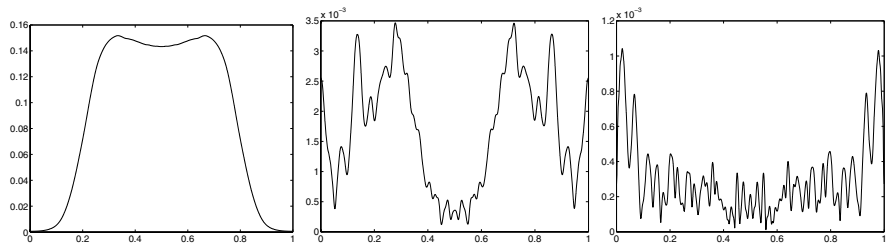
- We finally show the numerical results obtained by combining external fields and a nonsmooth lattice potential given by Example 4.2. As before we include



$|\psi^{\rm ex}(t,x)|^2$, $|\psi^{\rm ex}(t,x)-\psi^{\rm ts}(t,x)|$ and $|\psi^{\rm ex}(t,x)-\psi^{\rm bd}(t,x)|$ at $t=0.1$

FIG. 3. *Numerical results for Example* 4.1 *with* $U(x)$ *given by* (4.2) *and* $\varepsilon=\frac{1}{1024}$. *We use* $\triangle t=\frac{1}{5000}$, $\triangle x=\frac{1}{16384}$ *for the TS method,* $\triangle t=\frac{1}{20}$, $\triangle x=\frac{1}{8192}$ *for the BD method, and* $\triangle t=\frac{1}{100000}$, $\triangle x=\frac{1}{131072}$ *for the "exact" solution.*

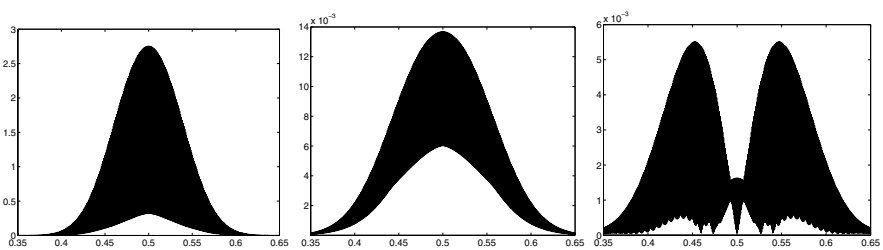
$$\Delta_\infty^{\rm ts}(t)=1.23{\rm E}-1,\ \Delta_\infty^{\rm bd}({\rm t})=1.20{\rm E}-1,\ \Delta_2^{\rm ts}({\rm t})=2.29{\rm E}-2,\ \Delta_2^{\rm bd}({\rm t})=2.31{\rm E}-2.$$



$|\psi^{\rm ex}(t,x)|^2$, $|\psi^{\rm ex}(t,x)-\psi^{\rm ts}(t,x)|$ and $|\psi^{\rm ex}(t,x)-\psi^{\rm bd}(t,x)|$ at $t=1.0$

FIG. 4. *Numerical results for Example* 4.1 *with* $U(x)$ *given by* (4.3) *and* $\varepsilon=\frac{1}{2}$. *We use* $\triangle t=\frac{1}{10}$, $\triangle x=\frac{1}{32}$ *for the TS and BD methods, and* $\triangle t=\frac{1}{100000}$, $\triangle x=\frac{1}{8192}$ *for the "exact" solution.*

$$\Delta_\infty^{\rm ts}(t)=3.47{\rm E}-3,\ \Delta_\infty^{\rm bd}({\rm t})=1.04{\rm E}-3,\ \Delta_2^{\rm ts}({\rm t})=1.96{\rm E}-3,\ \Delta_2^{\rm bd}({\rm t})=3.65{\rm E}-4.$$

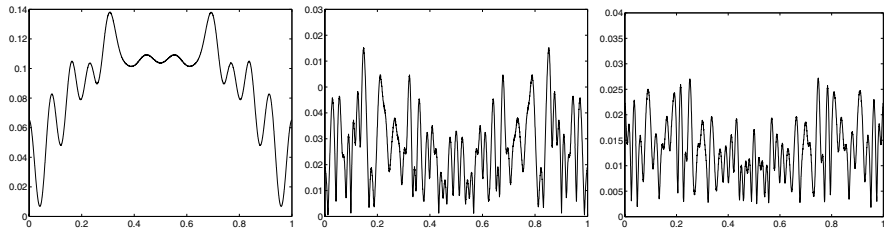


$|\psi^{\rm ex}(t,x)|^2$, $|\psi^{\rm ex}(t,x)-\psi^{\rm ts}(t,x)|$ and $|\psi^{\rm ex}(t,x)-\psi^{\rm bd}(t,x)|$ at $t=0.1$

FIG. 5. *Numerical results for Example* 4.1 *with* $U(x)$ *given by* (4.3) *and* $\varepsilon=\frac{1}{1024}$. *We use* $\triangle t=\frac{1}{10000}$, $\triangle x=\frac{1}{16384}$ *for the TS method,* $\triangle t=\frac{1}{100}$, $\triangle x=\frac{1}{16384}$ *for the BD method, and* $\triangle t=\frac{1}{100000}$, $\triangle x=\frac{1}{131072}$ *for the "exact" solution.*

$$\Delta_\infty^{\rm ts}(t)=1.37{\rm E}-2,\ \Delta_\infty^{\rm bd}({\rm t})=5.52{\rm E}-3,\ \Delta_2^{\rm ts}({\rm t})=2.76{\rm E}-3,\ \Delta_2^{\rm bd}({\rm t})=1.20{\rm E}-3.$$

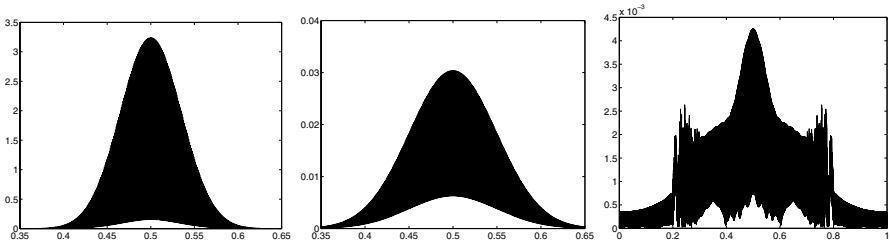
all three cases for the external potential U. The cases $\varepsilon=\frac{1}{2}$ and $\frac{1}{1024}$ are studied and the obtained results are given in Figures 8–13 and Table 4, respectively. We observe that the results of the Bloch decomposition are much better than the time-splitting spectral method, even if $\varepsilon=\frac{1}{2}$. Moreover, as ε gets smaller, the advantages of the Bloch decomposition method are even better visible.



$|\psi^{\rm ex}(t,x)|^2$, $|\psi^{\rm ex}(t,x)-\psi^{\rm ts}(t,x)|$ and $|\psi^{\rm ex}(t,x)-\psi^{\rm bd}(t,x)|$ at $t=1.0$

FIG. 6. *Numerical results for Example* 4.1 *with* $U(x)$ *given by* (4.4) *and* $\varepsilon=\frac{1}{2}$. *We use* $\triangle t=\frac{1}{10}$, $\triangle x=\frac{1}{32}$ *for the TS and BD methods, and* $\triangle t=\frac{1}{100000}$, $\triangle x=\frac{1}{8192}$ *for the "exact" solution.*

$$\Delta_\infty^{\rm ts}(t)=3.26{\rm E}-2,\ \Delta_\infty^{\rm bd}({\rm t})=2.72{\rm E}-2,\ \Delta_2^{\rm ts}({\rm t})=1.51{\rm E}-2,\ \Delta_2^{\rm bd}({\rm t})=1.45{\rm E}-2.$$



$|\psi^{\rm ex}(t,x)|^2$, $|\psi^{\rm ex}(t,x)-\psi^{\rm ts}(t,x)|$ and $|\psi^{\rm ex}(t,x)-\psi^{\rm bd}(t,x)|$ at $t=0.1$

FIG. 7. *Numerical results for Example* 4.1 *with* $U(x)$ *given by* (4.4) *and* $\varepsilon=\frac{1}{1024}$. *We use* $\triangle t=\frac{1}{10000}$, $\triangle x=\frac{1}{16384}$ *for the TS method,* $\triangle t=\frac{1}{10}$, $\triangle x=\frac{1}{8192}$ *for the BD method, and* $\triangle t=\frac{1}{100000}$, $\triangle x=\frac{1}{131072}$ *for the "exact" solution.*

$$\Delta_\infty^{\rm ts}(t)=3.04{\rm E}-2,\ \Delta_\infty^{\rm bd}({\rm t})=4.25{\rm E}-3,\ \Delta_2^{\rm ts}({\rm t})=5.35{\rm E}-3,\ \Delta_2^{\rm bd}({\rm t})=1.21{\rm E}-3.$$

To convince ourselves that only a few Bloch bands contribute to $\|\psi\|_{L^2(\mathbb{R})}$, even after time steps $\mathcal{O}(1)$, we show in Table 5 the numerical values of $\mathrm{M}_m^\varepsilon(t)=\left\|\mathbb{P}_m^\varepsilon\psi(t)\right\|^2_{L^2(\mathbb{R})}$ for $m=1,\dots,8$, corresponding to the solution of Example 4.1 with U given by (4.3).

We also check the conservation of the total (discrete) mass, i.e., $\|\psi(t)\|_{l^2(\mathcal{D})}$. We find that numerically it is of the order 10^{-6} for the smooth lattice potential (4.5) and 10^{-3} for the nonsmooth case (4.6). The latter, however, can be improved by using a refined spatial grid and more time steps.

In summary we find (at least for our one-dimensional computations) that, relying on the new Bloch decomposition–based algorithm, one can use much larger time steps, and sometimes even a coarser spatial grid, to achieve the same accuracy as that for the usual time-splitting spectral method. This is particularly visible in cases where the lattice potential is no longer smooth and $\varepsilon\ll 1$. Indeed in these cases the BD algorithm turns out to be [illegible] than the TS method.

[illegible] 4.1. In view of our results the earlier numerical studies based on TS methods [18, 17, 19] should be taken with some care, in particular when comparing the full Schrödinger solution to the semiclassical approximation beyond caustics.

5. Asymptotic analysis in the semiclassical regime. For completeness we shall also compare the numerical solution of the Schrödinger equation (1.1) with its semiclassical asymptotic description. To this end we shall rely on a multiple scales

TABLE 3
The results of Example 4.1 *with linear external potential* (4.2).

Spatial discretization error test at time $t = 0.1$ for $\varepsilon = 1/2$.
For TS $\triangle t = 0.0001$ and for BD $\triangle t = 0.01$.

Mesh size $\triangle x/\varepsilon$	1/2	1/4	1/8	1/16
$\left\|\psi^{\rm ts}_{\triangle x,\triangle t}(t,\cdot) - \psi^{\rm ex}(t,\cdot)\right\|_{l^2}$	2.73E-1	9.22E-2	5.78E-3	4.73E-6
Convergence order		1.6	4.0	10.3
$\left\|\psi^{\rm bd}_{\triangle x,\triangle t}(t,\cdot) - \psi^{\rm ex}(t,\cdot)\right\|_{l^2}$	3.15E-1	1.55E-1	1.32E-2	3.36E-6
Convergence order		1.0	3.6	11.9

Spatial discretization error test at time $t = 0.01$ for $\varepsilon = 1/1024$.
For TS $\triangle t = 0.00001$ and for BD $\triangle t = 0.001$.

Mesh size $\triangle x/\varepsilon$	1/2	1/4	1/8	1/16
$\left\|\psi^{\rm ts}_{\triangle x,\triangle t}(t,\cdot) - \psi^{\rm ex}(t,\cdot)\right\|_{l^2}$	5.22E-1	1.98E-1	1.53E-2	3.19E-5
Convergence order		1.4	3.7	8.9
$\left\|\psi^{\rm bd}_{\triangle x,\triangle t}(t,\cdot) - \psi^{\rm ex}(t,\cdot)\right\|_{l^2}$	4.71E-1	1.61E-1	9.17E-3	6.08E-6
Convergence order		1.5	4.1	10.6

Temporal discretization error test at $t = 0.1$ for $\varepsilon = 1/2$ and $\triangle x/\varepsilon = 1/128$.

Time step $\triangle t$	1/10	1/20	1/40	1/80
$\left\|\psi^{\rm ts}_{\triangle x,\triangle t}(t,\cdot) - \psi^{\rm ex}(t,\cdot)\right\|_{l^2}$	2.59E-4	6.47E-5	1.62E-5	4.04E-6
Convergence order		2.0	2.0	2.0
$\left\|\psi^{\rm bd}_{\triangle x,\triangle t}(t,\cdot) - \psi^{\rm ex}(t,\cdot)\right\|_{l^2}$	4.86E-5	1.23E-5	3.08E-6	7.60E-7
Convergence order		2.0	2.0	2.0

Temporal discretization error test at $t = 0.01$ for $\varepsilon = 1/1024$ and $\triangle x/\varepsilon = 1/128$.

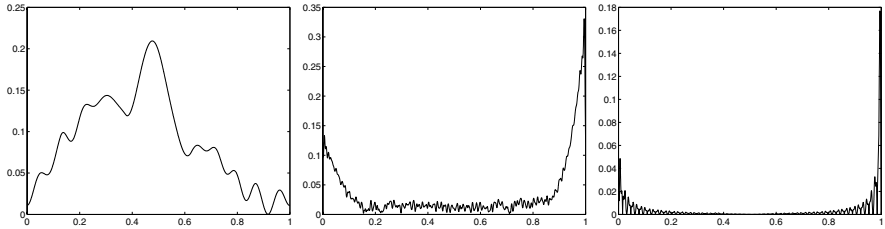
Time step $\triangle t$	1/1000	1/2000	1/4000	1/8000
$\left\|\psi^{\rm ts}_{\triangle x,\triangle t}(t,\cdot) - \psi^{\rm ex}(t,\cdot)\right\|_{l^2}$	6.60E-2	1.54E-2	3.81E-3	9.45E-4
Convergence order		2.1	2.0	2.0
time step $\triangle t$	1/100	1/200	1/400	1/800
$\left\|\psi^{\rm bd}_{\triangle x,\triangle t}(t,\cdot) - \psi^{\rm ex}(t,\cdot)\right\|_{l^2}$	3.32E-3	7.54E-4	1.42E-4	3.16E-5
Convergence order		2.1	2.4	2.2

WKB-type expansion methods, even though there are currently more advanced tools at hand; cf. [16, 29, 31]. The WKB method, however, has the advantage of giving a rather simple and transparent description of $\psi(t)$, solution to (1.1), for $\varepsilon \ll 1$ (at least locally in-time). Since the Bloch decomposition method itself is rather abstract we include this approximative description here too, so that the reader gets a better feeling for the appearing quantities. Moreover, this two-scale WKB method can also be used for [illegible] Schrödinger dynamics [11], a problem we shall study numerically in an upcoming work.

5.1. The WKB formalism. To this end let us suppose that the initial condition is of (two-scale) [illegible]. More precisely assume

$$\psi_{\rm in}(x) = \sum_{m=1}^{M} u_m\left(x, \frac{x}{\varepsilon}\right) {\rm e}^{i\phi(x)/\varepsilon}, \tag{5.1}$$

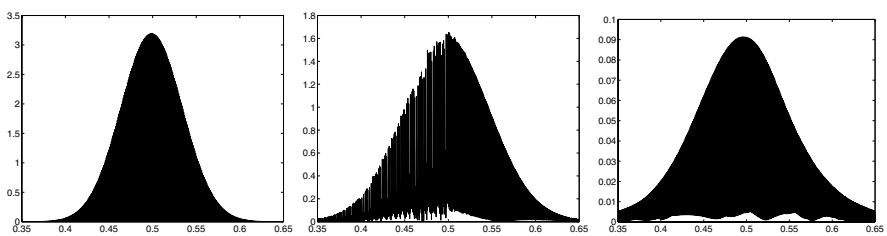
with some given [illegible] $\phi \in C^\infty(\mathbb{R})$ and some given initial (complex-valued) [illegible] $u_m(x, y + 2\pi) = u_m(x, y)$, each of which admits an asymptotic de-



$|\psi^{\rm ex}(t,x)|^2$, $|\psi^{\rm ex}(t,x)-\psi^{\rm ts}(t,x)|$ and $|\psi^{\rm ex}(t,x)-\psi^{\rm bd}(t,x)|$ at $t=1.0$

FIG. 8. *Numerical results for Example* 4.2 *with* $U(x)$ *given by* (4.2), $\varepsilon=\frac{1}{2}$. *We use* $\triangle t=\frac{1}{100}$, $\triangle x=\frac{1}{64}$ *for the TS method,* $\triangle t=\frac{1}{2}$, $\triangle x=\frac{1}{32}$ *for the BD method, and* $\triangle t=\frac{1}{100000}$, $\triangle x=\frac{1}{8192}$ *for the "exact" solution.*

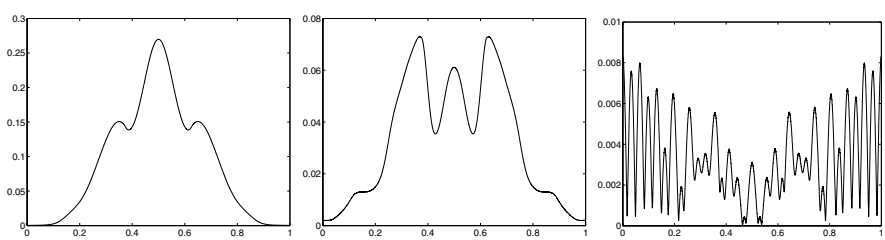
$$\Delta_\infty^{\rm ts}(t)=3.31{\rm E}-1,\ \Delta_\infty^{\rm bd}({\rm t})=1.77{\rm E}-1,\ \Delta_2^{\rm ts}({\rm t})=6.16{\rm E}-2,\ \Delta_2^{\rm bd}({\rm t})=1.38{\rm E}-2.$$



$|\psi^{\rm ex}(t,x)|^2$, $|\psi^{\rm ex}(t,x)-\psi^{\rm ts}(t,x)|$ and $|\psi^{\rm ex}(t,x)-\psi^{\rm bd}(t,x)|$ at $t=0.1$

FIG. 9. *Numerical results for Example* 4.2 *with* $U(x)$ *given by* (4.2), $\varepsilon=\frac{1}{1024}$. *We use* $\triangle t=\frac{1}{10000}$, $\triangle x=\frac{1}{65536}$ *for the TS method,* $\triangle t=\frac{1}{10}$, $\triangle x=\frac{1}{8192}$ *for the BD method, and* $\triangle t=\frac{1}{100000}$, $\triangle x=\frac{1}{131072}$ *for the "exact" solution.*

$$\Delta_\infty^{\rm ts}(t)=1.65,\ \Delta_\infty^{\rm bd}(t)=9.14{\rm E}-2,\ \Delta_2^{\rm ts}({\rm t})=2.63{\rm E}-1,\ \Delta_2^{\rm bd}({\rm t})=1.39{\rm E}-2.$$



$|\psi^{\rm ex}(t,x)|^2$, $|\psi^{\rm ex}(t,x)-\psi^{\rm ts}(t,x)|$ and $|\psi^{\rm ex}(t,x)-\psi^{\rm bd}(t,x)|$ at $t=1.0$

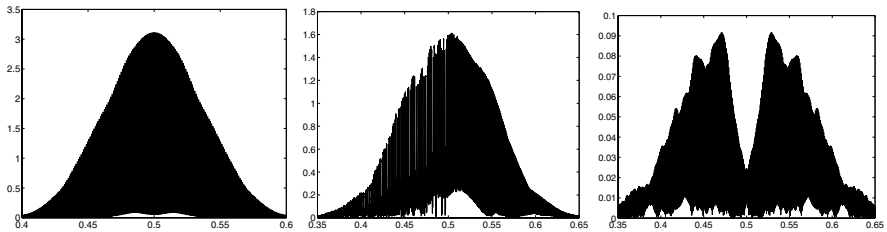
FIG. 10. *Numerical results for Example* 4.2 *with* $U(x)$ *given by* (4.3) *and* $\varepsilon=\frac{1}{2}$. *We use* $\triangle t=\frac{1}{200}$, $\triangle x=\frac{1}{64}$ *for the TS method,* $\triangle t=\frac{1}{5}$, $\triangle x=\frac{1}{32}$ *for the BD method, and* $\triangle t=\frac{1}{100000}$, $\triangle x=\frac{1}{8192}$ *for the "exact" solution.*

$$\Delta_\infty^{\rm ts}(t)=7.30{\rm E}-2,\ \Delta_\infty^{\rm bd}({\rm t})=8.30{\rm E}-3,\ \Delta_2^{\rm ts}({\rm t})=4.02{\rm E}-2,\ \Delta_2^{\rm bd}({\rm t})=3.89{\rm E}-3.$$

scription of the following form:

$$u_m(x,y)\sim u_m^0(x,y)+\varepsilon u_m^1(x,y)+\mathcal{O}(\varepsilon^2)\quad\forall\, m\in\mathbb{N}. \tag{5.2}$$

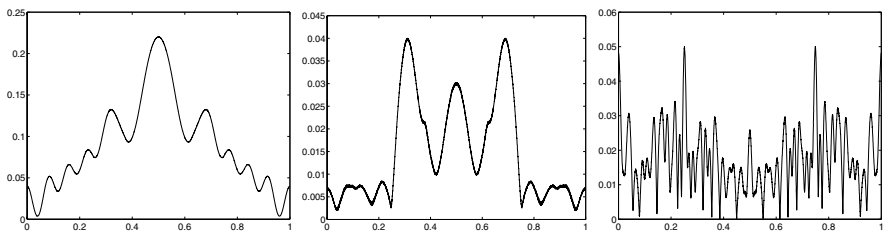
Here and in the following we shall be concerned only with the leading order asymptotic description.



$|\psi^{\rm ex}(t,x)|^2$, $|\psi^{\rm ex}(t,x)-\psi^{\rm ts}(t,x)|$ and $|\psi^{\rm ex}(t,x)-\psi^{\rm bd}(t,x)|$ at $t=0.1$

FIG. 11. *Numerical results for Example* 4.2 *with* $U(x)$ *given by* (4.3) *and* $\varepsilon=\frac{1}{1024}$. *We use* $\triangle t=\frac{1}{50000}$, $\triangle x=\frac{1}{65536}$ *for the TS method,* $\triangle t=\frac{1}{10}$, $\triangle x=\frac{1}{8192}$ *for the BD method, and* $\triangle t=\frac{1}{100000}$, $\triangle x=\frac{1}{131072}$ *for the "exact" solution.*

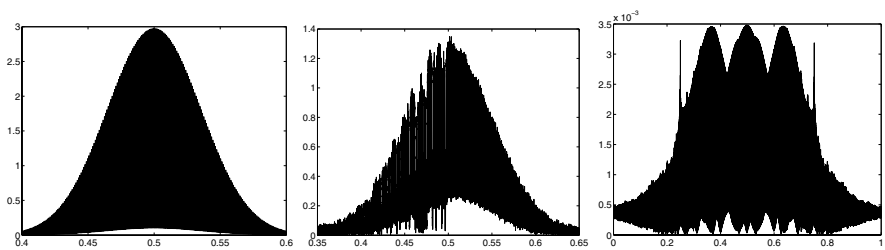
$$\Delta_\infty^{\rm ts}(t)=1.61,\ \Delta_\infty^{\rm bd}(t)=9.16{\rm E}-2,\ \Delta_2^{\rm ts}({\rm t})=2.63{\rm E}-1,\ \Delta_2^{\rm bd}({\rm t})=1.71{\rm E}-2.$$



$|\psi^{\rm ex}(t,x)|^2$, $|\psi^{\rm ex}(t,x)-\psi^{\rm ts}(t,x)|$ and $|\psi^{\rm ex}(t,x)-\psi^{\rm bd}(t,x)|$ at $t=1.0$

FIG. 12. *Numerical results for Example* 4.2 *with* $U(x)$ *given by* (4.4), $\varepsilon=\frac{1}{2}$. *We use* $\triangle t=\frac{1}{100}$, $\triangle x=\frac{1}{32}$ *for the TS method,* $\triangle t=\frac{1}{5}$, $\triangle x=\frac{1}{32}$ *for the BD method, and* $\triangle t=\frac{1}{100000}$, $\triangle x=\frac{1}{8192}$ *for the "exact" solution.*

$$\Delta_\infty^{\rm ts}(t)=4.01{\rm E}-2,\ \Delta_\infty^{\rm bd}({\rm t})=5.00{\rm E}-2,\ \Delta_2^{\rm ts}({\rm t})=1.85{\rm E}-2,\ \Delta_2^{\rm bd}({\rm t})=1.98{\rm E}-2.$$



$|\psi^{\rm ex}(t,x)|^2$, $|\psi^{\rm ex}(t,x)-\psi^{\rm ts}(t,x)|$ and $|\psi^{\rm ex}(t,x)-\psi^{\rm bd}(t,x)|$ at $t=0.1$

FIG. 13. *Numerical results for Example* 4.2. *Here* $U(x)$ *is given in* (4.4), $\varepsilon=\frac{1}{1024}$. *We use* $\triangle t=\frac{1}{10000}$, $\triangle x=\frac{1}{65536}$ *for the TS method,* $\triangle t=\frac{1}{10}$, $\triangle x=\frac{1}{8192}$ *for the BD method, and* $\triangle t=\frac{1}{100000}$, $\triangle x=\frac{1}{131072}$ *for "exact" solution.*

$$\Delta_\infty^{\rm ts}(t)=1.35,\ \Delta_\infty^{\rm bd}(t)=3.48{\rm E}-3,\ \Delta_2^{\rm ts}({\rm t})=2.23{\rm E}-1,\ \Delta_2^{\rm bd}({\rm t})=1.14{\rm E}-3.$$

5.1. Note that we consider only a single initial WKB phase $\phi(x)$ for all bands $m\in\mathbb{N}$. We could of course also allow for more general cases, like one WKB phase for each band or even a superposition of WKB states within each band. However, in order to keep the presentation clean we hesitate to do so. The standard WKB approximation, for nonperiodic problems, involves real-valued amplitudes $\tilde{u}^0(x),\tilde{u}^1(x),\dots$ which depend only on the slow scale.

TABLE 4
The results of Example 4.2 *with harmonic external potential* (4.3).

Spatial discretization error test at time $t = 0.1$ for $\varepsilon = 1/2$.
For TS $\triangle t = 0.0001$ and for BD $\triangle t = 0.01$.

Mesh size $\triangle x/\varepsilon$	1/2	1/4	1/8	1/16
$\left\|\psi^{\rm ts}_{\triangle x,\triangle t}(t,\cdot) - \psi^{\rm ex}(t,\cdot)\right\|_{l^2}$	2.71E-1	8.87E-2	5.19E-3	1.32E-4
Convergence order		1.6	4.1	5.3
$\left\|\psi^{\rm bd}_{\triangle x,\triangle t}(t,\cdot) - \psi^{\rm ex}(t,\cdot)\right\|_{l^2}$	3.23E-1	9.08E-2	7.03E-3	1.27E-4
Convergence order		1.8	3.7	5.8

Spatial discretization error test at time $t = 0.01$ for $\varepsilon = 1/1024$.
For TS $\triangle t = 0.00001$ and for BD $\triangle t = 0.001$.

Mesh size $\triangle x/\varepsilon$	1/2	1/4	1/8	1/16
$\left\|\psi^{\rm ts}_{\triangle x,\triangle t}(t,\cdot) - \psi^{\rm ex}(t,\cdot)\right\|_{l^2}$	3.99E-1	3.67E-1	2.19E-1	1.10E-1
Convergence order		0.1	0.7	1.0
$\left\|\psi^{\rm bd}_{\triangle x,\triangle t}(t,\cdot) - \psi^{\rm ex}(t,\cdot)\right\|_{l^2}$	2.06E-1	5.64E-2	8.16E-3	6.40E-4
Convergence order		1.9	2.8	3.7

Temporal discretization error test at $t = 0.1$ for $\varepsilon = 1/2$ and $\triangle x/\varepsilon = 1/128$.

Time step $\triangle t$	1/10	1/20	1/40	1/80
$\left\|\psi^{\rm ts}_{\triangle x,\triangle t}(t,\cdot) - \psi^{\rm ex}(t,\cdot)\right\|_{l^2}$	1.02E-3	6.41E-4	3.80E-4	2.18E-4
Convergence order		0.7	0.8	0.8
$\left\|\psi^{\rm bd}_{\triangle x,\triangle t}(t,\cdot) - \psi^{\rm ex}(t,\cdot)\right\|_{l^2}$	4.20E-6	1.02E-6	2.22E-7	5.56E-8
Convergence order		2.0	2.2	2.0

Temporal discretization error test at $t = 0.01$ for $\varepsilon = 1/1024$ and $\triangle x/\varepsilon = 1/128$.

Time step $\triangle t$	1/1000	1/2000	1/4000	1/8000
$\left\|\psi^{\rm ts}_{\triangle x,\triangle t}(t,\cdot) - \psi^{\rm ex}(t,\cdot)\right\|_{l^2}$	1.21E-1	1.18E-1	1.10E-1	1.10E-1
Convergence order		0.04	0.1	0.0
time step $\triangle t$	1/100	1/200	1/400	1/800
$\left\|\psi^{\rm bd}_{\triangle x,\triangle t}(t,\cdot) - \psi^{\rm ex}(t,\cdot)\right\|_{l^2}$	3.30E-5	5.21E-6	1.23E-6	3.16E-7
Convergence order		2.6	2.1	2.0

TABLE 5
The mass of $\psi(t,x)$*, solution to Example* 4.1 *with external potential* (4.3)*, decomposed into the Bloch bands for* $\varepsilon = \frac{1}{32}$ *at time* $t = 1$.

m	1	2	3	4
$\mathrm{M}^{\varepsilon}_m$	7.89E − 1	1.10E − 2	5.92E − 1	9.38E − 2
m	5	6	7	8
$\mathrm{M}^{\varepsilon}_m$	7.15E − 2	3.50E − 3	1.80E − 3	5.63E − 5

It is well known then (cf. [11, 20]) that the leading order term u^0_m, $m \in \mathbb{N}$, can be decomposed as

$$u^0_m\left(x, \frac{x}{\varepsilon}\right) = f_m(x)\chi_m\left(\frac{x}{\varepsilon}, \partial_x\phi(x)\right), \tag{5.3}$$

where we assume $f_m \in \mathcal{S}(\mathbb{R})$. In other words, there is an adiabatic decoupling between the slow scale x and fast scale x/ε. Indeed, a lengthy calculation, invoking the classical stationary phase argument (cf. [6, Chapter 4.7]), shows that the band projection $\mathbb{P}^{\varepsilon}_m\psi$

can be approximated via

$$\mathbb{P}^{\varepsilon}_m \psi(x) \sim f_m(x)\, \chi_m\left(\frac{x}{\varepsilon}, \partial_x \phi(x)\right) \mathrm{e}^{i\phi(x)/\varepsilon} + \mathcal{O}(\varepsilon). \tag{5.4}$$

This approximate formula shows the origin of the high oscillations induced either by V_Γ, described by χ_m, or by the dispersion, described by $\phi(x)$. We note that in general the higher order terms (in ε), such as u^1_m, etc., are of a more complicated structure than (5.3), but we shall neglect these terms in what follows (see, e.g., [11] for more details). One consequently finds that $\psi(t)$ obeys a leading order asymptotic description of the form

$$\psi(t,x) \sim \sum_{m=1}^{M} a_m(t,x)\, \chi_m\left(\frac{x}{\varepsilon}, \partial_x \phi_m(t,x)\right) \mathrm{e}^{i\phi_m(t,x)/\varepsilon} + \mathcal{O}(\varepsilon), \tag{5.5}$$

where $\phi_m(t,x) \in C^\infty([0,t_c)\times\mathbb{R})$ satisfies the mth band

$$\begin{cases} \partial_t \phi_m(t,x) + E_m(\partial_x \phi_m) + U(x) = 0, \quad m \in \mathbb{N}, \\ \phi_m\big|_{t=0} = \phi(x). \end{cases} \tag{5.6}$$

Also, the (complex-valued) leading order WKB amplitude $a_m(t,x) \in C^\infty([0,t_c)\times\mathbb{R})$ satisfies the

$$\begin{cases} \partial_t a_m + \partial_k E_m(\partial_x\phi_m)\partial_x a_m + \dfrac{1}{2}\,\partial_x(\partial_k E_m(\partial_x\phi_m))a_m \\ \qquad -(\beta_m(t,x)\partial_x U(x))\, a_m = 0, \quad a_m\big|_{t=0} = f_m(x), \end{cases} \tag{5.7}$$

with $\beta_m(t,x) := \langle \chi_m(y,k), \partial_k \chi_m(y,k)\rangle_{L^2(\mathcal{C})}$, evaluated at $k = \partial_x\phi_m$, the so-called

5.2. Note that the Berry term is purely imaginary, i.e., $\beta_m(t,x) \in \mathrm{i}\mathbb{R}$, which implies the following conservation law:

$$\partial_t |a_m|^2 + \partial_x\left(\partial_k E_m(\partial_x\phi_m)|a_m|^2\right) = 0 \quad \forall\, m \in \mathbb{N}. \tag{5.8}$$

Of course the above WKB-type expansion method is valid only up to (in general finite) time $0 \le t_c < \infty$, the in the solution of (5.6). Here we shall simply assume that $t_c > 0$ holds, i.e., no caustic is formed at time $t = 0$, which is very well possible in general. We note that in the considered numerical examples below we indeed have $t_c > 0$ and we refer to [10] for a broader discussion on this. For $t \ge t_c$ one would need to superimpose several WKB-type solutions corresponding to the multivalued solutions of the flow map $(x,t) \mapsto X_t(x) \equiv X_t(x; \partial_x\phi(x))$, where

$$\begin{cases} \dot{X}_t = \partial_k E_m(\Xi_t), \quad X_0 = x, \\ \dot{\Xi}_t = -\partial_x U(X_t), \quad \Xi_0 = \partial_x\phi(x). \end{cases} \tag{5.9}$$

Numerically we shall use the relaxation method introduced in [24] to solve the Hamilton–Jacobi equation (5.6). Consequently we can solve the system of transport equations (5.7) by a time-splitting spectral scheme similar to the ones used above.

5.2. Numerical examples. We shall finally study the WKB approach, briefly described above, using some numerical examples. Denote by

$$\psi^{\mathrm{sc}}(t,x) := \sum_{m=1}^{M} f_m(t,x)\chi_m\left(\frac{x}{\varepsilon}, \partial_x\phi_m\right) e^{i\phi_m(t,x)/\varepsilon} \tag{5.10}$$

TABLE 6
Difference between the asymptotic solution and the Schrödinger equation for Example 5.1 *($\triangle t = 10^{-4}$, $\triangle x = 1/32768$).*

ε	1/32	1/1024
$\sup_{0\le t\le 1} \| \psi(t,x) - \psi^{\mathrm{sc}}(t,x) \|_{L^2(\mathbb{R})}$	6.68E − 3	3.08E − 4
$\sup_{0\le t\le 1} \| \psi(t,x) - \psi^{\mathrm{sc}}(t,x) \|_{L^\infty(\mathbb{R})}$	5.57E − 2	2.38E − 3

TABLE 7
Difference between the asymptotic solution and the Schrödinger equation for Example 5.2 *for initial condition* (5.11) *($\triangle t = 10^{-4}$, $\triangle x = 1/32768$).*

ε	1/32	1/1024
$\sup_{0\le t\le 0.1} \| \psi(t,x) - \psi^{\mathrm{sc}}(t,x) \|_{L^2(\mathbb{R})}$	1.18E − 2	1.08E − 3
$\sup_{0\le t\le 0.1} \| \psi(t,x) - \psi^{\mathrm{sc}}(t,x) \|_{L^\infty(\mathbb{R})}$	9.34E − 2	7.74E − 3

the approximate semiclassical solution to the Schrödinger equation (1.1). In the following examples we only take into account a harmonic external potential of the form (4.3).

EXAMPLE 5.1 (Mathieu's model). We first consider Mathieu's model (4.5) and choose the initial condition in the form

$$\psi_{\mathrm{in}}(x) = \mathrm{e}^{-5(x-\pi)^2} \chi_1\left(\frac{x}{\varepsilon}, 0\right), \tag{5.11}$$

i.e., we choose $\phi(x) = 0$ and restrict ourselves to the case of only one band with index $m = 1$. (The analytical results of [6, 11, 20] then imply that we can neglect the contributions from all other bands $m > 1$ up to errors of order $\mathcal{O}(\varepsilon)$ in $L^2(\mathbb{R}) \cap L^\infty(\mathbb{R})$, uniformly on compact time-intervals.) In this case, we numerically find that no caustic is formed within the solution of (5.6) at least up to $t = 1$, the largest time in our computation. Note that (5.11) concentrates at the minimum of the first Bloch band, where it is known that

$$E_m(k) \approx \frac{|k|^2}{2m^*} + E_m(0). \tag{5.12}$$

This is the so-called effective mass theorem, yielding an effective mass $m^* \in \mathbb{R}$. In Table 6, we show the results with an additional harmonic external potential (cf. (4.3)) for $\varepsilon = \frac{1}{32}$ and $\varepsilon = \frac{1}{1024}$, respectively. Note that these numerical experiments, together with those given below, confirm the analytical results given in [6, 11, 20].

EXAMPLE 5.2 (Kronig–Penney model). Here, we consider again the Kronig–Penney model (4.6). First we use the same initial condition as given in (5.11) but with $m = 2$. The corresponding numerical results for $\varepsilon = \frac{1}{32}$ and $\frac{1}{1024}$ are shown in Table 7. In a second case, we alternatively choose initial data of the form

$$\psi_{\mathrm{in}}(x) = \mathrm{e}^{-5(x-\pi)^2} \chi_2\left(\frac{x}{\varepsilon}, \sin(x)\right) \mathrm{e}^{-\mathrm{i}\cos(x)/\varepsilon}, \tag{5.13}$$

i.e., $\phi(x) = -\cos(x)$. Here we find (numerically) that the caustic onset time is roughly given by $t_c \approx 0.24$; cf. Figure 14. The corresponding numerical results are given in Figure 15 and Table 8.

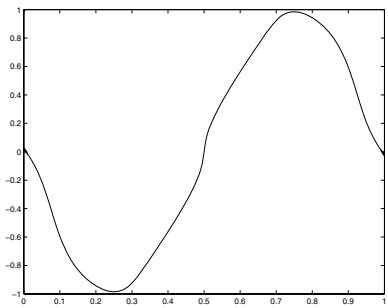
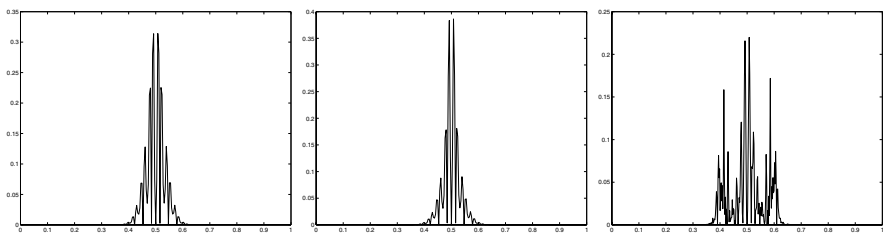
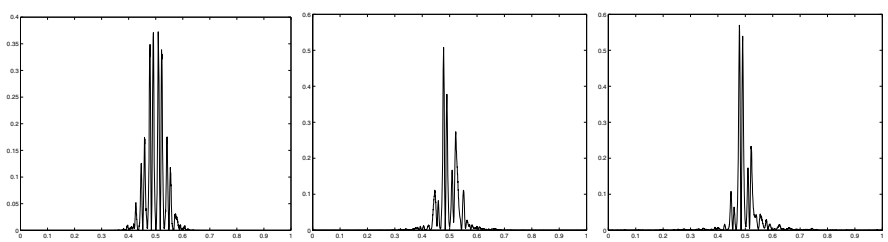


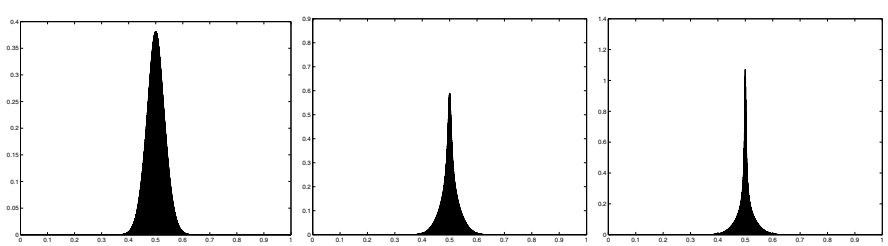
FIG. 14. *The graph of* $\partial_x \phi_2(t,x)$ *at* $t = 0.24$.



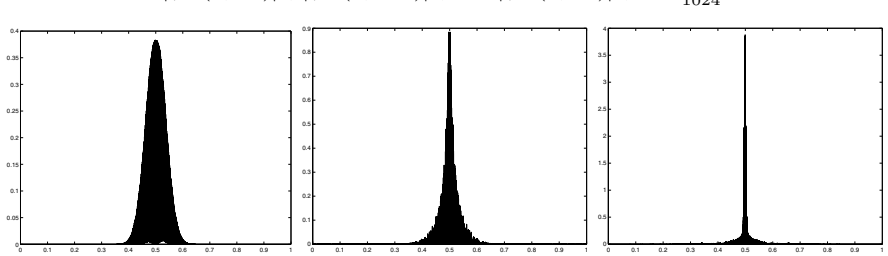
$|\psi^{\rm sc}(x,0.1)|^2$, $|\psi^{\rm sc}(x,0.25)|^2$, and $|\psi^{\rm sc}(x,0.5)|^2$, $\varepsilon = \frac{1}{32}$.



$|\psi^{\rm bd}(x,0.1)|^2$, $|\psi^{\rm bd}(x,0.25)|^2$, and $|\psi^{\rm bd}(x,0.5)|^2$, $\varepsilon = \frac{1}{32}$.



$|\psi^{\rm sc}(x,0.1)|^2$, $|\psi^{\rm sc}(x,0.25)|^2$, and $|\psi^{\rm sc}(x,0.5)|^2$, $\varepsilon = \frac{1}{1024}$.



$|\psi^{\rm bd}(x,0.1)|^2$, $|\psi^{\rm bd}(x,0.25)|^2$, and $|\psi^{\rm bd}(x,0.5)|^2$, $\varepsilon = \frac{1}{1024}$.

FIG. 15. *Numerical results for Example* 5.2 *with* $U(x)$ *given by* (4.3), $\triangle t = \frac{1}{10000}$, $\triangle x = \frac{1}{32768}$. *The left column shows the situation before the caustic, whereas the other two columns show, respectively, during the numerical results and after the caustic.*

TABLE 8
Difference between the asymptotic solution and the Schrödinger equation for Example 5.2 *for initial condition* (5.13) ($\triangle t = 10^{-4}$, $\triangle x = 1/32768$).

ε	1/32	1/1024
$\sup_{0\le t\le 0.1} \| \psi(t,x) - \psi^{\mathrm{sc}}(t,x) \|_{L^2(\mathbb{R})}$	1.68E − 2	3.19E − 3
$\sup_{0\le t\le 0.1} \| \psi(t,x) - \psi^{\mathrm{sc}}(t,x) \|_{L^\infty(\mathbb{R})}$	2.73E − 1	7.33E − 2

Acknowledgment. The authors are grateful to Prof. Christian Ringhofer for fruitful discussions on this work.

REFERENCES

[1] J. ASCH AND A. KNAUF, *Motion in periodic potentials*, Nonlinearity, 11 (1998), pp. 175–200.
[2] N. W. ASHCROFT AND N. D. MERMIN, *Solid State Physics*, Saunders, New York, 1976.
[3] W. Z. BAO, S. JIN, AND P. MARKOWICH, *On time-splitting spectral approximations for the Schrödinger equation in the semiclassical regime*, J. Comput. Phys., 175 (2002), pp. 487–524.
[4] W. Z. BAO, S. JIN, AND P. A. MARKOWICH, *Numerical study of time-splitting spectral discretizations of nonlinear Schrödinger equations in the semiclassical regimes*, SIAM J. Sci. Comput., 25 (2003), pp. 27–64.
[5] P. BECHOUCHE AND F. POUPAUD, *Semi-classical limit of a Schrödinger equation for a stratified material*, Monatsh. Math., 129 (2000), pp. 281–301.
[6] A. BENSOUSSAN, J. L. LIONS, AND G. PAPANICOLAOU, *Asymptotic Analysis for Periodic Structures*, North-Holland, Amsterdam, 1978.
[7] F. BLOCH, *Über die Quantenmechanik der Elektronen in Kristallgittern*, Z. Phys., 52 (1928), pp. 555–600.
[8] E. I. BLOUNT, *Formalisms of band theory*, in Solid State Physics, Vol. 13, Academic Press, New York, 1962, pp. 305–373.
[9] K. BUSCH, *Photonic band structure theory: Assessment and perspectives*, Compte Rendus Phys., 3 (2002), pp. 53–66.
[10] R. CARLES, *WKB analysis for nonlinear Schrödinger equations with a potential*, Comm. Math. Phys., 69 (2007), pp. 195–221.
[11] R. CARLES, P. A. MARKOWICH, AND C. SPARBER, *Semiclassical asymptotics for weakly nonlinear Bloch waves*, J. Statist. Phys., 117 (2004), pp. 369–401.
[12] C. CONCA, R. ORIVE, AND M. VANNINATHAN, *Bloch approximation in homogenization on bounded domains*, Asymptot. Anal., 41 (2005), pp. 71–91.
[13] C. CONCA, N. SRINIVASAN, AND M. VANNINATHAN, *Numerical solution of elliptic partial differential equations by Bloch waves method*, in Congress on Differential Equations and Applications/VII CMA, Universidad de Salamanca, 2001, pp. 63–83.
[14] C. CONCA AND M. VANNINATHAN, *Homogenization of periodic structures via Bloch decomposition*, SIAM J. Appl. Math., 57 (1997), pp. 1639–1659.
[15] M. V. FISCHETTI AND S. E. LAUX, *Monte Carlo analysis of electron transport in small semiconductor devices including band-structure and space-charge effects*, Phys. Rev. B, 38 (1998), pp. 9721–9745.
[16] P. GÉRARD, P. MARKOWICH, N. MAUSER, AND F. POUPAUD, *Homogenization limits and Wigner transforms*, Comm. Pure Appl. Math, 50 (1997), pp. 323–378.
[17] L. GOSSE, *Multiphase semiclassical approximation of an electron in a one-dimensional crystalline lattice.* II. *Impurities, confinement and Bloch oscillations*, J. Comput. Phys., 201 (2004), pp. 344–375.
[18] L. GOSSE AND P. A. MARKOWICH, *Multiphase semiclassical approximation of an electron in a one-dimensional crystalline lattice*—I. *Homogeneous problems*, J. Comput. Phys., 197 (2004), pp. 387–417.
[19] L. GOSSE AND N. MAUSER, *Multiphase semiclassical approximation of an electron in a one-dimensional crystalline lattice.* III. *From ab initio models to WKB for Schrödinger-Poisson*, J. Comput. Phys., 211 (2006), pp. 326–346.
[20] J. C. GUILLOT, J. RALSTON, AND E. TRUBOWITZ, *Semiclassical asymptotics in solid-state physics*, Comm. Math. Phys., 116 (1998), pp. 401–415.

[21] D. Hermann, M. Frank, K. Busch, and P. Wölfle, *Photonic band structure computations*, Optics Express, 8 (2001), pp. 167–173.
[22] R. Horn and C. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, UK, 1985.
[23] Z. Huang, S. Jin, P. Markowich, C. Sparber, and C. Zheng, *A time-splitting spectral scheme for the Maxwell-Dirac system*, J. Comput. Phys., 208 (2005), pp. 761–789.
[24] S. Jin and Z. Xin, *Numerical passage from systems of conservation laws to Hamilton–Jacobi equations, and relaxation schemes*, SIAM J. Numer. Anal., 35 (1998), pp. 2385–2404.
[25] J. D. Joannopoulos and M. L. Cohen, *Theory of short range order and disorder in tetrahedrally bonded semiconductors*, Solid State Phys., 31 (1974), pp. 71–148.
[26] H. J. Korsch and M. Glück, *Computing quantum eigenvalues made easy*, European J. Phys., 23 (2002), pp. 413–425.
[27] S. E. Laux, M. V. Fischetti, and D. J. Frank, *Monte Carlo analysis of semiconductor devices: The DAMOCLES program*, IBM J. Res. Develop., 34 (1990), pp. 466–494.
[28] J. M. Luttinger, *The effect of a magnetic field on electrons in a periodic potential*, Phys. Rev., 84 (1951), pp. 814–817.
[29] G. Panati, H. Spohn, and S. Teufel, *Effective dynamics for Bloch electrons: Peierls substitution and beyond*, Comm. Math. Phys., 242 (2003), pp. 547–578.
[30] M. Reed and B. Simon, *Methods of Modern Mathematical Physics* IV. *Analysis of Operators*, Academic Press, New York, 1978.
[31] S. Teufel, *Adiabatic Perturbation Theory in Quantum Dynamics*, Lecture Notes in Math. 1821, Springer-Verlag, Berlin, 2003.
[32] C. H. Wilcox, *Theory of Bloch waves*, J. Anal. Math., 33 (1978), pp. 146–167.
[33] J. Zak, *Dynamics of electrons in solids in external fields*, Phys. Rev., 168 (1968), pp. 686–695.
[34] A. Zettel, *Sturm-Liouville problems*, in Spectral Theory and Computational Methods for Sturm-Liouville Problems, Lecture Notes in Pure and Appl. Math. 191, D. Hinton and P. W. Schäfer, eds., Dekker, New York, 1997, pp. 1–104.

SIAM J. SCI. COMPUT.
Vol. 29, No. 2, pp. 539-555
© 2007 Society for Industrial and Applied Mathematics

NUMERICAL APPROXIMATION OF A MACROSCOPIC MODEL OF PEDESTRIAN FLOWS*

CHRISTOPHE CHALONS†

Abstract. This paper is concerned with the numerical approximation of the solutions of a macroscopic model for the description of the flow of pedestrians. Solutions of the associated Riemann problem are known to be possibly *nonclassical* in the sense that the underlying discontinuities may well violate the Lax inequalities, which makes their numerical approximation very sensitive. This study proposes to extend the transport-equilibrium strategy proposed in [C. Chalons, *C. R. Acad. Sci. Paris Ser.* I, 342 (2006), pp. 623–626] and [C. Chalons, *Transport-equilibrium schemes for computing nonclassical shocks.* I. *Scalar conservation laws*, submitted] for computing the nonclassical solutions of scalar conservation laws with either a *concave-convex* or a *convex-concave* flux function and supplemented with an *invertible* kinetic function. These strong properties are not fulfilled in the present setting, since the flow function admits several inflection points and the kinetic function is not invertible. We nevertheless succeed in obtaining an efficient numerical scheme. Numerical evidence is provided.

Key words. macroscopic pedestrian flows, nonclassical shocks in conservation laws, kinetic function, transport-equilibrium schemes

AMS subject classifications. 35L60, 35L65, 35L67, 35R60, 76M25, 76L05

DOI. 10.1137/050641211

1. Introduction.

In this paper, we are interested in the numerical approximation of the weak solutions of a scalar conservation law arising in the description of the flow of pedestrians. The model reads

$$\begin{cases} \partial_t \rho + \partial_x q(\rho) = 0, \ \ \rho(x,t) \in \mathbb{R}, \ \ (x,t) \in \mathbb{R}\times]0,+\infty[, \\ \rho(x,0) = \rho_0(x), \ \ x \in \mathbb{R}, \end{cases} \tag{1}$$

where $\rho \geq 0$ is the pedestrian density and $q : \mathbb{R}^+ \to \mathbb{R}^+$ is the flow function. Generally speaking, the solutions of (1) may develop discontinuities in finite time, however smooth the function q and the initial data ρ_0 may be. Then it turns out that the weak solutions (in the integral sense) of (1) are not uniquely determined by initial data ρ_0. In order to single out the solution of interest, various admissibility principles are used in the literature, depending on the geometric properties of the function q. For instance, one may require that the discontinuities of (1) separating two constant states ρ_- and ρ_+ and propagating with speed σ given by the Rankine–Hugoniot relation, $\sigma = [q(\rho_+) - q(\rho_-)] / [\rho_+ - \rho_-]$, obey Oleinik's inequalities:

$$\frac{q(\rho) - q(\rho_-)}{\rho - \rho_-} \geq \frac{q(\rho_+) - q(\rho_-)}{\rho_+ - \rho_-} \quad \text{for all } \rho \text{ between } \rho_- \text{ and } \rho_+. \tag{2}$$

Observe that (2) coincides with Liu's criterion [16] for scalar conservation laws. Another usual criterion is based on the Lax inequalities:

$$q'(\rho_-) \geq \sigma \geq q'(\rho_+). \tag{3}$$

*Received by the editors September 26, 2005; accepted for publication (in revised form) October 3, 2006; published electronically March 15, 2007.
http://www.siam.org/journals/sisc/29-2/64121.html

†Université Paris 7 & Laboratoire JLL, U.M.R. 7598, Boîte courrier 187, 75252 Paris Cedex 05, France (chalons@math.jussieu.fr).

In particular, we note that Oleinik's inequalities (2) imply the Lax inequalities (3). Discontinuities satisfying (3) are said to be [illegible].

When q is convex (or concave), conditions (2) and (3) are equivalent and select a unique solution of the so-called Riemann problem associated with (1), i.e., problem (1) when considering the particular initial data

$$\rho_0(x) = \begin{cases} \rho_l & \text{if } x < 0, \\ \rho_r & \text{if } x > 0. \end{cases} \tag{4}$$

If q is neither convex nor concave, Oleinik's inequalities still lead to a unique solution, and, following [12], this solution is called the [illegible]. Recall that the classical solution of (1)–(4) satisfies a maximum principle, which means that for all $x \in \mathbb{R}$ and $t \geq 0$, $\rho(x,t) \in [\min(\rho_l, \rho_r), \max(\rho_l, \rho_r)]$. However, PDE (1) naturally generates shock waves violating the Lax inequalities (3) (and then Oleinik's inequalities) when q fails to be convex (or concave), and these are to be considered. Such waves are referred to in [12] as [illegible] or as [illegible], and a solution which contains such a shock is said to be [illegible]. The uniqueness of the [illegible] weak solution of problem (1)–(4) is generally restored by means of a [illegible] imposed along each undercompressive discontinuity connecting a left state ρ_- to a right state ρ_+. It takes the form $\rho_+ = \psi(\rho_-)$, where ψ is the so-called [illegible]. In this context, note that the solution no longer obeys a [illegible] and has in general a total variation larger than its initial data. We refer the reader to [12] for a general theory of the classical and nonclassical solutions.

The precise model that we consider in this study was introduced recently by Colombo and Rosini [8]. It is based on the well-known Lighthill–Whitam [15] and Richards [17] model and takes the form of (1) as an immediate consequence of two basic assumptions, namely the conservation of the total number of pedestrians and a given speed law which depends only on the density $\rho \in [0, R]$ (R denotes the maximal density). However, this model was first dedicated to car flows, and so it is not able to reproduce important features of pedestrian flows, at least when considering typical concave increasing-decreasing flow functions. For instance, let us mention the [illegible] phenomenon in a crowd, or the fall of pedestrians in the outflow through a door of a crowd in panic. In order to overcome this difficulty, Colombo and Rosini [8] first proposed modifying the typical shape of the flow function q by introducing another characteristic density $R^\star > R$ for the maximal density in exceptional situations of panic. The flow function now looks like a concave-convex and increasing-decreasing function on $[0, R]$ and a convex-concave and increasing-decreasing function on $[R, R^\star]$ (like in Figure 1 (left)). In particular, discontinuities satisfying the usual Rankine–Hugoniot condition but violating standard admissibility conditions such as the Lax inequalities (3) are present in the model. Then, the same authors defined a unique Riemann solver using such undercompressive shocks. The main motivation in considering nonclassical solutions is to allow panic states ($\rho \in]R, R^\star]$) to appear in a calm situation ($\rho \in [0, R]$), because of a sharp increase in the density, for instance. Note that the [illegible] in classical solutions prevents such panic situations from arising.

From a numerical point of view, the approximation of nonclassical solutions is known to be very challenging and still constitutes (generally speaking) an open problem nowadays (see, for instance, [9], [10], [13], [4], [5], [6], but also [1] and the references within). The main difficulty is to enforce the validity of the kinetic relation at the discrete level. Very recently, a new efficient numerical strategy has been proposed in

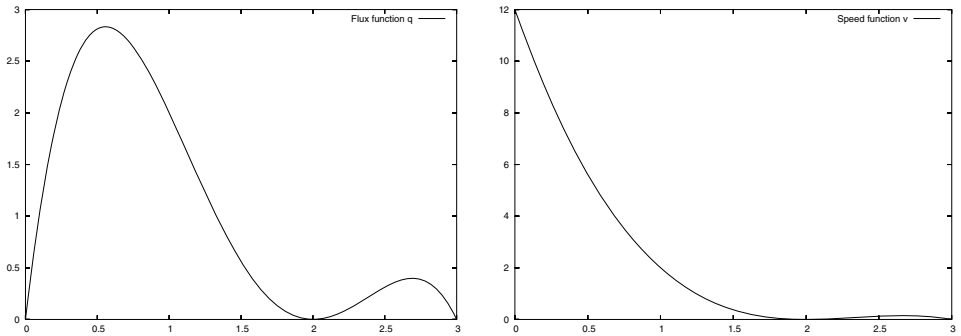


FIG. 1. *Closure relations:* $\rho \to q(\rho)$ *(left) and* $\rho \to v(\rho) = q(\rho)/\rho$ *(right).*

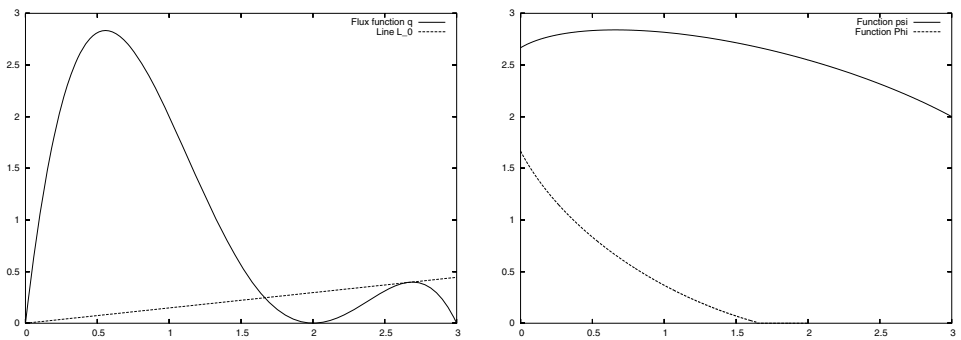


FIG. 2. *Function* q *and line* L_0 *(left); functions* ψ *and* Φ *(right).*

[3] for computing nonclassical solutions of scalar conservation laws in the situation when the flux function is concave-convex or convex-concave and, in an important way, when the kinetic function is an invertible function from $\mathbb{R}$ to $\mathbb{R}$. The algorithm proposed in [3] heavily relies on the latter property. Roughly speaking, the corresponding finite volume scheme is based on two steps, namely an [illegible], which treats each undercompressive shock as a nonmoving discontinuity, and a [illegible] for propagating these discontinuities. Here, it turns out that, on the one hand, the flow function q admits several inflexion points and, on the other hand, the kinetic function ψ (defined in (10) and represented in Figure 2 (right)) is neither defined on the whole domain $\mathbb{R}$ nor invertible. In this paper, we thus propose to adapt the transport-equilibrium scheme developed in [3] to the present setting. The resulting scheme still provides numerical solutions in full agreement with the exact ones.

2. Governing equation and closure relation. In this section, we present the model we use for describing the flow of pedestrians. Our approach considers the well-known Lighthill–Whitam [15] and Richards [17] model as a basic model. According to (1), it takes the form

$$\begin{cases} \partial_t \rho + \partial_x q(\rho) = 0, \quad q(\rho) = \rho v(\rho), \ \ (x,t) \in \mathbb{R} \times]0, +\infty[, \\ \rho(x,0) = \rho_0(x), \ \ x \in \mathbb{R}, \end{cases} \tag{5}$$

where ρ denotes the pedestrian density, v the pedestrian speed, and q the flux function. This equation expresses the conservation of the number of pedestrians in the space

domain, while the speed v is assumed to depend only on the density ρ. Let us recall that this model was first dedicated to car flows and so is not a priori in position to account for specific features of human flows, like the [illegible] phenomenon due to panic. For that reason, following [8], we introduce two remarkable values R and $R^\star$ ($R < R^\star$) for the density ρ. The first one represents a natural bound of ρ in situations with little or no panic ($\rho \in [0, R]$), and the second one is a maximal value of ρ in situations of great panic ($\rho \in]R, R^\star]$). In each of these two regions, the flow function q typically increases at first and then decreases. More precisely, the following assumptions are made in [8]:

$$
(6)\quad \begin{cases} \text{(i)} & q : [0, R^\star] \to [0, +\infty[\text{ belongs to } \mathcal{C}^0([0, R^\star]) \cap \mathcal{C}^2([0, R[\cup]R, R^\star]), \\ \text{(ii)} & q(\rho) = 0 \text{ if and only if } q \in \{0, R, R^\star\}, \\ \text{(iii)} & q' \text{ is bounded and vanishes exactly twice, once in } R_M \in]0, R[\\ & \text{and once in } R_M^\star \in]R, R^\star[, \\ \text{(iv)} & q \text{ has at most one inflection point } R_I \in]R_M, R[, \text{ and at most} \\ & \text{another one } R_I^\star \in]R, R_M^\star[. \end{cases}
$$

For the sake of illustration, we mention here that the numerical experiments in section 5 are performed with

$$
(7)\qquad q(\rho) = -\rho(\rho - R)^2(\rho - R^\star), \quad R = 2, \quad R^\star = 3,
$$

so that

$$
(8)\qquad R_M \simeq 0.5570, \quad R_M^\star \simeq 2.6930, \quad R_I \simeq 1.1208, \quad R_I^\star \simeq 2.3792.
$$

Both the graphs of this flux function $\rho \to q(\rho)$ and the corresponding speed function $\rho \to v(\rho) = q(\rho)/\rho$ are plotted in Figure 1. In particular, note that the speed function $\rho \to v(\rho)$ is decreasing on $[0, R]$, while it is first increasing, then decreasing on $[R, R^\star]$.

3. The Riemann solver. This section aims at defining a (unique) weak solution of (5)–(6) defined for all $(x, t) \in \mathbb{R} \times \mathbb{R}^{+,*}$ when initial data $x \to \rho_0(x)$ is made of only two constant states ρ_l and ρ_r, unspecified in $[0, R^\star]$ and separated by a discontinuity located at $x = 0$:

$$
(9)\qquad \rho_0(x) = \begin{cases} \rho_l & \text{if } x < 0, \\ \rho_r & \text{if } x > 0. \end{cases}
$$

With this in mind and following [8] and [12], we define two functions $\psi : [0, R^\star] \to [R, R^\star]$ and $\Phi : [0, R] \to [0, R]$ related to the graph of the function q in the (ρ, q)-plane. We proceed as follows: as soon as it is possible, $\psi(\rho)$ is such that the line that passes through the points with coordinates $(\rho, q(\rho))$ and $(\psi(\rho), q(\psi(\rho)))$ is tangent to the graph of the function q at point $(\psi(\rho), q(\psi(\rho)))$, while $\Phi(\rho)$ is such that this line intersects the curve $q = q(\rho)$ at a further point with coordinates $(\Phi(\rho), q(\Phi(\rho)))$; otherwise, these functions are extended by continuity as follows:

$$
(10)\qquad \psi(\rho) = \begin{cases} r \text{ such that } q'(r) = \frac{q(r) - q(\rho)}{r - \rho} & \text{if possible,} \\ R_I^\star & \text{if } \rho = R_I^\star, \\ R \text{ or } R^\star & \text{otherwise} \end{cases}
$$

and

$$
(11)\qquad \Phi(\rho) = \begin{cases} r \text{ such that } \frac{q(r) - q(\rho)}{r - \rho} = \frac{q(\rho) - q(\psi(\rho))}{\rho - \psi(\rho)} & \text{if possible,} \\ 0 & \text{otherwise.} \end{cases}
$$

Note that the required continuity property of the function $\rho \to \psi(\rho)$ actually selects a unique value between R and $R^\star$ in the last situation of (10) (see [8] for an example). Moreover, the inflection point $R_I^\star$ does not necessarily exist in (6)(iv). In this case, we set $R_I^\star = R$.

Figure 2 is concerned with the functions ψ and Φ when q is given by (7). On the left, we have plotted both the function q and line L_0 passing through the points with coordinates $(0, q(0) = 0)$, $(\Phi(0), q(\Phi(0)))$, and $(\psi(0), q(\psi(0)))$. Of course, we observe that L_0 is actually tangent to the graph of the function q at point $(\psi(0), q(\psi(0)))$ and cuts this graph at point $(\Phi(0), q(\Phi(0)))$. From a numerical point of view, we have

$$\psi(0) = \frac{8}{3} \simeq 2.6667 \quad \text{and} \quad \Phi(0) = \frac{5}{3} \simeq 1.6667. \tag{12}$$

On the right are represented the functions $\rho \to \psi(\rho)$ and $\rho \to \Phi(\rho)$ on their whole interval of definition.

The next theorem, first stated and proved by Colombo and Rosini [8], now defines a unique Riemann solver for (5)–(6).

THEOREM 1. [illegible] Δs [illegible]

$$s \in]0, R_M[\qquad \Delta s \in]0, R - s[. \tag{13}$$

[illegible] $(\rho_l, \rho_r) \in [0, R^\star]^2$ [illegible] $(x,t) \in \mathbb{R} \times \mathbb{R}^+ \to \rho(x,t) \in [0, R^\star]$ [illegible] (5) (6) (9) [illegible]

A. ρ_l [illegible] ρ_r [illegible] (ρ_l, ρ_r) [illegible] A [illegible]

$$A = \{(\rho_l, \rho_r) \in [0, R^\star]^2 \,/\, s \le \rho_l \le R,\ \Phi(\rho_l) < \rho_r \le R,\ (\rho_r - \rho_l) > \Delta s\}. \tag{14}$$

[illegible] ρ_l [illegible] $\psi(\rho_l)$ [illegible] [illegible] $\psi(\rho_l)$ [illegible] ρ_r

B. ρ_l [illegible] ρ_r [illegible] (ρ_l, ρ_r) [illegible] B [illegible]

$$B = \{(\rho_l, \rho_r) \in [0, R^\star]^2 \,/\, \rho_r > R,\ \rho_r > \rho_l,\ \rho_r < \psi(\rho_l)\}. \tag{15}$$

[illegible] ρ_l [illegible] $\psi(\rho_l)$ [illegible] [illegible] $\psi(\rho_l)$ [illegible] ρ_r

C. ρ_l [illegible] ρ_r [illegible] (ρ_l, ρ_r) [illegible] C [illegible]

$$C = \{(\rho_l, \rho_r) \in [0, R^\star]^2 \,/\, \rho_r > R,\ \rho_r > \rho_l,\ \rho_r \ge \psi(\rho_l)\}. \tag{16}$$

[illegible] ρ_l [illegible] ρ_r

Before addressing the numerical approximation of the solutions of (5)–(6), let us briefly comment on this theorem. First, it is important to notice that the conditions (14)–(15)–(16) in the Riemann solver separate data leading to classical solutions from those generating nonclassical behaviors. In other words, and following [12], [14], or [3], these conditions represent in some sense a [illegible], which determines when the solution becomes nonclassical. As may be expected, this nucleation condition will play a central part in the numerical strategy. Then, a physical interpretation

may be given to situation A. In this case, ρ_r is sufficiently close to the border between calm and panic, and ρ_l is sufficiently both large and far from ρ_r to create a panic state when facing ρ_r. An undercompressive shock between ρ_l and $\psi(\rho_l)$ expresses the corresponding violation of the usual maximum principle. In cases B and C, we have $\rho_r \in]0, R^\star]$. Thus, we observe that if $\rho_l \in [0, R]$, the two distinct regions associated with calm and panic are joined by a nonclassical behavior, using an undercompressive shock, either from ρ_l to $\psi(\rho_l)$ if possible, or from ρ_l to ρ_r otherwise. In some sense, one thus sees that Theorem 1 makes the function ψ play the part of [illegible] as soon as possible. Finally, we refer the reader to [8] for additional properties of interest satisfied (or not) by the Riemann solver proposed in Theorem 1.

4. Numerical approximation. In this section, we derive a relevant algorithm for approximating the nonclassical solutions of (5)–(6). To that end, the only information we rely on about the Riemann solver given in the previous section is the nucleation condition (14)–(15)–(16). In particular, and contrary to Godunov's or Glimm's scheme, for instance, we are not explicitly using the knowledge of the Riemann solutions. We follow the same approach as in [3]. The scheme is composed of two steps: an equilibrium step and a transport step. In the equilibrium step, our aim is to put at equilibrium all the admissible undercompressive discontinuities by modifying any given consistent and conservative scheme for (5). Then, the transport step takes into account the dynamics of these discontinuities and propagates them.

Let Δt and Δx be the time and the space steps. Introducing the interfaces $x_{j+1/2} = j\Delta x$ for $j \in \mathbb{Z}$ and the intermediate times $t^n = n\Delta t$ for $n \in \mathbb{N}$, we classically seek at each time t^n an approximation ρ_j^n of the solution $x \to \rho(x, t^n)$ on each interval $C_j = [x_{j-1/2}; x_{j+1/2})$, $j \in \mathbb{Z}$. The middle point of C_j is noted $x_j = 0.5\,(x_{j-1/2} + x_{j+1/2})$. In this context, we assume as given a two-point (without loss of generality) numerical flux function $(u, v) \to g(u, v)$ consistent with the flux function q, and we set $\lambda = \Delta t/\Delta x$. See also Remark 2 below. Let us start by motivating our algorithm.

4.1. Preliminary motivation. Our scheme is designed on the basis of the following observations. A natural discretization of the Riemann initial data (9) consists of setting

$$\rho_j^0 = \begin{cases} \rho_l & \text{if } j \leq 0, \\ \rho_r & \text{if } j \geq 1. \end{cases} \tag{17}$$

By Theorem 1, we know that if (ρ_l, ρ_r) obeys (16), the solution is an undercompressive shock connecting ρ_l to ρ_r and propagating with speed $\sigma(\rho_l, \rho_r)$ given by the Rankine–Hugoniot jump relation

$$\sigma(\rho_l, \rho_r) = \frac{q(\rho_r) - q(\rho_l)}{\rho_r - \rho_l}. \tag{18}$$

In particular, $\rho(x, t)$ always equals either ρ_l or ρ_r. From a numerical standpoint, we are tempted to take the usual conservative update formula associated with the numerical flux g, that is,

$$\rho_j^{n+1} = \rho_j^n - \lambda(g_{j+1/2} - g_{j-1/2}), \quad j \in \mathbb{Z}, \tag{19}$$

with $g_{j+1/2} = g(\rho_j^n, \rho_{j+1}^n)$ for all $j \in \mathbb{Z}$. It gives

$$\rho_j^1 = \begin{cases} \rho_l - \lambda(g(\rho_l, \rho_l) - g(\rho_l, \rho_l)) = \rho_l & \text{if } j \leq -1, \\ \rho_l - \lambda(g(\rho_l, \rho_r) - g(\rho_l, \rho_l)) & \text{if } j = 0, \\ \rho_r - \lambda(g(\rho_r, \rho_r) - g(\rho_l, \rho_r)) & \text{if } j = 1, \\ \rho_r - \lambda(g(\rho_r, \rho_r) - g(\rho_r, \rho_r)) = \rho_r & \text{if } j \geq 2, \end{cases}$$

with, generally speaking, $\rho_0^1 \notin \{\rho_l, \rho_r\}$ and $\rho_1^1 \notin \{\rho_l, \rho_r\}$. After the first time iteration, two intermediate states have thus been created by the conservative scheme (19). In order to avoid such spurious values, we are thus led to replace (19). We suggest this nonconservative formula:

$$\rho_j^{n+1} = \rho_j^n - \lambda(g_{j+1/2}^L - g_{j-1/2}^R), \quad j \in \mathbb{Z}, \tag{20}$$

where the numerical fluxes $g_{j+1/2}^L = g_{j+1/2}^L(\rho_j^n, \rho_{j+1}^n)$ and $g_{j+1/2}^R = g_{j+1/2}^R(\rho_j^n, \rho_{j+1}^n)$ have to be suitably defined. First, it is natural to set

$$g_{j+1/2}^L(\rho, \rho) = g_{j+1/2}^R(\rho, \rho) = g(\rho, \rho), \quad j \in \mathbb{Z},$$

for all ρ. Then, if (ρ_l, ρ_r) satisfies (16) with, in addition, $\rho_r = \psi(\rho_l)$, setting

$$g_{j+1/2}^L(\rho_l, \rho_r) = g(\rho_l, \rho_l) \quad \text{and} \quad g_{j+1/2}^R(\rho_l, \rho_r) = g(\psi(\rho_l), \rho_r), \quad j \in \mathbb{Z}, \tag{21}$$

is sufficient to avoid intermediate values since we get

$$\begin{cases} \rho_0^1 = \rho_l - \lambda(g_{1/2}^L(\rho_l, \rho_r) - g_{-1/2}^R(\rho_l, \rho_l)) = \rho_l - \lambda(g(\rho_l, \rho_l) - g(\rho_l, \rho_l)) = \rho_l, \\ \rho_1^1 = \rho_r - \lambda(g_{3/2}^L(\rho_r, \rho_r) - g_{1/2}^L(\rho_l, \rho_r)) = \rho_r - \lambda(g(\rho_r, \rho_r) - g(\psi(\rho_l), \rho_r)) = \rho_r. \end{cases}$$

In the same way, if (ρ_l, ρ_r) satisfies (16) but $\rho_r \neq \psi(\rho_l)$, setting

$$g_{j+1/2}^L(\rho_l, \rho_r) = g(\rho_l, \rho_l) \quad \text{and} \quad g_{j+1/2}^R(\rho_l, \rho_r) = g(\rho_r, \rho_r), \quad j \in \mathbb{Z}, \tag{22}$$

is fitting since

$$\begin{cases} \rho_0^1 = \rho_l - \lambda(g_{1/2}^L(\rho_l, \rho_r) - g_{-1/2}^R(\rho_l, \rho_l)) = \rho_l - \lambda(g(\rho_l, \rho_l) - g(\rho_l, \rho_l)) = \rho_l, \\ \rho_1^1 = \rho_r - \lambda(g_{3/2}^L(\rho_r, \rho_r) - g_{1/2}^L(\rho_l, \rho_r)) = \rho_r - \lambda(g(\rho_r, \rho_r) - g(\rho_r, \rho_r)) = \rho_r. \end{cases}$$

Thanks to the new update formula (20), we are thus able to remove nondesired values. Nevertheless, it is clear at this stage that the initial discontinuity (17) is made stationary by our new update formula. Instead, it should be moving at speed $\sigma(\rho_l, \rho_r)$ given in (18). This is the reason for which a transport step must be included in our algorithm. A sampling strategy will be proposed in the next section, again in order to avoid the emergence of spurious values (see just below for more details).

4.2. Transport-equilibrium scheme. We now describe the two steps of our numerical strategy in details. As motivated in the previous subsection, the central idea is to modify the numerical flux $g_{j+1/2}(\rho_j^n, \rho_{j+1}^n)$ by means of two fluxes $g_{j+1/2}^L(\rho_j^n, \rho_{j+1}^n)$ and $g_{j+1/2}^R(\rho_j^n, \rho_{j+1}^n)$ each time that an undercompressive shock appears in the solution of the Riemann problem (5)–(6)–(9) associated with $\rho_l = \rho_j^n$ and $\rho_r = \rho_{j+1}^n$. According to whether this undercompressive shock connects the states ρ_j^n and $\psi(\rho_j^n)$ or not, that is, depending on whether $(\rho_j^n, \rho_{j+1}^n) \in A \cup B$ or $(\rho_j^n, \rho_{j+1}^n) \in C$, we will use the formulas (21) or (22), respectively. However, we have shown in the previous section that with such a modification, the undercompressive discontinuities are made stationary. We will then introduce a transport step in order to make moving these discontinuities with the right speed.

Equilibrium step ($t^n \to t^{n+1-}$). This first step aims at making stationary the admissible undercompressive discontinuities of (5)–(6) (see Theorem 1). We thus consider the nonconservative version of (19) given by (20), where the numerical fluxes $g_{j+1/2}^L$ and

$g^R_{j+1/2}$ are defined as follows with the aid of the nucleation condition (14)–(15)–(16) and for all $j \in \mathbb{Z}$:

$$g^L_{j+1/2} = \begin{cases} g(\rho^n_j, \rho^n_j) & \text{if } (\rho^n_j, \rho^n_{j+1}) \in A \cup B \cup C, \\ g(\rho^n_j, \rho^n_{j+1}) & \text{otherwise,} \end{cases} \tag{23}$$

and

$$g^R_{j+1/2} = \begin{cases} g(\psi(\rho^n_j), \rho^n_{j+1}) & \text{if } (\rho^n_j, \rho^n_{j+1}) \in A \cup B, \\ g(\rho^n_{j+1}, \rho^n_{j+1}) & \text{if } (\rho^n_j, \rho^n_{j+1}) \in C, \\ g(\rho^n_j, \rho^n_{j+1}) & \text{otherwise.} \end{cases} \tag{24}$$

With these definitions, we easily check (see also previous subsection) that discontinuities separating two states ρ_- and ρ_+ such that $(\rho_-, \rho_+) \in C$ (that is, in particular when $\rho_+ = \psi(\rho_-)$) are kept at stationary equilibrium during this first step.

$(t^{n+1-} \to t^{n+1})$. This step is concerned with the dynamics of the discontinuities left stationary during the first step. Recall that the speed of propagation $\sigma(\rho_-, \rho_+)$ of a discontinuity between ρ_- and ρ_+ is given by the Rankine–Hugoniot relation (18). We then decide to define at each interface $x_{j+1/2}$ a speed of propagation $\sigma_{j+1/2}$:

$$\sigma_{j+1/2} = \begin{cases} \sigma(\rho^{n+1-}_j, \rho^{n+1-}_{j+1}) & \text{if } (\rho^n_j, \rho^n_{j+1}) \in A \cup B \cup C, \\ 0 & \text{otherwise,} \end{cases} \tag{25}$$

and solve locally (at each discontinuity $x_{j+1/2}$) a transport equation with speed $\sigma_{j+1/2}$. If Δt is chosen sufficiently small to avoid wave interactions, we can glue together the corresponding solutions to define a solution in the whole domain. In order to get a new approximation ρ^{n+1}_j at time $t^{n+1} = t^n + \Delta t$, we propose to pick up randomly on the interval $[x_{j-1/2}, x_{j+1/2}[$ a value at time Δt in this solution. In particular, such a sampling strategy prevents the emergence of spurious intermediate values with respect to those obtained at time t^{n+1-}. Given a well distributed random sequence (a_n) within the interval $(0, 1)$, this amounts to setting

$$\rho^{n+1}_j = \begin{cases} \rho^{n+1-}_{j-1} & \text{if } a_{n+1} \in [0, \lambda\sigma^+_{j-1/2}[, \\ \rho^{n+1-}_j & \text{if } a_{n+1} \in [\lambda\sigma^+_{j-1/2}, 1 + \lambda\sigma^-_{j+1/2}[, \\ \rho^{n+1-}_{j+1} & \text{if } a_{n+1} \in [1 + \lambda\sigma^-_{j+1/2}, 1[, \end{cases} \tag{26}$$

with $\sigma^+_{j+1/2} = \max(\sigma_{j+1/2}, 0)$ and $\sigma^-_{j+1/2} = \min(\sigma_{j+1/2}, 0)$ for all $j \in \mathbb{Z}$. The description of our numerical strategy is now completed.

1. Let us consider the Riemann initial data (9), naturally discretized by (17), such that ρ_l and ρ_r are joined by an undercompressive discontinuity. Then, the definitions of $g^L_{j+1/2}$ and $g^R_{j+1/2}$ make invisible the first step of the method ($\rho^{n+1-}_j = \rho^n_j$ for all j), and only the second step operates. This second step propagates the discontinuity using a sampling strategy. We conclude that, in this particular case, our method reduces to Glimm's random choice scheme. In addition, note that the speed of propagation $\sigma(\rho_l, \rho_r)$ intervening in the second step is exact since it is calculated from the Rankine–Hugoniot relation between ρ_l and ρ_r.

Let us now consider the Riemann initial data (9), again naturally discretized by (17), but with ρ_l and ρ_r such that the Riemann solution is classical, that is, $(\rho_l, \rho_r) \notin A \cup B \cup C$. Then $g^L_{j+1/2} = g^R_{j+1/2} = g_{j+1/2}$ and $\sigma_{j+1/2} = 0$ at each time

step. In this particular case, our method thus reduces to the usual conservative scheme (19).

We conclude this section by highlighting the main difference between the present algorithm and the one proposed in [3]. The latter actually uses the property that the kinetic function is invertible to define $g^L_{j+1/2}$ when an undercompressive shock is present in the Riemann solution associated with the initial states ρ^n_j and ρ^n_{j+1}. More precisely and using the notation of the present paper, it writes $g^L_{j+1/2} = g(\rho^n_j, \psi^{-1}(\rho^n_{j+1}))$ instead of $g^L_{j+1/2} = g(\rho^n_j, \rho^n_j)$ in (23). Basically, it turns out that this definition of $g^L_{j+1/2}$ is actually crucial in [3] for obtaining good numerical results, but only in the situation when the flux is convex-concave. In this case, the undercompressive shock in the Riemann solution associated with states ρ^n_j and ρ^n_{j+1} indeed always connects the states $\psi^{-1}(\rho^n_{j+1})$ and ρ^n_{j+1}. It is then important to involve ψ^{-1} in the definition of the numerical scheme. But in the present situation, we remark (see Theorem 1) that the undercompressive shocks always get ρ^n_j as their left state, and then (necessarily) $\psi(\rho^n_j)$ as their right state. We thus expect that the definition of $g^R_{j+1/2}$ in (24) involving ψ is sufficient to properly capture such discontinuities.

5. Numerical experiments. This section is devoted to the validation of the transport-equilibrium scheme we have proposed. To that purpose and without restriction, we consider a relaxation scheme as a basic numerical flux g, that is,

$$g(u,v) = \frac{1}{2}(q(u)+q(v)) + \frac{a(u,v)}{2}(u-v) \quad \text{with} \quad a(u,v) = \max_{[\min(u,v),\max(u,v)]} |q'| \tag{27}$$

(see [11], for instance), and we use the following standard CFL condition for computing the time step Δt at each time iteration:

$$\Delta t = \frac{1}{2} \times \frac{\Delta x}{\max_j |a(\rho^n_j, \rho^n_{j+1})|}.$$

. . .. 2. At this stage, it is important to notice that, generally speaking, both the numerical flux g and the time step Δt must be chosen so that the standard conservative scheme (19) (i) generates the classical solution of (5), in order to be relevant in situations where the solution is classical and the transport-equilibrium scheme simply reduces to (19); (ii) obeys a discrete version of the maximum principle; that is, for all n, j we have $\rho^n_j \in [\min_j \rho^0_j, \max_j \rho^0_j]$. Indeed, imagine that ρ_0 has the particular form (9) with $(\rho_l, \rho_r) \in [0, R]^2$ but $(\rho_l, \rho_r) \notin A$; the corresponding solution to (5) is classical by Theorem 1 and stays in the interval $[0, R]$. Then, if the conservative scheme (19) fails in satisfying a discrete maximum principle, a value $\rho^n_j > R$ may well appear for some n, j with the (very bad) consequence of starting the nonconservative transport-equilibrium scheme (that is, g^L and g^R no longer equal g in the whole domain) so as to generate a nonclassical solution.

Following a proposal by Colella [7], we consider the van der Corput random sequence (a_n) defined by

$$a_n = \sum_{k=0}^{m} i_k 2^{-(k+1)},$$

where $n = \sum_{k=0}^{m} i_k 2^k$, $i_k = 0, 1$, denotes the binary expansion of the integers $n = 1, 2, \ldots$. We easily find that the first few elements of this sequence are

$$\begin{array}{llll} a_1 = 0.5, & a_2 = 0.25, & a_3 = 0.75, & a_4 = 0.125, \\ a_5 = 0.625, & a_6 = 0.375, & a_7 = 0.875, & a_8 = 0.0625. \end{array}$$

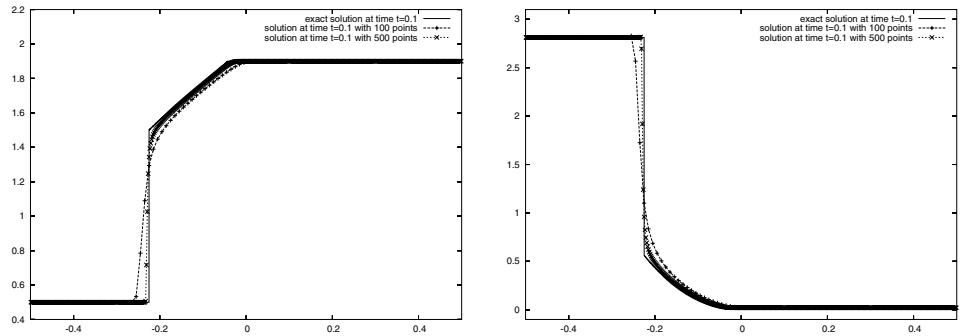


FIG. 3. *Test* 1—*classical solution:* $x \to \rho(x)$ *(left) and* $x \to q(\rho)(x)$ *(right).*

This sequence is actually well distributed within the interval $(0, 1)$. Moreover, one can prove, for instance, that $a_i < 0.5$ for i even and $a_i > 0.5$ for i odd. This well-known sequence is often preferred since it leads to very good results in the smooth parts of the solutions (see [7] and [6] for an illustration).

The closure relations for the numerical simulations are as follows. First of all, the flux function q is chosen as in (7) (see also (8) and Figure 1 for the graphs of the functions q, ψ, and Φ). Then, the thresholds s and Δs in the Theorem 1 are chosen to be

$$\Delta s = \Phi(0) = \frac{5}{3}, \quad s = \frac{1}{2}(R - \Delta s) = \frac{1}{6},$$

so that the condition (13) holds true. Finally, we mention that the computations are performed on two grids, containing respectively 100 ($\Delta x = 0.01$) and 500 ($\Delta x = 0.002$) points per unit interval. Let us now consider several typical behaviors of the Riemann solution given in the Theorem 1. Note that the discontinuity in the Riemann initial data is still located at $x = 0$ (see (9)).

The first test (Test 1) corresponds to the choice $\rho_l = 0.5$ and $\rho_r = 1.9$, so that $(\rho_l, \rho_r) \in [0, R]^2$ but $(\rho_l, \rho_r) \notin A$ since $(\rho_r - \rho_l) < \Delta s$. The solution thus coincides with the classical one. More precisely, we observe in Figure 3 that it is made of a classical shock attached to a rarefaction wave.

For Test 2, we choose $\rho_l = 0.2$ and $\rho_r = 1.9$ so that it is easily checked that $(\rho_l, \rho_r) \in [0, R]^2$ and $(\rho_l, \rho_r) \in A$. In such a situation, panic arises, and the solution is composed of an undercompressive discontinuity between $\rho_l = 0.2$ and $\psi(\rho_l) \simeq 2.7744$, followed by a classical part made of a rarefaction wave and a classical shock attached to the rarefaction. We observe in Figure 4 that our algorithm properly captures this nonclassical solution. Note also that the undercompressive discontinuity from ρ_l to $\psi(\rho_l)$ is sharp: there is no point in the profile. For the sake of comparison, Figure 5 shows that the usual relaxation scheme defined by the update formula (19) generates a (classical) solution which lies entirely in the interval $[0, R]$ and so is far from the expected one. This proves both the need for modifying classical conservative approaches and the good design of our strategy.

In the third test case (Test 3), we choose $\rho_l = 5/2$ and $\rho_r = 1$ so that $\rho_r < \rho_l$, and the solution is classical. In Figure 6, we see that the panic state ρ_l is connected to ρ_r by means of a single classical shock.

For Test 4, we choose $\rho_l = 0.2$ and $\rho_r = 5/2$ so that $(\rho_l, \rho_r) \in B$ (recall that $\psi(\rho_l) \simeq 2.7744 > \rho_r$). By Theorem 1, we know that the solution is composed of an

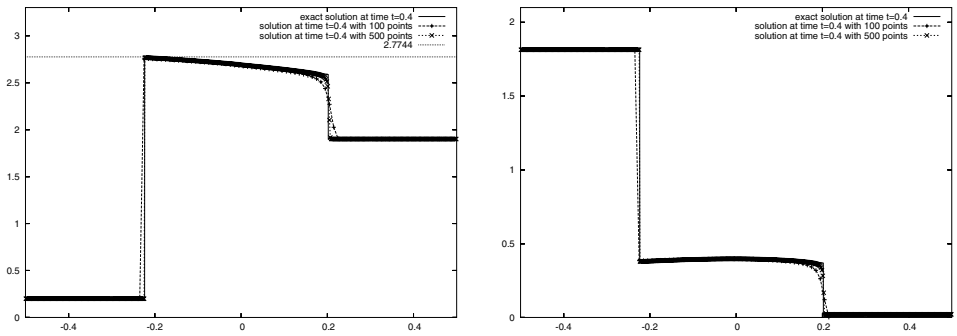


FIG. 4. *Test* 2—*nonclassical solution:* $x \to \rho(x)$ (*left*) *and* $x \to q(\rho)(x)$ (*right*).

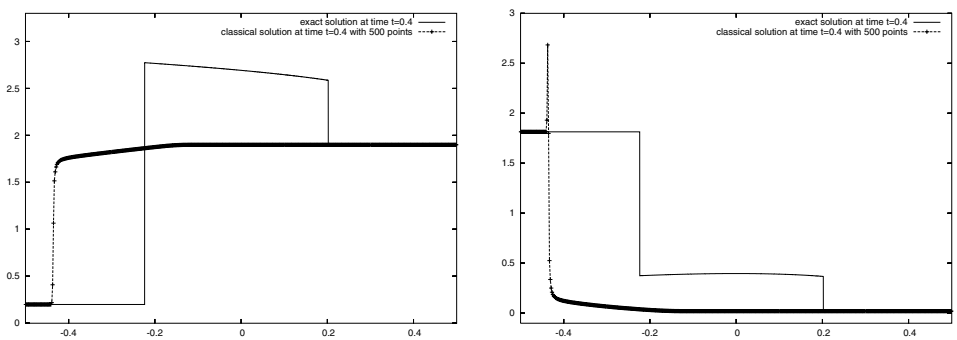


FIG. 5. *Test* 2, (*classical*) *solution generated by standard relaxation method:* $x \to \rho(x)$ (*left*) *and* $x \to q(\rho)(x)$ (*right*).

undercompressive discontinuity between ρ_l and $\psi(\rho_l)$, followed by a rarefaction wave. Here again, we observe in Figure 7 that our algorithm provides a numerical solution in agreement with the exact one, with, in addition, a sharp transition from ρ_l to $\psi(\rho_l)$. In contrast, the usual relaxation scheme would generate a classical behavior really different from the nonclassical one (the result is not presented here).

In the fifth test case (Test 5), we take $\rho_l = 0.2$ and $\rho_r = 2.9$ so that we now have $(\rho_l, \rho_r) \in C$. By Theorem 1, the solution is a single undercompressive shock connecting ρ_l to ρ_r. Figure 8 shows that our algorithm again sharply captures this undercompressive discontinuity. Note that it and Glimm's random choice scheme are identical for this test case since the first step is clearly transparent.

. . . 3. In certain figures, some undershoots and overshoots are observed near the shocks for the q component. Recall that q is calculated as a function of ρ (the conservative unknown). Then, these undershoots and overshoots are due to the numerical diffusion of the method and to the property that the function $\rho \to q(\rho)$ is not monotone between the left and right states of the discontinuity in general. Then, depending on the densities that appear in the numerical profile of the discontinuity, an oscillation is created. Observe that having removed the numerical diffusion of the undercompressive shocks with our algorithm implies that there are not oscillations around these discontinuities.

We finish this section by giving particular attention to the speed of propagation

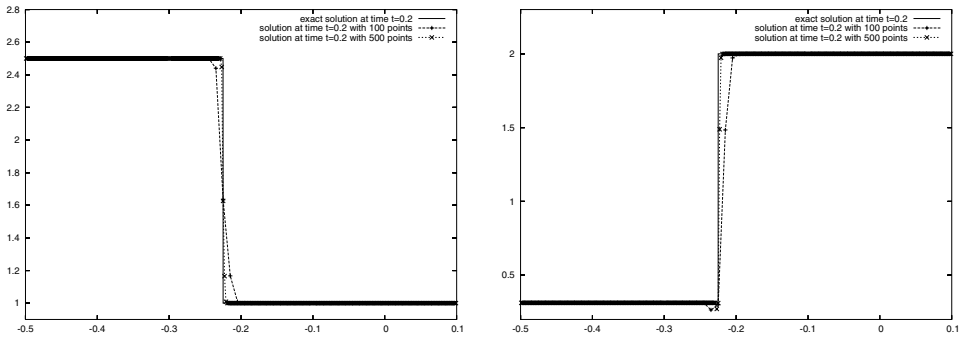


FIG. 6. *Test* 3—*classical solution:* $x \to \rho(x)$ (*left*) *and* $x \to q(\rho)(x)$ (*right*).

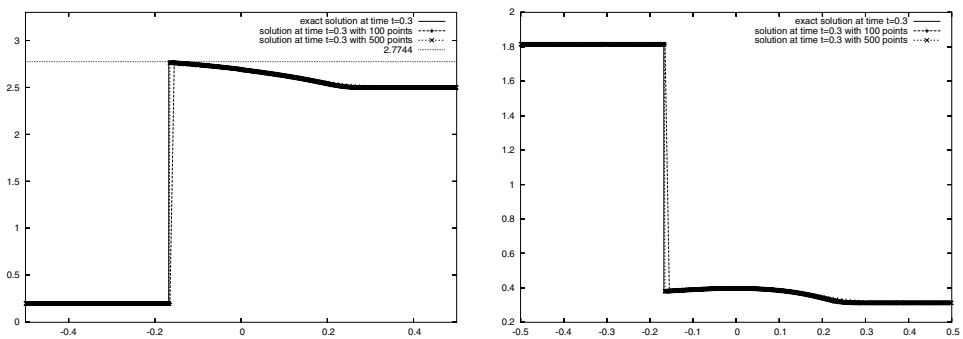


FIG. 7. *Test* 4—*nonclassical solution:* $x \to \rho(x)$ (*left*) *and* $x \to q(\rho)(x)$ (*right*).

provided by the second step of the algorithm. This speed is calculated at each time iteration and may well vary, while in reality the speed of propagation of each undercompressive shock is constant. Let us first consider the fifth test case, for which the left and right states ρ_l and ρ_r are connected by a single undercompressive discontinuity. In this case, the speed of propagation of the method is [illegible] in time and rightly calculated; see Remark 1 above. In Tests 2 and 4, the situation is different since only the left state $\rho_l = 0.2$ of the undercompressive shock is present in the initial data, but not the right state $\psi(\rho_l) \simeq 2.7744$. The latter is created by the method after some time iterations (this state behaves like an equilibrium point of the strategy), and during this process, the corresponding speed of propagation is changing to eventually reach the expected value. In Figure 9, we have plotted for Tests 2 (left) and 4 (right) the values taken by this speed of propagation in the course of the computations (from time $t = 0$ to the final time) and for several meshes. We observe that the method needs only a few times for correctly approaching the exact speed of propagation. For instance, the relative error is around 0.01% at the final time, which explains the good results of the method.

6. Quantitative evaluation of the method. In this section, we propose to evaluate the numerical solutions obtained for the five Riemann problems we have considered above through quantitative tests. We begin with the computation of the conservation errors, and then we study the order of convergence of the method through

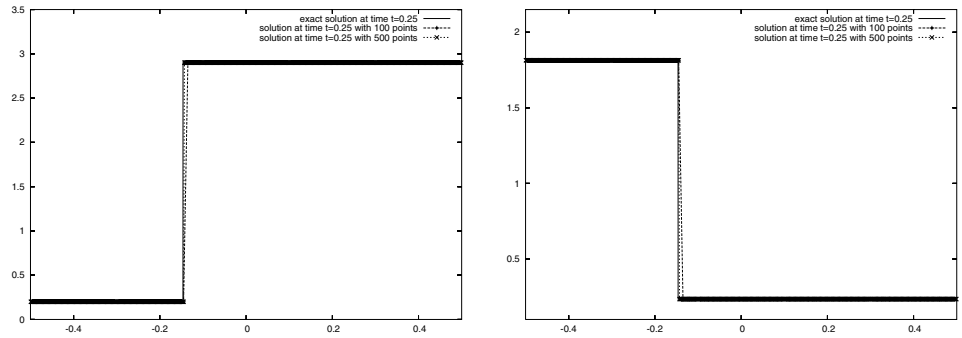


FIG. 8. *Test* 5*—nonclassical solution:* $x \to \rho(x)$ (*left*) *and* $x \to q(\rho)(x)$ (*right*).

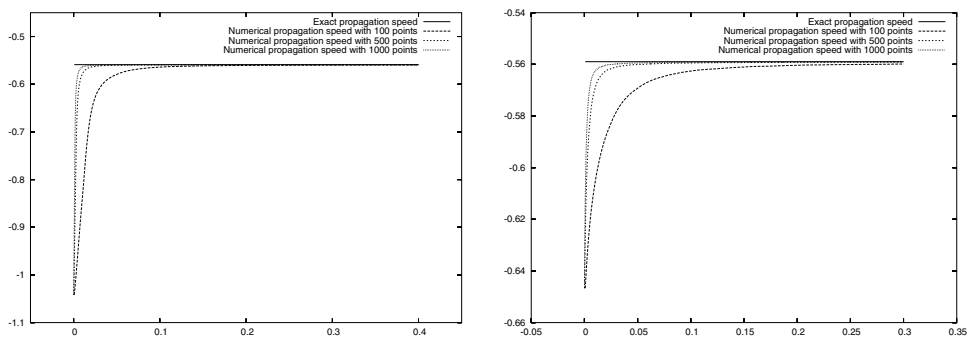


FIG. 9. *Numerical propagation speed of the undercompressive shock versus time: Test* 2 (*left*) *and Test* 4 (*right*).

the L^1-norm.

6.1. Conservation errors. It is obvious that our method is nonconservative since, first, two numerical fluxes g^L and g^R are used in the first step and, second, the transport step is based on a random sampling. We then propose to measure the conservation error on piecewise constant numerical solution ρ_λ defined as

$$\rho_\lambda(x,t) = \rho_j^n \quad \text{if } (x,t) \in [x_{j-1/2}, x_{j+1/2}) \times [t^n, t^{n+1}),$$

between the times $t = 0$ and, let us say, $t = T > 0$. For that, we denote by $[x_m, x_M]$ the computational domain, and we proceed exactly as in [3] by comparing with 0 the function $E : T \in \mathbb{R}^+ \to E(T) \in \mathbb{R}$, with $E(T)$ defined by the relation

$$\int_{x_m}^{x_M} \rho_\lambda(x,T)dx \times E(T) = \int_{x_m}^{x_M} \rho_\lambda(x,T)dx - \int_{x_m}^{x_M} \rho_\lambda(x,0)dx + \int_0^T q(\rho_\lambda(x_M,t))dt - \int_0^T q(\rho_\lambda(x_m,t))dt. \tag{28}$$

Figures 10, 11, 12, 13 and 14 show the function $T \to E(T)$ for the five test cases considered in the previous section. We used $x_m = -0.5$ and $x_M = 0.5$.

For the first test case (Figure 10), no conservation error is made since the solution is classical, so that our algorithm exactly reduces to the standard (conservative)

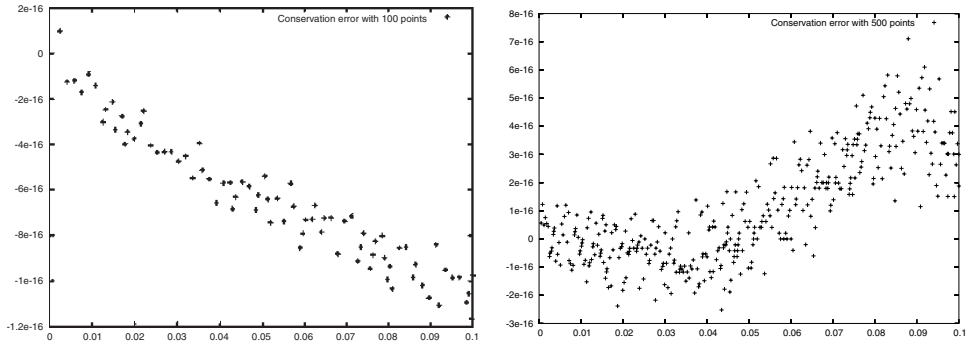


FIG. 10. *Test* 1—*conservation error with* 100 (*left*) *and* 500 (*right*) *points in the mesh.*

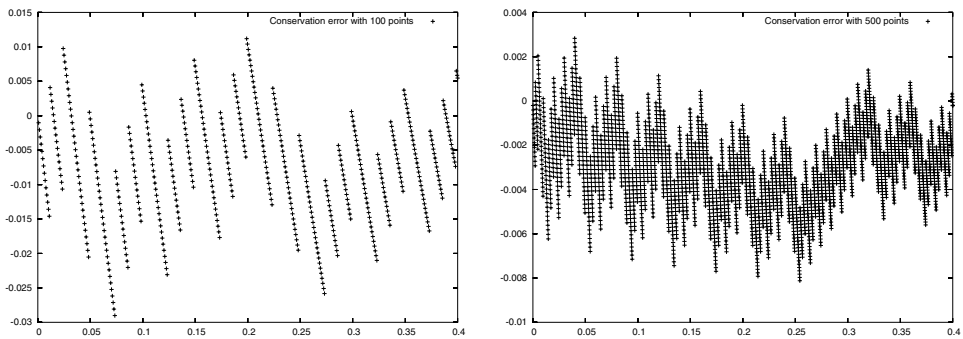


FIG. 11. *Test* 2—*conservation error with* 100 (*left*) *and* 500 (*right*) *points in the mesh.*

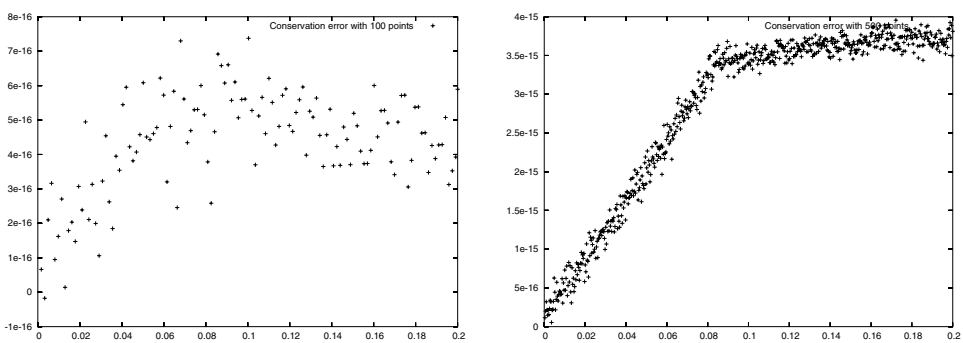


FIG. 12. *Test* 3—*conservation error with* 100 (*left*) *and* 500 (*right*) *points in the mesh.*

relaxation scheme. The same happens for Test 3; see Figure 12.

In Tests 2 and 4, for which the solution is nonclassical, we observe in Figures 11 and 13 that we have a very small conservation error: it seems to stay around 1% and 2% with 100 points in the mesh, and around 0.3% and 0.5% with 500 points in the mesh.

Finally, in the last test case (Figure 14), we observe that the conservation error is once more low and stays around 2.2% with 100 points and around 0.5% with 500 points.

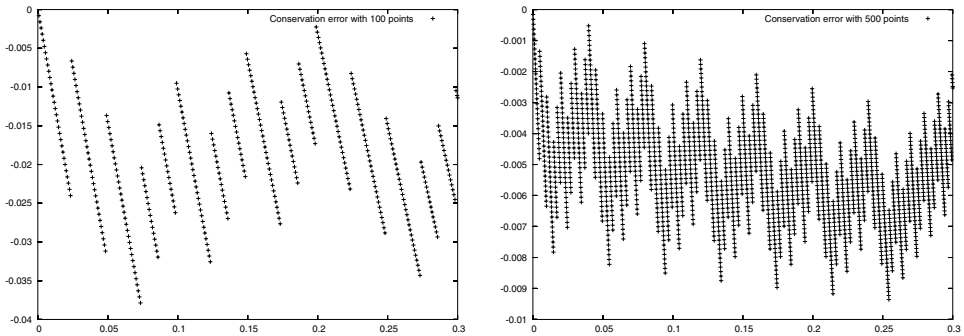


FIG. 13. *Test* 4—*conservation error with* 100 (*left*) *and* 500 (*right*) *points in the mesh.*

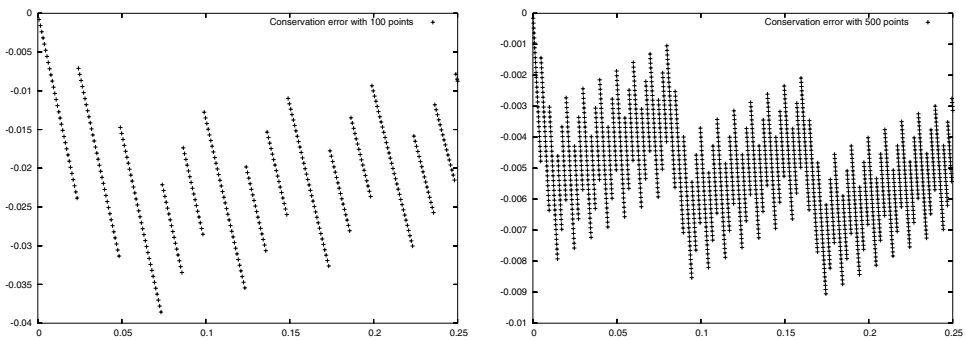


FIG. 14. *Test* 5—*conservation error with* 100 (*left*) *and* 500 (*right*) *points in the mesh.*

To conclude this section, we now address this lack of conservation in term of the number of pedestrians. Let us first notice that the exact solution obeys

$$0 = \int_{x_m}^{x_M} \rho(x,T)dx - \int_{x_m}^{x_M} \rho(x,0)dx + \int_0^T q(\rho(x_M,t))dt - \int_0^T q(\rho(x_m,t))dt,$$

so that, provided the dynamics of the numerical solution has not reached x_m and x_M (which means $q(\rho_\lambda(x_m,t)) = q(\rho(x_m,t))$ and $q(\rho_\lambda(x_M,t)) = q(\rho(x_M,t))$), we get

$$E(T) = \frac{\int_{x_m}^{x_M} \rho_\lambda(x,T)dx - \int_{x_m}^{x_M} \rho(x,T)dx}{\int_{x_m}^{x_M} \rho_\lambda(x,T)dx}.$$

Then, $E(T)$ represents the relative conservation error of ρ at time T on the interval $[x_m, x_M]$, and $\int_{x_m}^{x_M} \rho_\lambda(x,T)dx \times E(T)$ gives the number of persons lost in each computation. At the final time, we found that this number equals 0.0097, -0.0203, and -0.0169 for Tests 2, 4, and 5, respectively (and for 100 points only). This represents only a small portion of a person. Since to get integer numbers of people we have to round these results, we then eventually recover the conservation property (nobody is lost).

6.2. Convergence rate. We now numerically investigate the convergence rate of the method for all the Riemann problems considered until now. We measure the

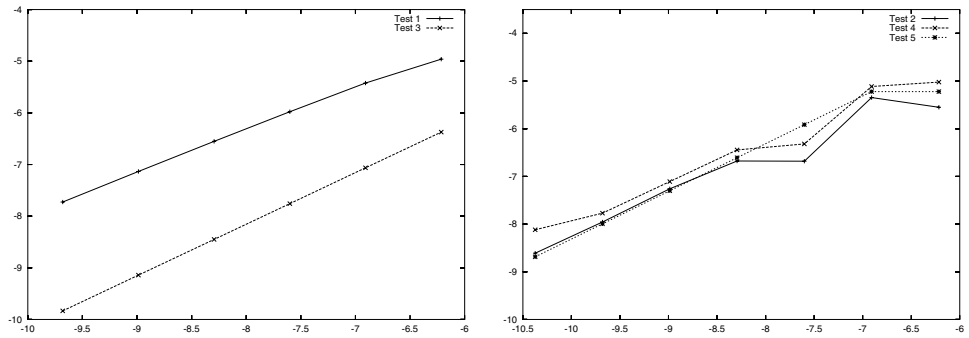


FIG. 15. $\ln(E_{L^1})$ *versus* $\ln(\Delta x)$.

L^1-norm E_{L^1} of the difference between the exact and the numerical solutions by the following usual formula:

$$E_{L^1} = \Delta x \sum_{j \in \mathbb{Z}} |\rho_\lambda(x_j, T_f) - \rho(x_j, T_f)|,$$

where T_f is the final time of the computation. This quantity is evaluated for several space steps Δx, and the results are presented on a graph relating $\ln(E_{L^1})$ to $\ln(\Delta x)$.

Figure 15(left) is concerned with Tests 1 and 3, for which the solutions are classical. In this case, we have already underlined (see Remark 1) that our algorithm is equivalent to the conservative scheme given by the numerical flux g. Then what we are measuring here is nothing but the convergence rate of the numerical solution to the exact solution for the relaxation scheme (27). The meshes we have used contain $2^i \times 500$ nodes per unit interval, with an integer i varying from 0 to 5. The solution of Test 3 is a pure classical shock, and we obtain a convergence rate of approximately 0.999. In contrast, for Test 1 the solution is a classical shock followed by an attached rarefaction wave. The convergence rate observed for this wave is around 0.845, less than for the pure shock.

Figure 15(right) shows the results for Tests 2, 4, and 5, for which the solutions develop an undercompressive shock. The meshes now contain $2^i \times 500$ nodes per unit interval, with i varying from 0 to 6. Let us begin with Test 5. The solution is a pure undercompressive shock, and our method is identical to Glimm's random choice scheme in this case (see again Remark 1). We get a convergence rate of 1, which is expected for this scheme. For Tests 2 and 4 the patterns of the solutions are more complicated, since the undercompressive shock is not pure anymore but attached to a rarefaction wave. Around this undercompressive discontinuity, our method then actually blends Glimm's random choice scheme and the relaxation scheme. We observe that the experimental order of convergence for Test 2 is close to 1, while for Test 4 it is not yet established but convergence seems secured.

7. Conclusion. We have proposed a very efficient numerical strategy for computing the nonclassical solutions of a particular nongenuinely nonlinear scalar conservation law (q is neither convex nor concave), recently introduced by Colombo and Rosini [8] for simulating human flows. The validation has been (successfully) carried out using several relevant Riemann initial data. The fact is that our approach is nonconservative, but measurements have shown that the loss of mass is extremely low,

while the numerical solutions fully agree with the exact ones. Now it would be of interest to use the method on initial data associated with concrete situations.

Acknowledgments. The author is very grateful to the referees and the editor of this paper, which benefited hugely from their reports and comments.

REFERENCES

[1] C. Chalons, *Bilans d'entropie discrets dans l'approximation numérique des chocs non classiques. Application aux équations de Navier-Stokes multi-pression* $2D$ *et à quelques systèmes visco-capillaires*, Ph.D. Thesis, Ecole Polytechnique, Palaiseau, France, 2002.

[2] C. Chalons, *Transport-equilibrium schemes for computing nonclassical shocks*, C. R. Acad. Sci. Paris Ser. I, 342 (2006), pp. 623–626.

[3] C. Chalons, *Transport-Equilibrium schemes for computing nonclassical shocks.* I. *Scalar conservation laws*, submitted.

[4] C. Chalons and P. G. LeFloch, *A fully discrete scheme for diffusive-dispersive conservation laws*, Numer. Math., 89 (2001), pp. 493–509.

[5] C. Chalons and P. G. LeFloch, *High-order entropy conservative schemes and kinetic relations for van der Waals fluids*, J. Comput. Phys., 167 (2001), pp. 1–23.

[6] C. Chalons and P. G. LeFloch, *Computing undercompressive waves with the random choice scheme. Nonclassical shock waves*, Interfaces and Free Boundaries, 5 (2003), pp. 129–158.

[7] P. Colella, *Glimm's method for gas dynamics*, SIAM J. Sci. Stat. Comput., 3 (1982), pp. 76–110.

[8] R. M. Colombo and M. D. Rosini, *Pedestrian flows and nonclassical shocks*, Math. Methods Appl. Sci., 28 (2005), pp. 1553–1567.

[9] B. T. Hayes and P. G. LeFloch, *Nonclassical shocks and kinetic relations: Scalar conservation laws*, Arch. Ration. Mech. Anal., 139 (1997), pp. 1–56.

[10] B. T. Hayes and P. G. LeFloch, *Nonclassical shocks and kinetic relations: Finite difference schemes*, SIAM J. Numer. Anal., 35 (1998), pp. 2169–2194.

[11] C. Lattanzio and D. Serre, *Convergence of a relaxation scheme for hyperbolic systems of conservation laws*, Numer. Math., 88 (2001), pp. 121–134.

[12] P. G. LeFloch, *Hyperbolic Systems of Conservation Laws: The Theory of Classical and Nonclassical Shock Waves*, E.T.H. Lecture Notes Series, Birkhäuser Boston, Cambridge, MA, 2002.

[13] P. G. LeFloch and C. Rohde, *High-order schemes, entropy inequalities, and nonclassical shocks*, SIAM J. Numer. Anal., 37 (2000), pp. 2023–2060.

[14] P. G. LeFloch and M. Shearer, *Nonclassical Riemann solvers with nucleation*, Proc. Royal Soc. Edinburgh, 134A (2004), pp. 941–964.

[15] M. J. Lighthill and G. B. Whitham, *On kinematic waves.* II. *A theory of traffic flow on long crowded roads*, Proc. Royal Soc. London Ser. A., 229 (1955), pp. 317–345.

[16] T. P. Liu, *Admissible Solutions of Hyperbolic Conservation Laws*, Mem. Amer. Math. Soc. 30, AMS, Providence, RI, 1981.

[17] P. I. Richards, *Shock waves on the highway*, Oper. Res., 4 (1956), pp. 42–51.

SIAM J. SCI. COMPUT.
Vol. 29, No. 2, pp. 556–578
© 2007 Society for Industrial and Applied Mathematics

ANALYSIS OF THE PARAREAL TIME-PARALLEL TIME-INTEGRATION METHOD*

MARTIN J. GANDER† AND STEFAN VANDEWALLE‡

Abstract. The parareal algorithm is a method to solve time-dependent problems parallel in time: it approximates parts of the solution later in time simultaneously to parts of the solution earlier in time. In this paper the relation of the parareal algorithm to space-time multigrid and multiple shooting methods is first briefly discussed. The focus of the paper is on new convergence results that show superlinear convergence of the algorithm when used on bounded time intervals, and linear convergence for unbounded intervals.

Key words. time-parallel time-integration, parareal, convergence analysis, shooting, multigrid, deferred correction

AMS subject classifications. 65R20, 45L05, 65L20

DOI. 10.1137/05064607X

1. Introduction.

The parareal algorithm was presented by Lions, Maday, and Turinici in [19] as a numerical method to solve evolution problems in parallel. The name was chosen to indicate that the algorithm is well suited for parallel real time computations of evolution problems whose solution cannot be obtained in real time using one processor only. The method approximates successfully the solution later in time before having fully accurate approximations from earlier times. The algorithm has received a lot of attention over the past few years, especially in the domain decomposition literature [18]. Extensive experiments can be found for fluid and structure problems in [8], for the Navier–Stokes equations in [9], and for reservoir simulation in [10]. Several variants of the method have been proposed, e.g., in [11, 8]. The algorithm has been further analyzed in [21, 22], and its stability is investigated in [29, 2].

Parareal is not the first algorithm to propose the solution of evolution problems in a time-parallel fashion. Already in 1964, Nievergelt suggested a parallel time-integration algorithm [24], which eventually developed into the multiple shooting method for boundary value problems; see [16]. The idea is to decompose the time-integration interval into subintervals, to solve an initial value problem on each subinterval concurrently, and to force continuity of the solution branches on successive intervals by means of a Newton procedure. This approach has also been proposed at a fine grain level, where the shooting intervals correspond to the step size of a one-step method [4]. Parallel multiple shooting for initial value problems was further studied in [6, 17].

In 1967, Miranker and Liniger [23] proposed a family of predictor-corrector methods, in which the prediction and correction steps can be performed in parallel over a number of time steps. Their idea was to "widen the computational front," i.e., to allow processors to compute solution values on several time steps concurrently. A

*Received by the editors November 25, 2005; accepted for publication (in revised form) October 2, 2006; published electronically March 22, 2007.
http://www.siam.org/journals/sisc/29-2/64607.html
†Section de Mathématiques, University of Geneva, 2-4 rue du Lièvre, CP 64 CH-1211 Geneva, Switzerland (Martin.Gander@math.unige.ch).
‡Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, B-3001 Leuven, Belgium (Stefan.Vandewalle@cs.kuleuven.be).

similar motivation led to the block time-integration methods by Shampine and Watts [27]. More recently, [26] and [33] considered the time-parallel application of iterative methods to the system of equations derived with implicit time-integration schemes applied to parabolic partial differential equations (PDEs). Instead of iterating until convergence over each time step before moving on to the next, they showed that it is possible to iterate over a number of time steps at once. Thus a different processor can be assigned to each time step and they all iterate simultaneously. This class of methods was analyzed for its parallel potential in [7]. The authors of that paper showed that the increase in parallel efficiency is often offset by an increased amount of computational work, because of slow information propagation forward in time. The acceleration of such methods by means of a multigrid technique led to the class of parabolic multigrid methods, as introduced by Hackbusch in [12]. The time-parallel multigrid method [13], the multigrid waveform relaxation method [20, 32], and the space-time multigrid method [14] also belong to that class. In [31], a time-parallel variant was shown to achieve excellent speedups on a computer with 512 processors; while run as a sequential algorithm the method is comparable to the best classical time marching schemes. Experiments with time-parallel parabolic multigrid methods on a system with 2^{14} processors are reported in [15].

Our paper is organized as follows. In section 2, we present a new and natural derivation of the parareal algorithm and some of its variants. It is shown that these techniques can be seen as practical implementations of the multiple shooting method. In section 3, we show that the algorithm also fits into the framework of time-multigrid methods. In particular the method is identified as a two-level full approximation scheme with an aggressive time-coarsening. In section 4, we develop new convergence results for scalar linear model problems, on bounded and unbounded time intervals. A number of practically relevant parareal methods are discussed in detail. In section 5, these convergence results are extended to two model PDEs: the heat equation and the pure advection equation. Numerical experiments that illustrate and clarify the theory are reported in section 6. We end in section 7 with some concluding remarks.

2. Parareal as a multiple shooting method. The parareal algorithm is a parallel method for computing the numerical solution for general systems of ordinary differential equations (ODEs) of the form

$$\boldsymbol{u}' = \boldsymbol{f}(\boldsymbol{u}), \quad \boldsymbol{u}(0) = \boldsymbol{u}_0, \quad t \in [0, T]. \tag{2.1}$$

The parareal algorithm is defined using two propagation operators. The operator $G(t_2, t_1, \boldsymbol{u}_1)$ provides a rough approximation to $\boldsymbol{u}(t_2)$ of the solution of (2.1) with initial condition $\boldsymbol{u}(t_1) = \boldsymbol{u}_1$, whereas operator $F(t_2, t_1, \boldsymbol{u}_1)$ provides a more accurate approximation of $\boldsymbol{u}(t_2)$. The algorithm starts with an initial approximation $\boldsymbol{U}_n^0$, $n = 0, 1, \ldots, N$, at time $t_0, t_1, \ldots, t_N$ given, for example, by the sequential computation of $\boldsymbol{U}_{n+1}^0 = G(t_{n+1}, t_n, \boldsymbol{U}_n^0)$, with $\boldsymbol{U}_0^0 = \boldsymbol{u}_0$, and then performs for $k = 0, 1, 2, \ldots$ the correction iteration

$$\boldsymbol{U}_{n+1}^{k+1} = G(t_{n+1}, t_n, \boldsymbol{U}_n^{k+1}) + F(t_{n+1}, t_n, \boldsymbol{U}_n^k) - G(t_{n+1}, t_n, \boldsymbol{U}_n^k). \tag{2.2}$$

Note that for $k \to \infty$ the parareal algorithm (2.2) will upon convergence generate a series of values $\boldsymbol{U}_n$ that satisfy $\boldsymbol{U}_{n+1} = F(t_{n+1}, t_n, \boldsymbol{U}_n)$. That is, the approximation at the time-points t_n will have achieved the accuracy of the F-propagator.

The algorithm can be interpreted most naturally as a classical deferred correction method [28]. In such a method, the solution to a "hard" problem $A(\boldsymbol{u}) = \boldsymbol{0}$ is

computed iteratively by solving a series of "easier" problems of the form $B(\boldsymbol{u}) = \boldsymbol{g}$. More precisely, starting with some approximation $\boldsymbol{u}^0$ to $\boldsymbol{u}$, one solves

$$B(\boldsymbol{u}^{k+1}) = B(\boldsymbol{u}^k) - A(\boldsymbol{u}^k) \quad \text{for } k = 0, 1, \ldots. \tag{2.3}$$

This iteration can be identified with (2.2) if $\boldsymbol{u} = (\boldsymbol{U}_1, \boldsymbol{U}_2, \ldots, \boldsymbol{U}_N)^T$ and if $A(\boldsymbol{u})$ and $B(\boldsymbol{u})$ are vectors of length N times the size of system (2.1), whose components $A(\boldsymbol{u})_{n+1}$ and $B(\boldsymbol{u})_{n+1}$, for $n = 0, \ldots, N-1$, are vectors given, respectively, by

$$A(\boldsymbol{u})_{n+1} = \boldsymbol{U}_{n+1} - F(t_{n+1}, t_n, \boldsymbol{U}_n) \quad \text{and} \quad B(\boldsymbol{u})_{n+1} = \boldsymbol{U}_{n+1} - G(t_{n+1}, t_n, \boldsymbol{U}_n).$$

The parareal method first appeared in [19] for a linear scalar model problem, and it was applied to a linear PDE, and to a nonlinear one by linearization. A different formulation for the nonlinear case appeared first in [3] and then in the mathematically equivalent but simplified form (2.2) in [1]. The original linearized formulation of [19] was rederived in a more general setting, analyzed, and tested on realistic problems in [8]. Here, we will not recall how the different variants have been derived in the literature. Instead, we will present a new, simple, and unified approach for their derivation based on the multiple shooting method applied to (2.1).

In the multiple shooting method the time interval $[0, T]$ is partitioned into N subintervals, not necessarily of equal length, determined by the time-points $0 = t_0 < t_1 < \cdots < t_{N-1} < t_N = T$. A system of N separate initial value problems is posed,

$$\left\{ \begin{array}{rclrcl} \boldsymbol{u}_0' &=& \boldsymbol{f}(\boldsymbol{u}_0), & \boldsymbol{u}_0(0) &=& \boldsymbol{U}_0, \\ \boldsymbol{u}_1' &=& \boldsymbol{f}(\boldsymbol{u}_1), & \boldsymbol{u}_1(t_1) &=& \boldsymbol{U}_1, \\ & \vdots & & & \vdots & \\ \boldsymbol{u}_{N-1}' &=& \boldsymbol{f}(\boldsymbol{u}_{N-1}), & \boldsymbol{u}_{N-1}(t_{N-1}) &=& \boldsymbol{U}_{N-1}, \end{array} \right. \tag{2.4}$$

together with the matching conditions

$$\boldsymbol{U}_0 - \boldsymbol{u}_0 = 0, \quad \boldsymbol{U}_1 - \boldsymbol{u}_0(t_1, \boldsymbol{U}_0) = 0, \quad \ldots, \quad \boldsymbol{U}_N - \boldsymbol{u}_{N-1}(T, \boldsymbol{U}_{N-1}) = 0. \tag{2.5}$$

These matching conditions form a nonlinear system of equations

$$\boldsymbol{F}(\boldsymbol{U}) = \boldsymbol{0}, \quad \boldsymbol{U} = (\boldsymbol{U}_0, \boldsymbol{U}_1, \ldots, \boldsymbol{U}_N)^T. \tag{2.6}$$

Solving this system with Newton's method leads to

$$\boldsymbol{U}^{k+1} = \boldsymbol{U}^k - J_F^{-1}(\boldsymbol{U}^k)\boldsymbol{F}(\boldsymbol{U}^k), \tag{2.7}$$

where J_F denotes the Jacobian of $\boldsymbol{F}$. Using the particular structure of the function $\boldsymbol{F}$ in (2.5), we can write the Newton update $J_F^{-1}(\boldsymbol{U}^k)\boldsymbol{F}(\boldsymbol{U}^k)$ in the form

$$\begin{bmatrix} I & & & & \\ -\frac{\partial u_0}{\partial U_0}(t_1, U_0^k) & I & & & \\ & -\frac{\partial u_1}{\partial U_1}(t_2, U_1^k) & I & & \\ & & \ddots & \ddots & \\ & & & -\frac{\partial u_{N-1}}{\partial U_{N-1}}(t_{n-1}, U_{N-1}^k) & I \end{bmatrix}^{-1} \begin{pmatrix} U_0^k - u_0 \\ U_1^k - u_1(t_1, U_0^k) \\ U_2^k - u_1(t_2, U_1^k) \\ \vdots \\ U_N^k - u_{N-1}(T, U_1^k) \end{pmatrix}.$$

Rearranging (2.7) by multiplying through by the Jacobian, we find the basic recurrence of the multiple shooting method applied to initial value problems,

$$\left\{ \begin{array}{rcl} \boldsymbol{U}_0^{k+1} &=& \boldsymbol{u}_0, \\ \boldsymbol{U}_{n+1}^{k+1} &=& \boldsymbol{u}_n(t_{n+1}, \boldsymbol{U}_n^k) + \frac{\partial \boldsymbol{u}_n}{\partial \boldsymbol{U}_n}(t_{n+1}, \boldsymbol{U}_n^k)(\boldsymbol{U}_n^{k+1} - \boldsymbol{U}_n^k). \end{array} \right. \tag{2.8}$$

While the formulation of (2.8) is entirely at the continuous level, to apply the multiple shooting algorithm one usually needs to discretize the differential equations on each subinterval. That is, one needs to select a numerical method for computing $\boldsymbol{u}_n(t_{n+1}, \boldsymbol{U}_n^k)$. In addition, one has to decide on how to compute or approximate the terms $\frac{\partial \boldsymbol{u}_n}{\partial \boldsymbol{U}_n}(t_{n+1}, \boldsymbol{U}_n^k)$, or the action of those terms on the difference $\boldsymbol{U}_n^{k+1} - \boldsymbol{U}_n^k$.

We will show in a series of remarks that different choices of these approximations have led to different parareal versions.

. . .. 2.1. If in the multiple shooting method one approximates the subinterval solves by $\boldsymbol{u}_n(t_{n+1}, \boldsymbol{U}_n^k) = F(t_{n+1}, t_n, \boldsymbol{U}_n^k)$, and if one approximates the second term in the right-hand side of (2.8) in a finite difference way using the G-propagator,

$$\frac{\partial \boldsymbol{u}_n}{\partial \boldsymbol{U}_n}(t_{n+1}, \boldsymbol{U}_n^k)(\boldsymbol{U}_n^{k+1} - \boldsymbol{U}_n^k) = G(t_{n+1}, t_n, \boldsymbol{U}_n^{k+1}) - G(t_{n+1}, t_n, \boldsymbol{U}_n^k),$$

then the multiple shooting algorithm (2.8) and the parareal algorithm (2.2) coincide.

. . .. 2.2. In [4], Bellen and Zennaro describe a parallel solver for the computation of a general first order recurrence relation of the form $y_{n+1} = F_{n+1}(y_n)$. The function F_{n+1} represents, for example, a one-step numerical discretization of an initial value problem. The method of [4] can be recovered from the multiple shooting algorithm (2.8) if the computation of $\boldsymbol{u}_n(t_{k+1}, \boldsymbol{U}_n^k)$ is approximated as $F_{n+1}(\boldsymbol{U}_n^k)$ and if the columns of the matrix $\frac{\partial \boldsymbol{u}_n}{\partial \boldsymbol{U}_n}(t_{n+1}, \boldsymbol{U}_n^k)$ are approximated individually by a finite difference method. As the authors point out, these columns can be approximated concurrently on d processors, where d is the dimension of (2.1), by applying the function F_{n+1} to d vectors that are each single component perturbations of $\boldsymbol{U}_n^k$.

. . .. 2.3. In shooting methods, the Jacobian may be computed by solving the so-called variational equations, which, in our case, are given by

$$\left(\frac{\partial \boldsymbol{u}_n}{\partial \boldsymbol{U}_n}\right)' = \boldsymbol{f}'(\boldsymbol{u}_n)\frac{\partial \boldsymbol{u}_n}{\partial \boldsymbol{U}_n}, \quad \frac{\partial \boldsymbol{u}_n}{\partial \boldsymbol{U}_n}(t_n) = I, \qquad n = 0, 1, \ldots, N-1. \tag{2.9}$$

Here, I denotes the identity matrix. These linear equations are in general matrix equations, where the matrix size corresponds to the size of the system of equations (2.1). They are classically solved simultaneously with the shooting equations (2.4) using the same method, since the trajectory values $\boldsymbol{u}_n$ are needed in (2.9). If one approximates these equations on the coarse mesh t_n, $n = 0, 1, \ldots, N$, only, then one obtains the original version of the parareal algorithm proposed for nonlinear problems in [19]. Since (2.9) is a matrix equation and one needs only the action of the Jacobian on the vector of differences in (2.8), it is computationally more effective to approximate on the coarse grid the vector valued equation

$$\boldsymbol{v}_n' = \boldsymbol{f}'(\boldsymbol{u}_n)\boldsymbol{v}_n, \quad \boldsymbol{v}_n(t_n) = \boldsymbol{U}_n^{k+1} - \boldsymbol{U}_n^k, \qquad n = 0, 1, \ldots, N-1, \tag{2.10}$$

to obtain $\boldsymbol{v}_n(t_{n+1}) = \frac{\partial \boldsymbol{u}_n}{\partial \boldsymbol{U}_n}(t_{n+1}, \boldsymbol{U}_n^k)(\boldsymbol{U}_n^{k+1} - \boldsymbol{U}_n^k)$. This is done in the so-called PITA algorithm for nonlinear problems proposed in [8] and is mathematically equivalent to approximating (2.9) on the coarse mesh, before applying the matrix valued solution to $\boldsymbol{U}_n^{k+1} - \boldsymbol{U}_n^k$. For linear problems, this approach coincides with formulation (2.2) of the parareal algorithm.

3. Parareal as a time-multigrid method. We will show in this section that iteration (2.2) can be cast into the classical framework of nonlinear multigrid methods. Consider a nonlinear system of equations of the form

$$A_h(\boldsymbol{u}) = \boldsymbol{b}. \tag{3.1}$$

Typically, this system is derived by discretization of a PDE on a mesh Ω_h. The two-grid variant of the so-called full approximation storage (FAS) multigrid method for this problem can be written as follows (see [5]):

$$\left\{\begin{array}{rcl} \tilde{\boldsymbol{u}}^k &=& S(\boldsymbol{u}^k, \boldsymbol{b}), \\ A_H(\boldsymbol{U}^{k+1}) &=& I_h^H(\boldsymbol{b} - A_h(\tilde{\boldsymbol{u}}^k)) + A_H(I_h^H \tilde{\boldsymbol{u}}^k), \\ \boldsymbol{u}^{k+1} &=& \tilde{\boldsymbol{u}}^k + I_H^h(\boldsymbol{U}^{k+1} - I_h^H \tilde{\boldsymbol{u}}^k). \end{array}\right. \tag{3.2}$$

The algorithm is used to compute a series of approximations $\boldsymbol{u}^k$ for $k = 1, 2, \ldots$, starting from a given initial estimate $\boldsymbol{u}^0$, to the solution of problem (3.1). The first step in the two-grid algorithm is called the smoothing step and is fully defined by the smoothing operator S. The second equation in (3.2) defines a coarse grid problem and is characterized by the operator A_H. This operator is usually a discretization of a partial differential operator on a coarse mesh Ω_H. Operator I_h^H is called the restriction operator and transfers discrete functions from the fine mesh Ω_h to the coarse one. Finally, the third step of the algorithm is the correction step. There, the approximation $\tilde{\boldsymbol{u}}^k$ generated by the smoother is updated by transferring a correction from the coarse mesh to the fine mesh by using a prolongation operator I_H^h.

We will show that the FAS components can be selected in such a way that the iteration defined by (3.2) is identical to the parareal iteration given in (2.2). First, we consider the operators A_h and A_H. They are defined by a discretization of the possibly nonlinear evolution equation (2.1) on a fine mesh and on a coarse mesh, respectively. For notational convenience, we will assume those meshes to be regular, with step size h for Ω_h and with step size Mh for Ω_H for some integer $M > 1$. We introduce two arbitrary one-step methods: for the fine grid approximation,

$$\boldsymbol{u}_m = \phi_m(\boldsymbol{u}_{m-1}) + \boldsymbol{b}_m, \quad \boldsymbol{u}_m \approx \boldsymbol{u}(mh), \quad m = 1, 2, \ldots, MN, \tag{3.3}$$

and for the coarse grid approximation,

$$\boldsymbol{U}_n = \Phi_n(\boldsymbol{U}_{n-1}) + \boldsymbol{B}_n, \quad \boldsymbol{U}_n \approx \boldsymbol{u}(nH), \quad n = 1, 2, \ldots, N. \tag{3.4}$$

In the case of a linear problem, the function $\phi_m(\cdot)$ corresponds to a matrix-vector product, with a matrix which we shall denote as ϕ_m. The vector $\boldsymbol{b}_m$ is a vector that collects known, solution independent values. We will extend the sets of equations (3.3) and (3.4) with an additional equation of the form $\boldsymbol{u}_0 = \boldsymbol{b}_0$, or $\boldsymbol{U}_0 = \boldsymbol{B}_0$, where both $\boldsymbol{b}_0$ and $\boldsymbol{B}_0$ are equal to the initial condition. Collecting the vectors of the fine grid approximation $\boldsymbol{u}_m$ in one large vector $\boldsymbol{u}$ and the vectors of the coarse grid approximation in one large vector $\boldsymbol{U}$, we can write the recurrence relation for the fine and coarse grid approximations in the form $A_h(\boldsymbol{u}) = \boldsymbol{b}$ and $A_H(\boldsymbol{u}) = \boldsymbol{B}$, respectively. In the case of a linear problem, we can write the former as a linear matrix equation of the form $A_h \boldsymbol{u} = \boldsymbol{b}$, with

$$A_h := \begin{bmatrix} I & & & \\ -\phi_1 & I & & \\ & \ddots & \ddots & \\ & & -\phi_{MN} & I \end{bmatrix}, \quad \boldsymbol{u} = \begin{pmatrix} \boldsymbol{u}_0 \\ \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_{NM} \end{pmatrix}, \quad \text{and } \boldsymbol{b} = \begin{pmatrix} \boldsymbol{b}_0 \\ \boldsymbol{b}_1 \\ \vdots \\ \boldsymbol{b}_{NM} \end{pmatrix}. \tag{3.5}$$

A similar notation could be used for the coarse mesh problem $A_H(\boldsymbol{U}) = \boldsymbol{B}$.

We will now demonstrate that parareal coincides with a time-multigrid method for a particular—be it somewhat unusual—choice of smoother, restriction, and interpolation operators. The smoother is defined recursively as follows:

$$\begin{cases} S(\boldsymbol{u}^k, \boldsymbol{b})_{nM} := \boldsymbol{u}^k_{nM} \quad \text{for } n = 0, 1, \ldots, N, \\ S(\boldsymbol{u}^k, \boldsymbol{b})_{nM+m} := \phi_{nM+m}(S(\boldsymbol{u}^k, \boldsymbol{b})_{nM+m-1}) + \boldsymbol{b}_{nM+m} \\ \qquad\qquad \text{for } m = 1, \ldots, M-1 \text{ and } n = 0, 1, \ldots, N-1. \end{cases} \tag{3.6}$$

That is, the smoother leaves the components of $\boldsymbol{u}^k$ with indices nM for $n = 0, 1, \ldots, N$ unchanged and fills out the intermediate components by using the recurrence (3.3). In the linear case, this smoother can be written in a more classical matrix formulation. To this end, we define a block Jacobi splitting of A_h,

$$A_h = M_{jac} + N_{jac}, \tag{3.7}$$

where the blocks are of size $M \times M$, except for the first block, which is of size $(M+1) \times (M+1)$, since it also includes the initial condition. We also introduce the matrix E, which is the identity of the same size as A_h, except for zeros on the diagonal at positions $0, M, 2M, \ldots, NM$. With this notation, we have

$$S(\boldsymbol{u}^k, b) := \boldsymbol{u}^k + EM^{-1}_{jac}(\boldsymbol{b} - A_h \boldsymbol{u}^k). \tag{3.8}$$

The method is based on a classical block Jacobi method, but the update of the $\boldsymbol{u}^k_{nM}$ values is prevented by means of the multiplication with the matrix E. Alternatively, the smoother can be interpreted as one phase of a classical two-color scheme, e.g., a red-black method. Let the mesh points be partitioned into a set of "black" points, in our case the points with indices nM, and let the remaining points be in the set of "red" points. One iteration of a standard red-black relaxation method would consist of an update of the red points, followed by an update of the black points. The smoother as defined in (3.6) corresponds to the first, i.e., red, step only.

For the analogy with the parareal scheme to go through, we need the restriction and prolongation operators to be, respectively, the standard injection operator and its transpose. More precisely,

$$(I^H_h \boldsymbol{u})_n := \boldsymbol{u}_{nM} \text{ and } (I^h_H \boldsymbol{U})_{nM+m} := \begin{cases} \boldsymbol{U}_n & \text{for } m = 0, \\ 0 & \text{for } m = 1, 2, \ldots, M-1. \end{cases} \tag{3.9}$$

That is, the restriction selects the entries at positions $0, M, \ldots, NM$ in the vector to which it is applied. The prolongation operator applied to a vector of length $N+1$ copies its entries into a vector of length $NM+1$ at the positions $0, M, \ldots, NM$, and fills in zero values in between. With all of the operators of the FAS method defined, we can now specify the components of the parareal algorithm. We consider $F(t_{n+1}, t_n, \cdot)$ to be composed of a sequence of steps of the form (3.3). More precisely,

$$\begin{cases} F(t_{n+1}, t_n, \boldsymbol{U}_n) := \boldsymbol{u}_{(n+1)M}, \text{ with } \quad \boldsymbol{u}_{nM} = \boldsymbol{U}_n, \\ \text{and } \quad \boldsymbol{u}_m = \phi_m(\boldsymbol{u}_{m-1}) + \boldsymbol{b}_m \text{ for } m = nM+1, \ldots, (n+1)M. \end{cases} \tag{3.10}$$

In a similar way, the propagator $G(t_{n+1}, t_n, \cdot)$ is defined as a single step of (3.4), i.e.,

$$G(t_{n+1}, t_n, \boldsymbol{U}_n) := \Phi_{n+1}(\boldsymbol{U}_n) + \boldsymbol{B}_{n+1}. \tag{3.11}$$

THEOREM 3.1. F G (3.10) (3.11). (3.8) (3.9), $A_h(\cdot)$ $A_H(\cdot)$ (3.3) (3.4). $\boldsymbol{u}^0$ $\boldsymbol{u}^0_{nM} = \boldsymbol{U}^0_n$ $n = 0, 1, \ldots, N$ (2.2) (2.1) (3.2) (3.1), $\boldsymbol{U}^k_n = \boldsymbol{u}^k_{nM}$ $n = 0, 1, \ldots, N$ $k \geq 0$.

. We shall prove the result by induction on k; i.e., we show that $\boldsymbol{U}^{k+1}_n = \boldsymbol{u}^{k+1}_{nM}$ if $\boldsymbol{U}^k_n = \boldsymbol{u}^k_{nM}$. For $k = 0$, the equality holds by the assumption of the theorem.

We rewrite each of the terms in the second equation of the FAS scheme (3.2). For the nth element of the left-hand side, we find

$$\left(A_H(\boldsymbol{U}^{k+1})\right)_n = -\Phi_n(\boldsymbol{U}^{k+1}_{n-1}) + \boldsymbol{U}^{k+1}_n.$$

The first term on the right-hand side leads to

$$\left(I^H_h(\boldsymbol{b} - A_h(\tilde{\boldsymbol{u}}^k))\right)_n = \left(\boldsymbol{b} - A_h(\tilde{\boldsymbol{u}}^k)\right)_{nM} = \boldsymbol{b}_{nM} + \phi_{nM}(\tilde{\boldsymbol{u}}^k_{nM-1}) - \tilde{\boldsymbol{u}}^k_{nM}.$$

For the second term on the right-hand side we find

$$\left(A_H(I^H_h\tilde{\boldsymbol{u}}^k)\right)_n = -\Phi_n((I^H_h\tilde{\boldsymbol{u}}^k)_{n-1}) + (I^H_h\tilde{\boldsymbol{u}}^k)_n = -\Phi_n(\tilde{\boldsymbol{u}}^k_{(n-1)M}) + \tilde{\boldsymbol{u}}^k_{nM}.$$

Hence, the second equation of (3.2) can be rewritten componentwise as

$$-\Phi_n(\boldsymbol{U}^{k+1}_{n-1}) + \boldsymbol{U}^{k+1}_n = \boldsymbol{b}_{nM} + \phi_{nM}(\tilde{\boldsymbol{u}}^k_{nM-1}) - \Phi_n(\boldsymbol{u}^k_{(n-1)M}),$$

where we have used the property that the smoother leaves the elements at positions nM, for $n = 0, 1, \ldots, N$, unchanged. Adding and subtracting $\boldsymbol{B}_n$, we obtain, after rearranging terms,

$$\boldsymbol{U}^{k+1}_n = \Phi_n(\boldsymbol{U}^{k+1}_{n-1}) + \boldsymbol{B}_n + \phi_{nM}(\tilde{\boldsymbol{u}}^k_{nM-1}) + \boldsymbol{b}_{nM} - \Phi_n(\boldsymbol{u}^k_{(n-1)M}) - \boldsymbol{B}_n.$$

By using the induction hypothesis, i.e., $\boldsymbol{U}^k_n = \boldsymbol{u}^k_{nM}$, and the definition of the F- and G-propagators and the smoother, we get

$$\boldsymbol{U}^{k+1}_n = G(t_n, t_{n-1}, \boldsymbol{U}^{k+1}_{n-1}) + F(t_n, t_{n-1}, \boldsymbol{U}^k_{n-1}) - G(t_n, t_{n-1}, \boldsymbol{U}^k_{n-1}).$$

This proves that the coarse grid FAS solution is precisely the solution generated by the parareal algorithm (2.2). Finally, we consider the last equation of (3.2),

$$\boldsymbol{u}^{k+1}_{nM} = \left(\tilde{\boldsymbol{u}}^k + I^h_H(\boldsymbol{U}^{k+1} - I^H_h\tilde{\boldsymbol{u}}^k)\right)_{nM} = (I^h_H\boldsymbol{U}^{k+1})_{nM} = \boldsymbol{U}^{k+1}_n,$$

where the second equality follows from the fact that the operator $I^h_H I^H_h$ acts as the identity on the components $\boldsymbol{u}_{nM}$ of any vector $\boldsymbol{u}$ to which it is applied. This proves the theorem. □

. . . 3.2. The coarse grid problem defined in the second equation of (3.2) is of the same type as the fine grid problem $A_h(\boldsymbol{u}) = \boldsymbol{b}$. Hence, it could be solved by a recursive application of the same method, as in classical multigrid.

. . . 3.3. Note that the smoother (3.6) is not convergent (but not divergent either). Actually, in the linear case, it can easily be shown that the spectral radius of

the iteration matrix is equal to one. This type of smoother, which operates only on the fine grid points and leaves the coarse points unchanged, is sometimes called an F-smoother; see [30].

[illegible] 3.4. With the multigrid components as defined above, the iteration (3.2) is consistent with the fine grid problem. That is, the solution of $A_h(\boldsymbol{u}) = \boldsymbol{b}$ is a fixed point of the iteration. The convergence of the algorithm towards that solution is proved in the next two sections.

4. Convergence analysis: The scalar ODE case. In this section, we first recall a convergence result presented earlier in the literature and then derive two new convergence theorems. The first result is valid on bounded time intervals, $T < \infty$, whereas the second one also holds for unbounded time intervals.

4.1. Some results from the literature. The purpose of our study is to derive the conditions under which the algorithm will converge and to derive an expression for the convergence speed. Our results are different from those in the literature, where typically the number of iterations of (2.2) is fixed to a finite value, and the convergence is studied as ΔT goes to zero. In the fixed number of iterations case, (2.2) defines a new time-integration scheme. The accuracy of the $\boldsymbol{U}_n^k$ values is characterized in a proposition from the first publication on the parareal algorithm [19], which applies for a scalar linear problem of the form

$$u' = au, \quad u(0) = u_0, \quad t \in [0, T] \ \text{ with } \ a \in \mathbb{C}. \tag{4.1}$$

PROPOSITION 4.1. [illegible] $\Delta T = T/N$ $t_n = n\Delta T$ [illegible] $n = 0, 1, \ldots, N$ [illegible] (4.1) [illegible] $a \in \mathbb{R}$ [illegible] $F(t_{n+1}, t_n, U_n^k)$ [illegible] [illegible] t_{n+1} [illegible] (4.1) [illegible] $u(t_n) = U_n^k$ [illegible] $G(t_{n+1}, t_n, U_n^k)$ [illegible] [illegible] ΔT [illegible]

$$\max_{1 \le n \le N} |u(t_n) - U_n^k| \le C_k \Delta T^{k+1}. \tag{4.2}$$

Hence, for a fixed iteration step k, the algorithm behaves in ΔT like an $O(\Delta T^{k+1})$ method. This theorem was extended to more general and, in particular, higher order time-integration schemes in [3, 2]. In those papers it was shown that parareal is a method of $O(\Delta T^{p(k+1)})$ when a method of order p is used as the coarse propagator, combined with an exact solver or a sufficiently accurate solver as the fine propagator.

In our analysis we will fix ΔT and study the algorithm's behavior as k goes to infinity. This case is not covered by the above theorem, because the constant C_k in (4.1) grows with k in the estimate of the proof in [19].

4.2. Some preparatory lemmas. In our analysis an important role will be played by a strictly lower triangular Toeplitz matrix $M := M(\beta)$ of size N. Its elements are fully defined by the values of the elements in its first column,

$$M_{i1} = \begin{cases} 0 & \text{if } \ i = 1, \\ \beta^{i-2} & \text{if } \ 2 \le i \le N \end{cases} \tag{4.3}$$

for some $\beta \in \mathbb{C}$. The analysis will involve the powers of the matrix M. Those matrices maintain the strictly lower triangular Toeplitz structure of M. The elements of those matrices, their norms, and a bound on their norms are identified in the lemmas below.

LEMMA 4.2. [illegible] k [illegible] M [illegible]

$$M_{i,1}^k = \begin{cases} 0 & \text{if } 1 \le i \le k, \\ \binom{i-2}{k-1}\beta^{i-1-k} & \text{if } k+1 \le i \le N. \end{cases} \tag{4.4}$$

[illegible]. The proof is based on induction on k. The matrix M is strictly lower triangular with a nonzero first subdiagonal. Hence, the multiplication of M^{k-1} with M generates exactly one additional zero subdiagonal. This leads to a total of k zero subdiagonals for the matrix M^k, which proves the first part of the lemma.

The ith element in the first column of M^k, for $i \ge k+1$, can be computed as follows:

$$M_{i,1}^k = \sum_{j=1}^{N} M_{i,j}^{k-1} \cdot M_{j,1} = \sum_{j=2}^{i-k+1} M_{i+1-j,1}^{k-1} \cdot M_{j,1} = \sum_{j=0}^{i-k-1} M_{i-1-j,1}^{k-1} \cdot M_{j+2,1},$$

where we have used the zero structure of M^{k-1} and M, and the Toeplitz nature of M^{k-1}. Using the induction hypothesis and (4.3), we can assign values to the matrix elements in the above sum; then we reverse the summation order to find

$$M_{i,1}^k = \sum_{j=0}^{i-k-1} \binom{i-3-j}{k-2} \beta^{i-k-1} = \sum_{J=0}^{i-k-1} \binom{J+k-2}{k-2} \beta^{i-k-1}.$$

The result now follows by applying the well-known summation formula

$$\sum_{s=0}^{r} \binom{n+s}{n} = \binom{n+r+1}{n+1}, \tag{4.5}$$

with $s = J$, $r = i-k-1$, and $n = k-2$. □

LEMMA 4.3. [illegible] k [illegible] M [illegible]

$$||M^k||_\infty = \begin{cases} \binom{N-1}{k} & \text{[illegible] } |\beta| = 1, \\ \dfrac{1}{(k-1)!} \dfrac{d^{k-1}}{dz^{k-1}} \left(\dfrac{z^{N-1}-1}{z-1} \right) \text{[illegible] } z = |\beta| & \text{[illegible] } |\beta| \ne 1. \end{cases} \tag{4.6}$$

[illegible]. The largest row sum of the entries in modulus, i.e., $||M^k||_\infty$, is the row sum in modulus of the last row, since all of the other rows contain less of the same entries than the last one by the structure of M^k. Because of the strictly lower triangular Toeplitz structure, we actually have $||M^k||_\infty = ||M^k||_1$. Hence,

$$||M^k||_\infty = \sum_{i=k+1}^{N} |M_{i,1}^k| = \sum_{i=k+1}^{N} \binom{i-2}{k-1} |\beta|^{i-1-k} = \sum_{i=0}^{N-k-1} \binom{i+k-1}{k-1} |\beta|^i . \tag{4.7}$$

The result for $|\beta| = 1$ now follows by applying (4.5) with $n = k-1$ and $r = N-k-1$. If $|\beta| \ne 1$, we proceed as follows:

$$||M^k||_\infty = \frac{1}{(k-1)!} \sum_{i=0}^{N-k-1} (i+k-1)(i+k-2)\dots(i+1)|\beta|^i$$

$$= \frac{1}{(k-1)!} \sum_{i=0}^{N-k-1} \frac{d^{k-1}}{dz^{k-1}} z^{i+k-1} \quad \text{for } z = |\beta|$$

$$= \frac{1}{(k-1)!} \frac{d^{k-1}}{dz^{k-1}} \sum_{i=0}^{N-k-1} z^{i+k-1} \quad \text{for } z = |\beta|$$

$$= \frac{1}{(k-1)!} \frac{d^{k-1}}{dz^{k-1}} \sum_{I=0}^{N-2} z^{I} \quad \text{for } z = |\beta|.$$

Summing the geometric series leads to the desired result. ∎

LEMMA 4.4. [illegible] k [illegible] M [illegible]

$$||M^k||_\infty \le \begin{cases} \min\left\{ \left(\dfrac{1-|\beta|^{N-1}}{1-|\beta|} \right)^k, \dbinom{N-1}{k} \right\} & \text{[illegible]} \quad |\beta| < 1, \\ |\beta|^{N-k-1} \dbinom{N-1}{k} & \text{[illegible]} \quad |\beta| \ge 1. \end{cases} \tag{4.8}$$

[illegible] The bound for $|\beta| \ge 1$ follows from (4.7) by replacing $|\beta|^i$ by the larger value $|\beta|^{N-k-1}$ and by using (4.5) to evaluate the summation.

Part of the bound for $|\beta| < 1$ follows from (4.7) by replacing $|\beta|^i$ by its upper bound 1. The other part can be derived as follows:

$$||M^k||_\infty \le ||M||_\infty^k = \left(\sum_{i=0}^{N-2} |\beta|^i \right)^k = \left(\frac{1-|\beta|^{N-1}}{1-|\beta|} \right)^k. \quad \blacksquare$$

4.3. A superlinear convergence result on bounded intervals. We derive the conditions for convergence of the parareal algorithm and an estimate for the convergence speed. We consider the classical model problem (4.1) and, initially, assume that the F-propagator is an exact solver. This restriction will be removed in section 4.5, where we consider the use of an approximate F-propagator. The coarse G-propagator is assumed to be a one-step method $G(t_{n+1}, t_n, U_n^k) = R(a\Delta T)U_n^k$, characterized by its stability function $R(z)$. For example, for a Runge–Kutta method with the Butcher tableau coefficients contained in the matrix $A = [a_{ij}]$ and the vector $\boldsymbol{b} = [b_j]$, we have

$$R(z) = 1 + z\boldsymbol{b}^T(I - zA)^{-1}\mathbf{1},$$

where I denotes the identity matrix, and $\mathbf{1}$ the vector containing all 1's.

THEOREM 4.5. [illegible] $T < \infty$ $\Delta T = T/N$ [illegible] $t_n = n\Delta T$ [illegible] $n = 0, 1, \ldots, N$ [illegible] $F(t_{n+1}, t_n, U_n^k)$ [illegible] t_{n+1} [illegible] (4.1) [illegible] $u(t_n) = U_n^k$ [illegible] $G(t_{n+1}, t_n, U_n^k) = R(a\Delta T)U_n^k$ [illegible] $M(\beta)$ [illegible] (4.3), [illegible]

$$\max_{1 \le n \le N} |u(t_n) - U_n^k| \le |e^{a\Delta T} - R(a\Delta T)|^k \|M^k(R(a\Delta T))\|_\infty \max_{1 \le n \le N} |u(t_n) - U_n^0|. \tag{4.9}$$

[illegible] We denote by e_n^k the error at iteration step k of the parareal algorithm at time t_n; i.e., $e_n^k := u(t_n) - U_n^k$. With (2.2) and an induction argument on n, this error

satisfies

$$e_n^{k+1} = R(a\Delta T)e_{n-1}^{k+1} + (e^{a\Delta T} - R(a\Delta T))e_{n-1}^k$$
$$= (e^{a\Delta T} - R(a\Delta T)) \sum_{j=1}^{n-1} R(a\Delta T)^{n-j-1} e_j^k.$$

This relation can be written in matrix form by collecting the error components of the kth iterate e_n^k in the vector $\boldsymbol{e}^k = (e_1^k, e_2^k \ldots, e_N^k)^T$, which leads to

$$\boldsymbol{e}^{k+1} = (e^{a\Delta T} - R(a\Delta T))M(R(a\Delta T))\boldsymbol{e}^k. \tag{4.10}$$

By induction on k in (4.10), we obtain

$$\boldsymbol{e}^k = (e^{a\Delta T} - R(a\Delta T))^k \left(M(R(a\Delta T))\right)^k \boldsymbol{e}^0, \tag{4.11}$$

and taking norms, the result follows. ∎

The bound in (4.9) can be computed by replacing the norm of $M^k(R(a\Delta T))$ by its value as given in Lemma 4.3 or by the bound given in Lemma 4.4. Of particular interest is the case of a one-step method used in its region of absolute stability, i.e., with $|R(a\Delta T)| < 1$.

COROLLARY 4.6. [illegible] $T < \infty$ $\Delta T = T/N$ [illegible] $t_n = n\Delta T$ [illegible] $n = 0, 1, \ldots, N$ [illegible] $F(t_{n+1}, t_n, U_n^k)$ [illegible] t_{n+1} [illegible] (4.1) [illegible] $u(t_n) = U_n^k$ [illegible] $G(t_{n+1}, t_n, U_n^k) = R(a\Delta T)U_n^k$ [illegible]

$$\max_{1\le n\le N} |u(t_n) - U_n^k| \le \frac{|e^{a\Delta T} - R(a\Delta T)|^k}{k!} \prod_{j=1}^{k} (N-j) \max_{1\le n\le N} |u(t_n) - U_n^0|. \tag{4.12}$$

[illegible] G [illegible] $C\Delta T^{p+1}$ [illegible] $p > 0$ [illegible] C [illegible] ΔT [illegible]

$$\max_{1\le n\le N} |u(t_n) - U_n^k| \le \frac{(CT)^k}{k!} \Delta T^{pk} \max_{1\le n\le N} |u(t_n) - U_n^0|. \tag{4.13}$$

[illegible] The first result follows from (4.9) and the norm estimate in Lemma 4.4 for $\beta := R(a\Delta T)$, considering that $|\beta| < 1$. The bound (4.13) follows from the bound on the local truncation error together with a simple estimate of the product,

$$\frac{|e^{a\Delta T} - R(a\Delta T)|^k}{k!} \prod_{j=1}^{k} (N-j) \le \frac{C^k \Delta T^{(p+1)k}}{k!} N^k = \frac{(CT)^k}{k!} \Delta T^{pk}. \qquad \blacksquare$$

[illegible] 4.7. The product term in (4.12) shows that the parareal algorithm terminates with a converged solution for any ΔT on any bounded time interval in at most $N-1$ steps. This property actually holds without restriction on the magnitude of $R(a\Delta T)$ or the quality of the G-operator. Typically, however, one would like to stop the iteration with a converged or sufficiently accurate solution well before $N-1$ iteration steps, since otherwise there is no speedup in the parallel process.

[illegible] 4.8. Observe that the algorithm converges superlinearly, as the division by $k!$ in (4.12) shows. This follows also from the value of the spectral radius of the error propagation operator in (4.10), which is zero.

4.4. A linear convergence result on long time intervals. Next we consider the parareal algorithm as a time-integration method on an infinitely long time interval. In the first theorem of this section we show that the convergence is not worse than linear, and we construct an upper bound for the iteration error. In the second theorem we show that the bound is asymptotically sharp. That is, the asymptotic convergence rate is exactly linear.

THEOREM 4.9. [illegible] ΔT [illegible] $t_n = n\Delta T$, [illegible] $n = 0, 1, \ldots$ [illegible] $F(t_{n+1}, t_n, U_n^k)$ [illegible] t_{n+1} [illegible] (4.1) [illegible] $u(t_n) = U_n^k$ [illegible] $G(t_{n+1}, t_n, U_n^k) = R(a\Delta T)U_n^k$ [illegible]

$$\sup_{n>0} |u(t_n) - U_n^k| \leq \left(\frac{|e^{a\Delta T} - R(a\Delta T)|}{1 - |R(a\Delta T)|} \right)^k \sup_{n>0} |u(t_n) - U_n^0|. \tag{4.14}$$

[illegible] G [illegible] $C\Delta T^{p+1}$ [illegible] $p > 0$ [illegible] C [illegible] ΔT [illegible]

$$\sup_{n>0} |u(t_n) - U_n^k| \leq \left(\frac{C\Delta T^p}{\Re(-a) + O(\Delta T)} \right)^k \sup_{n>0} |u(t_n) - U_n^0|. \tag{4.15}$$

[illegible] In the present case M, as defined in (4.3), is an infinite dimensional Toeplitz operator. Its infinity norm is given by

$$||M(a\Delta T)||_\infty = \sum_{j=0}^{\infty} |R(a\Delta T)|^j = \frac{1}{1 - |R(a\Delta T)|},$$

where we have used that $|R(a\Delta T)| < 1$; i.e., G operates in its region of absolute stability. Using (4.10), we obtain for the error vectors $\boldsymbol{e}^k$ of infinite length the relation

$$||\boldsymbol{e}^k||_\infty \leq |e^{a\Delta T} - R(a\Delta T)|^k ||M||_\infty^k ||\boldsymbol{e}^0||_\infty = \left(\frac{|e^{a\Delta T} - R(a\Delta T)|}{1 - |R(a\Delta T)|} \right)^k ||\boldsymbol{e}^0||_\infty,$$

which proves the first result.

Next, we can use the bound on the local truncation error in order to estimate the numerator of (4.14), i.e., $|e^{a\Delta T} - R(a\Delta T)| \leq C\Delta T^{p+1}$. This bound implies for a strictly positive integer p that $R(a\Delta T) = 1 + a\Delta T + O(\Delta T^2)$. From this, we find

$$\begin{aligned} |R(a\Delta T)| &= R(a\Delta T)^{1/2} \overline{R(a\Delta T)}^{1/2} \\ &= (1 + 1/2 a\Delta T + O(\Delta T^2))(1 + 1/2 \bar{a}\Delta T + O(\Delta T^2)), \end{aligned}$$

which implies $1 - |R(a\Delta T)| = \Re(-a)\Delta T + O(\Delta T^2)$. Thus, (4.15) follows. ∎

The parareal algorithm can be seen as an iteration which maps infinite sequences into infinite sequences. More precisely, it maps the discrete sequence $U^k := \{U_n^k\}_{n=0}^\infty$ into the discrete sequence $U^{k+1} := \{U_n^{k+1}\}_{n=0}^\infty$. In operator notation we can write the iteration compactly as $U^{k+1} = \mathcal{K}(a\Delta T)U^k$, with $\mathcal{K}(a\Delta T)$ the infinite dimensional iteration operator. When applied to model problem (4.1), the parareal iteration operator becomes an infinite dimensional triangular Toeplitz operator or, equivalently, a discrete convolution operator. Such an operator is fully characterized by the elements of its first column a_i, $i = 1, 2, \ldots$. The spectral radius ρ of a Toeplitz operator $\mathcal{T}$ is

known to be related to the operator $T(z) := \sum_{i=1}^{\infty} a_i z^{i-1}$. In particular, when $\sum_{i=1}^{\infty} |a_i| < \infty$, we have that

$$\rho(\mathcal{T}) = \max_{|z|\le 1} | T(z) | = \max_{|z|=1} | T(z) | ; \tag{4.16}$$

see, e.g., [25, Theorem 2.1] for a proof.

THEOREM 4.10. ΔT $t_n = n\Delta T$ $n = 0, 1, \ldots$ $F(t_{n+1}, t_n, U_n^k)$ t_{n+1} (4.1) $u(t_n) = U_n^k$ $G(t_{n+1}, t_n, U_n^k) = R(a\Delta T)U_n^k$

$$\rho(\mathcal{K}(a\Delta T)) = \frac{|e^{a\Delta T} - R(a\Delta T)|}{1 - |R(a\Delta T)|}. \tag{4.17}$$

The parareal iteration operator is characterized by the elements a_i in its first column. From (4.10) we have

$$a_i = (e^{a\Delta T} - R(a\Delta T))M_{i,1}(a\Delta T), \ i = 1, 2, \ldots .$$

Hence, using (4.3) and (4.16) we find

$$\begin{aligned}
\rho(\mathcal{K}(a\Delta T)) &= \max_{|z|=1} \left| (e^{a\Delta T} - R(a\Delta T)) \sum_{i=2}^{\infty} R(a\Delta T)^{i-2} z^{i-1} \right| \\
&= \max_{|z|=1} \frac{|e^{a\Delta T} - R(a\Delta T)|}{|1 - R(a\Delta T)z|} \\
&= \max_{\theta\in[0,2\pi]} \frac{|e^{a\Delta T} - R(a\Delta T)|}{|1 - |R(a\Delta T)|e^{i(\arg R(a\Delta T)+\theta)}|}.
\end{aligned}$$

The maximum is found for $\theta = -\arg R(a\Delta T)$, which proves the result. ∎

4.11. Note that (4.17) is not necessarily smaller than 1. We have that $\rho(\mathcal{K}(a\Delta T)) < 1$ if and only if $|e^{a\Delta T} - R(a\Delta T)| + |R(a\Delta T)| < 1$ for $a \in \mathbb{R}_0^-$. This is equivalent to the condition

$$\frac{e^{a\Delta T} - 1}{2} < R(a\Delta T) < \frac{e^{a\Delta T} + 1}{2}. \tag{4.18}$$

Note that (4.18) is essentially equivalent to the stability condition derived in [29].

4.5. Discussion of the fully discrete case. We consider the case where instead of an exact solution on the subintervals an approximate solver is used as the F-propagator. Typically, this would be a fine grid approximation obtained by repeated application of the G-propagator on a series of small time steps. Alternatively, it could also be the application of one or more steps of a method of higher order than the G-propagator. In those cases the proofs of the theorems derived in sections 4.3 and 4.4 remain valid with some minor modifications. It suffices to replace the term $e^{a\Delta T}$ representing the exact solution by a term of the form $R_f(a\Delta T)$. Here, the function $R_f(z)$ denotes the stability function of the F-propagator as a solver for (4.1) over an interval in time of length ΔT. For example, if the approximation is obtained by M steps of the fine solver, we have $R_f(a\Delta T) = R(a\Delta T/M)^M$.

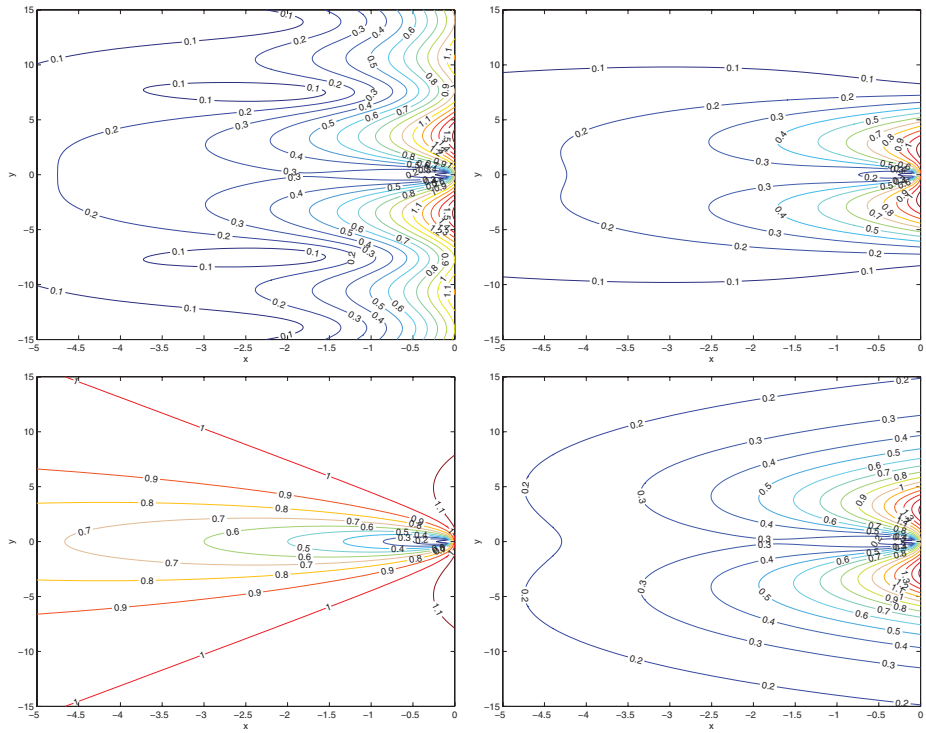


FIG. 4.1. *Contour plots of the convergence factor* $\rho(\mathcal{K}(z))$ *with* $z := a\Delta T \in \mathbb{C}^-$, *for* 4 *different F-propagator/G-propagator combinations: exact solution/backward Euler (top left), backward Euler* ($M = 10$)*/backward Euler (top right), trapezoidal rule* ($M = 1$)*/backward Euler (bottom left), and Radau* IIA ($M = 1$)*/backward Euler (bottom right).*

In Figure 4.1 on the top left, we show a contour plot of the function $\rho(\mathcal{K}(z))$ for $z := a\Delta T \in \mathbb{C}^-$, as defined in (4.17), with $R(z)$ the stability function of the backward Euler method. Contour plots of the linear convergence factor of the fully discrete parareal operator, i.e.,

$$\rho(\mathcal{K}(z)) = \frac{|R_f(z) - R(z)|}{1 - |R(z)|}, \tag{4.19}$$

as a function of $z \in \mathbb{C}^-$, are given in the other three parts of the figure. Each of these illustrates a combination of the backward Euler scheme as the G-propagator with a different F-propagator. On the top right, the effect of using an F-propagator consisting of ten backward Euler steps is shown. On the bottom left, the case of a single step of the trapezoidal rule is illustrated, while on the bottom right, the case of a single step of the three-stage Radau IIA method is considered. Note that when the number M is increased and, hence, the F-propagator becomes more accurate, the value of the fully discrete convergence factor will approach that of (4.17).

Of special interest is the case $a \in \mathbb{R}_0^-$. For this case, four combinations of propagators are considered in Figure 4.2. The top left picture illustrates the use of backward Euler as a coarse solver, and M steps of backward Euler as a fine solver for $M = 2, \ldots, 10$. The top curve in that picture corresponds to an exact fine solve. On the top right, we show the same plot with Radau IIA as a fine solver. On the bottom left, we show the trapezoidal rule as fine solver. On the bottom right, we illustrate

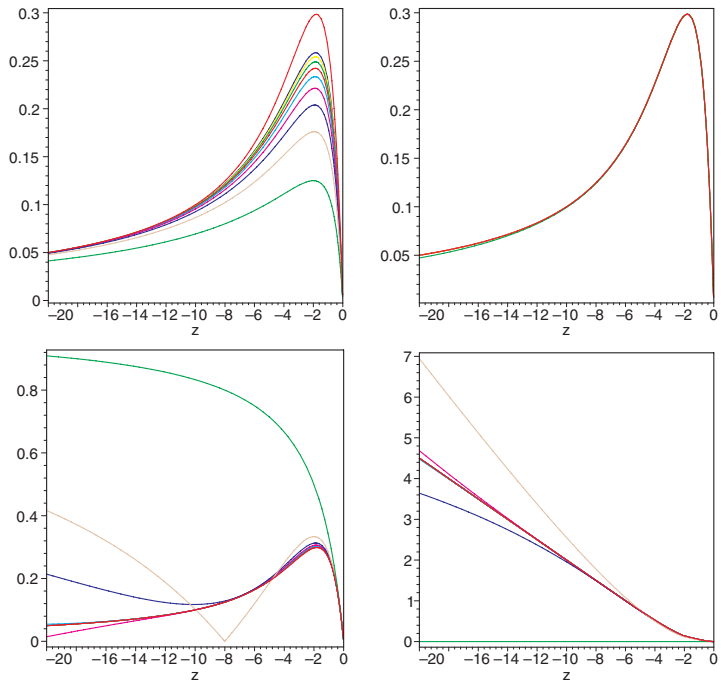


FIG. 4.2. *Value of the fully discrete convergence factor* $\rho(\mathcal{K}(z))$ *with* $z := a\Delta T \in \mathbb{R}_0^-$ *for* 4 *different* F*-propagator/*G*-propagator combinations: backward Euler/backward Euler (top left), Radau* IIA*/backward Euler (top right), trapezoidal rule/backward Euler (bottom left), and trapezoidal rule/trapezoidal rule (bottom right).*

the case where the trapezoidal rule is used for both the coarse and fine solve.

REMARK 4.12. These experiments indicate that for certain combinations, the continuous result derived in the earlier sections is an upper bound for the fully discretized algorithm. This can be shown rigorously in certain cases, but does not hold in general. For example, for the backward Euler scheme we have that $R(a\Delta T) = 1/(1 - a\Delta T)$. For $a \in \mathbb{R}_0^-$ it can easily be verified that

$$\frac{|R(a\Delta T/M)^M - R(a\Delta T)|}{1 - |R(a\Delta T)|} \leq \frac{|e^{a\Delta T} - R(a\Delta T)|}{1 - |R(a\Delta T)|}.$$

Hence, in that case the convergence rate of the fully discrete parareal operator is bounded by the convergence rate of the parareal solver with an exact solve as F-propagator. This is illustrated in the top left picture of Figure 4.2. This does not, however, continue to hold for all $a \in \mathbb{C}^-$.

REMARK 4.13. The stability function of the trapezoidal rule is known to be $R(z) = \frac{1+z/2}{1-z/2}$. From this, it is readily seen that the convergence rate of the combination trapezoidal rule/backward Euler tends to 1 when $z \to -\infty$. For a fixed value of $a\Delta T$, however, the convergence rate tends to the continuous limit with increasing m. This can be observed in the bottom left picture of Figure 4.2. For the trapezoidal rule/trapezoidal rule combination in the bottom right picture of Figure 4.2, the convergence rate actually becomes bigger than 1 for z sufficiently negative.

REMARK 4.14. The necessary and sufficient condition for convergence on infinite length time windows as given in Remark 4.11 can be generalized to the fully discrete

case. With (4.19), we see that convergence is guaranteed for $a \in \mathbb{R}_0^-$ if and only if $|R_f(a\Delta T) - R(a\Delta T)| + |R(a\Delta T)| < 1$. This, in turn, is equivalent to the conditions

$$-1 < R_f(a\Delta T) < 1 \quad \text{and} \quad \frac{R_f(a\Delta T) - 1}{2} < R(a\Delta T) < \frac{R_f(a\Delta T) + 1}{2}. \tag{4.20}$$

The first condition is equivalent to a requirement of stability for the F-propagator and will naturally be satisfied.

5. Convergence analysis for partial differential equations. We now use the results derived in section 4 to investigate the performance of the parareal algorithm on PDEs. We consider two model problems, a diffusion problem and an advection problem.

5.1. Convergence analysis for a diffusion equation. For the diffusion case, we consider the heat equation, without loss of generality in one dimension,

$$u_t = u_{xx} \quad \text{in } \Omega = \mathbb{R}, \quad u(0,x) \in L^2(\Omega). \tag{5.1}$$

Using a Fourier transform in space, this equation becomes a system of decoupled ODEs for each Fourier mode ω,

$$\hat{u}_t = -\omega^2 \hat{u}, \tag{5.2}$$

and hence the convergence results for scalar ODEs can be directly applied. In the theorem below, we assume that the heat equation is discretized in time using a one-step method that is stable on the negative real axis (A_0-stable). In the present case, this is a minimal necessary requirement for any time-integrator to produce meaningful results.

THEOREM 5.1. $F(t_{n+1}, t_n, U_n^k)$ t_{n+1} (5.1) $u(t_n) = U_n^k$ $G(t_{n+1}, t_n, U_n^k)$ A_0

$$\max_{1\le n\le N} ||u(t_n) - U_n^k||_2 \le \frac{\gamma_s^k}{k!} \prod_{j=1}^{k} (N-j) \max_{1\le n\le N} ||u(t_n) - U_n^0||_2, \tag{5.3}$$

$||\cdot||_2$ γ_s $|R(-\infty)| = c < 1$

$$\sup_{n>0} ||u(t_n) - U_n^k||_2 \le \gamma_l^k \sup_{n>0} ||u(t_n) - U_n^0||_2, \tag{5.4}$$

γ_l The numerator in the superlinear bound (4.12) when applied to (5.2) is a continuous function of $z := -\omega^2 \Delta T$. It is zero at $z = 0$ and, because of the A_0-stability assumption, it is bounded for $z \to -\infty$ along the real axis. Hence, the numerator is uniformly bounded, and we can define its supremum as

$$\gamma_s := \sup_{z \in \mathbb{R}^-} |e^z - R(z)|. \tag{5.5}$$

This leads to (5.3) by using the Parseval–Plancherel identity.

TABLE 5.1
Superlinear and linear convergence constants.

	Backward Euler	Trapezoidal rule	SDIRK	Radau IIA
γ_s	0.2036321888	1	0.1717941220	0.0634592650
γ_l	0.2984256075	∞	0.2338191487	0.0677592165

The convergence factor in the linear bound (4.15) applied to (5.2) can be bounded as a function of $z := -\omega^2 \Delta T$,

$$\gamma_l := \sup_{z \in \mathbb{R}^-} \frac{|e^z - R(z)|}{1 - |R(z)|}. \tag{5.6}$$

This bound is finite because of the assumptions on the one-step method. This leads to (5.4) using the Parseval–Plancherel identity. □

The constants γ_s and γ_l can easily be computed numerically. The values of these constants for the backward Euler method, the trapezoidal rule, the two-stage singly diagonally implicit Runge–Kutta (SDIRK) method, and the three-stage Radau IIA method are presented in Table 5.1. The assumptions on the one-step method specified in the theorem are necessary but not sufficient to guarantee convergence of the parareal algorithm as a solver on unbounded time intervals. That is, the constant γ_l may well be greater than 1. A necessary and sufficient condition for convergence follows from Remark 4.11.

COROLLARY 5.2.

$$\frac{e^z - 1}{2} < R(z) < \frac{e^z + 1}{2} \qquad z \in \mathbb{R}^-. \tag{5.7}$$

5.3. A necessary and sufficient condition for convergence in the fully discrete case is found by replacing e^z by $R_f(z)$ in (5.7) and by adding the condition $|R_f(z)| < 1$.

5.4. In practice, the heat equation is usually posed on a finite spatial domain, and a spatial discretization is applied. If the spatial discretization is such that the resulting ODE system has only negative real eigenvalues, the bound derived in Theorem 5.1 continues to hold, as it is a uniform bound for all $-\omega^2 \Delta T$.

5.5. From the results in Table 5.1, we see that the SDIRK and Radau IIA methods have better convergence factors than the lower order methods. This shows that using these higher order methods on the coarse grid is speeding up the convergence of the parareal algorithm. However, this comes at the cost of a more expensive evaluation of the G-propagator.

5.2. Convergence of parareal for an advection equation. Next, we consider a pure advection problem,

$$u_t = u_x \quad \text{in } \Omega = \mathbb{R}, \quad u(0, x) \in L^2(\Omega). \tag{5.8}$$

Using a Fourier transform in space, this equation becomes a decoupled system of differential equations for each Fourier mode ω,

$$\hat{u}_t = -i\omega \hat{u}. \tag{5.9}$$

TABLE 5.2
Superlinear and linear convergence constants for the advection equation.

	Backward Euler	Trapezoidal rule	SDIRK	Radau IIA
α_s	1.224353426	2	1.185652097	1.362526017
α_l	1.632645559	∞	∞	2.231320732

We will again apply the convergence results obtained earlier for scalar ODEs. We assume that the advection equation is discretized in time using a one-step method that is stable on the whole of the imaginary axis. In the present case, this is a necessary requirement for the method to produce bounded results.

THEOREM 5.6. $F(t_{n+1}, t_n, U_n^k)$ (5.9) $u(t_n) = U_n^k$ $G(t_{n+1}, t_n, U_n^k)$ $|R(z)| \leq 1$ z

$$\max_{1\leq n\leq N} ||u(t_n) - U_n^k||_2 \leq \frac{\alpha_s^k}{k!} \prod_{j=1}^{k} (N-j) \max_{1\leq n\leq N} ||u(t_n) - U_n^0||_2, \tag{5.10}$$

α_s The numerator in the superlinear bound (4.12) when applied to (5.2) is a continuous function of $z := -\omega\Delta T$. It is zero at $z = 0$ and, because of the boundedness of $|R(z)|$ along the imaginary axis, it is bounded at infinity. Hence, the numerator is uniformly bounded, and we can define its supremum as

$$\alpha_s := \sup_{z\in\mathbb{R}} |e^{iz} - R(iz)|. \tag{5.11}$$

This leads to (5.10) by using the Parseval–Plancherel identity. □

5.7. Typically, there is no convergence result for (5.8) on unbounded intervals. Formally, we can compute the constant α_l defined as

$$\alpha_l := \sup_{z\in\mathbb{R}} \frac{|e^{iz} - R(iz)|}{1 - |R(iz)|}. \tag{5.12}$$

However, this value will mostly be greater than 1, if not infinite.

As before, the constants α_s and α_l can be computed numerically. The values of these constants for a number of one-step methods are given in Table 5.2.

5.8. In practice, the advection problem is usually posed on a finite spatial interval, and a spatial discretization is applied. If the spatial discretization is such that the resulting ODE system has only purely imaginary eigenvalues, the bound derived in Theorem 5.6 continues to hold, as it is a uniform bound for all $i\omega\Delta T$.

5.9. The bound in Theorem 5.6 needs to become small for k less than N for the algorithm to be interesting. To see if this is possible, we study the function

$$g(\alpha, N, k) := \frac{\alpha^k}{k!} \prod_{j=1}^{k} (N-j) = \frac{(-\alpha)^k \, \Gamma(k+1-N)}{k! \, \Gamma(1-N)},$$

where Γ denotes the Gamma-function. In Figure 5.1, we show, for two values of α and various values of N, how the function $g(\alpha, N, k)$ becomes small as k grows from 1 up to N. This shows that even though there is superlinear convergence in the advection

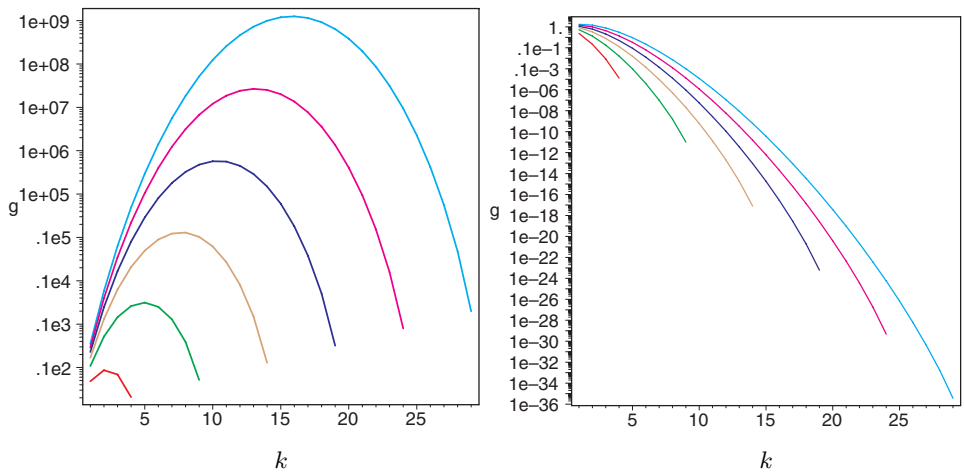


FIG. 5.1. *Behavior of the superlinear convergence factor for* $\alpha = 1.2$ *on the left and* $\alpha = 0.06$ *on the right, as a function of the iteration number* k*, for the values of* $N = 5, 10, 15, 20, 25, 30$.

case, the constants α_s of the various methods do not permit a speedup of the parareal algorithm. In the diffusion case, however, there is substantial speedup, as the constant for the Radau IIA method shows.

. . .. 5.10. If one considers the pure advection case on a sufficiently bounded domain with a Dirichlet boundary condition, it is the boundary condition which largely determines the solution. In such a case, our numerical experiments show that the parareal algorithm converges linearly.

6. Numerical experiments. In order to illustrate the theoretical results, we first show some numerical experiments for the scalar model problem (4.1) with $f = 0$, $a = -1$, $u_0 = 1$. These results are graphically displayed in Figure 6.1. The backward Euler method is chosen for both the coarse approximation and the fine approximation, with time step ΔT and $\Delta T/M$, respectively. In the top left, top right, and bottom left pictures, we show the convergence results obtained for $T = 1$, $T = 10$, and $T = 50$, respectively, using $N = 10$ and $M = 20$ in each case. In each picture the L^∞-norm of the true error is plotted as a function of the iteration index k, together with the bounds as presented in Proposition 4.1 (earlier bound), in Corollary 4.6 (superlinear bound) and in Theorem 4.9 (linear bound).

One can clearly see that parareal has two different convergence regimes: for $T = 1$, the algorithm converges superlinearly, and the superlinear bound is quite sharp; for $T = 10$, the convergence rate is initially linear, and then a transition occurs to the superlinear convergence regime. Finally, for $T = 50$, the algorithm is in the linear convergence regime and the linear bound from Theorem 4.9 is quite sharp. Note also that the earlier bound from Proposition 4.1 indicates stagnation for $T = 10$, since $\Delta T = 1$, and divergence for $T = 50$, since then $\Delta T > 1$. The parareal algorithm does, however, also converge for $\Delta T \geq 1$.

From our analysis, one can also see that the convergence factor is small for small ΔT, then becomes bigger and reaches a maximum, and finally becomes smaller again for large ΔT; see Figure 4.2, for example. This effect is demonstrated in the bottom right picture of Figure 6.1 for the same model problem as before. Again, the backward Euler discretization is used for both the coarse and fine solver. For each case the same F-propagator is used, defined by $\Delta t = 1/20$.

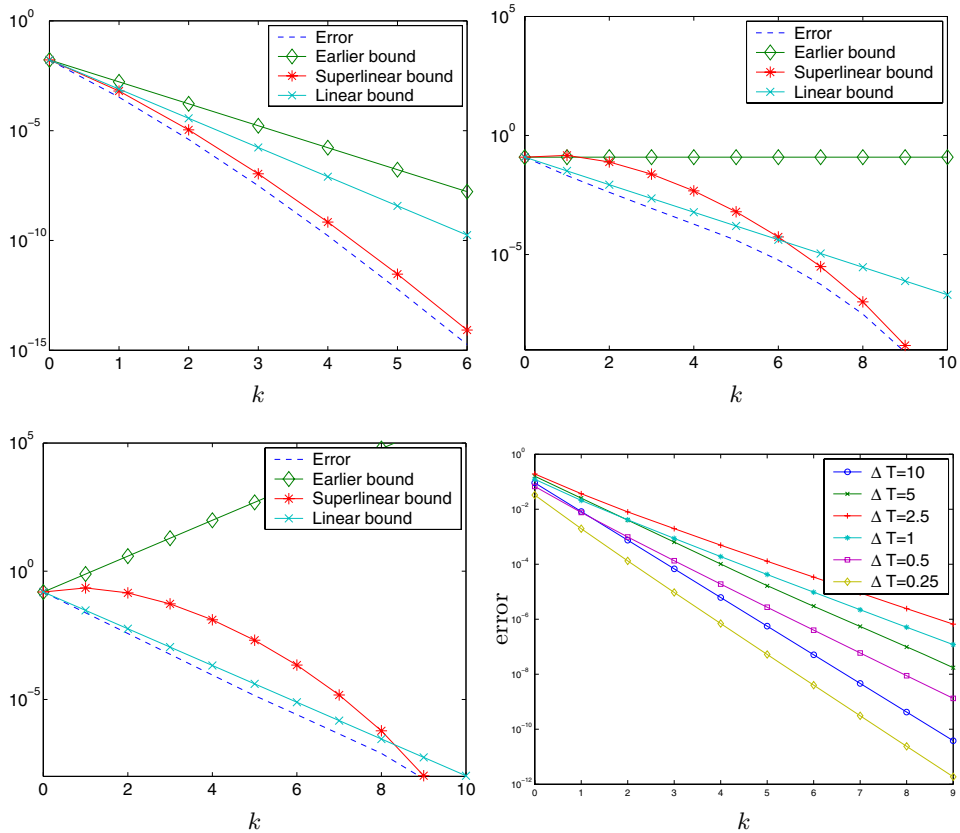


FIG. 6.1. *Parareal convergence for* (4.1): L^∞*-norm of the error as a function of the iteration index* k *on a short (top left), medium (top right), and long (bottom left) time interval; dependence of the convergence speed on* ΔT *(bottom right).*

We now turn our attention to the PDE case and show some experiments for the heat equation $u_t = u_{xx} + f$ in $(0, L) \times (0, T]$ with homogeneous initial and boundary conditions and with $f = x^4(1-x) + t^2$. We use backward Euler for the integration in time and choose the domain size L such that the problem contains a frequency where the linear convergence bound for backward Euler is attained; see Figure 4.2, top left. With $\Delta T = 1/2$ and $M = 10$ we obtain the results shown in Figure 6.2. On the left, results are shown for $T = 4$, where the algorithm with $\Delta T = 1/2$ will converge in 8 steps. One can see that this is clearly the case. Before that, the algorithm is in the superlinear convergence regime, as predicted by the superlinear bound. Note that the latter bound indicates zero as the error at the eighth step and thus cannot be plotted on the logarithmic scale. On the right, the error is shown for $T = 8$, and the algorithm is now in the linear convergence regime.

We finally show four experiments for the advection equation $u_t = u_x + f$ in $(0, L) \times (0, T]$. To emulate the situation of an unbounded domain, we impose in the first two experiments periodic boundary conditions. We use as initial condition $u(x, 0) = e^{-20(x-\frac{L}{2})^2}$ and the forcing function $f \equiv 0$. With $\Delta T = 1/2$, $T = 4$, and $M = 30$, we obtain the results in Figure 6.3 on the left.

In the top left panel of Figure 6.3, we have chosen the domain size L for the worst-case behavior of our superlinear bound, and in the bottom left panel of Figure 6.3 we

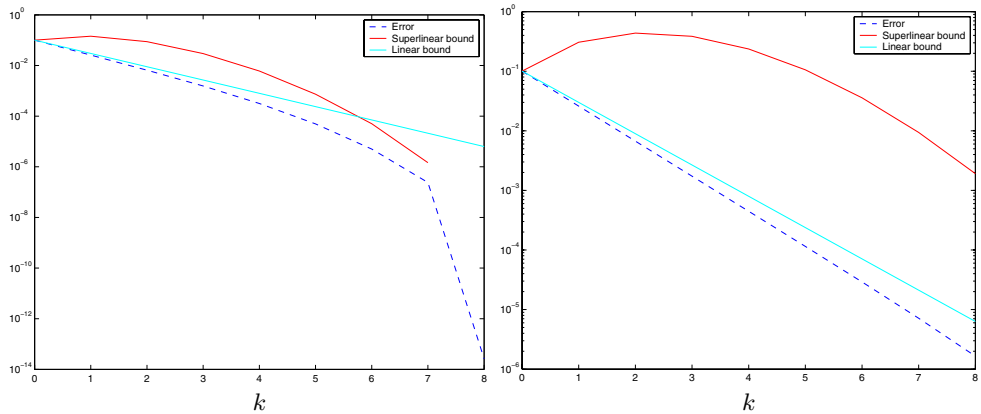


FIG. 6.2. *Error in the L^∞-norm in time and L^2-norm in space for the parareal algorithm applied to the heat equation, on a short (left) and long (right) interval.*

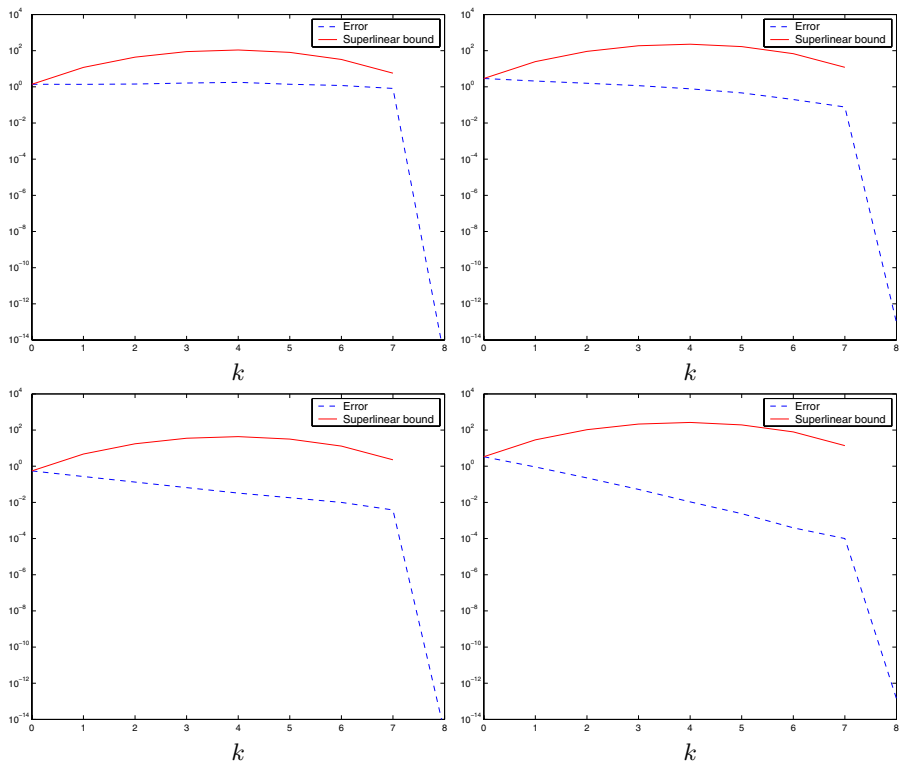


FIG. 6.3. *Error in the L^∞-norm in time and L^2-norm in space for the parareal algorithm applied to the advection equation, with periodic boundary conditions in the graphs on the left, and a Dirichlet condition in the graphs on the right. In the top row the domain size L is chosen for worst-case behavior, and in the bottom row $L = \frac{1}{2}$.*

set $L = \frac{1}{2}$. In the former case, the algorithm stagnates and converges only in step 8, when the fine solution has been computed through sequentially. In the latter case we observe slow, linear convergence. We now repeat the same two experiments but with a boundary condition $u(L,t) = \sin(5t)$. As one can see in Figure 6.3 on the right,

the boundary condition improves the behavior of the parareal algorithm applied to the advection equation, as discussed in Remark 5.10. This is expected based on the information propagation along characteristics.

7. Concluding remarks. We have pointed out how the parareal algorithm relates to some earlier algorithms that have appeared in the literature on time-parallel time-integration methods. We showed that parareal can be seen as a variant of the multiple shooting method, with a Jacobian matrix that is computed approximately by finite differences on a coarse mesh. As a Newton-type method for solving a nonlinear system of equations, its superlinear convergence on bounded time intervals is not unexpected. This convergence behavior was studied in detail in section 4.3.

We also showed that the algorithm can be rewritten as a two-level multigrid-in-time method of full approximation storage type. It is characterized by a possibly aggressive coarsening in time: the coarsening ratio M is not restricted to a small integer for the method to be effective. The algorithm uses a rather unusual smoother which corresponds to a single phase in a two-color relaxation scheme, combined with most simple restriction and prolongation operators. As a multigrid method for general time-dependent problems, parareal can be interpreted as a type of parabolic multigrid method, a class of solvers for parabolic PDEs that includes the space-time multigrid, time-parallel multigrid, and multigrid waveform relaxation methods. All of these can be written in the framework of formula (3.2) and differ only in the details of the multigrid components. With those methods, parareal has in common the linear convergence rate on long time intervals, as studied in section 4.4. In contrast to those methods, which are only effective for parabolic PDEs, parareal is applicable to a much wider range of problems.

Recently new parareal variants have appeared, e.g., genuinely multigrid versions, with more than two grid levels and methods that combine coarsening in time with coarsening in space. Promising results with such methods have already been reported in the literature, e.g., in [9, 11]. Possibly, insights from the present paper may prove to be useful in the study of the theoretical properties of these new algorithms; perhaps, they may direct us to still more general or effective time-parallel time-integration methods. This will be the subject of further investigation.

REFERENCES

[1] L. Baffico, S. Bernard, Y. Maday, G. Turinici, and G. Zérah, *Parallel-in-time molecular-dynamics simulations*, Phys. Rev. E, 66 (2002), pp. 057706–1–4.

[2] G. Bal, *On the convergence and the stability of the parareal algorithm to solve partial differential equations*, in Proceedings of the 15th International Domain Decomposition Conference, Lect. Notes Comput. Sci. Eng. 40, R. Kornhuber, R. H. W. Hoppe, J. Péeriaux, O. Pironneau, O. B. Widlund, and J. Xu, eds., Springer, Berlin, 2003, pp. 426–432.

[3] G. Bal and Y. Maday, *A "parareal" time discretization for non-linear PDE's with application to the pricing of an American put*, in Recent Developments in Domain Decomposition Methods, Lect. Notes Comput. Sci. Eng. 23, Springer, Berlin, 2002, pp. 189–202.

[4] A. Bellen and M. Zennaro, *Parallel algorithms for initial value problems for difference and differential equations*, J. Comput. Appl. Math., 25 (1989), pp. 341–350.

[5] A. Brandt, *Multi-level adaptive solutions to boundary value problems*, Math. Comp., 31 (1977), pp. 333–390.

[6] P. Chartier and B. Philippe, *A parallel shooting technique for solving dissipative ODEs*, Computing, 51 (1993), pp. 209–236.

[7] A. Deshpande, S. Malhotra, C. C. Douglas, and M. H. Schultz, *A rigorous analysis of time domain parallelism*, Parallel Algorithms Appl., 6 (1995), pp. 53–62.

[8] C. Farhat and M. Chandesris, *Time-decomposed parallel time-integrators: Theory and feasibility studies for fluid, structure, and fluid-structure applications*, Internat. J. Numer. Methods Engrg., 58 (2003), pp. 1397–1434.

[9] P. F. Fischer, F. Hecht, and Y. Maday, *A parareal in time semi-implicit approximation of the Navier-Stokes equations*, in Proceedings of the 15th International Domain Decomposition Conference, Lect. Notes Comput. Sci. Eng. 40, R. Kornhuber, R. H. W. Hoppe, J. Péeriaux, O. Pironneau, O. B. Widlund, and J. Xu, eds., Springer, Berlin, 2003, pp. 433–440.

[10] I. Garrido, M. S. Espedal, and G. E. Fladmark, *A convergence algorithm for time parallelization applied to reservoir simulation*, in Proceedings of the 15th International Domain Decomposition Conference, Lect. Notes Comput. Sci. Eng. 40, R. Kornhuber, R. H. W. Hoppe, J. Péeriaux, O. Pironneau, O. B. Widlund, and J. Xu, eds., Springer, Berlin, 2003, pp. 469–476.

[11] I. Garrido, B. Lee, G. E. Fladmark, and M. E. Espedal, *Convergent iterative schemes for time parallelization*, Math. Comp., 26 (2006), pp. 1403–1428.

[12] W. Hackbusch, *Parabolic multigrid methods*, in Computing Methods in Applied Sciences and Engineering, VI, R. Glowinski and J.-L. Lions, eds., North–Holland, Amsterdam 1984, pp. 189–197.

[13] G. Horton, *The time-parallel multigrid method*, Comm. Appl. Numer. Methods, 8 (1992), pp. 585–595.

[14] G. Horton and S. Vandewalle, *A space-time multigrid method for parabolic partial differential equations*, SIAM J. Sci. Comput., 16 (1995), pp. 848–864.

[15] G. Horton, S. Vandewalle, and P. Worley, *An algorithm with polylog parallel complexity for solving parabolic partial differential equations*, SIAM J. Sci. Comput., 16 (1995), pp. 531–541.

[16] H. B. Keller, *Numerical Methods for Two-Point Boundary Value Problems*, Blaisdell, Waltham, MA, 1968.

[17] M. Kiehl, *Parallel multiple shooting for the solution of initial value problems*, Parallel Comput., 20 (1994), pp. 275–295.

[18] R. Kornhuber, R. H. W. Hoppe, J. Péeriaux, O. Pironneau, O. B. Widlund, and J. Xu, eds., *Proceedings of the 15th International Domain Decomposition Conference*, Lect. Notes Comput. Sci. Eng. 40, Springer, Berlin, 2003.

[19] J.-L. Lions, Y. Maday, and G. Turinici, *A "parareal" in time discretization of PDE's*, C. R. Acad. Sci. Paris Sér. I Math., 332 (2001), pp. 661–668.

[20] C. Lubich and A. Ostermann, *Multi-grid dynamic iteration for parabolic equations*, BIT, 27 (1987), pp. 216–234.

[21] Y. Maday and G. Turinici, *A parareal in time procedure for the control of partial differential equations*, C. R. Math. Acad. Sci. Paris. Sér. I Math., 335 (2002), pp. 387–391.

[22] Y. Maday and G. Turinici, *The parareal in time iterative solver: A further direction to parallel implementation*, in Proceedings of the 15th International Domain Decomposition Conference, Lect. Notes Comput. Sci. Eng. 40, R. Kornhuber, R. H. W. Hoppe, J. Péeriaux, O. Pironneau, O. B. Widlund, and J. Xu, eds., Springer, Berlin, 2003, pp. 441–448.

[23] W. L. Miranker and W. Liniger, *Parallel methods for the numerical integrating of ordinary differential equations*, Math. Comp., 91 (1967), pp. 303–320.

[24] J. Nievergelt, *Parallel methods for integrating ordinary differential equations*, Comm. ACM, 7 (1964), pp. 731–733.

[25] L. Reichel and L. N. Trefethen, *Eigenvalues and pseudo-eigenvalues of Toeplitz matrices*, Linear Algebra Appl., 162/164 (1992), pp. 153–185.

[26] J. H. Saltz and V. K. Naik, *Towards developing robust algorithms for solving partial differential equations on MIMD machines*, Parallel Comput., 6 (1988), pp. 19–44.

[27] L. F. Shampine and H. A. Watts, *Block implicit one-step methods*, Math. Comp., 23 (1969), pp. 731–740.

[28] R. D. Skeel, *A theoretical framework for proving accuracy results for deferred corrections*, SIAM J. Numer. Anal., 19 (1982), pp. 171–96.

[29] G. A. Staff and E. M. Rønquist, *Stability of the parareal algorithm*, in Proceedings of the 15th International Domain Decomposition Conference, Lect. Notes Comput. Sci. Eng. 40, R. Kornhuber, R. H. W. Hoppe, J. Péeriaux, O. Pironneau, O. B. Widlund, and J. Xu, eds., Springer, Berlin, 2003, pp. 449–456.

[30] K. Stueben, *A review of algebraic multigrid*, J. Comput. Appl. Math., 198 (2001), pp. 281–309.

[31] S. Vandewalle and E. V. de Velde, *Space-time concurrent multigrid waveform relaxation*, Ann. Numer. Math., 1 (1994), pp. 347–360.

[32] S. Vandewalle and R. Piessens, *Efficient parallel algorithms for solving initial-boundary value and time-periodic parabolic partial differential equations*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 1330–1346.

[33] D. E. Womble, *A time-stepping algorithm for parallel computers*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 824–837.

SIAM J. SCI. COMPUT.
Vol. 29, No. 2, pp. 579–597
© 2007 Society for Industrial and Applied Mathematics

HIGH-RESOLUTION FLUX-BASED LEVEL SET METHOD*

PETER FROLKOVIČ† AND KAROL MIKULA‡

Abstract. A new high-resolution flux-based finite volume method for general advection equations in nondivergent form including a level set equation for moving interfaces is introduced. The method is applicable to the case of nondivergence free velocity and to general unstructured grids in higher dimensions. We show that the method is consistent and that the numerical solution fulfills the discrete minimum/maximum principle. Numerical experiments show its second order accuracy for smooth solutions as well as for solutions with discontinuous derivatives and on general unstructured meshes. Numerical examples for passive transport and shrinking of dynamic interfaces, including examples with topological changes, are presented using locally adapted two-dimensional and three-dimensional grids.

Key words. level set method, finite volume method, high resolution, evolving interfaces, unstructured grids

AMS subject classifications. 65M25, 76M12

DOI. 10.1137/050646561

1. Introduction.

The basic task of level set methods is to describe the evolution of dynamic interfaces. Particular forms of the level set equations and corresponding computational simulations have a tremendous number of different applications; see [31, 32, 38, 39] for details and references.

The level set equation for the moving interface advected by a velocity field $\vec{V}$ can be written in the form

$$\partial_t \phi + \vec{V} \cdot \nabla \phi = 0\,, \quad \phi(t^0, x) = \phi^0(x)\,, \tag{1.1}$$

where the level set function $\phi = \phi(t, x)$ describes the evolving interface implicitly by its zero level set (contour line in two dimensions and isosurface in three dimensions).

Depending on the type of application, the velocity $\vec{V}$ in (1.1) can be given in several ways. For instance, in tracking of dynamic interfaces between two immiscible fluids, the velocity field $\vec{V} = \vec{V}(x)$ describes some external fluid movement; see [44, 45, 16, 17, 24] for some examples. Typically, $\vec{V}$ is obtained as a numerical solution of some PDE describing a conservation law for fluid flow, e.g., $\nabla \cdot \vec{V} = 0$.

Further applications result from problems of geometric evolution equations, appearing among others in crystal growth, combustion, or image segmentation, where $\vec{V} = \delta \vec{N}$ with $\vec{N}$ being the unit normal vector to the interface given by $\vec{N} = \nabla\phi/|\nabla\phi|$, and $\delta = \delta(x)$ describes the speed of motion in normal direction $\vec{N}$. In this case, (1.1) is usually written in the form

$$\partial_t \phi + \delta |\nabla \phi| = 0\,, \tag{1.2}$$

*Received by the editors December 2, 2005; accepted for publication (in revised form) October 5, 2006; published electronically March 30, 2007.
http://www.siam.org/journals/sisc/29-2/64656.html
†Simulations in Technology Center, University of Heidelberg, Im Neuenheimer Feld 368, 69120 Heidelberg, Germany (peter.frolkovic@uni-hd.de).
‡Department of Mathematics and Descriptive Geometry, Slovak University of Technology, Radlinskeho 11, 81368 Bratislava, Slovakia (mikula@math.sk). The work of this author was supported by grants VEGA 1/3321/06 and APVT-20-040902.

and even higher order terms such as (mean) curvature or intrinsic Laplacian of curvature can be included in the definition of δ; cf. [39].

Finally, if there is a nonzero right-hand side f in (1.2) and one is searching for its stationary solutions, the so-called Eikonal equation

$$\delta|\nabla\phi| = f, \tag{1.3}$$

with some appropriate boundary conditions, has to be solved.

These important applications of (1.1) with $\vec{V} = \vec{V}(x, \nabla\phi)$ are solved using different numerical techniques. The main objective of this paper is to derive a consistent high-resolution flux-based finite volume method that can be used for such applications (including nondivergence free velocities) on general unstructured computational grids in any space dimension.

Nowadays, the finite volume techniques are very popular discretization methods for conservation laws. They preserve many useful properties of numerical solutions known from the analytical case, such as the mass conservation and the minimum/maximum principle.

The level set equations that model the movement of interface in normal direction are very often solved by the finite difference methods. Such numerical techniques can be described as solutions to the so-called Hamilton–Jacobi equation,

$$\partial_t\phi + H(\nabla\phi) = 0, \tag{1.4}$$

with $H(\nabla\phi) = \delta|\nabla\phi|$ for (1.2). Numerical discretizations of (1.4) can be then formally written in the form

$$\phi_i^{n+1} = \phi_i^n - \Delta t^n \, \mathcal{H}_i(\nabla_i^n\phi)\,, \tag{1.5}$$

where $\mathcal{H}_i$ is numerical Hamiltonian and numerical gradients $\nabla_i^n\phi$ are determined typically by some special finite difference approximations of $\nabla\phi$ in the time-space grid point (t^n, x_i). Popularity of such finite difference level set methods for structured grids dates from [33], and these methods have reached high level quality using the Hamilton–Jacobi essentially nonoscillatory (ENO) [34], weighted ENO (WENO) [19], or high-resolution central schemes [23].

Later, several discretization techniques were developed for two-dimensional unstructured triangular meshes that are described in the form (1.5). In [1, 2] the so-called monotonic Lax–Friedrichs Hamiltonian $\mathcal{H}_i$ was proposed, and similar ideas were used to derive WENO schemes [47] or central WENO schemes [22] for triangular meshes. In [20] the approximative gradients $\nabla_i^n\phi$ in (1.5) are found by an averaging over finite volumes that are dual to finite element mesh, and in [4] by an averaging over finite elements surrounding the nodes. In all these papers, only triangular meshes are considered.

A different approach to solve level set problems is taken by semi-Lagrangian [11, 10, 41] or particle tracking methods [9]. The main idea of such methods is to work directly with the formulation (1.1) by approximating the related characteristic curves. Note that these methods can still be viewed as pointwise (particle) based numerical techniques.

One drawback of all mentioned finite difference based level set methods is that they are not conservative if applied with passive transport having divergence free velocity $\vec{V}$. For such applications, some combinations of level set methods with other methods to improve the mass preservation were proposed, such as reinitialization

techniques to recover numerical signed distance function [42], the coupled level set and volume-of-fluid method [43], or the hybrid particle level set method [8]; see also a comprehensive review in [24]. Recently, a few papers have proposed using standard numerical conservative methods for level set equation (1.1) with divergence free velocity such as finite volume methods [30] or discontinuous Galerkin discretizations [35, 48, 25].

Finite volume or finite element methods, which are very useful for conservation laws on unstructured grids, cannot be used directly for (1.1) or (1.4), because the Hamilton–Jacobi equation "cannot be written in a conservation form, suitable for integration by parts, which is the backbone of finite volume and finite element methods" [47]. A new approach was proposed in [14, 15] to overcome this difficulty, where the level set problem (1.1) was reformulated to a conservation form with a source term. At the same time, a flux-based finite volume scheme using integration by parts was described, and the standard CFL restriction on time step was relaxed using a recursive algorithm.

The flux-based level set method proposed in [15] is based on the first order accurate approximation. In this paper, we extend it to a high-resolution method based on second order approximations using the concept of "high-resolution finite volume methods" as discussed in detail in, e.g., [21]. Our high-resolution scheme involves in a natural way also the first order scheme given in [15] and, in such a way, is able to approximate well not only problems with smooth solutions but also problems that exhibit solutions with discontinuous derivatives. We can prove that the high-resolution flux-based level set method is consistent on general grids and that the discrete minimum/maximum principle is valid under natural assumptions. To make valid such assumptions in general, well-known limiting procedures can be applied.

Several numerical experiments will be given that illustrate the robustness and order of convergence of our new method, including examples with smooth solutions, solutions with discontinuous derivatives, and solutions with topological changes. Among others, unstructured grids made of rectangles and triangles in two dimensions, and hexahedra, pyramids, and tetrahedra in three dimensions are used. No reinitialization techniques are applied to recover a numerical signed distance function. A comparison with an ENO scheme (second order in space and time) using [26] for an example of a rotating solid body shows better performance of our method. Finally, more complex examples of Zalesak's disk rotation and vortex-in-a-box are presented.

The paper is organized as follows. In section 2 we derive a general consistent flux-based finite volume discretization scheme for solving (1.1). In section 3 we present particular (first and second order accurate) discretization schemes. In section 4, experimental order of convergence of the method and various numerical experiments are discussed.

2. General consistent discretization scheme. In this section we derive a finite volume discretization method for the model problem of the advection equation in nondivergent form,

$$\partial_t \phi + \vec{V} \cdot \nabla \phi = 0\,, \quad \phi(t^0, x) = \phi^0(x)\,, \tag{2.1}$$

where the function $\phi = \phi(t, x)$ has to be determined for $t > t^0 \in \mathbf{R}$ and for $x \in \Omega \subset \mathbf{R}^d$ with $d = 2$ or $d = 3$. The domain Ω is supposed to be polygonal with boundary $\partial\Omega$. For the moment, we skip a description of boundary conditions.

The advection equation (2.1) is solved on subintervals (t^n, t^{n+1}) of the time interval of interest and is related to (1.1) by considering $\vec{V}(x) = \vec{V}(x, \nabla\phi(t^n, x))$.

If one introduces characteristic curves $X = X_x^n(t)$ defined in a backward manner,

$$X_x^n(t) = x - \int_{t^n}^{t} \vec{V}(X_x^n(s))\, ds\,,$$

one can express the solution of (2.1) using the simple formula

$$\phi(t,x) = \phi(t^n, X_x^n(t))\,. \tag{2.2}$$

Several numerical methods, such as the modified method of characteristics [7] or semi-Lagrangian methods [10], treat (2.1) by solving (or approximating) the relation (2.2) in each grid point $x = x_i$ of a discrete mesh.

To apply a finite volume method to approximate (2.1), we suppose that Ω is covered by a mesh of nonempty, nonoverlapping, polygonal computational cells (finite volumes) $\Omega_i \subset \Omega$, $i = 1, 2, \ldots, I$. Furthermore, $\Delta t^n = t^{n+1} - t^n$, $n = 0, 1, \ldots$.

There are several types of finite volume schemes for solving advective equations; cf. [21, 3]. The cell-based finite volume method can be obtained by integrating (2.2) over Ω_i in $t = t^{n+1}$; see [21, 37, 27, 36] for some reviews and references. In order to derive the flux-based finite volume discretization, we rewrite (2.1) to the divergent form with nonzero right-hand side,

$$\partial_t \phi + \nabla \cdot \left(\vec{V}\phi\right) = \phi \nabla \cdot \vec{V}\,. \tag{2.3}$$

If one integrates (2.3) over Ω_i and time interval (t^n, t^{n+1}), the following integral equation can be derived:

$$\int_{\Omega_i} \phi(t^{n+1}, x) = \int_{\Omega_i} \phi(t^n, x) - \int_{t^n}^{t^{n+1}} \int_{\partial\Omega_i} \vec{n}_i \cdot \vec{V}\phi(t,\gamma) + \int_{t^n}^{t^{n+1}} \int_{\Omega_i} \phi(t,x) \nabla \cdot \vec{V}\,, \tag{2.4}$$

where $\vec{n}_i = \vec{n}_i(\gamma)$, $\gamma \in \partial\Omega_i$ is the normal outward unit vector. The integral form (2.4) can be viewed formally as a conservation law with a source term. In what follows we will derive a general discretization scheme for solving (2.1) by applying numerical quadrature rules for integrals in (2.4). To do so, let us suppose that the numerical solution at $t = t^n$ is given as a function $\phi^n(x) \approx \phi(t^n, x), x \in \Omega$.

The first integral on the right-hand side of (2.4) is approximated by

$$\int_{\Omega_i} \phi(t^n, x)\, dx \approx |\Omega_i| \phi_i^n\,, \tag{2.5}$$

where $\phi_i^n = \phi^n(x_i)$ for some point $x_i \in \Omega_i$ that will be specified later. An analogous approximation is used for the integral on the left-hand side of (2.4).

Before describing an approximation of the second integral in the right-hand side of (2.4), we introduce some notation. The common boundary (of nonzero measure) of two neighboring finite volumes Ω_i and Ω_j will be denoted by $\Gamma_{ij} := \partial\Omega_i \cap \partial\Omega_j$, $|\Gamma_{ij}| > 0$, and $\Gamma_{i0} := \partial\Omega_i \cap \partial\Omega$, $|\Gamma_{i0}| > 0$. Moreover, we suppose that Γ_{ij} consists of (possibly several) $\Gamma_{ij}^e \subset \Gamma_{ij}$ such that Γ_{ij}^e is a straight line if $d = 2$, or a flat polygon if $d = 3$. Consequently, one has

$$\partial\Omega_i = \bigcup_{(e,j)\in\Lambda_i} \overline{\Gamma_{ij}^e}\,, \quad \Lambda_i := \left\{(e,j) \,:\, |\Gamma_{ij}^e| > 0\right\}.$$

The integral over $\partial\Omega_i$ can now be expressed as the sum of boundary integrals over Γ_{ij}^e. Let $\gamma_{ij}^e \in \Gamma_{ij}^e$ be the barycenter of Γ_{ij}^e, i.e., the integration point of a quadrature

formula for integrals over Γ_{ij}^e that is exact for linear polynomials. In what follows, we will use the approximation

$$\int_{\Omega_i} \nabla \cdot \vec{V}(x)\, dx = \int_{\partial\Omega_i} \vec{n}_i(\gamma) \cdot \vec{V}(\gamma)\, d\gamma \approx \sum_{(e,j)\in\Lambda_i} |\Gamma_{ij}^e| \vec{n}_{ij}^e \cdot \vec{V}_{ij}^e\,, \tag{2.6}$$

where $\vec{n}_{ij}^e := \vec{n}_i(\gamma_{ij}^e) = -\vec{n}_j(\gamma_{ij}^e) = -\vec{n}_{ji}^e$ and $\vec{V}_{ji}^e \equiv \vec{V}_{ij}^e := \vec{V}(\gamma_{ij}^e)$.

Similarly to (2.6), for the second integral in the right-hand side of (2.4), we get the following approximation:

$$\int_{t^n}^{t^{n+1}} \int_{\partial\Omega_i} \vec{n}_i \cdot \vec{V} \phi(t,\gamma) d\gamma dt \approx \Delta t^n \sum_{(e,j)\in\Lambda_i} |\Gamma_{ij}^e| \vec{n}_{ij}^e \cdot \vec{V}_{ij}^e\, \phi_{ij}^e(t^{n+1/2}), \tag{2.7}$$

where

$$\phi_{ij}^e(t^{n+1/2}) \approx \phi^n(X_{\gamma_{ij}^e}^n(t^{n+1/2}))\,, \quad t^{n+1/2} := t^n + \Delta t^n/2\,. \tag{2.8}$$

Finally, the third integral in the right-hand side of (2.4) is approximated by

$$\int_{t^n}^{t^{n+1}} \int_{\Omega_i} \phi(t,x)\, \nabla \cdot \vec{V}(x) \approx \Delta t^n\, \phi_i^{n+1/2} \sum_{(e,j)\in\Lambda_i} |\Gamma_{ij}^e| \vec{n}_{ij}^e \cdot \vec{V}_{ij}^e\,, \tag{2.9}$$

where $\phi_i^{n+1/2}$ approximates the averaged value of $\phi(t^{n+1/2}, x)$, $x \in \Omega_i$. The simplest choice is $\phi_i^{n+1/2} = \phi_i^n$, which can be seen as the first order accurate approximation of the integral in (2.9). Other choices will be introduced later.

Combining approximations (2.5), (2.7), and (2.9), we end up with the following [illegible] for solving (2.1):

$$\phi_i^{n+1} = \phi_i^n - \frac{\Delta t^n}{|\Omega_i|} \sum_{(e,j)\in\Lambda_i} |\Gamma_{ij}^e| \vec{n}_{ij}^e \cdot \vec{V}_{ij}^e \left(\phi_{ij}^e(t^{n+1/2}) - \phi_i^{n+1/2}\right). \tag{2.10}$$

Note that the scheme (2.10) represents a fully explicit discretization method.

Similarly to [6, 4], we define the notion of consistency for a numerical scheme approximating (2.1).

DEFINITION 2.1. [illegible] (2.10) [illegible] (2.1) [illegible] $\vec{V}$ [illegible] $\phi^n(x)$ $x \in \Omega$ [illegible]

$$\phi_i^{n+1} = \phi_i^n - \Delta t^n\, \vec{V} \cdot \nabla \phi^n\,. \tag{2.11}$$

To derive a specific form of general scheme (2.10) that has the consistency property, we use piecewise linear approximating functions $\phi^n(x)$ of the form

$$\phi^n(x) = \phi_i^n + \nabla\phi_i^n \cdot (x - x_i)\,, \quad x \in \Omega_i\,, \tag{2.12}$$

where $\nabla\phi_i^n \approx \nabla\phi^n(x_i)$. Then we introduce a particular form of (2.8) by defining the function $\phi_{ij}^e(t)$ for $t \in [t^n, t^{n+1})$ as follows:

$$\phi_{ij}^e(t) := \begin{cases} \phi_i^n + \nabla\phi_i^n \cdot (\gamma_{ij}^e - x_i - (t-t^n)\vec{V}_i), & \vec{n}_{ij}^e \cdot \vec{V}_{ij}^e > 0, \\ \phi_j^n + \nabla\phi_j^n \cdot (\gamma_{ij}^e - x_j - (t-t^n)\vec{V}_j), & \vec{n}_{ij}^e \cdot \vec{V}_{ij}^e < 0, \end{cases} \tag{2.13}$$

and $\vec{V}_i := \vec{V}(x_i) = \vec{V}(x_i, \nabla\phi_i^n)$. Note that the approximation (2.13) follows the conservation law principle as $\phi_{ij}^e(t) = \phi_{ji}^e(t)$. For the case $\gamma_{i0}^e \in \partial\Omega$, one can extend the definition (2.13) by choosing

$$\phi_{i0}^e(t) := \phi_i^n + \nabla\phi_i^n \cdot \left(\gamma_{i0}^e - x_i - (t - t^n)\vec{V}_i\right), \quad \vec{n}_{i0}^e \cdot \vec{V}_{i0}^e < 0\,. \tag{2.14}$$

LEMMA 2.2. [illegible] (2.13)–(2.14) [illegible] (2.10) [illegible] (2.1) [illegible] After substituting constant vectors $\vec{V}$ and $\nabla\phi^n$ in (2.10) and using (2.13) to obtain $\phi_{ij}^e(t^n) = \phi_i^n + \nabla\phi^n \cdot (\gamma_{ij}^e - x_i) = \phi^n(\gamma_{ij}^e)$, one has

$$\phi_i^{n+1} = \phi_i^n - \frac{\Delta t^n}{|\Omega_i|}\vec{V} \cdot \sum_{(e,j)\in\Lambda_i} |\Gamma_{ij}^e| \vec{n}_{ij}^e \phi^n(\gamma_{ij}^e)$$
$$+ \frac{\Delta t^n}{|\Omega_i|}\left(\frac{\Delta t^n}{2}\vec{V} \cdot \nabla\phi^n + \phi_i^{n+1/2}\right) \sum_{(e,j)\in\Lambda_i} |\Gamma_{ij}^e| \vec{n}_{ij}^e \cdot \vec{V} = \phi_i^n - \Delta t^n \vec{V} \cdot \nabla\phi^n,$$

because of (2.6) and

$$\sum_{(e,j)\in\Lambda_i} |\Gamma_{ij}^e| \vec{n}_{ij}^e \phi^n(\gamma_{ij}^e) = \int_{\partial\Omega_i} \vec{n}_i(\gamma)\phi^n(\gamma)d\gamma = \int_{\Omega_i} \nabla\phi^n dx = |\Omega_i|\nabla\phi^n\,. \qquad \square$$

3. Particular forms of general discretization scheme (2.10).

3.1. First order discretization scheme. Formally, the general discretization scheme (2.10) also covers the case of piecewise constant numerical solution $\phi^n(x)$, when (2.12) takes the simple form

$$\phi^n(x) = \phi_i^n\,, \quad x \in \Omega_i \quad \Rightarrow \quad \nabla\phi^n \equiv \vec{0}\,, \quad x \in \Omega_i\,.$$

Consequently, one obtains from (2.13) that for $t \in [t^n, t^{n+1})$

$$\phi_{ij}^e(t) = \begin{cases} \phi_i^n, & \vec{n}_{ij}^e \cdot \vec{V}_{ij}^e > 0, \\ \phi_j^n, & \vec{n}_{ij}^e \cdot \vec{V}_{ij}^e < 0, \end{cases} \tag{3.1}$$

which is nothing else but the standard first order upwind approximation. If one also chooses $\phi_i^{n+1/2} = \phi_i^n$, the scheme (2.10) takes the form

$$\phi_i^{n+1} = \phi_i^n - \frac{\Delta t^n}{|\Omega_i|} \sum_{(e,j)\in\Lambda_i^{in}} |\Gamma_{ij}^e| \vec{n}_{ij}^e \cdot \vec{V}_{ij}^e \left(\phi_j^n - \phi_i^n\right), \tag{3.2}$$

where

$$\Lambda_i^{in} := \{(e,j) \in \Lambda_i\,,\, \vec{n}_{ij}^e \cdot \vec{V}_{ij}^e < 0\}\,, \quad \Lambda_i^{out} := \Lambda_i \setminus \Lambda_i^{in}\,. \tag{3.3}$$

Of course, one cannot expect in general that the discretization scheme (3.2) is consistent in the sense of Definition 2.1. The lack of consistency can be illustrated by a simple one-dimensional example with $\Omega_i = (x_i - 0.5h_i, x_i + 0.5h_i)$, $\vec{V} = V = \text{const} < 0$, when it can be written as

$$h_i\phi_i^{n+1} = h_i\phi_i^n - \Delta t^n V(\phi_{i+1}^n - \phi_i^n)\,, \tag{3.4}$$

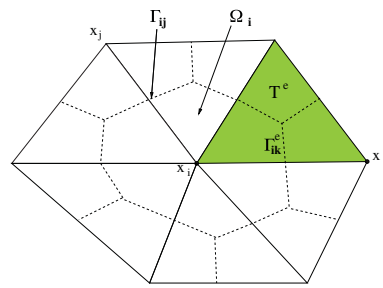


FIG. 3.1. *Illustration of 2D vertex-centered finite volume mesh.*

$i = 1, \ldots, I$. It is easy to see that the method (3.4) is consistent only if $h_i \equiv h$. Similarly, the scheme (3.2) in higher dimensional cases is consistent only for uniform grids with $\Omega_i = (x_i - h/2, x_i + h/2)^d$, i.e., when $|\Omega_i| \equiv h^d$ and $|\Gamma_{ij}| \equiv h^{d-1}$.

Due to stability reasons, the CFL restriction on Δt^n in (3.2) must be fulfilled. It can be formulated by requiring that

$$\Delta t^n \leq \min_i \tau_i \,, \tag{3.5}$$

where τ_i represents the local residence time for Ω_i (or critical time step; cf. [15]) given by

$$\tau_i := -\frac{|\Omega_i|}{V_i^{in}} \,, \quad V_i^{in} := \sum_{(e,j)\in\Lambda_i^{in}} |\Gamma_{ij}^e| \vec{n}_{ij}^e \cdot \vec{V}_{ij}^e \,, \quad i = 1, 2, \ldots, I \,. \tag{3.6}$$

If (3.5) is fulfilled, then the local discrete minimum/maximum principle is valid:

$$\min\{\phi_i^n,\ \phi_j^n,\ j \in \Lambda_i^{in}\} \leq \phi_i^{n+1} \leq \max\{\phi_i^n,\ \phi_j^n,\ j \in \Lambda_i^{in}\} \,. \tag{3.7}$$

The method (3.2) was developed by the authors in [15] and applied on uniform grids to nontrivial examples of curve and surface evolution. The method was realized with the vertex-centered finite volume method that is described in the next section. Moreover, a recursive procedure removing the CFL restriction (3.5) has been introduced to the method in [15].

3.2. High-resolution discretization scheme. First, we reconstruct the piecewise linear approximating function $\phi^n(x)$ in (2.12) by following, e.g., [40]. To do so, we suppose that the finite volume mesh Ω_i, $i = 1, \ldots, I$, is complementary to some conforming finite element mesh $T^e \subset \Omega$ with $e = 1, \ldots, E$ and vertices x_i, $i = 1, \ldots, I$. We use the barycentered form of such vertex-centered control volumes Ω_i, where the polygonal boundary $\partial\Omega_i$ is obtained by connecting the barycenters of elements with middle points of element edges; see, e.g., [12, 3] for some details and Figure 3.1 for an illustration.

The standard finite element interpolation of nodal values ϕ_i^n is used to define

$$\hat{\phi}^n(x) := \sum_{i=1}^{I} \phi_i^n N_i(x) \,, \tag{3.8}$$

where $N_i(x)$ denote finite element basis functions with $N_i(x_j) = \delta_{ij}$.

To reconstruct the piecewise constant gradient of $\phi^n(x)$ in (2.12), we apply

$$\nabla\phi_i^n := \sum_e \frac{\beta_{i,e}^n}{|\Omega_i \cap T^e|} \int_{\Omega_i \cap T^e} \nabla\hat{\phi}^n(x)\, dx\,. \tag{3.9}$$

The preferable choice for $\beta_{i,e}^n \in [0,1]$ (such that $\sum_e \beta_{i,e}^n \leq 1$) is

$$\beta_{i,e}^n = |\Omega_i \cap T^e|/|\Omega_i|\,, \tag{3.10}$$

but other choices can be used in a limiting procedure (see Remark 1).

The velocity $\vec{V}$ is approximated at integration points γ_{ij}^e by $\vec{V}_{ij}^e := \vec{V}(\gamma_{ij}^e, \nabla\hat{\phi}^n(\gamma_{ij}^e))$.

To define $\phi_i^{n+1/2}$ in the general scheme (2.10), we introduce the total outflow and inflow fluxes (see also (3.3)),

$$V_i^{out} := \sum_{(e,j)\in\Lambda_i^{out}} |\Gamma_{ij}^e| \vec{n}_{ij}^e \cdot \vec{V}_{ij}^e\,, \quad V_i^{in} := \sum_{(e,j)\in\Lambda_i^{in}} |\Gamma_{ij}^e| \vec{n}_{ij}^e \cdot \vec{V}_{ij}^e\,. \tag{3.11}$$

Furthermore, if $V_i^{out} > 0$, we define time-dependent averaged outflow value,

$$\phi_i^{out}(t) := \frac{1}{V_i^{out}} \sum_{(e,j)\in\Lambda_i^{out}} |\Gamma_{ij}^e| \vec{n}_{ij}^e \cdot \vec{V}_{ij}^e\, \phi_{ij}^e(t), \tag{3.12}$$

and analogously for averaged inflow value (if $V_i^{in} < 0$),

$$\phi_i^{in}(t) := \frac{1}{V_i^{in}} \sum_{(e,j)\in\Lambda_i^{in}} |\Gamma_{ij}^e| \vec{n}_{ij}^e \cdot \vec{V}_{ij}^e\, \phi_{ij}^e(t). \tag{3.13}$$

Finally, let us define $\phi_i^{n+1/2} = \alpha_i \phi_i^{out}(t^{n+1/2}) + (1-\alpha_i)\phi_i^{in}(t^{n+1/2})$, $\alpha_i \in [0.5, 1]$, and

$$V_i := (1-\alpha_i)V_i^{out} - \alpha_i V_i^{in}\,, \tag{3.14}$$

and the scheme (2.10) can be written as

$$\phi_i^{n+1} = \phi_i^n - \frac{\Delta t^n V_i}{|\Omega_i|} \left(\phi_i^{out}(t^{n+1/2}) - \phi_i^{in}(t^{n+1/2})\right)\,. \tag{3.15}$$

The preferable choice of α_i in (3.14) is $\alpha_i = 0.5$. Note that the parameters α_i in (3.14) and β_i in (3.9) can be chosen independently of each other.

Denoting the residence time $\tau_i := |\Omega_i|/V_i$, the scheme (3.15) also can be written in the form

$$\phi_i^{n+1} = \phi_i^n - \Delta t^n \frac{\phi_i^{out}(t^{n+1/2}) - \phi_i^{in}(t^{n+1/2})}{\tau_i}\,.$$

Note that the first order accurate method (3.2) can be obtained from (3.15) by setting $\beta_{i,e}^n = 0$ in (3.9) and $\alpha_i = 1$ in (3.14), $i = 1, \ldots, I$. Such a form of (3.15) can be used only locally, when it takes the form

$$\phi_i^{n+1} = \phi_i^n - \frac{\Delta t^n V_i^{in}}{|\Omega_i|} \left(\phi_i^n - \phi_i^{in}(t^{n+1/2})\right)\,. \tag{3.16}$$

We choose (3.16) instead of (3.15) for special cases $V_i^{out} = 0$ or $V_i^{in} = 0$, when (3.15) is not well defined. Note that for $V_i^{in} = 0$, (3.16) turns to $\phi_i^{n+1} = \phi_i^n$.

The following local discrete minimum/maximum principle can be proved for ϕ_i^{n+1} by showing that (3.15) and (3.16) can be viewed as positive coefficient schemes [3, 4].

THEOREM 3.1.

$$\phi_{i,\min}^n := \min\{\phi_i^n, \{\phi_j^n, (j,e) \in \Lambda_i \quad \exists (k,e) \in \Lambda_i^{in}\}\},$$

$$\phi_{i,\max}^n := \max\{\phi_i^n, \{\phi_j^n, (j,e) \in \Lambda_i \quad \exists (k,e) \in \Lambda_i^{in}\}\}.$$

$$\phi_{i,\min}^n < \phi_i^{in}(t^n) < \phi_{i,\max}^n \tag{3.17}$$

$$\phi_{i,\min}^n < \phi_i^n < \phi_{i,\max}^n, \tag{3.18}$$

ϕ_i^{n+1} (3.15)

$$\phi_{i,\min}^n \le \phi_i^{n+1} \le \phi_{i,\max}^n \tag{3.19}$$

$\Delta t^n > 0$
(3.17) ϕ_i^{n+1} (3.16) (3.19)
. First, we prove (3.19) for the solution given by (3.15). To do so, let us denote

$$\omega(t) := \frac{\phi_i^n - \phi_{i,\min}^n}{\phi_i^{out}(t) - \phi_{i,\min}^n}. \tag{3.20}$$

If $\phi_i^{out}(t^n) > \phi_i^n$ or $\phi_i^{out}(t) \ge \phi_i^n$ for $t \ge t^n$, one obtains that $\omega(t^{n+1/2}) \in (0,1]$ for sufficiently small $\Delta t^n > 0$. From (3.20) one has

$$\phi_i^{out}(t) = \frac{1}{\omega(t)}\phi_i^n - \frac{1-\omega(t)}{\omega(t)}\phi_{i,\min}^n, \tag{3.21}$$

and (3.15) can be written in the form of positive coefficient scheme

$$\begin{aligned}\phi_i^{n+1} = &\left(1 - \frac{\Delta t^n}{\omega(t^{n+1/2})\tau_i}\right)\phi_i^n \\ &+ \frac{(1-\omega(t^{n+1/2}))\Delta t^n}{\omega(t^{n+1/2})\tau_i}\phi_{i,\min}^n + \frac{\Delta t^n}{\tau_i}\phi_i^{in}(t^{n+1/2})\end{aligned} \tag{3.22}$$

if $\Delta t^n \le \min\{\omega(t^n), \omega(t^{n+1/2})\}\tau_i$. Consequently, the property (3.19) is valid.

For the case that $\phi_i^{out}(t^n) < \phi_i^n$ or $\phi_i^{out}(t) \le \phi_i^n$ for $t \ge t^n$, one can proceed analogously by defining

$$\omega(t) := \frac{\phi_i^n - \phi_{i,\max}^n}{\phi_i^{out}(t) - \phi_{i,\max}^n}. \tag{3.23}$$

For ϕ_i^{n+1} obtained from (3.16) the proof is straightforward. □

1. The results of Theorem 3.1 can be used to control any standard limiter procedure; see, e.g., [21, 3, 13] for flux or slope limiter procedures. For instance, the assumption (3.17) can be made valid by setting (in an extreme case) $\beta_{j,e}^n = 0$, $j \in \Lambda_i^{in}$, in (3.10).

TABLE 4.1

Errors in $L_2((0,T),L_2(S^2))$-norm and EOC for the example of a shrinking sphere using the first order scheme (3.2) *(third and fourth columns) and using the high-resolution scheme* (3.15) *(fifth and sixth columns). Analogously, the errors in $L_1((0,T),L_1(S^2))$-norm and EOC for the example of a shrinking cube are presented (seventh and eighth columns). In both cases, the high-resolution scheme gives the second order accuracy.*

E	N	Error	EOC	Error	EOC	Error	EOC
10	10	3.871E-2		6.289E-3		2.412E-3	
20	20	1.867E-2	1.05	1.470E-3	2.09	7.800E-4	1.63
40	40	9.181E-3	1.02	3.546E-4	2.05	2.128E-4	1.87
80	80	4.528E-3	1.02	8.763E-5	2.01	5.395E-5	1.98
160	160	2.243E-3	1.01	2.167E-5	2.01	1.380E-5	1.97

4. Numerical experiments. In the following subsections, we present various numerical experiments to illustrate the behavior of the high-resolution flux-based level set method. Note that in all examples no limiting procedures and no reinitializations of the level set function were used.

4.1. Comparison with exact solutions on a three-dimensional uniform grid. In this subsection we give a comparison with simple exact solutions of the surface evolution problem.

First we show experimental order of convergence (EOC) for an evolving sphere computed on a subsequently refined uniform rectangular grid in three dimensions. For $\delta = -1$ in (1.2), the exact radius of a shrinking sphere is given by $r(t) = r(0) - t$, $t \in [0,T]$, where $r(0)$ is an initial radius and the solution is well defined for $T \leq r(0)$. We use the spatial domain $\Omega = [-1.25, 1.25]^3$, which is split into E^3 cubic elements T^e with the side length $h = 2.5/E$. The time step is given by $\tau = T/N$.

To compare numerical and exact solutions, we find all zero crossing points $x_k^n, k = 1,\dots,K$, of a piecewise linear representation of the numerical solution with finite element grid lines in every discrete time step $n = 0,\dots,N$. Then, in the case of a shrinking sphere, we compute Euclidean distances $r_k^n = |x_k^n|$ from the origin of all $x_k^n, k = 1,\dots,K$, and compare them with the exact radius of the shrinking sphere. Summing for all discrete time steps, we get the formula

$$\text{Error} = \left(\sum_{n=0}^{N} \tau \frac{1}{K} \sum_{k=1}^{K} (r_k^n - r(n\tau))^2 \right)^{\frac{1}{2}} \tag{4.1}$$

which is used as the $L_2((0,T),L_2(S^2))$-norm of the difference of exact and numerical surfaces, where S^2 is a unit sphere. In the experiment presented in Table 4.1 we use $r(0) = 1$ and $T = 0.4$.

In the case of a shrinking cube centered at origin, we compute $m_k^n = \max(|x_1^n|, |x_2^n|, |x_3^n|)$ for all zero crossing points $x_k^n = (x_1^n, x_2^n, x_3^n)_k$, $k = 1,\dots,K$. The value m_k^n is compared with the corresponding exact quantity, a distance $m(t)$ of the faces of the exactly shrinking cube from the origin. Note that $m(t) = m(0) - t$. Points on the exactly and numerically evolved surfaces correspond to points on the unit sphere (using the radial coordinate). Therefore, analogously to (4.1), we can define the formula

$$\text{Error} = \sum_{n=0}^{N} \tau \frac{1}{K} \sum_{i=1}^{K} |m_i^n - m(n\tau)| \tag{4.2}$$

which is used as the $L_1((0,T), L_1(S^2))$-norm of the error in surface motion. This norm is used for nonsmooth solutions due to the corners in the evolving isosurface. In the experiment presented in Table 4.1 we use $m(0) = 1$, $T = 0.4$, and further parameters are the same as in the case of the shrinking sphere.

Table 4.1 shows that the method (3.15) is $O(h^2)$ for both examples, compared to the method from (3.2), which is only first order accurate.

4.2. Comparison with exact solutions on two-dimensional unstructured grids. We present two-dimensional examples with known analytical solutions to demonstrate the high-resolution accuracy of our method on unstructured grids. Implementations were realized with software toolbox UG [5]. The domain is a square $\Omega := (-0.5, 0.5)^2$. The velocity is given as a sum of the divergence free part (the rotation) and the part causing shrinking or expanding of the interface in normal direction,

$$\vec{V} = \begin{pmatrix} -y \\ x \end{pmatrix} + \delta \frac{\nabla \phi}{|\nabla \phi|}, \tag{4.3}$$

where δ is a constant to be specified later. The initial condition and the Dirichlet boundary conditions will be prescribed using the exact solution Φ of a particular problem.

First, an example of a circle that rotates and shrinks is considered:

$$\Phi(t, x_1, x_2) = \sqrt{\tilde{x}_1^2 + \tilde{x}_2^2} - r_0 + \delta t \tag{4.4}$$

and

$$\tilde{x}_1 := x_1 \cos(t) + x_2 \sin(t) + 0.25, \;\; \tilde{x}_2 := x_2 \cos(t) - x_1 \sin(t). \tag{4.5}$$

The initial radius is chosen as $r_0 = 0.2$. The parameter δ is chosen such that after half rotation, i.e., $t = \pi$, the radius of circle is halved; i.e., $\delta = -0.1/\pi$.

Figure 4.1 shows an initial finite element mesh of 28 triangles representing the grid level 0. The grid is subsequently refined by dividing each triangle into 4 subtriangles. The finest grid is obtained after 6 uniform refinements (i.e., the grid level equals 6) and contains 114688 triangles. The picture in the middle of Figure 4.1 presents the numerical solution at $t = 0$ and $t = \pi$ for the grid level 6.

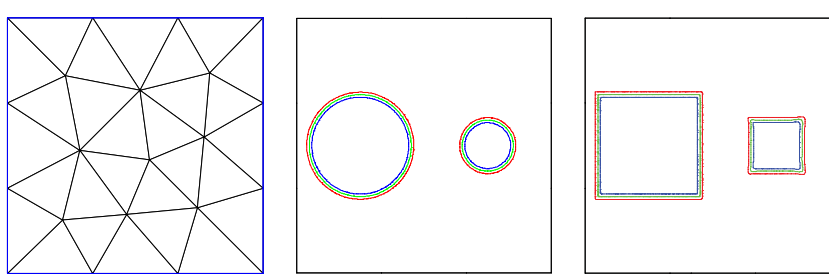


FIG. 4.1. *Initial coarse grid (left) and three contour lines ($\phi = -0.01, 0.0$ and 0.01) of numerical solutions at $t = 0$ and $t = \pi$ for the circle example (middle) and the square example (right). In this resolution, almost no difference between exact and numerical zero isolines would be visible.*

Table 4.2 shows an L_1-norm of numerical error at $t = t^N = T$,

$$\text{Error} = \int_\Omega |\Phi(T, x_1, x_2) - \sum_{i=1}^{I} \phi_i^N N_i(x_1, x_2)| \, dx_1 dx_2, \tag{4.6}$$

TABLE 4.2
Numerical errors and EOC for the shrinking and rotating circle example.

Level	N	Error (4.6)	EOC	Error (4.7)	EOC	Area error	EOC
3	80	3.5268E-4		5.8035E-4		3.5455E-4	
4	160	0.9099E-4	1.95	1.2370E-4	2.23	0.1953E-4	4.18
5	320	0.2330E-4	1.97	0.3203E-4	1.95	0.0629E-4	1.63
6	640	0.0598E-4	1.96	0.0830E-4	1.95	0.0211E-4	1.58

TABLE 4.3
Numerical errors and EOC for the shrinking and rotating square example.

Level	N	Error (4.6)	EOC	Error (4.7)	EOC	Area error	EOC
3	80	1.7461E-3		3.8785E-3		3.3658E-3	
4	160	0.5986E-3	1.54	1.3725E-3	1.50	1.0503E-4	1.64
5	320	0.1678E-3	1.83	0.3698E-3	1.89	0.2677E-4	1.97
6	640	0.0443E-3	1.92	0.0982E-3	1.91	0.0669E-4	2.00

for each grid level; see also (3.8). Note that the error (4.6) is computed on the whole domain and not only on the interface. The L_1 error of the interface position at $t = T$ is computed using

$$\text{Error} = \sum_{k=1}^{K} |\Gamma^k| \left| r^k - r_{\text{exact}} \right|, \tag{4.7}$$

where the numerical interface is reconstructed as a polygon made of K segments Γ^k, and the Euclidean distance r^k is computed between the middle point (x_1^k, x_2^k) of Γ^k and the center (x_1^c, x_2^c). For this example, one has $r_{\text{exact}} = 0.1$ and $(x_1^c, x_2^c) = (0.25, 0.0)$. Moreover, the error between the exact area 0.01π of the circle at $t = \pi$ and the area given by the numerical solution is presented in Table 4.2.

An analogous numerical experiment is realized using a level set function (not a distance function) describing a square that is transported by the same velocity (4.3), the results of which are presented in Figure 4.1 (right) and Table 4.3. The exact solution is given by

$$\Phi(t, x_1, x_2) := \begin{cases} \tilde{x}_2 - r_0 - \delta t, & \tilde{x}_2 \geq 0 \quad \text{and} \quad \tilde{x}_2 \geq |\tilde{x}_1|, \\ -\tilde{x}_2 - r_0 - \delta t, & \tilde{x}_2 \leq 0 \quad \text{and} \quad -\tilde{x}_2 \geq |\tilde{x}_1|, \\ \tilde{x}_1 - r_0 - \delta t, & \tilde{x}_1 \geq 0 \quad \text{and} \quad \tilde{x}_1 \geq |\tilde{x}_2|, \\ -\tilde{x}_1 - r_0 - \delta t, & \tilde{x}_1 \leq 0 \quad \text{and} \quad -\tilde{x}_1 \geq |\tilde{x}_2|, \end{cases} \tag{4.8}$$

where $r_0 = 0.2$ (see also (4.5)) and the corners of the square remain sharp, producing discontinuous derivatives of the solution. The error (4.7) on the interface ($r_{\text{exact}} = 0.1$ and $(x_1^c, x_2^c) = (0.25, 0.0)$), where $r^k = \max(|x_1^k - x_1^c|, |x_2^k - x_2^c|)$, and the error between the exact area 0.04 and the area determined by the numerical solution are included in Table 4.3.

Next, analogous numerical experiments with expanding characteristics are presented. First, an example of the expanding and rotating circle is chosen where the exact solution $\Phi(\pi, x_1, x_2)$ is given by (4.4) with $r_0 = 0.1$, $\delta = 0.1/\pi$ for $(x_1, x_2) \notin S((0.25, 0.0), 0.1)$, and $\Phi(\pi, x_1, x_2) \equiv -0.1$ otherwise, where $S((0.25, 0.0), 0.1)$ is a circle centered at $(0.25, 0.0)$ with the radius 0.1.

Note that for this example the high-resolution method gives EOC in the error (4.6) that is less than 2. One of the reasons is a constant region that is developed in the

TABLE 4.4
Numerical errors and EOC for the expanding and rotating circle example.

Level	N	Error (4.6)	EOC	Error (4.7)	EOC	Area error	EOC
3	80	1.5615E-3		1.8584E-3		7.338E-4	
4	160	0.5413E-3	1.53	0.4784E-3	1.96	1.434E-4	2.36
5	320	0.1989E-3	1.44	0.1100E-3	2.12	0.906E-4	0.66
6	640	0.0678E-3	1.55	0.0262E-3	2.07	0.080E-4	3.5

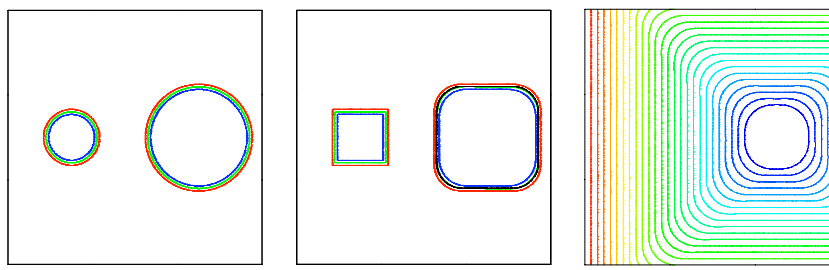


FIG. 4.2. *The left picture shows three contour lines ($\phi = -0.01, 0.0$, and 0.01) of numerical solutions at $t = 0$ and $t = \pi$ for the expanding and rotating circle example (in this resolution no difference between the exact and the numerical zero isolines could be visible). The middle picture presents analogous results for the expanding and rotating square, where additionally the exact zero level line (black) is presented. The right picture shows the level lines of the numerical solution distributed uniformly between -0.075 and 0.55.*

TABLE 4.5
Numerical errors and EOC for the expanding and rotating square example.

Level	N	Error (4.6)	EOC	Error (4.7)	EOC	Area error	EOC
3	80	3.9579E-3		9.9274E-3		6.3885E-3	
4	160	1.8124E-3	1.13	5.1273E-3	0.95	3.9632E-3	0.69
5	320	0.8203E-3	1.14	2.2505E-3	1.19	1.8900E-3	1.07
6	640	0.3733E-3	1.14	1.0654E-3	1.08	0.9554E-3	0.98

exact solution. Nevertheless, the error (4.7) on the interface diminishes quadratically; see Table 4.4, where the difference between the exact area 0.04π and the numerical one is also presented. The initial and final circular interface is presented in the picture at the left of Figure 4.2.

Finally, an example of the expanding and rotating square is computed. The initial function $\Phi(0, x_1, x_2)$ is given by (4.8) with $r_0 = 0.1$ and $\delta = 0.1\pi$. The exact solution can be simply constructed using the Huygens principle; see also the picture at the right of Figure 4.2. The global numerical error (4.6), the local one (4.7), and the difference between the exact area $0.12 + 0.01\pi$ and the numerical one are presented in Table 4.5. The numerical solution at $t = \pi$ is plotted in the middle picture of Figure 4.2 (three isolines where the interface is located) and in the right picture (isocontours plot of the final level set function). For this example, the high-resolution method seems to be slightly better than first order accurate.

4.3. Numerical experiments using locally adapted grids. The example with the rotating and shrinking square was repeated with a locally adapted grid by refining (and coarsening) in each time step only elements in the neighborhood of the zero contour line; see Figure 4.3. Note that mixed grids consisting of squares

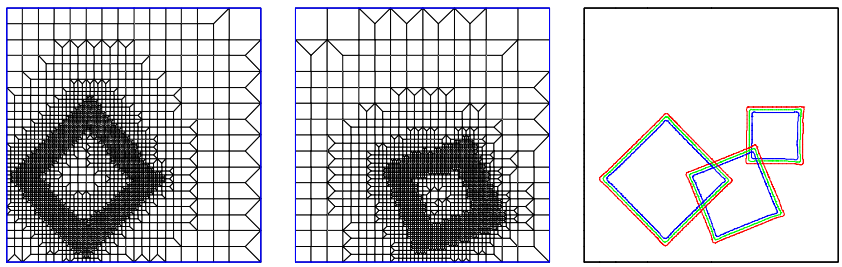


FIG. 4.3. *Locally adapted grid at $t = \pi/4$ (left) and $t = 5\pi/8$ (middle) and numerical solutions at $t = \pi/4$, $t = 5\pi/8$, and $t = \pi$ (right).*

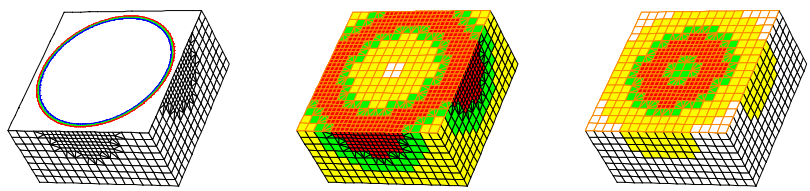


FIG. 4.4. *A cross section of numerical solution (left) and locally adapted grid used in the first (middle) and last (right) time step for the example of a shrinking sphere.*

and triangles were used. The simulation used 3290 finite elements at the beginning and 1547 at the end of computations and took considerably less time to realize than computations with corresponding uniformly refined grids.

To illustrate the capability of UG [5] for three-dimensional grids, Figure 4.4 shows the numerical solution (a shrinking sphere) and the grids before the first and last time step. Note that mixed grids consisting of hexahedra, pyramids, and tetrahedra were used.

Finally, in a nontrivial experiment we evolve an initial curve in the form of a shrinking quatrefoil [15]. The quatrefoil is the zero level set of the initial level set function (not a signed distance function) constructed by the formula

$$\phi^0(x_1, x_2) = -0.5 + \sqrt{x_1^2 + x_2^2}/r_L, \quad r_L = 0.6 + 0.4 \sin\left(4 \operatorname{arctg}\left(\frac{x_2}{x_1}\right)\right).$$

In this experiment, topological changes of the evolving curve occur; see Figure 4.5.

4.4. Benchmarks with divergence free velocity. In this section, we present numerical results for two standard benchmark examples with divergence free velocity field to illustrate the behavior of our method for such types of problems. The main application of level set equation (1.1) with externally given velocity $\vec{V}$ such that $\nabla \cdot \vec{V} = 0$ are problems of two-phase flow where the evolving level set function describes a moving interface between two incompressible fluids. Note that for this type of problem, the level set equation can be written in a conservative form (2.3) with zero right-hand side. For such a classical scalar linear hyperbolic equation describing

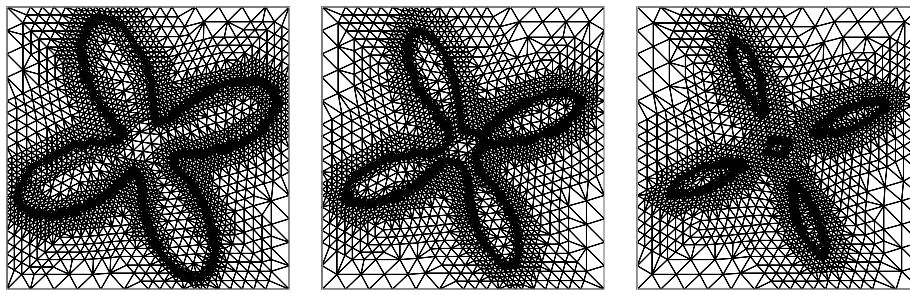


FIG. 4.5. *Shrinking of a quatrefoil with topological changes; the zero contour line of the numerical solution and the used grid are plotted for three different time steps.*

conservation law there exist an enormous number of sophisticated numerical methods and recently some of them were devoted directly to level set applications [30, 35, 25].

Before presenting our numerical results, we illustrate a better performance of the finite volume based scheme (3.15) as compared to the finite difference based ENO scheme [31] for a test example of a rotating circle. The high-resolution flux-based level set method (3.15) was implemented in the MATLAB toolbox of level set methods [26] and compared with the existing implementation of the ENO scheme of second order in time and space.

The example is defined on a square $(-1,1)^2$, and the rotation is given in the clockwise direction. Dirichlet boundary conditions given by the exact values of a signed distance function were used in both methods.

Figure 4.6 shows numerical solutions on a coarse grid with $(I+1)^2$ nodes, $I = 20$, where the differences between two methods can be clearly recognized. Table 4.6 shows the simple discrete L_1-norm

$$\text{Error} = \frac{4}{I^2} \sum_{i=1}^{(I+1)^2} |\phi_i^0 - \phi_i^N|,$$

which is computed from the differences of numerical solutions in grid points at $t = 0$ and the time after one rotation.

In what follows, we present results of our scheme in solving two complex examples known from the literature, namely for the Zalesak's disk rotation and the vortex-in-a-box examples. In the literature, to improve a finite difference level set

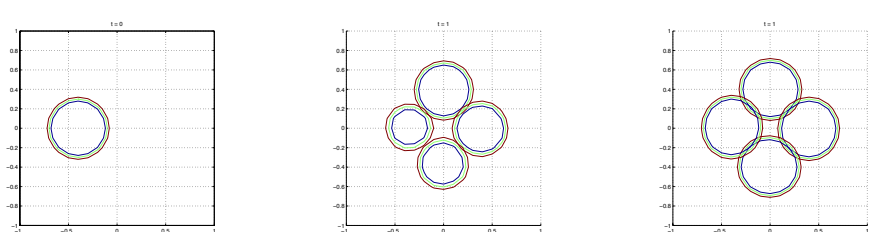


FIG. 4.6. *Initial condition (left), numerical solution obtained by the second order ENO scheme (middle) and by the high-resolution scheme FB-LSM* (3.15) *(right) for* 4 *different times on a coarse mesh of* 21×21 *grid points. The rotation occurs in the clockwise direction.*

TABLE 4.6
Numerical error of flux-based level set method (FB-LSM) and ENO scheme for a rotating circle.

I	N	Error FB-LSM	Error ENO
20	140	104.046E-4	708.131E-4
40	280	28.726E-4	235.063E-4
80	560	7.899E-4	71.060E-4

approximation of the interface, additional numerical techniques, such as redistancing algorithms [44, 42, 48], particle tracking techniques [8, 9, 18], volume-of-fluid methods [46], or discontinuous Galerkin methods [48, 25], were introduced; see also a review in [24].

In our opinion, the results obtained by the flux-based level set method (2.10), presented in this section, are qualitatively comparable to those of published studies, especially if one takes into the account that no additional numerical techniques, mentioned before, were applied in our computations to improve the level set approximation of the interface. In this way, the flux-based level set method can produce satisfactory results for reasonable meshes by preserving its simple form. Nevertheless, in contradiction to quoted papers, the primal field of applications of our flux-based level set method are problems that involve geometric evolution equations and that might also include the evolution described by external flow; see, e.g., the boat-sail distance problem [28, 29].

First, the example of the solid body rotation of the so-called Zalesak's disk is presented; see, e.g., [42, 8, 9, 18, 25, 46]. The initial interface is a circle with radius 0.15 centered at $(0.5, 0.75)$ with a vertical rectangular cut of the width and length equal to 0.05 and 0.25, respectively. The slotted disk rotates in the counterclockwise direction with a constant angular velocity and returns to the origin position after one rotation.

The example is simulated on a unit square. The initial level set function is given by a signed distance function to Zalesak's disk, and exact Dirichlet boundary conditions are used during simulations. The initial coarse grid (referred to later as the grid level 0) consists of 4 right triangles that are obtained by connecting the corners of the square with the middle point of the square. Afterwards, the grid is refined uniformly by dividing each triangle into 4 subtriangles, creating subsequent grid levels. Numerical results are presented for the grid levels 7 (consisting of 65536 triangles with 33025 vertices, 720 uniform time steps) and 8 (consisting of 262144 triangles with 131585 vertices, 1440 uniform time steps).

In Figure 4.7 a comparison of the numerically and exactly computed interface after one rotation can be found for grid levels 7 and 8. The corresponding L_1 errors (4.6) are presented in Table 4.7.

As expected, one can observe a smoothing of initially sharp corners in the numerical solution of Zalesak's disk problem. The area of the disk surrounded by the zero isoline of the numerical level set function at grid level 8 is 5.8216E-2 for $t = 0$ and 5.7893E-2 for $t = T$, which corresponds to the loss of 0.55%.

Finally, an example of a single vortex in a box is presented; see, e.g., [9, 18, 25, 46]. The time-dependent velocity with nonzero vorticity is given by

$$\vec{V} = 2\cos\left(\frac{\pi t}{8}\right)\left(-\sin^2(\pi x)\sin(\pi y)\cos(\pi y)\,,\ \sin^2(\pi y)\sin(\pi x)\cos(\pi x)\right). \tag{4.9}$$

The time dependency of $\vec{V}$ in (4.9) was resolved by evaluating it at $t = t^n + 0.5\Delta t$ for

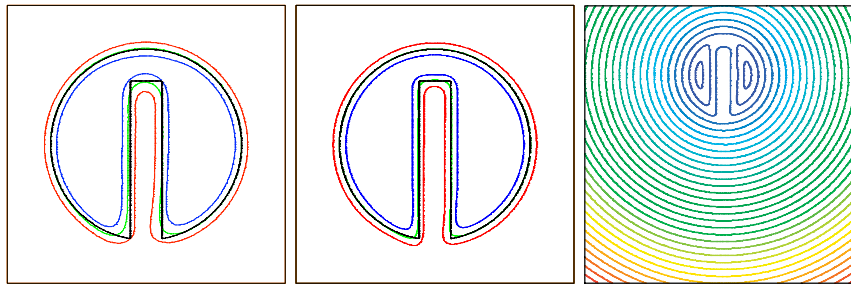


FIG. 4.7. *The left picture shows a detail of three level lines ($\phi = -0.01, 0.0$, and 0.01) of the numerical solution and the exact zero line (black) after one rotation at grid level 7 for Zalesak's example. The middle picture presents analogous results for grid level 8. The right picture shows level lines of the numerical solution in the whole domain (distributed uniformly from -0.05 to 0.75) after one rotation for grid level 8.*

TABLE 4.7
Numerical errors and EOC for the Zalesak example and the single vortex example.

Level	N	Error (4.6)	EOC	N	Error (4.6)	EOC
		Zalesak ex.			Single vortex ex.	
6	360	6.3206E-4		800	1.2309E-2	
7	720	2.2406E-4	1.50	1600	0.4461E-2	1.46
8	1440	0.9224E-4	1.28	3200	0.1246E-2	1.84

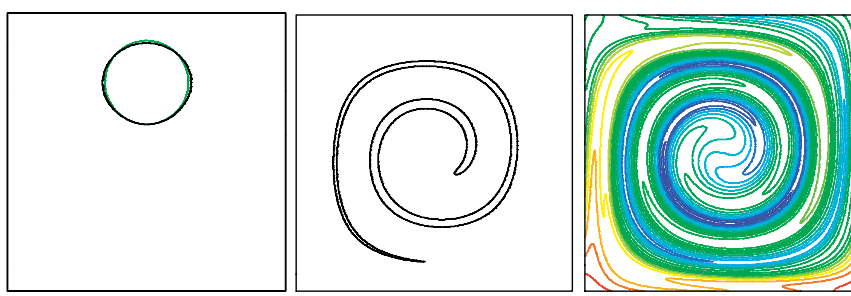


FIG. 4.8. *The left picture shows the exact circular interface (green) and the computed interface (black) at $t = 8$ of the single vortex example for grid level 8. The middle picture presents the zero level line of the numerical solution at $t = 4$, and the right picture shows level lines in the whole domain distributed uniformly between -0.1 and 0.75.*

each time step to obtain $\vec{V}$ in (1.1).

The initial interface is a circle with radius 0.15 and centered at $(0.5, 0.75)$ in a unit square $(0, 1)^2$. The interface stretched during simulations by wrapping around the center of the box, reaching a maximal deformation at $t = 4$ and returning to the initial circular form at $t = 8$.

The computations were realized using a signed distance function as an initial level set function and exact Dirichlet boundary conditions during simulations. The coarse grid and subsequent grid levels were chosen analogously to Zalesak's disk example. A uniform time step was chosen by setting $\Delta t^n = \Delta t = 8/N$.

Numerical results for the single vortex example are plotted in Figure 4.8, and the numerical L_1 error (4.6) is presented in Table 4.7. The area of the circle at $t = 8$ equals 7.2302E-2, which corresponds to a gain of 2.3% with respect to the exact area

7.0686E-2. Note that again no extension to our level set method was used. For instance, no redistancing was applied in simulations; see the picture at the right of Figure 4.7 where the numerical level set function at $t = 4$ is presented.

Acknowledgment. The authors would like to thank Christian Wehner, a student at the University of Heidelberg, for providing numerical results comparing the flux-based level set method and the ENO scheme.

REFERENCES

[1] R. Abgrall, *Numerical discretization of the first-order Hamilton-Jacobi equation on triangular meshes*, Comm. Pure Appl. Math., 49 (1996), pp. 1339–1373.

[2] S. Augoula and R. Abgrall, *High order numerical discretization for Hamilton–Jacobi equations on triangular meshes*, J. Sci. Comput., 15 (2000), pp. 197–229.

[3] T. Barth and M. Ohlberger, *Finite volume methods: Foundation and analysis*, in Encyclopedia of Computational Mechanics, Vol. 1, John Wiley & Sons, New York, 2004, pp. 439–474.

[4] T. J. Barth and J. A. Sethian, *Numerical schemes for the Hamilton-Jacobi and level set equations on triangulated domains*, J. Comput. Phys., 1 (1998), pp. 1–40.

[5] P. Bastian, K. Birken, K. Eckstein, K. Johannsen, S. Lang, N. Neuss, and H. Rentz-Reichert, *UG—a flexible software toolbox for solving partial differential equations*, Comput. Vis. Sci., 1 (1997), pp. 27–40.

[6] M. G. Crandall and P.-L. Lions, *Two approximations of solutions of Hamilton-Jacobi equations*, Math. Comp., 43 (1984), pp. 1–19.

[7] J. Douglas, Jr., and T. F. Russell, *Numerical methods for convection-dominated diffusion problems based on combining the method of characteristics with finite element or finite difference procedures*, SIAM J. Numer. Anal., 19 (1982), pp. 871–885.

[8] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell, *A hybrid particle level set method for improved interface capturing*, J. Comput. Phys., 183 (2002), pp. 83–116.

[9] D. Enright, F. Losasso, and R. Fedkiw, *A fast and accurate semi-Lagrangian particle level set method*, Comput. & Structures, 83 (2005), pp. 479–490.

[10] M. Falcone and R. Ferretti, *Convergence analysis for a class of high-order semi-Lagrangian advection schemes*, SIAM J. Numer. Anal., 35 (1998), pp. 909–940.

[11] M. Falcone, T. Giorgi, and P. Loreti, *Level sets of viscosity solutions: Some applications to fronts and rendez-vous problems*, SIAM J. Appl. Math., 54 (1994), pp. 1335–1354.

[12] P. Frolkovič, *Discretization*, in d^3f—ein Programmpaket zur Modellierung von Dichteströmungen, GRS-139, E. Fein, ed., Gesellschaft für Anlagen und Reaktorsicherheit, Braunschweig, Germany, 1998; available online from http://sit.uni-hd.de.

[13] P. Frolkovič, *Flux-based method of characteristics for coupled transport equations in porous media*, Comput. Vis. Sci., 6 (2004), pp. 173–184.

[14] P. Frolkovič and K. Mikula, *Flux-based level set method: A finite volume method for evolving interfaces*, Preprint 15, IWR, Universität Heidelberg, Heidelberg, Germany; available online from http://www.iwr.uni-heidelberg.de/organization/sfb359/PP/Preprint2003-15.pdf (September 2003).

[15] P. Frolkovič and K. Mikula, *Flux-based level set method: A finite volume method for evolving interfaces*, Appl. Numer. Math., 57 (2007), pp. 436–454.

[16] I. Ginzburg and G. Wittum, *Two-phase flows on interface refined grids modeled with VOF, staggered finite volumes, and spline interpolants*, J. Comput. Phys., 166 (2001), pp. 302–335.

[17] S. Gross, V. Reichelt, and A. Reusken, *A finite element based level set method for two-phase incompressible flows*, Comput. Vis. Sci., 9 (2006), pp. 239–257.

[18] S. E. Hieber and P. Koumoutsakos, *A Lagrangian particle level set method*, J. Comput. Phys., 210 (2005), pp. 342–367.

[19] G.-S. Jiang and D. Peng, *Weighted ENO schemes for Hamilton–Jacobi equations*, SIAM J. Sci. Comput., 21 (2000), pp. 2126–2143.

[20] G. Kossioris, Ch. Makridakis, and P. E. Souganidis, *Finite volume schemes for Hamilton-Jacobi equations*, Numer. Math., 83 (1999), pp. 427–442.

[21] R. J. LeVeque, *Finite Volume Methods for Hyperbolic Problems*, Cambridge Texts Appl. Math., Cambridge University Press, Cambridge, UK, 2002.

[22] D. Levy, S. Nayak, C.-W. Shu, and Y.-T. Zhang, *Central WENO schemes for Hamilton–Jacobi equations on triangular meshes*, SIAM J. Sci. Comput., 28 (2006), pp. 2229–2247.

[23] C.-T. Lin and E. Tadmor, *High-resolution nonoscillatory central schemes for Hamilton–Jacobi equations*, SIAM J. Sci. Comput., 21 (2000), pp. 2163–2186.
[24] F. Losasso, R. Fedkiw, and S. Osher, *Spatially adaptive techniques for level set methods and incompressible flow*, Comput. & Fluids, 35 (2006), pp. 995–1010.
[25] E. Marchandise, J.-F. Remacle, and N. Chevaugeon, *A quadrature-free discontinuous Galerkin method for the level set equation*, J. Comput. Phys., 212 (2006), pp. 338–357.
[26] I. M. Mitchell, *A Toolbox of Level Set Methods*, Technical report TR-2004-09, Department of Computer Science, University of British Columbia, Vancouver, Canada, 2004; available online from http://www.cs.ubc.ca/~mitchell/ToolboxLS.
[27] T. Neubauer and P. Bastian, *On a monotonicity preserving Eulerian-Lagrangian localized adjoint method for advection-diffusion equations*, Adv. Water Resources, 28 (2005), pp. 1292–1309.
[28] T. Nishida and K. Sugihara, *Approximation of the boat-sail Voronoi diagram and its application*, in Computational Science and Its Applications—ICCSA 2004, A. Laganà et. al., eds., Springer, Berlin, 2004, pp. 227–236.
[29] T. Nishida, K. Sugihara, and M. Kimura, *Stable marker-particle method for the Voronoi diagram in a flow field*, J. Comput. Appl. Math., to appear.
[30] E. Olsson and G. Kreiss, *A conservative level set method for two phase flow*, J. Comput. Phys., 210 (2005), pp. 225–246.
[31] S. Osher and R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*, Springer, New York, 2003.
[32] S. Osher and N. Paragios, *Geometric Level Set Methods in Imaging, Vision, and Graphics*, Springer, New York, 2003.
[33] S. Osher and J. A. Sethian, *Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton–Jacobi formulations*, J. Comput. Phys., 79 (1988), pp. 12–49.
[34] S. Osher and C.-W. Shu, *High-order essentially nonoscillatory schemes for Hamilton–Jacobi equations*, SIAM J. Numer. Anal., 28 (1991), pp. 907–922.
[35] D. A. Di Pietro, S. Lo Forte, and N. Parolini, *Mass preserving finite element implementations of the level set method*, Appl. Numer. Math., 56 (2006), pp. 1179–1195.
[36] M. Restelli, L. Bonaventura, and R. Sacco, *A semi-Lagrangian discontinuous Galerkin method for scalar advection by incompressible flows*, J. Comput. Phys., 216 (2006), pp. 195–215.
[37] T. F. Russell and M. A. Celia, *An overview of research on Eulerian-Lagrangian localized adjoint methods (ELLAM)*, Adv. Water Resources, 25 (2002), pp. 1215–1231.
[38] G. Sapiro, *Geometric Partial Differential Equations and Image Analysis*, Cambridge University Press, Cambridge, UK, 2001.
[39] J. Sethian, *Level Set Methods and Fast Marching Methods*, Cambridge University Press, Cambridge, UK, 1999.
[40] Th. Sonar, *On the design of an upwind scheme for compressible flow on general triangulations*, Numer. Algorithms, 4 (1993), pp. 135–149.
[41] J. Strain, *Semi-Lagrangian methods for level set equation*, J. Comput. Phys., 151 (1999), pp. 498–533.
[42] M. Sussman and E. Fatemi, *An efficient, interface-preserving level set redistancing algorithm and its application to interfacial incompressible fluid flow*, SIAM J. Sci. Comput., 20 (1999), pp. 1165–1191.
[43] M. Sussman and E. Puckett, *A coupled level set and volume-of-fluid method for improved interface capturing*, J. Comput. Phys., 162 (2000), pp. 301–337.
[44] M. Sussman, P. Smereka, and S. Osher, *A level set approach for computing solutions to incompressible two-phase flow*, J. Comput. Phys., 114 (1994), pp. 146–159.
[45] A.-K. Tornberg and B. Engquist, *A finite element based level set method for multiphase flow applications*, Comput. Vis. Sci., 3 (2000), pp. 93–101.
[46] X. Yang, A. J. James, J. Lowengrub, X. Zheng, and V. Cristini, *An adaptive coupled level-set/volume-of-fluid interface capturing method for unstructured triangular grids*, J. Comput. Phys., 217 (2006), pp. 364–394.
[47] Y.-T. Zhang and C.-W. Shu, *High-order WENO schemes for Hamilton–Jacobi equations on triangular meshes*, SIAM J. Sci. Comput., 24 (2003), pp. 1005–1030.
[48] X. Zheng, J. Lowengrub, A. Anderson, and V. Cristini, *Adaptive unstructured volume remeshing.* II. *Application to two- and three-dimensional level-set simulations of multiphase flow*, J. Comput. Phys., 208 (2005), pp. 626–650.

SIAM J. SCI. COMPUT.
Vol. 29, No. 2, pp. 598–621
© 2007 Society for Industrial and Applied Mathematics

CONSTRUCTING ROBUST GOOD LATTICE RULES FOR COMPUTATIONAL FINANCE*

XIAOQUN WANG†

Abstract. The valuation of many financial derivatives leads to high-dimensional integrals. The constructions of *robust* or *universal* good lattice rules for financial applications are both important and challenging. An important *common* feature of the integrands in computational finance is that they can often be well approximated by a sum of low-dimensional functions, i.e., functions that depend on only a small number of variables (usually just two variables). For numerical integration of such functions the quality of the low-order (i.e., low-dimensional) projections of the node set is crucial. In this paper we propose methods to construct good lattice points with "optimal" low-order projections. The quality of a point set is measured by a new measure called *elementary order-ℓ discrepancy*, which measures the quality of all order-ℓ projections and is more informative than usual measures. Two constructions, namely the Korobov and the component-by-component constructions, are studied such that the low-order projections are optimized. Numerical experiments demonstrate that even in high dimensions it is possible to construct new good lattice points with order-2 projections that are better than those of the Sobol' points and random points and with higher-order projections that are no worse (while the Sobol' points lost the advantage over random points in order-2 projections on the average). The new lattice rules have the potential to improve upon the accuracy for favorable functions, while doing no harm for unfavorable ones. Their applications for pricing path-dependent options and American options (based on the least-square Monte Carlo method) are studied and their high efficiency is demonstrated. A nice surprise revealed is the *robustness property* of such lattice rules: the good projection property and the suitability for a large range of problems. The potential possibility and limitations of good lattice points in achieving good quality of moderate- and high-order projections is investigated. The reason why classical lattice rules may not be efficient for high-dimensional finance problems is also discussed.

Key words. quasi-Monte Carlo methods, good lattice rules, multivariate integration, option pricing, American options

AMS subject classifications. 65C05, 65D30, 65D32

DOI. 10.1137/060650714

1. Introduction.

Many practical problems can be transformed into the computations of multivariate integrals:

$$I_s(f) = \int_{[0,1]^s} f(\mathbf{x})\, d\mathbf{x}.$$

Typical examples are the valuations of financial derivatives. In principle, any stochastic simulation whose purpose is to estimate an expectation fits this framework. High-dimensional integrals are usually approximated by Monte Carlo (MC) or quasi-Monte Carlo (QMC) algorithms:

$$Q_{n,s}(f) := Q_{n,s}(f; P) := \frac{1}{n} \sum_{k=0}^{n-1} f(\mathbf{x}_k),$$

*Received by the editors January 24, 2006; accepted for publication (in revised form) October 11, 2006; published electronically March 30, 2007. This work was supported by the National Science Foundation of China.

http://www.siam.org/journals/sisc/29-2/65071.html

†Department of Mathematical Sciences, Tsinghua University, Beijing 100084, China (xwang@math.tsinghua.edu.cn).

where $P := \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{n-1}\}$ is a set of random points in MC or deterministic [illegible] in QMC. The MC methods converge as $O(n^{-1/2})$, independently of s. QMC methods try to take the points more uniformly distributed and have the potential to improve the convergence rate for some classes of functions. Digital nets and good lattice points are two important classes of QMC; see [18, 20]. For example, a rank-1 lattice rule has the form

$$Q_{n,s}(f) = \frac{1}{n} \sum_{k=0}^{n-1} f\left(\left\{\frac{k\mathbf{z}}{n}\right\}\right),$$

where $\mathbf{z} = (z_1, z_2, \ldots, z_s)$ is the [illegible] with no factor in common with n and the notation $\{\mathbf{x}\}$ means the vector whose components are the fractional part of x_j. The central topic in the field of lattice rules is to find a "good" generating vector; see [12, 18, 20].

In financial applications, it is often observed that while the nominal dimension of the problems can be very large, the [illegible] is often small; i.e., the underlying functions are nearly a superposition of low-dimensional functions, especially when dimension reduction techniques are used (see [3, 27]). More precisely, the analysis of variance (ANOVA) expansion up to the second order,

$$f(\mathbf{x}) \approx f_0 + \sum_{j=1}^{s} f_j(x_j) + \sum_{1 \le i < j \le s} f_{ij}(x_i, x_j),$$

can usually provide a quite satisfactory approximation, where $f_j(x_j)$ and $f_{ij}(x_i, x_j)$ represent the individual and the cooperative effects. For numerical integration of such functions the quality of low-order (especially the order-1 and order-2) projections of the node set is crucial. QMC point sets should be constructed to be as well distributed as possible in their low-order projections. We believe that good quality in order-2 projections of a lattice point set is necessary and sufficient for many problems in finance. Point sets with optimal order-1 projections are not difficult to obtain (an arbitrary lattice point set has perfect order-1 projections). However, it is not so easy (or even impossible) to obtain point sets with optimal projections of various orders simultaneously.

It was shown in [29] that in high dimensions (say, $s \ge 50$) common quasi-random numbers, such as the Sobol's [25], have excellent order-1 projections (as expected), but often exhibit patterns in order-2 and higher-order projections and that these projections are [illegible] on the average than those of random points for practical n (which may contradict common intuition). In fact, this nonuniformity is typical for quasi-random numbers [16]. It is interesting to know whether good lattice points could provide better order- 2 and higher-order projections than digital nets do. What is the optimal situation (i.e., the logical limit) which can be achieved by a construction, say, the Korobov construction [12]? How does one construct lattice points with "optimal" low-order (or some specific order) projections? How efficient and robust are they for practical applications? The method in [6] provides a theoretical basis for answering these questions (but no generators are published there).

Confronted with such issues, the objectives of this paper are to explore the potential possibility of good lattice points in achieving good low-order projections (aiming at constructing robust lattice rules) and to study their applications to finance. Lattice rules are usually found by minimizing some quality measures; thus the choice of measure is important [7, 14, 8, 9, 28, 30]. To construct lattice points with low-order

projections as well distributed as possible, one needs a suitable measure. The classical quality measure P_α (see [20]) was found to be not suitable in high dimensions for this purpose (though it may not be bad in small dimensions, say $s \leq 8$), since it treats implicitly higher-order projections as more important than lower-order ones and does not put special emphasis on low-order projections, which may cause serious problems in high dimensions; see [28]. It was shown in [28] that the lattice rules constructed based on P_α may not give good results for high-dimensional problems (in fact, they can be even worse than MC, and this is why lattice rules are not so popular for finance problems). The reason will become clear in section 4.

In [28] an attempt was made to choose suitable weights involved in the weighted version of P_α by relating the weights to the sensitivity indices of a particular problem. The weighted version of P_α is related to the worst-case error of certain [illegible]. We refer to [23] for the theory of weighted function spaces. A principal difficulty of using the theory of weighted space for practical applications is in choosing appropriate weights. The lattice rules constructed in [28] are extremely effective, but they are [illegible]. The high efficiency of those rules is achieved by focusing on important dimensions and low-order projections. Based on a similar idea, in this paper we attempt to construct [illegible] lattice points, which are of acceptable quality for a wide range of problems.

This paper is organized as follows. In section 2 we introduce weighted Sobolev spaces and discuss the relationship of the uniformity of the projections with the quadrature error. In section 3 we discuss quality measures for lattice rules and present methods to construct lattice rules with "optimal" low-order (or some specific order) projections. Numerical comparisons on the quality of projections of various orders are presented in section 4. The potential possibility and the limitations of lattice rules (i.e., what lattice rules can do and cannot do) are investigated. Applications to the valuation of path-dependent options and American options (based on the least-square MC [15]) are demonstrated in section 5. Numerical experiments in sections 4 and 5 show the robustness property of our lattice rules. Concluding remarks are presented in section 6. Some generators of lattice rules with optimal order-2 projections are provided in the appendix.

2. Weighted Sobolev space and quadrature error. In this section we present some background on weighted function spaces and discuss the factors that affect QMC quadrature errors.

2.1. Background on reproducing kernel Hilbert space. Let $H(K)$ be a reproducing kernel Hilbert space (RKHS) with reproducing kernel $K(\mathbf{x}, \mathbf{y})$. The kernel $K(\mathbf{x}, \mathbf{y})$ has the following properties:

$$K(\mathbf{x}, \mathbf{y}) = K(\mathbf{y}, \mathbf{x}), \quad K(\cdot, \mathbf{y}) \in H(K) \qquad \forall\, \mathbf{x}, \mathbf{y} \in [0,1]^s$$

and

$$f(\mathbf{y}) = \langle f, K(\cdot, \mathbf{y}) \rangle \qquad \forall\, f \in H(K),\ \mathbf{y} \in [0,1]^s,$$

where $\langle \cdot, \cdot \rangle$ is the inner product of $H(K)$ and $||f||_{H(K)} = \langle f, f \rangle^{1/2}$. Let

$$h_s(\mathbf{x}) = \int_{[0,1]^s} K(\mathbf{x}, \mathbf{y})\, d\mathbf{y} \qquad \forall\, \mathbf{x} \in [0,1]^s.$$

We assume that $h_s \in H(K)$. Then I_s is a continuous linear functional with the reproducer h_s, i.e.,

$$I_s(f) \,=\, \langle f, h_s \rangle \qquad \forall\, f \in H(K).$$

Let P be a set of n points in the s-dimensional unit cube:

$$P := \{\mathbf{x}_k = (x_{k,1}, \ldots, x_{k,s}) : \ k = 0, 1, \ldots, n-1\} \subset [0,1]^s. \tag{1}$$

The [illegible] of the algorithm $Q_{n,s}(f;P)$ over the unit ball of $H(K)$, or the [illegible] of the set P with respect to the kernel K, is defined by

$$D(P; H(K)) = \sup\{|I_s(f) - Q_{n,s}(f;P)| : \ f \in H(K), \ ||f||_{H(K)} \le 1\}.$$

Since $H(K)$ is an RKHS, the square worst-case error can be expressed in terms of the reproducing kernel (see [9, 23]):

$$D^2(P; H(K)) = \int_{[0,1]^{2s}} K(\mathbf{x}, \mathbf{y})\, d\mathbf{x} d\mathbf{y} - \frac{2}{n} \sum_{i=0}^{n-1} \int_{[0,1]^s} K(\mathbf{x}_i, \mathbf{x})\, d\mathbf{x} + \frac{1}{n^2} \sum_{i,k=0}^{n-1} K(\mathbf{x}_i, \mathbf{x}_k). \tag{2}$$

Due to the linearity of $I_s(f) - Q_{n,s}(f)$, we have the error bound

$$|I_s(f) - Q_{n,s}(f)| \le D(P; H(K))\ ||f||_{H(K)} \qquad \forall f \in H(K). \tag{3}$$

2.2. A weighted Sobolev space. Weighted Sobolev space was introduced in [23]. Here we specify a particular RKHS with the following reproducing kernel:

$$K_{s,\gamma}(\mathbf{x}, \mathbf{y}) = 1 + \sum_{\emptyset \ne u \subseteq \mathcal{A}} K_u(\mathbf{x}_u, \mathbf{y}_u), \tag{4}$$

where $\mathcal{A} = \{1, \ldots, s\}$ and

$$K_u(\mathbf{x}_u, \mathbf{y}_u) = \gamma_{s,u} \prod_{j \in u} \eta(x_j, y_j),$$

with

$$\eta(x, y) = \frac{1}{2} B_2(\{x - y\}) + \left(x - \frac{1}{2}\right)\left(y - \frac{1}{2}\right),$$

where $B_2(x) = x^2 - x + 1/6$ is the Bernoulli polynomial of degree 2 and $\gamma := \{\gamma_{s,u}\}$ is a sequence of positive numbers (the weights sequence). One can take $\gamma_{s,u} = 0$ as the limiting case of positive $\gamma_{s,u}$'s. The weighted Sobolev space with [illegible] weights is related to the weighted Korobov space with general weights studied in [6]; see section 3.

The inner product in the space $H(K_{s,\gamma})$ is

$$\langle f, g \rangle_{H(K_{s,\gamma})} = \sum_{u \subseteq \mathcal{A}} \gamma_{s,u}^{-1} \int_{[0,1]^{|u|}} \left(\int_{[0,1]^{s-|u|}} \frac{\partial^{|u|} f}{\partial \mathbf{x}_u} d\mathbf{x}_{-u} \right) \left(\int_{[0,1]^{s-|u|}} \frac{\partial^{|u|} g}{\partial \mathbf{x}_u} d\mathbf{x}_{-u} \right) d\mathbf{x}_u,$$

where $\mathbf{x}_u$ denotes the vector of the coordinates of $\mathbf{x}$ with indices in u, $-u$ denotes the complementary set $\mathcal{A} - u$, and $|u|$ denotes the cardinality. The term corresponding to $u = \emptyset$ in the inner product is interpreted as $\int_{[0,1]^s} f(\mathbf{x}) d\mathbf{x} \int_{[0,1]^s} g(\mathbf{x}) d\mathbf{x}$. (We assume that $\gamma_{s,\emptyset} = 1$.) The weight $\gamma_{s,u}$ measures the importance of the group of variables in u. If $||f||_{H(K_{s,\gamma})} \le 1$ and $\gamma_{s,u}$ is small, then $\frac{\partial^{|u|} f}{\partial \mathbf{x}_u}$ is small.

Based on (2), the square worst-case error is

$$D^2(P; H(K_{s,\gamma})) = \frac{1}{n^2} \sum_{i,k=0}^{n-1} \sum_{\emptyset \neq u \subseteq \mathcal{A}} \gamma_{s,u} \prod_{j \in u} \eta(x_{i,j}, x_{k,j}).$$

The most well-studied form of weights are the product weights:

$$\gamma_{s,u} = \prod_{j \in u} \gamma_{s,j} \quad \text{for} \quad \text{some} \quad \{\gamma_{s,j}\}. \tag{5}$$

In this case, the kernel (4) can be written as a product

$$K_{s,\gamma}(\mathbf{x}, \mathbf{y}) = \prod_{j=1}^{s} [1 + \gamma_{s,j}\, \eta(x_j, y_j)],$$

and the square worst-case error is

$$D^2(P; H(K_{s,\gamma})) = -1 + \frac{1}{n^2} \sum_{i,k=0}^{n-1} \prod_{j=1}^{s} [1 + \gamma_{s,j}\, \eta(x_{i,j}, x_{k,j})].$$

2.3. Projections of a point set and QMC quadrature error. We discuss how the uniformity of the projections of a point set affects the quadrature error (see [9]). Observe that $K_u(\mathbf{x}_u, \mathbf{y}_u)$ is also a reproducing kernel. The associated RKHS is denoted by $H(K_u)$. The inner product in $H(K_u)$ is

$$\langle f, g \rangle_{H(K_u)} = \gamma_{s,u}^{-1} \int_{[0,1]^{|u|}} \frac{\partial^{|u|} f(\mathbf{x}_u)}{\partial \mathbf{x}_u} \frac{\partial^{|u|} g(\mathbf{x}_u)}{\partial \mathbf{x}_u}\, d\mathbf{x}_u \qquad \forall f, g \in H(K_u).$$

It is known (see [10]) that the original RKHS $H(K_{s,\gamma})$ can be written as the direct sum of the spaces $H(K_u)$:

$$H(K_{s,\gamma}) = \bigoplus_{u \subseteq \mathcal{A}} H(K_u).$$

Therefore, an arbitrary function $f \in H(K_{s,\gamma})$ has a unique projection decomposition (similar to the ANOVA decomposition described below):

$$f(\mathbf{x}) = \sum_{u \subseteq \mathcal{A}} f_u(\mathbf{x}_u) \qquad \text{with} \quad f_u \in H(K_u), \tag{6}$$

and

$$||f||^2_{H(K_{s,\gamma})} = \sum_{u \subseteq \mathcal{A}} ||f_u||^2_{H(K_{s,\gamma})}.$$

Note that for any $f \in H(K_u)$, we have $||f||_{H(K_u)} = ||f||_{H(K_{s,\gamma})}$.

For functions of $H(K_{s,\gamma})$ the projection decomposition (6) coincides with the ANOVA decomposition; see [5]. The ANOVA decomposition of a function is defined recursively by

$$f_u(\mathbf{x}) = \int_{[0,1]^{s-|u|}} f(\mathbf{x}) d\mathbf{x}_{-u} - \sum_{v \subset u} f_v(\mathbf{x}) \quad \text{for} \quad \emptyset \neq u \subseteq \mathcal{A},$$

with $f_\emptyset(\mathbf{x}) = I_s(f)$. The ANOVA decomposition is used in QMC to explore the dimension structure of functions such as the effective dimensions; see [3, 26, 27].

For a point set P, let P_u denote its projection on the space with coordinate indices in u. From (3) it is obvious that

$$|I_s(f_u) - Q_{n,s}(f_u)| \le D(P_u; H(K_u))\ ||f_u||_{H(K_{s,\gamma})},$$

where

$$D^2(P_u; H(K_u)) = \frac{\gamma_{s,u}}{n^2} \sum_{i,k=0}^{n-1} \prod_{j\in u} \eta(x_{i,j}, x_{k,j}).$$

Therefore, for $f \in H(K_{s,\gamma})$ we have

$$\begin{aligned} |I_s(f) - Q_{n,s}(f)| &\le \sum_{\emptyset\ne u\subseteq\mathcal{A}} |I_s(f_u) - Q_{n,s}(f_u)| \\ &\le \sum_{\emptyset\ne u\subseteq\mathcal{A}} D(P_u; H(K_u))\ ||f_u||_{H(K_{s,\gamma})}. \end{aligned} \tag{7}$$

This demonstrates a relationship of the quadrature error with the uniformity of the projections P_u and the ANOVA terms f_u. If a QMC point set has good projections P_u for the subsets u that correspond to important terms f_u, then a good result can be expected. It would be nice to find a point set P such that $D(P_u; H(K_u))$ is minimized for each nonempty subset $u \subseteq \mathcal{A}$. However, there are $2^s - 1$ nonempty subsets of $\mathcal{A}$, making the optimization problem too difficult.

2.4. Measuring the uniformity of projections of the same order. By defining

$$\mathcal{D}^2_{(\ell)}(P) := \sum_{u\subseteq\mathcal{A},\,|u|=\ell} D^2(P_u; H(K_u)), \quad \ell = 1, 2, \ldots, s,$$

the integration error $I_s(f) - Q_{n,s}(f)$ can be bounded in terms of $\mathcal{D}_{(\ell)}(P)$ (after using the Cauchy–Schwarz inequality in (7)):

$$|I_s(f) - Q_{n,s}(f)| \le \sum_{\ell=1}^{s} \mathcal{D}_{(\ell)}(P)\ ||f_{(\ell)}||,$$

where

$$||f_{(\ell)}||^2 = \sum_{u\subseteq\mathcal{A},\,|u|=\ell} ||f_u||^2_{H(K_{s,\gamma})}. \tag{8}$$

The quantity $\mathcal{D}_{(\ell)}(P)$ is called the [illegible] ; see [29]. It measures the uniformity of the projections of order-ℓ taken together and is the worst-case error of the algorithm $Q_{n,s}(f)$ over the unit ball of the RKHS $H(K_{(\ell)})$, i.e.,

$$\mathcal{D}_{(\ell)}(P) = D(P; K_{(\ell)}),$$

where $K_{(\ell)}(\mathbf{x}, \mathbf{y})$ is the reproducing kernel

$$K_{(\ell)}(\mathbf{x}, \mathbf{y}) := \sum_{u\subseteq\mathcal{A},\,|u|=\ell} K_u(\mathbf{x}_u, \mathbf{y}_u) \tag{9}$$

and (8) gives the square norm of functions in the RKHS $H(K_{(\ell)})$.

Instead of trying to find point sets P such that $D(P_u; H(K_u))$ is small for all $u \subseteq \mathcal{A}$, one may try to find P such that $\mathcal{D}_{(\ell)}(P)$ is small for $\ell = 1, 2, \ldots, s$ or at least for $\ell \le 3$. It is shown that for relatively small n (in 1000's) and large s (say, $s = 64$), the Sobol' points have order-2 and higher-order discrepancies no smaller than those of random points [29]. It is challenging to construct point sets with smaller order-2 and higher-order discrepancies. We are interested in whether good lattice points can do better.

The order-ℓ discrepancy has two drawbacks. First, for an arbitrary point set P and arbitrary weights $\gamma_{s,u}$, computing $\mathcal{D}_{(\ell)}(P)$ is expensive: direct computation based on its definition is equivalent to computing $D(P_u; H(K_u))$ for each subset $u \subseteq \mathcal{A}$ with $|u| = \ell$, the number of which is $O(s^\ell)$. Second, the order-ℓ discrepancy does not have the shift-invariant property: for a point set P, its shifted version $P + \Delta := \{\{\mathbf{x}_k + \Delta\}, k = 1, \ldots, n-1\}$, where $\Delta \in [0,1)^s$, may have a different order-ℓ discrepancy. For example, consider a point set $P := \{k/n : k = 0, \ldots, n-1\}$ in $[0,1]$. It is easy to calculate that $\mathcal{D}_{(1)}(P) = \sqrt{3}/(3n)$. However, for $\Delta = 1/(2n)$, the shifted point set is $P + \Delta = \{(2k+1)/(2n) : k = 0, \ldots, n-1\}$ and $\mathcal{D}_{(1)}(P+\Delta) = \sqrt{3}/(6n) = \mathcal{D}_{(1)}(P)/2$.

The first drawback can be overcome if we use product weights (5) or [illegible]; see the next section. The second deficiency can be avoided by using the [illegible] (the mean is taken over all random shifts), defined by

$$\int_{[0,1]^s} \mathcal{D}^2_{(\ell)}(P + \Delta)\, d\Delta.$$

It turns out that this is related to the theory of the shift-invariant kernel.

3. The construction of good lattice rules. We are interested in constructing good lattice points such that their mean-squared order-ℓ discrepancies (as defined above) are small, at least for $\ell \le 3$. A suitable quality measure is crucial for this purpose. The theory of the shift-invariant kernel is useful.

3.1. Shift-invariant kernel. For an arbitrary reproducing kernel $K(\mathbf{x}, \mathbf{y})$, its associated shift-invariant kernel is defined as

$$K^{\mathrm{sh}}(\mathbf{x}, \mathbf{y}) := \int_{[0,1]^s} K(\{\mathbf{x} + \Delta\}, \{\mathbf{y} + \Delta\})\, d\Delta.$$

By shift-invariant, we mean that for arbitrary $\Delta \in [0,1)^s$,

$$K^{\mathrm{sh}}(\mathbf{x}, \mathbf{y}) = K^{\mathrm{sh}}(\{\mathbf{x} + \Delta\}, \{\mathbf{y} + \Delta\}) \qquad \forall \mathbf{x}, \mathbf{y} \in [0,1]^s.$$

It is shown in [9, 11] that for a point set P in (1), we have

$$\int_{[0,1]^s} D^2(P + \Delta; H(K))\, d\Delta = D^2(P; H(K^{\mathrm{sh}})). \tag{10}$$

This implies that there exists a shift $\Delta \in [0,1)^s$ such that

$$D(P + \Delta; H(K)) \le D(P; H(K^{\mathrm{sh}})).$$

In other words, $D(P; H(K^{\mathrm{sh}}))$ gives an upper bound on the value of $D(P+\Delta; H(K))$ with a good choice of Δ and measures the average performance of $D(P + \Delta; H(K))$

with respect to the random shifts Δ. Moreover, $D(P; H(K^{\rm sh}))$ is shift-invariant: i.e., $D(P+\Delta; H(K^{\rm sh})) = D(P; H(K^{\rm sh}))$ for any shift $\Delta \in [0,1)^s$. There is an added benefit to using the shift-invariant kernel: for a rank-1 lattice point set

$$P := P(\mathbf{z}) := \left\{ \left\{ \frac{k\mathbf{z}}{n} \right\} : k = 0, \ldots, n-1 \right\}, \tag{11}$$

the computational cost of evaluating the worst-case error $D(P(\mathbf{z}); H(K^{\rm sh}))$ is reduced. Indeed, from (2) and from the shift-invariant property of $K^{\rm sh}(\mathbf{x}, \mathbf{y})$, we have (see [9])

$$D^2(P(\mathbf{z}); H(K^{\rm sh})) = -\int_{[0,1]^s} K^{\rm sh}(\mathbf{x}, \mathbf{0})\, d\mathbf{x} + \frac{1}{n} \sum_{k=0}^{n-1} K^{\rm sh}(\mathbf{x}_k, \mathbf{0}). \tag{12}$$

Although $D(P; H(K))$ typically takes an $O(n^2)$ operation to evaluate for a general point set (see (2)), $D(P(\mathbf{z}); H(K^{\rm sh}))$ takes only an $O(n)$ operation to evaluate for a lattice point set.

3.2. Quality measures of lattice rules. In dimension $s \geq 3$ one normally finds lattice rules by computer searches based on some criterion of "goodness." There are a number of such criteria available; see [9, 18, 20]. Here we will use a criterion based on the worst-case error of the RKHS, whose kernel is the shift-invariant kernel of the kernel given in (4).

For the kernel (4), its associated shift-invariant kernel can easily be found to be

$$K^{\rm sh}_{s,\gamma}(\mathbf{x}, \mathbf{y}) = 1 + \sum_{\emptyset \neq u \subseteq \mathcal{A}} \gamma_{s,u} \prod_{j \in u} B_2(\{x_j - y_j\}),$$

where we have used $\int_0^1 \eta(\{x+\Delta\}, \{y+\Delta\}) d\Delta = B_2(\{x-y\})$. For an arbitrary point set P in (1), from (2) it follows that

$$D^2(P; H(K^{\rm sh}_{s,\gamma})) = \frac{1}{n^2} \sum_{i,k=0}^{n-1} \sum_{\emptyset \neq u \subseteq \mathcal{A}} \gamma_{s,u} \prod_{j \in u} B_2(\{x_{i,j} - x_{k,j}\}).$$

For the rank-1 lattice point set $P(\mathbf{z})$ in (11), based on (12) we have a simplification

$$D^2(P(\mathbf{z}); H(K^{\rm sh}_{s,\gamma})) = \frac{1}{n} \sum_{k=0}^{n-1} \sum_{\emptyset \neq u \subseteq \mathcal{A}} \gamma_{s,u} \prod_{j \in u} B_2\left(\left\{ \frac{kz_j}{n} \right\} \right).$$

Note that if the weights $\gamma_{s,u}$ are of the product form (5), the computation of the quantity $D^2(P; H(K^{\rm sh}_{s,\gamma}))$ is simpler. For example, for the rank-1 lattice point set $P(\mathbf{z})$ in (11), we have

$$D^2(P(\mathbf{z}); H(K^{\rm sh}_{s,\gamma})) = -1 + \frac{1}{n} \sum_{k=0}^{n-1} \prod_{j=1}^{s} \left[1 + \gamma_{s,j} B_2\left(\left\{ \frac{kz_j}{n} \right\} \right) \right]. \tag{13}$$

From the general relation (10) we have

$$\int_{[0,1]^s} D^2(P+\Delta; H(K_{s,\gamma}))\, d\Delta = D^2(P; H(K^{\rm sh}_{s,\gamma})).$$

We will use the worst-case error corresponding to the shift-invariant kernel $K^{\rm sh}_{s,\gamma}(\mathbf{x},\mathbf{y})$, i.e., $D(P;H(K^{\rm sh}_{s,\gamma}))$, as a quality measure for lattice rules. It depends on the weights $\gamma_{s,u}$ and is a mixture of the uniformity of various projections. Judged by such a measure, the goodness of a lattice point set depends not only on the point set itself, but also on the weights. A lattice point set with small $D(P;H(K^{\rm sh}_{s,\gamma}))$ does not necessarily have good low-order projections, say, order-2 projections; see section 4. Our purpose is to choose appropriate weights $\gamma_{s,u}$, such that good low-order projections are achieved, or such that the projections of some specific orders are optimized.

. . .. 1. Since the Bernoulli polynomial of degree 2 can be expressed as

$$B_2(x) = \frac{1}{2\pi^2} \sum_{h=-\infty}^{\infty}{}' \frac{e^{2\pi i h x}}{h^2}, \qquad x \in [0,1], \tag{14}$$

where the prime on the sum indicates that the $h = 0$ term is omitted, the shift-invariant kernel $K^{\rm sh}_{s,\gamma}(\mathbf{x},\mathbf{y})$ can be written as

$$K^{\rm sh}_{s,\gamma}(\mathbf{x},\mathbf{y}) = 1 + \sum_{\emptyset\ne u\subseteq\mathcal{A}} \hat\gamma_{s,u} \prod_{j\in u} \sum_{h=-\infty}^{\infty}{}' \frac{e^{2\pi i h(x_j-y_j)}}{h^2},$$

where

$$\hat\gamma_{s,u} = \frac{\gamma_{s,u}}{(2\pi^2)^{|u|}}.$$

This is the reproducing kernel of a [illegible] with the weights $\hat\gamma_{s,u}$ and $\alpha = 2$; see [6]. If the weights $\gamma_{s,u}$ are of the product form (5), then the weights $\hat\gamma_{s,u}$ are also of the product form

$$\hat\gamma_{s,u} = \prod_{j\in u} \hat\gamma_{s,j} \quad \text{with} \quad \hat\gamma_{s,j} = \frac{\gamma_{s,j}}{2\pi^2},$$

and the kernel $K^{\rm sh}_{s,\gamma}(\mathbf{x},\mathbf{y})$ can be written as

$$K^{\rm sh}_{s,\gamma}(\mathbf{x},\mathbf{y}) = \prod_{j=1}^{s} \left(1 + \hat\gamma_{s,j} \sum_{h=-\infty}^{\infty}{}' \frac{e^{2\pi i h(x_j-y_j)}}{h^2}\right).$$

This is a special case of Korobov kernel considered in [24].

Now consider the reproducing kernel $K_{(\ell)}(\mathbf{x},\mathbf{y})$ given in (9), which is related to the order-ℓ discrepancy defined in section 2.4. Its associated shift-invariant kernel is

$$K^{\rm sh}_{(\ell)}(\mathbf{x},\mathbf{y}) = \sum_{u\subseteq\mathcal{A},\,|u|=\ell} \gamma_{s,u} \prod_{j\in u} B_2(\{x_j - y_j\}).$$

For an arbitrary point set P in (1), from (2) we have

$$D^2(P;H(K^{\rm sh}_{(\ell)})) = \frac{1}{n^2} \sum_{i,k=0}^{n-1} \sum_{u\subseteq\mathcal{A},\,|u|=\ell} \gamma_{s,u} \prod_{j\in u} B_2(\{x_{i,j} - x_{k,j}\}). \tag{15}$$

For the rank-1 lattice point set $P(\mathbf{z})$ in (11), from (12) we have a simplification

$$D^2(P(\mathbf{z});K^{\rm sh}_{(\ell)}) = \frac{1}{n} \sum_{k=0}^{n-1} \sum_{u\subseteq\mathcal{A},\,|u|=\ell} \gamma_{s,u} \prod_{j\in u} B_2\left(\left\{\frac{kz_j}{n}\right\}\right).$$

Based on (10), it is obvious that $D(P;K^{\rm sh}_{(\ell)})$ gives an upper bound on the order-ℓ discrepancy of the shifted point set $P+\Delta$ by a good choice of Δ.

3.3. The elementary order-ℓ discrepancy. In the rest of this paper we are interested in the [illegible] weights (see [6]), that is,

$$\gamma_{s,\emptyset} = 1,\ \gamma_{s,u} := \Gamma_{|u|}, \tag{16}$$

for some nonnegative numbers $\Gamma_1, \ldots, \Gamma_s$ (they may depend on the dimension s). The order-dependent weights depend on u only through its order. A benefit of using order-dependent weights is that there exists a fast algorithm to compute $D(P; H(K^{\mathrm{sh}}_{(\ell)}))$ as shown below. The order-dependent weights are also of the product form if and only if $\Gamma_\ell = \beta^\ell$ for some constant β.

For an arbitrary point set P in (1), its [illegible] ℓ [illegible], denoted by $G_{(\ell)}(P)$, is defined as

$$G^2_{(\ell)}(P) := \frac{1}{n^2} \sum_{i,k=0}^{n-1} \sum_{u \subseteq \mathcal{A},\, |u|=\ell} \prod_{j \in u} B_2(\{x_{i,j} - x_{k,j}\}).$$

The value of $G_{(\ell)}(P)$ is equal to $D(P; H(K^{\mathrm{sh}}_{(\ell)}))$ in (15) with $\gamma_{s,u} = 1$ for all $u \subseteq \mathcal{A}$ with $|u| = \ell$. If P is a lattice point set (11), then

$$G^2_{(\ell)}(P(\mathbf{z})) = \frac{1}{n} \sum_{k=0}^{n-1} \sum_{u \subseteq \mathcal{A},\, |u|=\ell} \prod_{j \in u} B_2\left(\left\{\frac{kz_j}{n}\right\}\right).$$

If P is a set of independent samples from the uniform distribution over $[0,1]^s$, then

$$E[G^2_{(\ell)}(P)] = \frac{1}{6^\ell n} \binom{s}{\ell}. \tag{17}$$

Obviously, for order-dependent weights (16), $D(P; H(K^{\mathrm{sh}}_{(\ell)}))$ given in (15) can be expressed in terms of the elementary order-ℓ discrepancies:

$$D^2(P; H(K^{\mathrm{sh}}_{(\ell)})) = \Gamma_\ell\, G^2_{(\ell)}(P),$$

and the worst-case error $D(P; H(K^{\mathrm{sh}}_{s,\gamma}))$ can then be expressed as the weighted sum of the elementary order-ℓ discrepancies:

$$D^2(P; H(K^{\mathrm{sh}}_{s,\gamma})) = \sum_{\ell=1}^{s} D^2(P; K^{\mathrm{sh}}_{(\ell)}) = \sum_{\ell=1}^{s} \Gamma_\ell\, G^2_{(\ell)}(P). \tag{18}$$

The elementary order-ℓ discrepancy $G_{(\ell)}(P)$ measures the quality of all order-ℓ projections taken together. Its advantage is that it does not depend on any weight (it depends only on the point set itself). We will use it to compare different (lattice) point sets. We will show that such a measure is more informative than the worst-case error.

The formulas for the elementary order-ℓ discrepancies involve quantities of the form

$$\sum_{u \subseteq \mathcal{A},\, |u|=\ell} \prod_{j \in u} C_j \quad \text{for some} \quad C_1, C_2, \ldots, C_s.$$

Such quantities can be efficiently computed using a recursive formula as shown in [6]. This recursive method will be used in our algorithms below.

3.4. The construction algorithms. Two constructions of lattice rules will be considered: the Korobov and the component-by-component (CBC) constructions. They are studied in [12] and [22] under the classical quality measure P_α. Here we use a weighted version of P_α with a good choice of weights. The theory of weighted function spaces yields rich results on the constructions of lattice rules. For practical applications, the principal difficulty is to choose appropriate weights. An attempt to choose product weights was made in [28] using a matching strategy. It should be emphasized that the resulting generator using a weighted version of P_α will almost always be quite different from that produced by using P_α (see section 4). We assume that n is a prime and the weights are order-dependent.

ALGORITHM 1 (Korobov construction).

For a fixed dimension s and for given order-dependent weights $\Gamma_1, \Gamma_2, \ldots$, find the optimal Korobov-form generator

$$\mathbf{z}(a_s) := (1, a_s, \ldots, a_s^{s-1}) \,(\mathrm{mod}\, n),$$

with $a_s \in \{1, \ldots, n-1\}$, by minimizing the square of the worst-case error

$$D^2(P(\mathbf{z}(a_s)); H(K^{\mathrm{sh}}_{s,\gamma})) = \frac{1}{n} \sum_{\ell=1}^{s} \sum_{k=0}^{n-1} \Gamma_\ell \sum_{u \subseteq \mathcal{A},\, |u|=\ell} \prod_{j \in u} B_2\left(\left\{\frac{k a_s^{j-1}}{n}\right\}\right).$$

ALGORITHM 2 (component-by-component construction).

For given weights $\Gamma_1, \Gamma_2, \ldots$, the generator $\mathbf{z}$ is found one component at a time:

1. Set z_1, the first component of $\mathbf{z}$, to 1.

2. For $s = 2, \ldots, s_{\max}$, with the components $z_1, \ldots, z_{s-1}$ fixed, find $z_s \in \{1, \ldots, n-1\}$ such that

$$D^2(P(z_1, \ldots, z_s); H(K^{\mathrm{sh}}_{s,\gamma})) = \frac{1}{n} \sum_{\ell=1}^{s} \sum_{k=0}^{n-1} \Gamma_\ell \sum_{u \subseteq \mathcal{A},\, |u|=\ell} \prod_{j \in u} B_2\left(\left\{\frac{k z_j}{n}\right\}\right) \tag{19}$$

is minimized.

The CBC lattice rules have the advantage of being extensible in dimension s if the weights are independent of s, and they achieve the (strong) tractability error bound under appropriate conditions on the weights [6]. A faster CBC algorithm was developed in [19]. The error bounds of Korobov rules in weighted spaces were studied in [30] for product weights (the same analysis applies to nonproduct weights). Faster Korobov algorithms are also desirable. We are mainly interested in the potential possibility of these constructions in achieving good elementary order-ℓ discrepancies by suitably choosing the weights $\Gamma_1, \Gamma_2, \ldots$.

If the order-dependent weights are also of the product form, i.e., $\Gamma_\ell = \beta^\ell$ for some β, then a more efficient way to compute $D^2(P(\mathbf{z}); H(K^{\mathrm{sh}}_{s,\gamma}))$ is to use the formula (13). In this case, the quality of low-order projections of the resulting lattice points can be [illegible] to the parameter β, and thus this parameter should be chosen carefully to reflect the relative importance of ANOVA terms of different orders (for this reason, product weights are not the focus of this paper).

4. Good lattice points in high dimensions: How well are their projections distributed? We take a completely different strategy from [6] to investigate the quality of good lattice points; namely, we focus on the [illegible] of different orders. This allows us to assess the quality of the projections of some selected orders

and to examine the possible advantages of lattice points over digital nets and random points. Note that an [illegible] lattice point set (11) with prime n has the same order-1 projections, namely, $\{k/n : k = 0, 1, \ldots, n-1\}$, which are [illegible] perfectly distributed. Thus we need only focus on order-2 and higher-order projections.

The formula (18) indicates that the square worst-case error $D^2(P; H(K^{\mathrm{sh}}_{s,\gamma}))$ is the [illegible] of the square elementary order-ℓ discrepancies. The weights allow greater or smaller emphasis on the projections of different orders and affect the goodness of projections of various orders. The choice of the weights should reflect the characteristic of the problems at hand. We consider two choices of order-dependent weights:

- [illegible] (A): $\Gamma_1 = \Gamma_2 = 1$, and $\Gamma_\ell = 0$ for $\ell \geq 3$.
- [illegible] (B): $\Gamma_1 = \Gamma_2 = \Gamma_3 = 1$, and $\Gamma_\ell = 0$ for $\ell \geq 4$.

The corresponding lattice rule (found by Algorithm 1 or 2) is known as (Korobov or CBC) lattice rule (A) or (B). The choice (A) puts all emphasis on order-2 projections and thus may result in lattice rules with optimal order-2 projections, while the choice (B) puts emphasis on both order-2 and order-3 projections. The relative size of Γ_3 with respect to Γ_2 is important. The choice in (B) is intended for use with finance problems in section 5 and is based on the matching strategy in [28].

We compare the Korobov or CBC lattice rule (A) or (B) with the Sobol' points [25] by comparing their elementary order-ℓ discrepancies. The root mean-squared elementary order-ℓ discrepancy (see (17)), is included as a benchmark.

We are also interested in the following problem: given a method of construction, say, the Korobov or CBC construction, with the freedom of choosing order-dependent weights, what is the [illegible] of the elementary order-ℓ discrepancy which can be achieved by this construction for a [illegible] ℓ? This value and the mean value for a random point set can be used as benchmarks. They indicate how good (or bad) the order-ℓ projections in the optimal (or average) case can be, implying the possibility or impossibility of good lattice points.

It turns out that for a [illegible] ℓ with $1 < \ell \leq s$, the optimal elementary order-ℓ discrepancy, which can be achieved by the Korobov or CBC method, can be easily found: one just chooses the weights to be

$$\Gamma_\ell = 1, \quad \text{but} \quad \Gamma_j = 0 \quad \text{for} \quad j \neq \ell. \tag{20}$$

This choice of weights in the Korobov or CBC algorithm results in lattice points achieving the minimum elementary order-ℓ discrepancy among all possible order-dependent weights using the Korobov or CBC algorithm, respectively. (An interesting problem is the following: what is the minimum value of the elementary order-ℓ discrepancy without restricting a method of construction?)

Table 1 presents the comparisons of the elementary order-ℓ discrepancies for $s = 64$. The optimal values of the elementary order-ℓ discrepancies (obtained as indicated above) for the Korobov or CBC construction are also included. The comparisons clearly indicate the potential possibility and limitations of good lattice points. We observe the following:

- For $\ell = 1$, lattice points (A) and (B) (and Sobol' points) have elementary order-1 discrepancies and convergence order much better than those of random points.
- For $\ell = 2$, lattice points (A) and (B) have elementary order-2 discrepancies much better than those of the Sobol' points and random points. The convergence order of the elementary order-2 discrepancies of lattice points (A)

TABLE 1

Shown are the elementary order-ℓ discrepancies and the convergence orders in dimension $s = 64$. For the Sobol' points, $n = 2^8, 2^{10}$, and 2^{12}. The convergence order (i.e., the value r in an expression of the form $O(n^{-r})$) is estimated from linear regression on the empirical data. "Mean" in the third column is the root mean square elementary order-ℓ discrepancy; see (17)*. The optimal values of the elementary order-ℓ discrepancies are obtained (for the Korobov and CBC constructions) using the weights* (20) *for each fixed ℓ.*

				Lattice (A)		Lattice (B)		Optimal values	
$G_{(\ell)}$	n	Mean	Sobol	Korobov	CBC	Korobov	CBC	Korobov	CBC
$G_{(1)}$	251	2.06e-1	2.13e-2	1.30e-2	1.30e-2	1.30e-2	1.30e-2	1.30e-2	1.30e-2
	1009	1.03e-1	5.66e-3	3.24e-3	3.24e-3	3.24e-3	3.24e-3	3.24e-3	3.24e-3
	4001	5.16e-2	1.34e-3	8.16e-4	8.16e-4	8.16e-4	8.16e-4	8.16e-4	8.16e-4
	r	0.50	1.00	1.00	1.00	1.00	1.00	1.00	1.00
$G_{(2)}$	251	4.72e-1	5.53e-1	3.09e-1	3.21e-1	3.11e-1	3.55e-1	3.09e-1	3.21e-1
	1009	2.36e-1	2.95e-1	9.56e-2	9.31e-2	9.56e-2	1.27e-1	9.56e-2	9.31e-2
	4001	1.18e-1	1.21e-1	2.48e-2	2.49e-2	3.39e-2	4.66e-2	2.48e-2	2.49e-2
	r	0.50	0.50	0.91	0.92	0.80	0.73	0.91	0.92
$G_{(3)}$	251	8.77e-1	8.13e-1	8.84e-1	8.82e-1	8.84e-1	8.39e-1	7.71e-1	6.77e-1
	1009	4.37e-1	4.10e-1	4.19e-1	4.41e-1	4.19e-1	3.91e-1	3.87e-1	3.12e-1
	4001	2.20e-1	2.13e-1	2.02e-1	2.29e-1	1.70e-1	1.67e-1	1.50e-1	1.32e-1
	r	0.50	0.48	0.53	0.49	0.59	0.58	0.59	0.59
$G_{(4)}$	251	1.40e00	1.43e00	1.40e00	1.40e00	1.40e00	1.42e00	1.40e00	1.40e00
	1009	6.97e-1	7.13e-1	6.99e-1	6.97e-1	6.99e-1	7.15e-1	6.92e-1	6.87e-1
	4001	3.50e-1	3.59e-1	3.50e-1	3.50e-1	3.61e-1	3.58e-1	3.41e-1	3.38e-1
	r	0.50	0.50	0.50	0.50	0.49	0.50	0.51	0.51

and (B) is much faster than that of the Sobol' points (the Sobol' points lost the advantage over random points in order-2 projections on average). Lattice points (A) give the best elementary order-2 discrepancies.

- For $\ell = 3$, the elementary order-3 discrepancies of lattice points (A) and (B) are similar to those of the Sobol' points and random points and are close to the optimal values. Lattice points (B) have elementary order-3 discrepancies better than those of lattice points (A).
- For $\ell \geq 4$, lattice points (A) and (B) have order-ℓ projections [illegible] than those of random points (this is a surprise). For $\ell \geq 4$, even the optimal elementary order-ℓ discrepancy, which can be achieved by the Korobov or CBC construction, is only slightly better than the mean values for random points (in Table 1 we do not give the numerical results for $\ell > 4$). It seems that for $\ell \geq 4$, if n is small (say, in 1000's) and s is large (say, $s > 50$), one should not expect to achieve much better order-ℓ projections than those of random points. There exists a limit for given n and s. This indicates the difficulty in achieving good higher-order projections.
- In all cases, the Korobov and CBC lattice points have similar behavior.

[illegible] 2. A lattice point set which achieves the optimal elementary order-ℓ discrepancy with $\ell > 2$ (see (20) for the construction of such a point set) may have very bad elementary order-2 discrepancy. For example, the lattice points which achieve the optimal elementary order-4 discrepancy have very bad elementary order-2 discrepancy. Thus the Korobov or CBC lattice points which achieve the optimal elementary order-ℓ discrepancy for $\ell \geq 4$ can be useful only when the integrand is dominated by order-ℓ ANOVA terms (and the possible advantage is very limited). Our interest in them is just due to the motivation to know what is the best possible order-ℓ discrepancy for a given ℓ and n. On the other hand, the lattice points found with the weights (A) and (B) have not only very good (order-1 and) order-2 projections but also higher-order projections no worse than those of random points. Note that an alternative

TABLE 2

Shown are the elementary order-ℓ discrepancies for the classical (Korobov or CBC) lattice rules constructed based on the quality measure P_2 and the mean values for random points in dimension $s = 64$.

n		$G_{(1)}$	$G_{(2)}$	$G_{(3)}$	$G_{(4)}$	$G_{(8)}$	$G_{(16)}$	$G_{(32)}$	$G_{(64)}$
251	Mean	2.06e-1	4.72e-1	8.77e-1	1.40e00	3.24e00	8.30e-1	3.03e-5	7.93e-27
	Korobov	1.30e-2	8.22e-1	8.70e-1	1.40e00	3.24e00	8.30e-1	3.03e-5	7.93e-27
	CBC	1.30e-2	2.55e00	2.70e00	4.68e00	6.88e00	1.16e00	3.16e-5	7.91e-27
1009	Mean	1.03e-1	2.36e-1	4.37e-1	6.97e-1	1.62e00	4.14e-1	1.51e-5	3.96e-27
	Korobov	3.24e-3	3.66e-1	4.14e-1	6.98e-1	1.62e00	4.14e-1	1.51e-5	3.96e-27
	CBC	3.24e-3	3.05e00	2.90e00	5.41e00	7.29e00	1.02e00	1.85e-5	4.02e-27
4001	Mean	5.16e-2	1.18e-1	2.20e-1	3.50e-1	8.12e-1	2.08e-1	7.59e-6	1.99e-27
	Korobov	8.16e-4	3.48e-1	2.60e-1	3.61e-1	8.12e-1	2.08e-1	7.59e-6	1.99e-27
	CBC	8.16e-4	2.95e00	2.83e00	5.08e00	6.29e00	6.92e-1	8.70e-6	2.00e-27

choice of weights $\Gamma_\ell = \beta^\ell$ with a "small" parameter β may also produce lattice points with good order-2 projections. The corresponding algorithm is more efficient but less robust (since the parameter β has strong influence on the quality as mentioned above).

The attempt to improve the elementary order-ℓ discrepancies for $\ell \geq 4$ does not turn out to be rewarding for small n in high dimensions. To achieve better higher-order projections, one has to enlarge n (or find better construction beyond Korobov or CBC). In practice, we recommend using lattice rules (A). Due to their good projection property, it is reasonable to expect that they have good performance for favorable functions (dominated by order-1 and order-2 terms), while doing no harm for unfavorable ones (say, functions dominated by high-order terms). In this sense they are robust, perhaps opening the way to wider use.

We turn to the classical lattice rules. By "classical lattice rules" we mean the rules found by using the classical quality measure P_α in searching algorithms. For simplicity of presentation, we assume $\alpha = 2$; then P_2 has the following form (see [20]):

$$P_2 = \sum_{\mathbf{h}\cdot\mathbf{z}\equiv 0}{}' \frac{1}{\overline{h}_1^{\,2} \cdots \overline{h}_s^{\,2}},$$

where $\mathbf{h} = (h_1, \ldots, h_s)$, $\mathbf{h} \cdot \mathbf{z} = h_1 z_1 + \cdots + h_s x_s$, and $\overline{h}_j = \max(1, |h_j|)$.

Algorithms 1 and 2 in section 3 have parallels when P_2 is used; see [22]. Table 2 gives a comparison of the elementary order-ℓ discrepancies for the classical lattice rules in dimension $s = 64$. We observe that both the Korobov and CBC lattice points constructed based on P_2 [illegible]. Moreover, the CBC lattice rules also have very bad elementary higher-order discrepancies. Such a phenomenon may occur even in a relatively small dimension, say $s = 15$.

Why does the use of the classical quality measure P_2 result in lattice rules with such a poor distribution property? The reason is that the measure P_2 is the square worst-case error $D^2(P(\mathbf{z}); H(K^{\mathrm{sh}}_{s,\gamma}))$ given in (19) with a special choice of order-dependent weights:

$$\Gamma_j = (2\pi^2)^j, \quad j = 1, \ldots, s. \tag{21}$$

(Note that this choice is equivalent to product weights: $\gamma_{s,j} = 2\pi^2, j = 1, \ldots, s$. See also [8].) Indeed, with this choice of order-dependent weights, from (19) and (14), we

have

$$
\begin{aligned}
D^2(P(\mathbf{z}); H(K^{\mathrm{sh}}_{s,\gamma})) &= -1 + \frac{1}{n}\sum_{k=0}^{n-1}\prod_{j=1}^{s}\left(1 + 2\pi^2 B_2\left(\left\{\frac{kz_j}{n}\right\}\right)\right) \\
&= -1 + \frac{1}{n}\sum_{k=0}^{n-1}\prod_{j=1}^{s}\left(1 + {\sum_{h=-\infty}^{\infty}}' \frac{\exp(2\pi i h k z_j/n)}{h^2}\right) \\
&= -1 + \frac{1}{n}\sum_{k=0}^{n-1}\sum_{\mathbf{h}\in\mathbf{Z}^s}\prod_{j=1}^{s}\frac{\exp(2\pi i h_j k z_j/n)}{\overline{h}_j^{\,2}} \\
&= -1 + \sum_{\mathbf{h}\in\mathbf{Z}^s}\frac{1}{n}\sum_{k=0}^{n-1}\frac{(\exp(2\pi i \mathbf{h}\cdot\mathbf{z}/n))^k}{\prod_{j=1}^{s}\overline{h}_j^{\,2}} \\
&= -1 + \sum_{\mathbf{h}\cdot\mathbf{z}\equiv 0}\frac{1}{\overline{h}_1^{\,2}\cdots\overline{h}_s^{\,2}} \\
&= {\sum_{\mathbf{h}\cdot\mathbf{z}\equiv 0}}' \frac{1}{\overline{h}_1^{\,2}\cdots\overline{h}_s^{\,2}} = P_2,
\end{aligned}
$$

where $\mathbf{Z}^s$ is the set of all s-dimensional vectors of integers.

Thus using P_2 as a quality measure in the optimizing search is equivalent to using the worst-case error $D(P(\mathbf{z}); H(K^{\mathrm{sh}}_{s,\gamma}))$ with the very special order-dependent weights given in (21). The choice of weights (21) puts much more significant emphasis on higher-order projections than on lower-order ones. As we showed, an emphasis on higher-order projections does not seem to improve the elementary higher-order discrepancies but may [illegible] 2 [illegible]. In other words, the classical weights are too large to yield good results. We believe that this is the reason why the classical lattice rules may be not suitable for high-dimensional finance problems, for which the order-1 and order-2 ANOVA terms play the dominant role [28]. The classical lattice rules are optimal for certain [illegible] function classes (see [20]), but they need not be good for practical use. The functions in such an artificial class have dimension structure quite different from that of the functions from finance as analyzed in [28].

5. Pricing path-dependent options and American options. The Korobov or CBC lattice points (A) and (B) constructed in the previous section have good performance in their low-order projections even in high dimensions: they have elementary order-1 and order-2 discrepancies much better than those of the Sobol' points and random points and have elementary order-3 and higher-order discrepancies no worse. It is known that in finance applications, while the nominal dimension can be very large, the effective dimension in the superposition sense is often very small; see [26, 27]. This strongly motivates us to use lattice points with good low-order projections. In this section, we demonstrate the practical performance of the new lattice points. The finance problems we consider consist in pricing arithmetic Asian options and pricing American put options and American-Bermudan-Asian options. We test the robustness of the new lattice points by comparing their performance on different problems and on different model parameters.

5.1. Arithmetic Asian options. Consider an Asian call option based on the arithmetic average of the underlying asset. The terminal payoff is

$$g_A(S_{t_1},\dots,S_{t_s}) = \max\left(0, \frac{1}{s}\sum_{j=1}^{s} S_{t_j} - K\right),$$

where K is the strike price at the expiration date T and S_{t_j} are the prices of the underlying asset at equally spaced times $t_j = j\Delta t,\ j = 1,\dots,s,\ \Delta t = T/s$. We assume that under the risk-neutral measure the asset follows geometric Brownian motion (Black–Scholes model):

$$dS_t = rS_t dt + \sigma S_t dB_t, \tag{22}$$

where r is the risk-free interest rate, σ is the volatility, and B_t is the standard Brownian motion. The value of the option at $t = 0$ is

$$C_0 = E_Q[e^{-rT} g_A(S_{t_1},\dots,S_{t_s})],$$

where $E_Q[\cdot]$ is the expectation under the risk-neutral measure Q. The simulation method is based on the analytical solution to (22)

$$S_t = S_0 \exp((r - \sigma^2/2)t + \sigma B_t)$$

and on a generation of the Brownian motion

$$(B_{t_1},\dots,B_{t_s})^T = A\,(z_1,\dots,z_s)^T,$$

where $z_j \sim N(0,1),\ j = 1,\dots,s$, are independent and identically distributed standard normal variables and A is an $s \times s$ matrix A satisfying $AA^T = V$ with $V = (\min(t_i,t_j))_{i,j=1}^{s}$. The standard construction (i.e., sequential sampling) takes A to be the Cholesky decomposition of V.

5.1.1. The efficiency improvement techniques. To improve QMC, we use dimension reduction techniques, such as the Brownian bridge (BB) (see [3, 17]), and the principal component analysis (PCA) (see [1]), and variance reduction techniques, such as antithetic variates, control variates, and their combination. We take the payoff of the geometric average Asian call option (which is analytically tractable) as a control variable: $g_G(S_{t_1},\dots,S_{t_s}) = \max(0, \prod_{j=1}^{s} S_{t_j}^{1/s} - K)$. Note that

$$E_Q[g_A(\cdot)] = E_Q\left[g_A(\cdot) - b\, g_G(\cdot)\right] + b\, E_Q[g_G(\cdot)],$$

where b is some multiplier b. The optimal b^* is obtained by minimizing the variance of $g_A - bg_G$. The same multiplier b^* will be used in QMC. Importance sampling and weighted importance sampling in QMC can also be effectively used by choosing a suitable importance density. This problem will be studied in the future.

5.1.2. The algorithms and the error estimation. We compare the efficiency of the following QMC algorithms with that of MC:

- the Korobov and CBC lattice rules (A) and (B);
- the classical Korobov or CBC lattice rules constructed based on P_2 (see [22]).
- the QMC algorithm based on the Sobol' points.

In order to get an estimation of the accuracy for each method, we compute the sample variance and the standard deviations. For the Sobol' points, we use a digital-scrambling method as used in [28]. For lattice point set $P := \{\mathbf{x}_k\}$, we use a random shift method; see [21]. Define

$$Q_{n,s}(f;\Delta) := \frac{1}{n}\sum_{k=0}^{n-1} f(\{\mathbf{x}_k + \Delta\})$$

and

$$\overline{Q}_{n,s}(f) := \frac{1}{m}\sum_{j=1}^{m} Q_{n,s}(f;\Delta_j), \tag{23}$$

where $\Delta_1, \cdots, \Delta_m$ are independent and identically distributed random vectors, uniformly distributed over $[0,1]^s$. The sample variance of the estimate (23) is estimated by

$$\widetilde{\mathrm{Var}}[\overline{Q}_{n,s}(f)] = \frac{1}{m(m-1)}\sum_{j=1}^{m}[Q_{n,s}(f;\Delta_j) - \overline{Q}_{n,s}(f)]^2.$$

The standard deviation is just the square root of the sample variance. The [illegible] or the [illegible] of the estimate (23) with respect to the crude MC estimate (i.e., without variance reduction and dimension reduction) is defined by

$$\text{Eff}\ [\overline{Q}_{n,s}] := \widetilde{\mathrm{Var}}(Q_{\mathrm{crudeMC}}(f))/\widetilde{\mathrm{Var}}[\overline{Q}_{n,s}(f)],$$

where $\widetilde{\mathrm{Var}}(Q_{\mathrm{crudeMC}}(f))$ is the sample variance of the crude MC estimate based on mn samples. The numerical results are presented in Tables 3 and 4.

5.1.3. The numerical results. Tables 3 and 4 present the comparisons for pricing arithmetic Asian options. The experiments show the following:

- The classical Korobov or CBC lattice rules constructed based on P_2 behave very badly. Their performance can be worse than that of MC. The reason is that they have very bad elementary order-2 discrepancies as shown in section 4. Thus such rules are unsuitable for functions for which order-2 ANOVA terms are important.
- The efficiency of lattice rules (A) and (B) is dramatically improved. They are more efficient than the Sobol' points when standard construction is used to generate the Brownian motion, and are at the same level of accuracy with the Sobol' points if dimension reduction techniques are used (since the possible bad order-2 and higher-order projections of Sobol' points in latter dimensions now have less influence due to the reduced effective dimensions).
- The relative efficiency of QMC algorithms is affected by the strike price K: the smaller the K is, the more efficient the QMC algorithms are. This is consistent with the fact that a smaller K leads to functions of lower superposition dimension (small K leads to integrands dominated mainly by order-1 ANOVA terms); see [27].
- Dimension reduction and variance reduction techniques provide further significant improvement for QMC (dimension reduction techniques are useless in MC). In most cases, lattice rules (A) and (B) combined with these techniques are more efficient than the Sobol' points combined with the same techniques.

TABLE 3

Shown are the standard deviations and relative efficiency (in parentheses) with respect to crude MC for the arithmetic Asian option with $m = 100$ replications for $s = 64$ ($n = 4001$ for MC and good lattice points, $n = 4096$ for the Sobol' points). The parameters are $S_0 = 100, \sigma = 0.2, T = 1.0, r = 0.1$. In the second column are the path generation methods: STD—standard construction (sequential sampling), BB—Brownian bridge, PCA—principal component analysis.

Strike price		MC	Sobol	Classical		Lattice (A)		Lattice (B)	
				Korobov	CBC	Korobov	CBC	Korobov	CBC
90	STD	1.68e-2 (1.00)	4.31e-3 (15)	1.21e-2 (1.92)	6.87e-2 (0.06)	1.92e-3 (77)	1.65e-3 (104)	2.14e-3 (62)	2.22e-3 (57)
	BB	1.68e-2 (0.998)	7.10e-4 (561)	2.11e-2 (0.63)	6.22e-3 (7.32)	1.10e-3 (235)	7.62e-4 (487)	1.55e-3 (118)	7.51e-4 (502)
	PCA	1.68e-2 (0.997)	5.82e-4 (835)	7.16e-3 (5.53)	8.73e-4 (372)	5.96e-4 (796)	5.53e-4 (927)	5.67e-4 (881)	5.64e-4 (888)
100	STD	1.37e-2 (1.00)	6.43e-3 (4.5)	1.91e-2 (0.51)	1.05e-1 (0.02)	2.91e-3 (22)	2.46e-3 (31)	2.83e-3 (23)	3.33e-3 (17)
	BB	1.37e-2 (0.999)	7.88e-4 (302)	2.81e-2 (0.24)	7.39e-3 (3.43)	1.28e-3 (115)	7.67e-4 (319)	2.17e-3 (40)	7.84e-4 (305)
	PCA	1.37e-2 (0.998)	5.28e-4 (674)	7.09e-3 (3.73)	8.40e-4 (266)	5.30e-4 (669)	5.06e-4 (734)	5.50e-4 (619)	5.03e-4 (741)
110	STD	9.07e-3 (1.00)	6.48e-3 (2)	1.85e-2 (0.24)	1.00e-1 (0.01)	3.01e-3 (9)	3.15e-3 (8)	3.16e-3 (8)	3.19e-3 (8)
	BB	9.08e-3 (0.999)	6.32e-4 (206)	2.87e-2 (0.10)	7.46e-3 (1.48)	1.29e-3 (50)	8.42e-4 (116)	2.19e-3 (17)	7.30e-4 (154)
	PCA	9.08e-3 (0.998)	4.45e-4 (416)	8.52e-3 (1.13)	9.38e-4 (94)	4.73e-4 (368)	4.40e-4 (425)	4.94e-4 (337)	4.36e-4 (434)

TABLE 4

The same as Table 3*, but with variance reduction techniques (Crude—crude estimate without variance reduction, AV—antithetic variates, CV—control variates, ACV—combination of AV and CV). The strike price is fixed to be $K = 100$. We give only the variance reduction factors (the standard deviation of the crude MC estimate is* 1.37*e*-2*).*

Path		MC	Sobol	Classical		Lattice (A)		Lattice (B)	
				Korobov	CBC	Korobov	CBC	Korobov	CBC
STD	Crude	1.0	4.5	0.5	0.017	22	31	23	17
	AV	5.6	4.8	0.5	0.017	26	33	31	21
	CV	1145	1331	110	5.3	4914	6015	4895	3910
	ACV	2113	1973	113	6.6	11716	13098	10421	6361
BB	Crude	1.0	302	0.2	3.4	115	319	40	305
	AV	5.6	756	0.2	3.4	192	690	252	697
	CV	1151	12849	61	177	13693	19282	12037	20584
	ACV	2143	16792	69	229	23535	36038	22590	38287
PCA	Crude	1.0	674	3.7	266	669	734	619	741
	AV	5.6	35868	4.2	463	8351	13851	8875	21090
	CV	1156	132610	29	6244	31161	48900	30316	69510
	ACV	2142	655760	90	13359	38484	89657	68602	89226

5.2. American options. American options are derivative securities for which the holder of the security can choose the time of exercise. The American option's additional complication—finding the optimal time for exercise—makes pricing it one of the most challenging problems in finance. Recently, several simulation methods are proposed for the valuation of American options (see, among others, [2, 15]). Our experiments are based on the least-square Monte Carlo (LSM) method [15].

Consider an American option which can be exercised only at discrete times $t_j = j\Delta t,\ j = 1, \ldots, s,\ \Delta t = T/s$ (such options are sometimes called Bermudan options). The payoff function is assumed to be $g(t, S_t)$, where S_t is the asset price. The risk-neutral dynamics for the asset price are the same as in (22). There are several major steps in the LSM method (see [15] for a more detailed description):

- Generate N realization paths $\{S_t^i, t = 0, t_1, \ldots, t_s; i = 1, \ldots, N\}$.

- Determine for each path i the optimal exercise time τ_i^*. This is done by comparing (i) the payoff from immediate exercise and (ii) the expected payoff from keeping the option alive (i.e., the continuation value). The continuation value is estimated by a least-square regression using the cross-sectional information provided by simulation. This procedure is repeated recursively going back in time from T.
- Estimate the option's price by

$$\hat{p} = \frac{1}{N} \sum_{i=1}^{N} e^{-r\tau_i^*} g(\tau_i^*, S_{\tau_i^*}),$$

where τ_i^* is the optimal exercise time for the ith path.

Note that the problem of pricing American options does not fit the framework of multivariate integration. However, we may still hope that the better distribution property of quasi-random numbers may lead to a more accurate estimate. Instead of using random points in the path generation in LSM, one may try to use quasi-random numbers and BB in path generation; see [4, 13]. In our experiments, we use good lattice points with good low-order projections and the Sobol' points. Moreover, we will also try to use principal component analysis (PCA) to generate the paths. The same method for error estimation as in section 5.1 is used here. All results for the examples below are obtained using the same basis functions as in [15].

We first look at American put options. The results are presented in Table 5 for $K = 100,\ \sigma = 0.2,\ T = 1.0,\ r = 0.1$ (with 64 exercise periods). We observe that with the standard construction the Korobov or CBC lattice points (A) and (B) consistently reduce the variance compared to crude MC (i.e., LSM based on random points) by factors of about 3 and are more efficient than the Sobol' points. Moreover, by using BB or PCA, their efficiency can be further increased in most cases, with the variance reduction factor up to 10 (but BB does not seem to improve Korobov lattice (B) in this experiment). In BB or PCA, the lattice points (A) and (B) have similar performance with the Sobol' points.

Following [15], we now consider an American-Bermudan-Asian option on the average of the stock prices during a given time horizon. It matures in one year and it cannot be exercised during the first quarter. The payoff function of this option at time t is given by $\max(0, A_t - K)$, where K is the strike price and A_t is the arithmetic average of the underlying asset during the period three months prior to time 0 (the valuation date) to time t. The results are presented in Table 6 for different pairs (A_0, S_0) and for $K = 100,\ \sigma = 0.2,\ T = 1.0,\ r = 0.1$ (the time is discretized into 64 steps). We observe, once more, that the Korobov or CBC lattice points (A) and (B) consistently reduce the variance compared to crude MC by factors up to 50 (with standard construction), and are more efficient than the Sobol' points. In the case of BB or PCA construction, the lattice points reduce the variance by factors up to 300 and have performance similar to that of the Sobol' points.

In both examples, quasi-random numbers and dimension reduction techniques in LSM have good performance. The potential difficulty with correlations between the paths did not turn to be much of a problem. The theoretical basis for using quasi-random numbers and dimension reduction in LSM is under investigation. More complicated American-type securities are worth considering, both empirically and theoretically.

A remarkable property of (Korobov or CBC) lattice points (A) is their robustness, since they are useful in a broad sense for financial simulations. Such lattice points

TABLE 5

Shown are the standard deviations and relative efficiency (in parentheses) with respect to crude MC for American put option with $m = 100$ replications ($n = 4001$ for MC and good lattice point, and $n = 4096$ for the Sobol' points). The parameters are $K = 100, \sigma = 0.2, T = 1.0, r = 0.1$. The exercise periods are $s = 64$. In the second column are the path generation methods (STD—standard construction).

Initial stock price		MC	Sobol	Lattice (A) Korobov	Lattice (A) CBC	Lattice (B) Korobov	Lattice (B) CBC
90	STD	8.78e-3 (1.00)	6.69e-3 (1.7)	4.55e-3 (3.7)	4.58e-3 (3.7)	4.45e-3 (3.9)	4.29e-3 (4.2)
	BB	8.26e-3 (1.1)	4.02e-3 (4.8)	4.83e-3 (3.3)	4.42e-2 (3.9)	4.63e-3 (3.6)	3.16e-3 (7.7)
	PCA	8.56e-3 (1.1)	4.16e-3 (4.5)	3.66e-3 (5.8)	3.73e-3 (5.5)	4.41e-3 (4.0)	3.50e-3 (6.3)
100	STD	9.09e-3 (1.00)	6.14e-3 (2.2)	4.47e-3 (4.1)	4.79e-3 (3.6)	5.41e-2 (2.8)	5.40e-3 (2.8)
	BB	9.89e-3 (0.8)	5.58e-3 (2.7)	3.19e-3 (8.1)	3.51e-3 (6.7)	7.05e-3 (1.7)	2.96e-3 (9.5)
	PCA	9.81e-3 (0.9)	2.97e-3 (9.40)	3.08e-3 (8.7)	3.10e-3 (8.6)	2.83e-3 (10.3)	2.89e-3 (9.9)
110	STD	6.54e-3 (1.00)	4.59e-3 (2.0)	3.93e-3 (2.8)	3.38e-3 (3.7)	3.31e-3 (3.9)	4.24e-3 (2.4)
	BB	7.51e-3 (0.8)	3.10e-3 (4.4)	2.46e-3 (7.1)	2.33e-3 (7.9)	6.09e-3 (1.2)	2.16e-3 (9.1)
	PCA	7.07e-3 (0.9)	1.97e-3 (11.1)	2.03e-4 (10.4)	2.29e-3 (8.2)	2.07e-3 (10.0)	1.92e-2 (11.6)

TABLE 6

Similar to Table 5, but for American-Bermudan-Asian call option. The parameters are $K = 100, \sigma = 0.2, T = 1.0, r = 0.1$. The time is discretized into 64 steps. The standard deviations are given only for crude MC estimates (in parentheses).

(A_0, S_0)		MC	Sobol	Lattice (A) Korobov	Lattice (A) CBC	Lattice (B) Korobov	Lattice (B) CBC
(90,90)	STD	(5.31e-3) 1.0	1.6	7.7	6.3	5.7	6.1
	BB	0.8	178.1	33.4	69.0	6.2	89.7
	PCA	0.7	275.0	207.4	318.7	220.6	288.2
(90,110)	STD	(1.43e-2) 1.0	9.7	52.9	56.3	41.7	32.5
	BB	0.8	238.1	158.7	96.8	103.6	271.4
	PCA	0.8	382.4	375.9	356.6	360.4	334.3
(100,90)	STD	(6.23e-3) 1.0	1.8	8.7	8.7	7.6	7.5
	BB	0.8	104.2	42.4	81.8	12.6	117.6
	PCA	0.7	221.6	123.4	177.4	157.7	249.1
(100,110)	STD	(1.43e-2) 1.0	11.6	41.6	39.1	31.9	31.2
	BB	0.8	153.9	108.7	96.4	104.4	178.3
	PCA	0.8	181.8	117.7	213.3	159.9	199.4
(110,90)	STD	(7.01e-3) 1.0	2.1	9.9	9.8	7.4	7.3
	BB	0.8	12.9	22.2	28.9	21.9	49.4
	PCA	0.9	50.6	36.4	47.9	25.2	40.9
(110,110)	STD	(1.34e-2) 1.0	9.0	25.5	23.7	25.7	17.2
	BB	0.9	67.3	43.6	66.2	83.0	129.0
	PCA	0.9	146.4	57.0	80.5	105.9	93.0

are different from the ones constructed in [28], which are problem-dependent (they have similar efficiency). Also, the difficulty of choosing suitable weights is partially avoided (since we could use the same weights regardless of the problems at hand, the dimension, and the methods of path generation). We thus recommend using lattice points (A) in practice and present in the appendix (see Tables 7 and 8) some generators (Korobov and CBC) for $n = 1009$ and $n = 4001$ up to dimensions $s = 128$ (faster searching for generators for larger n and s is an important future work). Note that once the generators are found (or given), lattice points are much easier to generate than random points and other quasi-random numbers.

6. Conclusions. The classical constructions of lattice points are not as good as they ought to be, since they do not put special emphasis on lower-order projections. Good lattice rules can be nonuniversal or universal. Nonuniversal rules depend on the specific problem at hand and thus are of limited interest. Universal rules do not depend on a particular problem (but may depend on some feature of a wide range of problems) and thus have great importance. This paper focuses on universal good lattice rules.

Because of the common feature of low superposition dimension for high-dimensional integrands in finance, we proposed methods to construct good lattice points whose low-order projections are well-behaved. The new lattice points are shown to have better two-dimensional projections than Sobol' points and random points even in high dimensions. It turns out that a superior distribution property in order-2 projections of a lattice point set is sufficient and necessary to achieve high efficiency for many high-dimensional finance problems. Indeed, the new lattice points are significantly more efficient than MC and are more efficient than the Sobol' points in most cases for pricing path-dependent options and American options. A nice surprise revealed is the robustness property of the new lattice rules; i.e., they have a good projection property and are useful for a wide range of problems. In searching for robust lattices, the lattice points (A) offer a good compromise, since they are efficient for problems for which the order-1 and order-2 ANOVA terms play the dominant role. Such problems occur naturally in finance (especially when a dimension reduction technique is used); see [27].

We stress that the quality measure is very important for searching good lattice points in high dimensions. If the worst-case error of certain weighted Korobov space is used as a quality measure, then the choice of weights is crucial [28]. A bad choice of weights may lead to bad lattice points. Large weights on the higher-order terms may lead to lattice points with poor quality of order-2 projections which are inappropriate for functions from finance. If there is no information a priori about the integrand, then it is safer to use lattice points with good low-order projections as constructed in this paper, since they can improve the accuracy for favorable functions, while doing no harm for unfavorable ones. In high dimensions, the lattice points constructed based on the classical measure P_α are especially dangerous because of the bad order-2 (and possibly higher-order) projections due to the large weights. Note that the uniformity on some specific order of projections is measured by the elementary order-ℓ discrepancy, which allows us to assess the quality of the projections of some selected orders and is more informative than usual quality measure. The new measure is especially useful for comparing the goodness of different point sets.

The construction of point sets with good low- and moderate-order projections deserves further attention. For American options, though the new lattice points and the dimension reduction techniques improve the LSM method, the theoretical basis of such procedures for the LSM method is yet unknown and is worth studying. This would be helpful for better understanding the question of which QMC point set is better adopted to American options, and would be helpful for finding better techniques to improve the efficiency.

Appendix. Generators of Korobov and CBC with the weights (A).

TABLE 7

Shown are generators with the optimal order-2 projections (with the weights (A)*) up to dimension* $s = 128$ *for* $n = 1009$. *The optimal Korobov generator* $(1, a_s, \ldots, a_s^{s-1}) \pmod n$ *is determined separately for each dimension* s. *The CBC generator* $(z_1, z_2, \ldots, z_s)$ *is found sequentially, one component at a time.*

s	a_s	z_s	s	a_s	z_s	s	a_s	z_s
1	1	1	44	380	388	87	95	499
2	390	390	45	77	37	88	95	465
3	382	295	46	380	12	89	95	165
4	382	221	47	380	305	90	95	81
5	309	110	48	380	152	91	95	479
6	160	187	49	380	26	92	95	23
7	156	350	50	77	240	93	95	402
8	156	138	51	77	492	94	430	159
9	156	316	52	380	31	95	237	95
10	156	213	53	77	193	96	430	440
11	156	323	54	380	206	97	430	389
12	156	281	55	380	242	98	430	39
13	156	131	56	425	273	99	237	297
14	304	54	57	380	500	100	237	355
15	156	225	58	380	34	101	430	189
16	304	468	59	342	284	102	237	322
17	156	420	60	342	177	103	237	119
18	304	143	61	342	421	104	430	219
19	304	264	62	342	257	105	430	406
20	249	291	63	342	275	106	430	83
21	249	89	64	342	126	107	430	223
22	249	494	65	342	436	108	237	263
23	249	107	66	342	198	109	430	183
24	249	320	67	342	266	110	430	148
25	249	353	68	342	154	111	430	186
26	466	435	69	262	10	112	430	147
27	177	180	70	181	298	113	430	85
28	177	472	71	181	113	114	237	7
29	177	397	72	181	135	115	430	271
30	57	141	73	181	487	116	233	248
31	177	57	74	262	317	117	498	44
32	177	16	75	262	371	118	498	116
33	57	106	76	181	192	119	498	188
34	177	313	77	181	293	120	233	381
35	65	497	78	181	181	121	233	432
36	65	400	79	262	101	122	498	241
37	65	244	80	342	235	123	233	409
38	326	329	81	342	150	124	233	38
39	380	385	82	342	280	125	233	168
40	380	172	83	342	337	126	233	199
41	77	490	84	95	151	127	233	123
42	77	28	85	95	72	128	233	338
43	77	67	86	308	315			

TABLE 8
The same as Table 5, but for $n = 4001$.

s	a_s	z_s	s	a_s	z_s	s	a_s	z_s
1	1	1	44	1452	1254	87	1495	1418
2	1478	1478	45	1375	122	88	372	112
3	1527	1446	46	1375	808	89	372	1850
4	823	1031	47	1397	1481	90	372	348
5	1134	555	48	358	1870	91	372	108
6	1134	1180	49	358	703	92	372	159
7	1563	390	50	358	1420	93	372	645
8	547	1902	51	1397	874	94	372	1362
9	547	689	52	358	1287	95	372	1854
10	1419	1471	53	358	51	96	372	1548
11	547	974	54	1397	1106	97	372	1442
12	547	1754	55	358	1931	98	1495	1441
13	933	1725	56	1247	450	99	1495	1228
14	933	957	57	1247	734	100	372	1901
15	1175	1746	58	1585	494	101	372	1919
16	933	1186	59	1585	666	102	1495	1516
17	933	372	60	1639	1759	103	372	240
18	1175	1477	61	1306	760	104	372	595
19	1175	517	62	1639	745	105	372	1324
20	1175	1093	63	1639	1875	106	905	536
21	1175	1451	64	1639	1558	107	905	970
22	933	775	65	1639	898	108	905	1764
23	1175	981	66	1306	458	109	84	1715
24	774	1543	67	1306	1364	110	905	621
25	336	855	68	1306	146	111	84	1078
26	336	476	69	367	1320	112	84	208
27	774	1572	70	556	997	113	84	335
28	336	534	71	556	1367	114	905	1019
29	336	453	72	367	376	115	84	1239
30	774	1006	73	556	1124	116	84	313
31	933	1747	74	367	1123	117	372	184
32	933	1492	75	367	189	118	372	1270
33	933	1206	76	556	1744	119	372	66
34	1175	809	77	556	166	120	372	303
35	933	639	78	1495	665	121	1495	918
36	933	64	79	1495	1658	122	1495	1233
37	933	724	80	372	863	123	372	1778
38	933	1960	81	372	1066	124	372	1500
39	933	214	82	372	299	125	387	1997
40	933	936	83	372	1462	126	387	1131
41	1375	688	84	1495	1205	127	387	1221
42	1452	895	85	272	1453	128	387	1985
43	1375	1541	86	272	1645			

Acknowledgment. The author would like to thank the referees for their valuable comments.

REFERENCES

[1] P. ACWORTH, M. BROADIE, AND P. GLASSERMAN, *A comparison of some Monte Carlo and quasi-Monte Carlo techniques for option pricing*, in Monte Carlo and Quasi-Monte Carlo Methods 1996, H. Niederreiter, P. Hellekalek, G. Larcher, and P. Zinterhof, eds., Springer-Verlag, New York, 1998, pp. 1–18.

[2] M. BROADIE AND P. GLASSERMAN, *Pricing American-style security using simulation*, J. Econom. Dynam. Control, 21 (1997), pp. 1323–1352.

[3] R. E. Caflisch, W. Morokoff, and A. B. Owen, *Valuation of mortgage backed securities using Brownian bridges to reduce effective dimension*, J. Comp. Finance, 1 (1997), pp. 27–46.

[4] S. K. Chaudhary, *American options and the LSM algorithm: Quasi-random sequences and Brownian bridges*, J. Comput. Finance, 8 (2005), pp. 101–115.

[5] J. Dick, I. H. Sloan, X. Wang, and H. Woźniakowski, *Liberating the weights*, J. Complexity, 20 (2004), pp. 593–623.

[6] J. Dick, I. H. Sloan, X. Wang, and H. Woźniakowski, *Good lattice rules in weighted Korobov spaces with general weights*, Numer. Math., 103 (2006), pp. 63–97.

[7] P. L'Ecuyer and C. Lemieux, *Variance reduction via lattice rules*, Management Sci., 46 (2000), pp. 1214–1235.

[8] F. J. Hickernell, *Quadrature error bounds with applications to lattice rules*, SIAM J. Numer. Anal., 33 (1996), pp. 1995–2016.

[9] F. J. Hickernell, *Lattice rules: How well do they measure up?*, in Random and Quasi-Random Point Sets, P. Hellekalek and G. Larcher, eds., Springer-Verlag, New York, 1998, pp. 109–168.

[10] F. J. Hickernell and X. Wang, *The error bounds and tractability of quasi-Monte Carlo algorithms in infinite dimension*, Math. Comp., 71 (2002), pp. 1641–1661.

[11] F. J. Hickernell and H. Woźniakowski, *Integration and approximation in arbitrary dimension*, Adv. Comput. Math., 12 (2000), pp. 25–58.

[12] N. M. Korobov, *The approximate computation of multiple integrals*, Dokl. Akad. Nauk SSSR, 124 (1959), pp. 1207–1210 (in Russian).

[13] C. Lemieux, *Randomized quasi-Monte Carlo: A tool for improving the efficiency of simulations in finance*, in Proceedings of the 2004 Winter Simulation Conference, R. G. Ingall et al., eds., Washington, D.C., 2004, pp. 1565–1573.

[14] C. Lemieux and P. L'Ecuyer, *On selection criteria of lattice rules and other quasi-Monte Carlo point sets*, Math. Comput. Simulation, 55 (2001), pp. 139–148.

[15] F. A. Longstaff and E. S. Schwartz, *Valuing American options by simulation: A least-squares approach*, Rev. Financial Stud., 14 (2001), pp. 113–148.

[16] W. J. Morokoff and R. E. Caflisch, *Quasi-random sequences and their discrepancies*, SIAM J. Sci. Comput., 15 (1994), pp. 1251–1279.

[17] B. Moskowitz and R. E. Caflisch, *Smoothness and dimension reduction in quasi-Monte Carlo methods*, Math. Comput. Modelling, 23 (1996), pp. 37–54.

[18] H. Niederreiter, *Random Number Generation and Quasi-Monte Carlo Methods*, CBMS-NSF Regional Conf. Ser. in Appl. Math. 63, SIAM, Philadelphia, 1992.

[19] D. Nuyen and R. Cools, *Fast algorithms for component-by-component construction of rank-1 lattice rules in shift-invariant reproducing kernel Hilbert spaces*, Math. Comp., 75 (2006), pp. 903–920.

[20] I. H. Sloan and S. Joe, *Lattice Methods for Multiple Integration*, Oxford University Press, Oxford, UK, 1994.

[21] I. H. Sloan, F. Y. Kuo, and S. Joe, *Constructing randomly shifted lattice rules in weighted Sobolev spaces*, SIAM J. Numer. Anal., 40 (2002), pp. 1650–1665.

[22] I. H. Sloan and A. V. Reztsov, *Component-by-component construction of good lattice points*, Math. Comp., 71 (2002), pp. 263–273.

[23] I. H. Sloan and H. Woźniakowski, *When are quasi-Monte Carlo algorithms efficient for high dimensional integrals?*, J. Complexity, 14 (1998), pp. 1–33.

[24] I. H. Sloan and H. Woźniakowski, *Tractability of multivariate integration for weighted Korobov classes*, J. Complexity, 17 (2001), pp. 697–721.

[25] I. M. Sobol', *On the distribution of points in a cube and the approximate evaluation of integrals*, Zh. Vychisli. Mat. Mat. Fiz., 7 (1967), pp. 784–802.

[26] X. Wang and K. T. Fang, *The effective dimensions and quasi-Monte Carlo integration*, J. Complexity, 19 (2003), pp. 101–124.

[27] X. Wang and I. H. Sloan, *Why are high-dimensional finance problems often of low effective dimension?*, SIAM J. Sci. Comput., 27 (2005), pp. 159–183.

[28] X. Wang and I. H. Sloan, *Efficient weighted lattice rules with application to finance*, SIAM J. Sci. Comput., 28 (2006), pp. 728–750.

[29] X. Wang and I. H. Sloan, *Low discrepancy sequences in high dimensions: How well are their projections distributed?*, J. Comput. Appl. Math., to appear.

[30] X. Wang, I. H. Sloan, and J. Dick, *On Korobov lattice rules in weighted spaces*, SIAM J. Numer. Anal., 42 (2004), pp. 1760–1779.

SIAM J. SCI. COMPUT.
Vol. 29, No. 2, pp. 622–634
© 2007 Society for Industrial and Applied Mathematics

FAST NUMERICAL METHODS FOR BERNOULLI FREE BOUNDARY PROBLEMS*

CHRISTOPHER M. KUSTER†, PIERRE A. GREMAUD†, AND RACHID TOUZANI‡

Abstract. The numerical solution of the free boundary Bernoulli problem is addressed. An iterative method based on a level-set formulation and boundary element method is proposed. Issues related to the implementation, the accuracy, and the generality of the method are discussed. The efficiency of the approach is illustrated by numerical results.

Key words. Bernoulli, free boundary, level set, boundary elements

AMS subject classifications. 65N38, 35R35, 76B99

DOI. 10.1137/06065444X

1. Introduction.

Bernoulli free boundary problems find their origin in the description of free surfaces for ideal fluids [12]. There are, however, numerous other applications leading to similar formulations; see, for instance, [11]. For concreteness, we focus on the exterior Bernoulli problem. Let Ω be a bounded domain in $\mathbb{R}^2$. The exterior Bernoulli problem consists in seeking a bounded domain $A \supset \Omega$ and a function u defined on $\bar{A} \setminus \Omega$ such that

$$\Delta u = 0 \qquad \text{in } A \setminus \overline{\Omega}, \tag{1.1}$$

$$u = 1 \qquad \text{on } \partial\Omega, \tag{1.2}$$

$$u = 0 \qquad \text{on } \partial A, \tag{1.3}$$

$$\frac{\partial u}{\partial n} = \mu \qquad \text{on } \partial A, \tag{1.4}$$

where μ is given. In the previous example, one can think of u as a streamfunction and of Ω as an obstacle. Taking into account (1.3), condition (1.4) can be written as $|\nabla u| = |\mu|$ and corresponds, for fluid applications, to Bernoulli's principle; see, for instance, [9].

The above problem has been extensively studied; see [11] for general remarks. For a convex simply connected bounded domain Ω, it is known that for any negative constant $\mu < 0$, the above problem admits a unique classical solution. Further, the free boundary ∂A has regularity $\mathcal{C}^{2,\alpha}$; see [21, Theorem 1.1].[1] The convexity assumption is necessary for uniqueness, as counterexamples show (see [11, Example 13]). The study of the interior Bernoulli problem is more delicate, and not even convexity can ensure uniqueness.

*Received by the editors March 16, 2006; accepted for publication (in revised form) October 26, 2006; published electronically March 30, 2007.

http://www.siam.org/journals/sisc/29-2/65444.html

†Department of Mathematics and Center for Research in Scientific Computation, North Carolina State University, Raleigh, NC 27695-8205 (cmkuster@ncsu.edu, gremaud@ncsu.edu). The work of the first author was partially supported by the National Science Foundation (NSF) through grant DMS-0244488. The work of the second author was partially supported by the NSF through grants DMS-0204578, DMS-0244488, and DMS-0410561.

‡Laboratoire de Mathématiques, CNRS UMR 6620, Université Blaise Pascal (Clermont-Ferrand), 63177 Aubière Cedex, France (Rachid.Touzani@math.univ-bpclermont.fr).

[1]The result is, in fact, established in the more general case of the p-Laplacian in [21] and is generalized to nonconstant μ in [22].

There are roughly two ways of tackling such problems numerically. First, a variational formulation may be considered and the corresponding cost function minimized [19, 23, 28]; this requires the calculations of shape gradients. Second, a fixed point-type approach can be set up where a sequence of elliptic problems is solved in a sequence of converging domains, those domains being obtained through some updating rule at each iteration [6, 11, 24]. The method studied in this paper falls in the latter category.

Two simple generic numerical tools are combined. First, the interface is represented through a level-set formulation as described in section 2. Second, the elliptic part of the problem is solved through the use of a boundary element method; see section 3 (see [14] for another example of a method involving those two tools in a different setting). The proposed method is thus conceptually simpler than the shape optimization approach. Since it requires, in principle, only the calculations of quantities being defined on the interface or close to it, it is potentially faster than the finite difference approaches of, for instance, [6, 11, 24] or more generally of immersed interface-type methods [27]. The feasibility of the algorithm and its complexity are investigated in section 4.

A general overview of the strategy is as follows: the potential problem is solved with one of the conditions on the free boundary omitted, and then the omitted condition is used to update the location of the free boundary. More specifically, given an initial domain $A_0 \supset \bar{\Omega}$, the simplest variant of this type consists in solving the sequence of problems

$$\Delta u_k = 0 \qquad \text{in } A_k \setminus \overline{\Omega}, \qquad k = 0, 1, 2, \ldots, \tag{1.5}$$

$$u_k = 1 \qquad \text{on } \partial\Omega, \tag{1.6}$$

$$\partial_n u_k = \mu \qquad \text{on } \partial A_k. \tag{1.7}$$

For a given domain A_k, it is well known that problem (1.5)–(1.7) admits a unique solution; see, for instance, [15, Theorem 5.1]. The new domain A_{k+1} is found by moving ∂A_k in its normal direction so that u_k vanishes there. Let $P_k \in \partial A_k$; to first order, we have

$$u_k(P_{k+1}) \approx u_k(P_k) + \mu\, d_k,$$

where $P_{k+1} = P_k + n_k d_k$, n_k being the outer unit normal to ∂A_k at P_k. The new point P_{k+1}, or similarly the distance d_k, is determined by the requirement $u_k(P_{k+1}) = 0$, i.e., $d_k = \frac{-u_k(P_k)}{\mu}$. The free boundary is thus updated according to[2]

$$\partial A_{k+1} = \partial A_k - \frac{u_k}{\mu}\, n_k. \tag{1.8}$$

2. Level-set representation of the interface. In the above iterative process, the interface ∂A_k has to be updated as long as the residual $u_k|_{\partial A_k}$ is not (numerically) zero. Relation (1.8) indicates that the boundary should be moved in the normal direction by an amount proportional to the residual.

[2]The present iterative process can be modified by imposing the Neumann condition (1.7) on the "next" boundary ∂A_{k+1} instead of ∂A_k [11, 13]. However, in the formulation adopted here, those modifications are of little use, as they require information such as curvature which complicates the calculation of the "evolution" of the interface.

If the residual is considered as a normal speed, a time-dependent problem can be set up for the "evolution" (or correction) of the domain A_k. We denote the domain so generated $A_k(t)$ with $A_k(0) = A_k$. Let $F : \mathbb{R}^2 \to \mathbb{R}$ be an extension of the residual away from A_k (see subsection 2.3 below). We want the interface $\Gamma(t) = \partial A_k(t)$ to be characterized by

$$\Gamma(0) = \partial A_k, \qquad \Gamma(t) = \{(x(t), t); x(0) = x_0, x_0 \in \partial A_k\} \quad \text{for } t > 0,$$

where

$$\frac{dx}{dt} = Fn, \qquad x(0) = x_0,$$

n being the unit outer normal to $\partial A_k(t)$. A level-set approach, as pioneered in [30] (see also [29, 34]), consists in representing the interface $\{\Gamma(t)\}_{t\geq 0}$ as the zero level set of family of level-set functions $\{\phi(\cdot, t)\}_{t\geq 0}$ with the property

$$A_k(t) = \{x \in \mathbb{R}^2; \phi(x, t) < 0\}, \qquad \Gamma(t) = \partial A_k(t) = \{x \in \mathbb{R}^2; \phi(x, t) = 0\},$$

where $A_k(t)$ denotes the domain stemming from the evolution of A_k through the above process. By taking the time derivative of the relation $\phi(x(t), t) = 0$, the level-set equation is obtained:

$$\partial_t \phi + F|\nabla \phi| = 0, \tag{2.1}$$

$$\phi(\cdot, 0) = \phi_0, \tag{2.2}$$

where ϕ_0 is a level-set function corresponding to ∂A_k.

Several points related to the implementation of the above method now have to be considered. The fixed boundary $\partial\Omega$ is approximated by a piecewise linear curve $\partial\Omega_h$ with N elements. The size of the smallest element of $\partial\Omega_h$ is denoted Δx. Let $B \subset \mathbb{R}^2$ be a square domain of size $M\Delta x \times M\Delta x$, where M is chosen large enough so that B contains A. We associate with B, in a natural way, a uniform Cartesian mesh B_h of size Δx. In the following, the level-set functions are characterized by their nodal values on the mesh B_h. Two kinds of interpolation operators are considered on B_h. In the contouring step (subsection 2.1), a classical $\mathcal{P}_1$ interpolation is used. In the projection step (subsection 2.2), a classical local $\mathcal{Q}_2$ interpolation is considered.

2.1. Contouring. In many applications of the level-set method, the actual reconstruction of the interfaces is not needed. This is not the case here, as the elliptic problem (1.5)–(1.7) has to be solved in a family of successive domains defined by those interfaces. Let ϕ be a given level-set function and $I_{\mathcal{P}}\phi$ its $\mathcal{P}_1$-interpolant on B_h. By contouring, we mean the operation that associates with the nodal values of ϕ the zero level set of $I_{\mathcal{P}}\phi$. More precisely, each square cell of the underlying Cartesian grid is divided into two triangular elements,[3] and on each of those triangles, the unique polynomial of degree 1 agreeing with the values of the level-set function at the vertices is considered. Contour lines are then constructed by connecting the zeros of $I_{\mathcal{P}}\phi$ on the grid lines. This process is facilitated by two facts: first, each triangle contains at most one contour line passing through it, and, second, the gradient of $I_{\mathcal{P}}\phi$ is piecewise constant on the grid elements, yielding an outward normal to $A_k(t)$. Note that both properties fail in general for the local $\mathcal{Q}_2$ interpolant considered in the projection step below.

If $I_{\mathcal{P}}\phi$ is uniformly equal to zero on a given triangle, then the problem is underresolved (Δx is too large) and the algorithm fails.

[3] Any such decomposition is acceptable.

2.2. Projection. As mentioned above, some quantities such as the normal speed need to be extended away from the interface. The first step in this process consists again in an accurate reconstruction of the boundary. A loop through the mesh is done to determine which nodes are "close to" the interface. Here a node is close to the interface if it has a primary neighbor where ϕ has opposite sign.

With each node x_i, we associate its eight closest neighbors and construct on this set of nine nodes the unique polynomial of degree 2 in each variable agreeing with the values of ϕ there. This local $\mathcal{Q}_2$ interpolant is denoted $I_{\mathcal{Q},x_i}\phi$. For each node x_i close to the interface, the closest point $x^\star$ to x_i on the zero level set of $I_{\mathcal{Q},x_i}\phi$ is computed. The square of the Cartesian distance, i.e., $|x - x_i|^2$, is minimized subject to the constraint $I_{\mathcal{Q},x_i}\phi = 0$. The Lagrangian for this problem is

$$\mathcal{L}(x,\lambda) = |x - x_i|^2 + \lambda\, I_{\mathcal{Q},x_i}\phi(x). \tag{2.3}$$

To find the point $x^\star$, the system

$$\nabla_{x,\lambda}\mathcal{L}(x,\lambda) = 0 \tag{2.4}$$

is solved by Newton's method with Armijo line search [25]. This projection method was introduced and discussed in [17]. It roughly corresponds to a simpler version of the method proposed in [8] which is based on cubic splines; see [26] for more comments and comparisons. The above algorithm can fail if the interface is underresolved.

The projection step could potentially be used to reconstruct the interface, i.e., as another contouring step. However, while it is locally more accurate than the above contouring algorithm (and is thus ideal when used in conjunction with the extension step described below), it has several disadvantages as a contouring tool (local character, nonconstant elementwise gradient).

2.3. Extension. This step extends the speed F away from the interface, so that (2.1), (2.2) can be solved. Let Γ be the interface obtained from the contouring step. By construction, Γ is a closed (for the problems considered here), possibly multiconnected, piecewise linear (on the triangular mesh derived from B_h) curve in B. The boundary nodes corresponding to Γ are denoted $\{\xi_j\}$; i.e., the ξ_j's are the end points of the line segments that form Γ. Further, through the elliptic step (see section 3), the value of F at the center point of each linear segment of Γ is known; the center point nodes are denoted $\{\bar{\xi}_j\}$.

First, the values of F at the midpoint boundary nodes $\{\bar{\xi}_j\}$ are extended to the set of Cartesian nodes $\{x_i\}$ that form the vertices of the triangles containing the nodes $\{\bar{\xi}_j\}$.

More precisely, consider the node $\bar{\xi}_j$ in Figure 1, left. This local extension step is based on the analysis of the "domain of influence" of the nodes $\{\bar{\xi}_j\}$. This domain of influence is taken here as the set of all the points in B whose orthogonal projection on the line containing the segment through $\bar{\xi}_j$ belongs to that segment. If a node x_i belongs to the domain of influence of $\bar{\xi}_j$, then F is extended at x_i by the value $F(\bar{\xi}_j)$ as in Figure 1, left. If instead the node x_i is in the domain of influence of more than one midpoint boundary node, as would be the case in the dark grey area in Figure 1, left, then the value at the midpoint of the closest segment to x_i is retained. Finally, if x_i does not belong to the domain of influence of any midpoint boundary node, as in the light grey area in Figure 1, right, then the value of F at x_i is taken as

$$F(x_j) = \frac{\beta}{\alpha+\beta}F_{j-1} + \frac{\alpha}{\alpha+\beta}F_j,$$

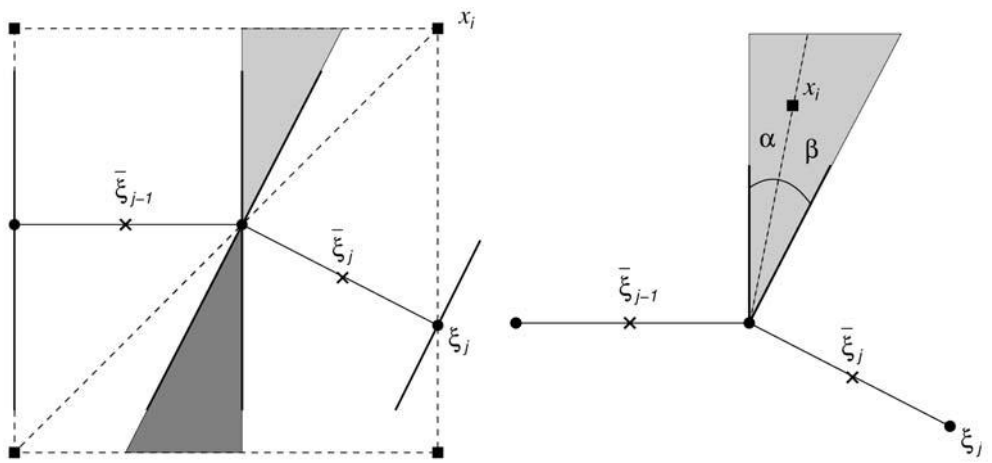


FIG. 1. *Left: general view of the geometry involved in the local extension step and corresponding "domains of influence" (dark grey area: shock-like domain, light grey area: rarefaction-like domain); right: definition of the local angles α and β.*

where F_{j-1} and F_j are the values of F at the midpoint boundary nodes $\bar{\xi}_{j-1}$ and $\bar{\xi}_j$, respectively, and where the angles α and β are defined as in Figure 1, right. This way of defining the local extension of F is compatible with the global extension (2.7).

Second, a renormalized function $\tilde{\phi}$ is initialized as a signed distance function at the same Cartesian nodes at which F has just been extended. The projection step is used to do this.

We emphasize that both of those local extension steps for F and $\tilde{\phi}$ take place only on the nodes adjacent to the interface; the corresponding values are then used as starting points for the extension to the rest of the Cartesian nodes. This is accomplished using the fast marching method [2] to solve

$$|\nabla\tilde{\phi}| = 1 \qquad \text{in } B, \tag{2.5}$$

$$\tilde{\phi} = 0 \qquad \text{on } \Gamma, \tag{2.6}$$

$$\nabla\tilde{F}\cdot\nabla\tilde{\phi} = 0 \qquad \text{in } B, \tag{2.7}$$

$$\tilde{F} = F \qquad \text{on } \Gamma. \tag{2.8}$$

A fully upwind mixed first/second order discretization of the above equations is applied on the mesh B_h; see [18, 35, 36] for more details.

2.4. Updating the interface. The interface is moved by updating the corresponding level-set function through (2.1), (2.2). More precisely, after the extension step, the level-set function $\tilde{\phi}$ corresponding to the current interface Γ is a signed distance function, and, in particular, $|\nabla\tilde{\phi}| = 1$. Therefore, (2.1), (2.2) here reads

$$\partial_t\phi + \tilde{F} = 0,$$

$$\phi(\cdot,0) = \tilde{\phi}.$$

The update is then trivially computed by taking one forward Euler step,

$$\phi_{\text{new}} = \tilde{\phi} - \Delta t\,\tilde{F}, \tag{2.9}$$

where $\Delta t = -\frac{1}{2\mu}$. Note that this corresponds to half the optimal time step given by (1.8); taking the "optimal" value from (1.8) may lead to overshoots in the position of the interface and may, in fact, result in slowing down the convergence of the global iterative process.

2.5. Initial interface. An initial guess of the interface's position, ∂A_0, needs to be provided. This can be done in an ad hoc way. In section 4, ∂A_0 is taken as a curve of constant distance to Ω.

3. Boundary element method. Consider again the problem (1.5)–(1.7). For the sake of simplicity, the subscript k is dropped in this section. We assume both ∂A and $\partial\Omega$ to be simple closed curves and let $\Gamma = \partial A \cup \partial\Omega$. The region of interest $A \setminus \bar{\Omega}$ being interior to ∂A and exterior to $\partial\Omega$, ∂A is oriented counterclockwise, while $\partial\Omega$ is clockwise.

Multiplying (1.5) by the fundamental solution $G(x,y) := -\frac{1}{2\pi}\log|x-y|$ and integrating twice by parts leads to

$$u(x) = \int_\Gamma G(x,y)\,\frac{\partial u}{\partial n_y}(y)\,ds(y) - \int_\Gamma \frac{\partial G}{\partial n_y}(x,y)\,u(y)\,ds(y), \tag{3.1}$$

where n is the unit outer normal to $A \setminus \bar{\Omega}$ at y. The above integral representation is valid for $x \in A \setminus \bar{\Omega}$. To treat the case $x \in \Gamma$, we define the linear operator $\mathcal{L} : L^2(\Gamma) \to L^2(\Gamma)$ by

$$\mathcal{L}v(x) = \begin{cases} \frac{v(x)}{2} + \displaystyle\int_{\partial A} \frac{\partial G}{\partial n_y}(x,y)\,v(y)\,ds(y) - \int_{\partial\Omega} G(x,y)\,v(y)\,ds(y) & \text{for } x \in \partial A, \\ \displaystyle\int_{\partial A} \frac{\partial G}{\partial n_y}(x,y)\,v(y)\,ds(y) - \int_{\partial\Omega} G(x,y)\,v(y)\,ds(y) & \text{for } x \in \partial\Omega \end{cases}$$

and the function $\mathcal{F} \in L^2(\Gamma)$ by

$$\mathcal{F}(x) = \begin{cases} \mu \displaystyle\int_{\partial A} G(x,y)\,ds(y) - \int_{\partial\Omega} \frac{\partial G}{\partial n_y}(x,y)\,ds(y) & \text{for } x \in \partial A, \\ \mu \displaystyle\int_{\partial A} G(x,y)\,ds(y) - \int_{\partial\Omega} \frac{\partial G}{\partial n_y}(x,y)\,ds(y) - \frac{1}{2} & \text{for } x \in \partial\Omega. \end{cases}$$

Taking into account the boundary conditions (1.6) and (1.7), it is then standard to check that if

$$w(x) = \begin{cases} u(x) & \text{for } x \in \partial A, \\ \frac{\partial u}{\partial n}(x) & \text{for } x \in \partial\Omega, \end{cases}$$

where u is the solution to (1.5)–(1.7), then

$$\mathcal{L}w(x) = \mathcal{F}(x) \qquad \forall x \in \Gamma. \tag{3.2}$$

Problem (3.2) is discretized as follows. The interface $\partial A = \partial A_h$ is obtained through contouring of a given level-set function, see section 2, and piecewise constant elements are considered. The function w solution to (3.2) is approximated by w_h such that

$$w_h(x) = w_e \qquad \forall x \in e,$$

where e is an edge of either ∂A_h or $\partial\Omega_h$ and where w_e is the constant value of w_h on that edge. Equation (3.2) is then collocated at the midpoints of the edges. In other words, for a generic piecewise constant function v_h, we define

$$\mathcal{L}_h v_h(\bar{\xi}_e) = \begin{cases} \frac{v_h(\bar{\xi}_e)}{2} + \displaystyle\int_{\partial A_h} \frac{\partial G}{\partial n_y}(\bar{\xi}_e, y)\, v_h(y)\, ds(y) - \int_{\partial\Omega_h} G(\bar{\xi}_e, y)\, v_h(y)\, ds(y) \\ \qquad \text{for } \bar{\xi}_e \in \partial A_h, \\ \displaystyle\int_{\partial A_h} \frac{\partial G}{\partial n_y}(\bar{\xi}_e, y)\, v_h(y)\, ds(y) - \int_{\partial\Omega_h} G(\bar{\xi}_e, y)\, v_h(y)\, ds(y) \\ \qquad \text{for } \bar{\xi}_e \in \partial\Omega_h, \end{cases}$$

where $\bar{\xi}_e$ is the midpoint of the edge e. Similarly, we also have

$$\mathcal{F}_h(\bar{\xi}_e) = \begin{cases} \mu \displaystyle\int_{\partial A_h} G(\bar{\xi}_e, y)\, ds(y) - \int_{\partial\Omega_h} \frac{\partial G}{\partial n_y}(\bar{\xi}_e, y)\, ds(y) & \text{for } \bar{\xi}_e \in \partial A_h, \\ \mu \displaystyle\int_{\partial A_h} G(\bar{\xi}_e, y)\, ds(y) - \int_{\partial\Omega_h} \frac{\partial G}{\partial n_y}(\bar{\xi}_e, y)\, ds(y) - \frac{1}{2} & \text{for } \bar{\xi}_e \in \partial\Omega_h. \end{cases}$$

The approximate solution w_h is the solution to

$$\mathcal{L}_h w_h(\bar{\xi}_e) = \mathcal{F}_h(\bar{\xi}_e) \qquad \forall \bar{\xi}_e \in \partial A_h \cup \partial\Omega_h. \tag{3.3}$$

The above integrals are computed exactly in the present implementation, in the sense of the Cauchy principal value where appropriate; see [4] for more details.

Both the integral equation (3.2) and the linear problem (3.3) are well conditioned with one exception: 0 is an eigenvalue of $\mathcal{L}$ if and only if the transfinite diameter of A is 1 [20]. This can be avoided by appropriate rescaling if necessary. It can be verified that $\mathcal{L}_h$ admits eigenvalues and singular values that are bounded independent of the mesh; see, e.g., [20] or [31] for explicit expressions of the eigenvalues in some specific cases. Further, in spite of the fact that the elements of ∂A_h are allowed to be arbitrarily small, the condition number of $\mathcal{L}_h$ has been numerically verified to be of order N, which would correspond to the uniform mesh case [3]. The condition number of the matrices corresponding to the numerical tests of section 4 are on the order of 100. The resulting linear system is solved by GMRES [25, 33], which is consequently expected to perform well here even without preconditioning; see also [7] for a general convergence result for GMRES applied to integral equations. GMRES is restarted after 20 steps (i.e., the solver is GMRES(20)) and is stopped on small relative residuals; more precisely, the stopping criterion is

$$\|\mathcal{F}_h - \mathcal{L}_h w\|_2 \leq 10^{-10} \|\mathcal{F}_h\|_2,$$

where w denotes the current iterate. With the above parameters, GMRES has been observed to perform slightly better than other CG-like methods such as QMR and Bi-CGSTAB [25] on the test problems of section 4.

4. Numerical results.

4.1. Algorithm. To solve the external Bernoulli problem, we use the following algorithm:

Input: a discretization of the boundary $\partial\Omega$, $\partial\Omega_h$, μ
Create the underlying Cartesian grid B_h
Create a level set function ϕ on B_h corresponding to ∂A_0
$k = 0$
$R_{-1} = 10^{10}$ (initial residual)
loop
  Contour ϕ (subsection 2.1) to find ∂A_k
  Solve (3.3) to get u
  $R_k = \max$ of $|u|$ on ∂A_k
  if $(R_{k-1} - R_k)/R_k < 10^{-3}$ (small residual decrease) **then**
    STOP
  end if
  Set $F = u$ on ∂A_k
  Extend ϕ and F (subsections 2.2 and 2.3)
  Move boundary (subsection 2.4)
  $k = k + 1$
end loop

Several remarks are in order.

- Profiling reveals that for reasonable meshes the relative costs of contouring and interface updating are very low (less than 1%), while most of the time is spent constructing and solving the linear system (3.3) (about 90%) and extending the solution (about 10%). Methods such as fast multipoles [5, 32] can be used to bring down the asymptotic complexity of the linear solver from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$, where N is the number of elements of $\partial\Omega_h$. However, this results in increased efficiency only for very large values of N; see [26, Appendix] for more details; see also [16] for a large-scale two-dimensional example. Fast summation methods should be considered for three-dimensional problems.
- Progressive mesh refinement can be considered; i.e., a coarse mesh solution can be used as the starting point. A strategy of this type is, for instance, used in [23] for a similar type of problem but for a different numerical approach.
- In solving (3.3), the "missing" condition (1.3) can be used when choosing the initial iterate for GMRES. This results in faster convergence (fewer GMRES iterates) as the algorithm progresses.
- The extension step through fast marching (subsection 2.3) is done in the whole computational domain B. The corresponding complexity is $\mathcal{O}(M^2 \log M)$, where M^2 is the total number of nodes in the Cartesian grid B_h. A narrow band/computational tube implementation [2, 37] could be considered to speed up the algorithm. However, the global complexity of the problem would not change: if the width of the band is a constant multiple of Δx, say $n\Delta x$, then by (2.9), Δt should be reduced from $\frac{-1}{2\mu}$ to a value less than $\frac{n\Delta x}{|\tilde{F}|}$ since the band has to contain the boundary.
- Higher order boundary element methods can be used [4]. While second order convergence is observed in section 4, this is a result of the solution being constant on the outer free boundary. Piecewise linear elements should be used to ensure second order convergence for more general cases. To the authors' knowledge, the present work is one of very few published results regarding the accuracy of a combined level-set boundary element method; see, for instance, [14].

TABLE 1
Convergence and complexity rates for Example 1 (radial case); N is number of elements on $\partial\Omega_h$, L^∞ error refers to the Hausdorff distance between exact and computed boundaries, k is the number of nonlinear iterations (see section 4.1), Time is the runtime in seconds.

N	$\max\|u\|$ on Γ	Rate	L^∞ error	Rate	k	Time	Rate
25	1.64(−2)	–	1.53(−2)	–	6	0.71	–
50	5.07(−3)	1.7	4.40(−3)	1.8	8	4.1	2.5
100	1.30(−3)	2.0	1.12(−3)	2.0	10	19	2.2
200	3.34(−4)	2.0	2.85(−4)	2.0	11	76	2.0
400	8.18(−5)	2.0	7.10(−5)	2.0	12	372	2.1
800	1.92(−5)	2.1	1.74(−5)	2.0	14	1465	2.0

- The use of second order fast marching is not crucial for the accuracy of the method which is dictated by both the contouring and boundary element methods. It was observed that first order fast marching may lead to more iterations (domain updates), and, hence, due to the very little difference in cost per iteration, the use of second order fast marching is advocated.
- The numerical solutions are observed to be remarkably independent of the initial domain ∂A_0 (for problems with unique solutions [11]). More iterations can be expected in case of poor initial guesses (for instance if the smoothness properties of ∂A_0 are vastly different from those of ∂A).

4.2. Example 1. Following [11], a quick look at the radial case is instructive. Let Ω be the unit ball. We consider the problem (1.1)–(1.4) with Ω as above and $\mu = -2$. The solution to (1.1)–(1.3) with A being the ball of radius R centered at the origin is

$$u(r) = -2\,R\log r + 1,$$

expressed in polar coordinates. An iterative process similar to the one above can then be considered. Taking (1.8) into account, the kth step of the algorithm reads

$$R_{k+1} = R_k - R_k \log R_k + \frac{1}{2}, \qquad k = 1, 2, \ldots.$$

Therefore, in the fully radial case, the problem amounts to finding a fixed point to the function $f(R) = R - R\log R + \frac{1}{2}$. The function f has a unique fixed point $\bar{R}$, where

$$\bar{R} = \frac{1}{2\,W(\frac{1}{2})},$$

the function W being the Lambert W function[4] [10].

Table 1 shows second order convergence in the L^∞ norm. Full second order convergence is obtained even though the elliptic solver is based on a piecewise constant discretization. This is due to the fact that the exact solution is constant on both the inner and outer boundaries. Further, when measured with respect to runtime, the complexity is also second order.

The convergence history is instructive. Figure 2, left, displays the error (maximum of $|u|$ on the free boundary) through the iterations. The behavior of the first iterates is governed by the geometry, see (2.9), and is only weakly dependent on the mesh size

[4]The Lambert W function is the inverse of $W \mapsto z = W\,e^W$.

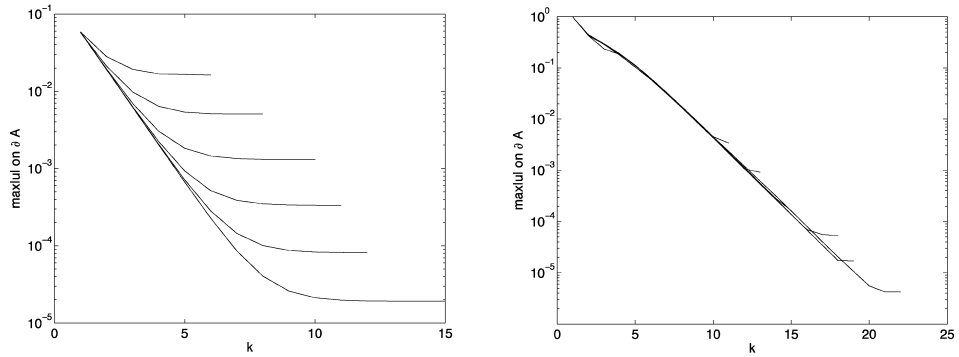


FIG. 2. *Convergence history of the iterations: maximum value of* $|u|$ *on the current interface as a function of the iteration number; left: Example* 1*, right: Example* 2*. The convergence curves are in the obvious order: from top to bottom,* $N = 25, 50, 100, 200, 400, 800$.

Δx. The later iterations during which the fine structure of the boundary is determined do depend on Δx. This explains the mesh dependency of the number of iterations observed in Table 1.

4.3. Example 2. We consider here the problem (1.1)–(1.4) with Ω consisting of two disks of radius 1, one centered at $(-2, 2)$ and one at $(2, -2)$; further, $\mu = -1/4$. The initial boundary is taken as two circles of radius 1.1, one around each of the inner disks. Note that for this choice of μ, the exact boundary is simply connected. A couple of iterates are displayed in Figure 3. One can note that after the first step the correct topology of the interface has already been achieved.

No exact solution is available for the present example. In Table 2, the maximum of u on the boundary is reported. By construction, this maximum should vanish for the converged solution. The complexity, as measured from the runtimes, is also reported. In both cases, the rates are about two.

Convergence history is displayed in Figure 2, right; a behavior similar as that of Example 1 is observed.

5. Conclusion. Solutions of the Bernoulli free boundary problem can be efficiently computed by the method presented here. Providing a Green's function is available, the method can be used to solve other free boundary problems. For instance, it can be applied with only minor modifications to the Prandtl–Batchelor problem (see [1] and the references therein), which consists in looking for a domain A which is now interior to the fixed domain Ω such that for a given function σ,

$$|\nabla u_A|^2 - |\nabla u_\Omega|^2 = \sigma \qquad \text{on } \partial A,$$

where u_A and u_Ω solve

$$\begin{cases} \Delta u_A = -\omega & \text{in } A, \\ u_A = 0 & \text{on } \partial A, \end{cases} \qquad \begin{cases} \Delta u_\Omega = 0 & \text{in } \Omega, \\ u_\Omega = 0 & \text{on } \partial A, \\ u_\Omega = \mu & \text{on } \partial\Omega, \end{cases}$$

μ and ω being positive constants.[5]

[5]To avoid the calculation of a bulk due to the inhomogeneity, a particular solution to $\Delta u_0 = -\omega$ (for instance $u_0 = -\frac{\omega}{4}(x^2 + y^2)$) can easily be subtracted from u_A. This restores homogeneity but renders the boundary condition nonconstant.

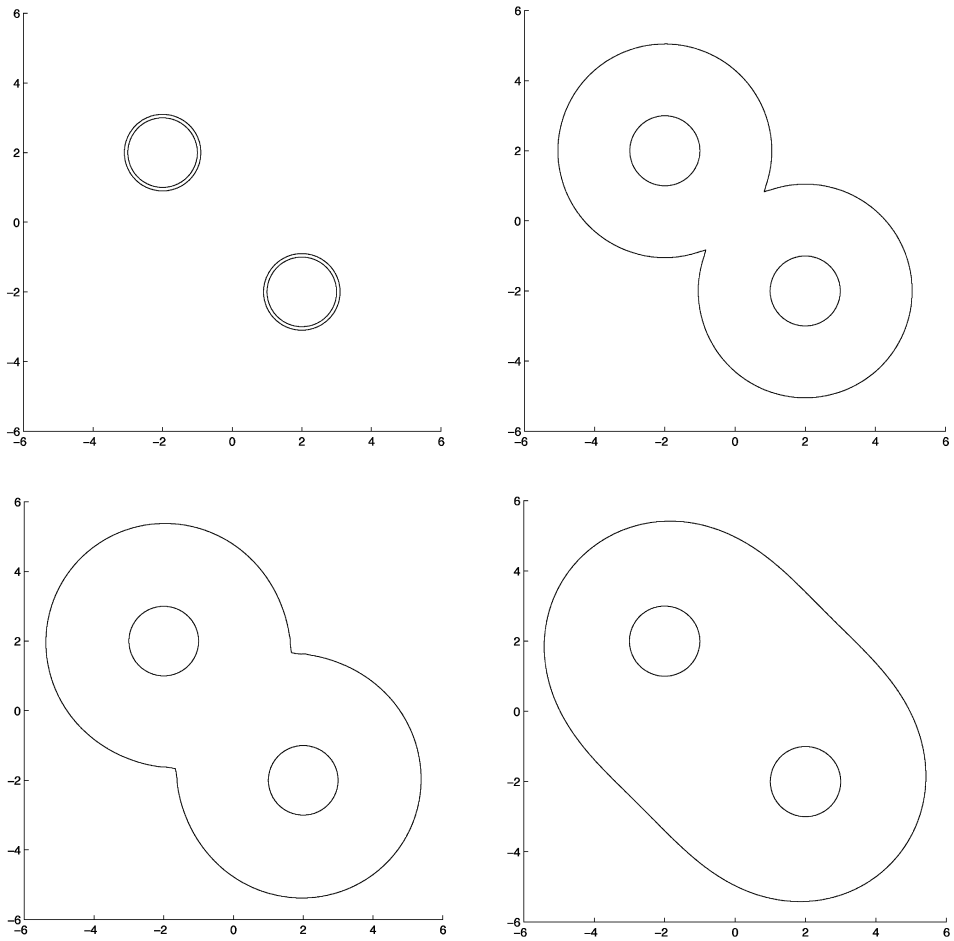


FIG. 3. *Evolution of the interface for Example* 2 *at the initial step and after steps* 1, 2, *and* 15 *(*$N = 100$*).*

TABLE 2

Convergence and complexity rates for Example 2*;* N *is number of elements on* $\partial\Omega_h$*,* k *is the number of nonlinear iterations (see section* 4.1*), Time is the runtime in seconds.*

N	$\max \lvert u \rvert$ on Γ	Rate	k	Time	Rate
26	3.43(−3)	–	11	4.4	–
50	9.16(−4)	2.0	13	19	2.2
100	1.64(−4)	2.5	15	85	2.2
200	5.31(−5)	1.6	18	573	2.8
400	1.71(−5)	1.6	19	1678	1.6
800	4.25(−6)	2.0	21	7613	2.2

REFERENCES

[1] A. ACKER, *On the existence of convex solutions to a generalized Prandtl-Batchelor free boundary problem*-II, Z. Angew. Math. Phys., 53 (2002), pp. 438–485.

[2] D. ADALSTEINSSON AND J. A. SETHIAN, *The fast construction of extension velocities in level set methods*, J. Comput. Phys., 148 (1999), pp. 2–22.

[3] M. Ainsworth, W. McLean, and T. Tran, *The conditioning of boundary element equations on locally refined meshes and preconditioning by diagonal scaling*, SIAM J. Numer. Anal., 36 (1999), pp. 1901–1932.

[4] K. E. Atkinson, *The Numerical Solutions of Integral Equations of the Second Kind*, Cambridge Monogr. Appl. Comput. Math. 4, Cambridge University Press, Cambridge, UK, 1997.

[5] R. Beatson and L. Greengard, *A short course on fast multipole methods*, in Wavelets, Multilevel Methods and Elliptic PDEs (Leicester, 1996), Numer. Math. Sci. Comput., Oxford University Press, New York, 1997, pp. 1–37.

[6] F. Bouchon, S. Clain, and R. Touzani, *Numerical solution of the free boundary Bernoulli problem using a level set formulation*, Comput. Methods Appl. Mech. Engrg., 194 (2005), pp. 3934–3948.

[7] S. L. Campbell, I. C. F. Ipsen, C. T. Kelley, C. D. Meyer, and Z. Q. Xue, *Convergence estimates for solution of integral equations with GMRES*, J. Integral Equations Appl., 8 (1996), pp. 19–34.

[8] D. L. Chopp, *Some improvements of the fast marching method*, SIAM J. Sci. Comput., 23 (2001), pp. 230–244.

[9] A. J. Chorin and J. E. Marsden, *A Mathematical Introduction to Fluid Mechanics*, Springer, New York, 1992.

[10] R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth, *On the Lambert W function*, Adv. Comput. Math., 5 (1996), pp. 329–359.

[11] M. Flucher and M. Rumpf, *Bernoulli's free-boundary problem, qualitative theory and numerical approximation*, J. Reine Angew. Math., 486 (1997), pp. 165–204.

[12] K. O. Friedrichs, *Über ein Minimumproblem für Potentialströmungen mit freiem Rand*, Math. Ann., 109 (1934), pp. 60–82.

[13] P. R. Garabedian, *Calculation of axially symmetric cavities and jets*, Pacific J. Math., 6 (1956), pp. 611–684.

[14] M. Garzon, D. Adalsteinsson, L. Gray, and J. A. Sethian, *A coupled level set-boundary integral method for moving boundary simulations*, Interfaces Free Bound., 7 (2005), pp. 277–302.

[15] R. González and R. Kress, *On the treatment of a Dirichlet–Neumann mixed boundary value problem for harmonic functions by an integral equation method*, SIAM J. Math. Anal., 8 (1977), pp. 504–517.

[16] A. Greenbaum, L. Greengard, and G. B. McFadden, *Laplace's equation and the Dirichlet-Neumann map in multiply connected domains*, J. Comput. Phys., 105 (1993), pp. 267–278.

[17] P. A. Gremaud, C. M. Kuster, and Z. Li, *A study of numerical methods for the level set approach*, Appl. Numer. Math., to appear.

[18] P. A. Gremaud and C. M. Kuster, *Computational study of fast methods for the eikonal equation*, SIAM J. Sci. Comput., 27 (2006), pp. 1803–1816.

[19] J. Haslinger, T. Kozubek, K. Kunisch, and G. Peichl, *Shape optimization and fictitious domain approach for solving free-boundary problems of Bernoulli type*, Comput. Optim. Appl., 26 (2003), pp. 231–251.

[20] J. Hayes and R. Kellner, *The eigenvalue problem for a pair of coupled integral equations arising in the numerical solution of Laplace's equation*, SIAM J. Appl. Math., 22 (1972), pp. 503–513.

[21] A. Henrot and H. Shahgholian, *Existence of classical solutions to a free boundary problem for the p-Laplace operator.* I. *The exterior convex case*, J. Reine Angew. Math., 521 (2000), pp. 85–97.

[22] A. Henrot and H. Shahgholian, *The one phase free boundary problem for the p-Laplacian with non-constant Bernoulli boundary condition*, Trans. Amer. Math. Soc., 354 (2002), pp. 2399–2416.

[23] K. Ito, K. Kunisch, and G. H. Peichl, *Variational approach to shape derivatives for a class of Bernoulli problems*, J. Math. Anal. Appl., 314 (2006), pp. 126–149.

[24] K. T. Kärkkäinen and T. Tiihonen, *Free surfaces: Shape sensitivity analysis and numerical methods*, Internat. J. Numer. Methods Engrg., 44 (1999), pp. 1079–1098.

[25] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Problems*, Frontiers Appl. Math. 16, SIAM, Philadelphia, 1995.

[26] C. M. Kuster, *Fast Numerical Methods for Evolving Interfaces*, Ph.D. thesis, Department of Mathematics, North Carolina State University, Raleigh, NC, 2006; http://www.lib.ncsu.edu/theses/available/etd-04262006-083221/.

[27] Z. Li and K. Ito, *The Immersed Interface Method: Numerical Solutions of PDEs Involving Interfaces and Irregular Domains*, Frontiers Appl. Math. 33, SIAM, Philadelphia, 2006.

[28] G. Mejak, *Numerical solution of Bernoulli-type free boundary value problems by variable domain method*, Internat. J. Numer. Methods Engrg., 37 (1994), pp. 4219–4245.

[29] S. J. Osher and R. P. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*, Appl. Math. Sci. 153, Springer, New York, 2002.
[30] S. Osher and J. Sethian, *Fronts propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations*, J. Comput. Phys., 56 (1988), pp. 12–49.
[31] G. J. Rodin and O. Steinbach, *Boundary element preconditioners for problems defined on slender domains*, SIAM J. Sci. Comput., 24 (2003), pp. 1450–1464.
[32] V. Rokhlin, *Rapid solution of integral equations of classical potential theory*, J. Comput. Phys., 60 (1983), pp. 187–207.
[33] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.
[34] J. A. Sethian, *Level Set Methods and Fast Marching Methods*, Cambridge University Press, Cambridge, UK, 1999.
[35] J. A. Sethian and A. Vladimirsky, *Fast methods for the Eikonal and related Hamilton-Jacobi equations on unstructured meshes*, Proc. Natl. Acad. Sci. USA, 97 (2000), pp. 5699–5703.
[36] J. A. Sethian and A. Vladimirsky, *Ordered upwind methods for static Hamilton–Jacobi equations: Theory and algorithms*, SIAM J. Numer. Anal., 41 (2003), pp. 325–363.
[37] L. Yatziv, A. Bartesaghi, and G. Sapiro, *O(N) implementation of the fast marching algorithm*, J. Comput. Phys., 212 (2006), pp. 393–399.

SIAM J. SCI. COMPUT.
Vol. 29, No. 2, pp. 635–655
© 2007 Society for Industrial and Applied Mathematics

COMPUTING THE NEAREST DOUBLY STOCHASTIC MATRIX WITH A PRESCRIBED ENTRY*

ZHENG-JIAN BAI†, DELIN CHU‡, AND ROGER C. E. TAN‡

Abstract. In this paper a nearest doubly stochastic matrix problem is studied. This problem is to find the closest doubly stochastic matrix with the prescribed $(1,1)$ entry to a given matrix. According to the well-established dual theory in optimization, the dual of the underlying problem is an unconstrained differentiable, but not twice differentiable, convex optimization problem. A Newton-type method is used for solving the associated dual problem, and then the desired nearest doubly stochastic matrix is obtained. Under some mild assumptions, the quadratic convergence of the proposed Newton method is proved. The numerical performance of the method is also demonstrated by numerical examples.

Key words. doubly stochastic matrix, generalized Jacobian, Newton's method, quadratic convergence

AMS subject classifications. 65K05, 15A51, 90C25

DOI. 10.1137/050639831

1. Introduction. A matrix $A \in \mathbb{R}^{n\times n}$ is called doubly stochastic if it is nonnegative and all its row and column sums are equal to one. Doubly stochastic matrices have found many important applications in probability and statistics, quantum mechanics, the study of hypergroups, economics and operation research, physical chemistry, communication theory and graph theory, etc.; see [15, 3, 17, 19, 23] and the references therein.

In this paper, we are interested in the best approximation problem related to doubly stochastic matrices: Given a matrix $T \in \mathbb{R}^{n\times n}$, find its nearest doubly stochastic matrix with the same $(1,1)$ entry as the given matrix T. This problem can be mathematically stated as follows:

$$\text{(1)} \qquad \begin{cases} \min & \frac{1}{2}\|M-T\|_F^2 \\ \text{s.t.} & M\mathbf{e}=\mathbf{e}, \quad \mathbf{e}^T M=\mathbf{e}^T, \\ & \mathbf{e}_1^T M\,\mathbf{e}_1=\mathbf{e}_1^T T\mathbf{e}_1, \\ & M\geq 0, \end{cases}$$

where

$$\mathbf{e}=(1,\ldots,1)^T\in\mathbb{R}^n, \qquad \mathbf{e}_1=(1,0,\ldots,0)^T\in\mathbb{R}^n,$$

and $M \geq 0$ means that M is nonnegative. Problem (1) was originally suggested by Professor Zhaojun Bai (Department of Computer Science, UC Davis). It arose from

*Received by the editors September 8, 2005; accepted for publication (in revised form) July 10, 2006; published electronically April 6, 2007. This work was partially supported by research grant R-146-000-047-112 of the National University of Singapore, research grant 0000-X07152 of Xiamen University, and National Natural Science Foundation of China grant 10601043.

http://www.siam.org/journals/sisc/29-2/63983.html

†Department of Information and Computational Mathematics, Xiamen University, Xiamen 361005, People's Republic of China (zjbai@xmu.edu.cn).

‡Department of Mathematics, National University of Singapore, 2 Science Drive 2, Singapore 117543 (matchudl@nus.edu.sg, scitance@nus.edu.sg).

numerical simulation of large (semiconductor, electronic) circuit networks. Padé approximation technique using the Lanczos process is very powerful for computing a lower order approximation to the linear system matrix describing the large linear network [1, 9]. The matrix T produced by the Lanczos process is in general not a doubly stochastic matrix. Suppose that the original system matrix is doubly stochastic; then we need to find the nearest doubly stochastic matrix M to T and at the same time match the moments.

Problem (1) has been studied in [10] based on the alternating projection method [2]. In [10, 14], problem (1) is simplified by removing the requirements on the $(1,1)$ entry and the nonnegativity of the matrix M. In this case, the solution can be obtained explicitly. We will revisit problem (1). Based on the dual approach in optimization [16], we will first reformulate (1) as an unconstrained differentiable but not twice differentiable convex optimization problem, next apply Newton's method to solve this convex problem, and then obtain the desired nearest doubly stochastic matrix. Under some mild assumptions, we will show that the proposed Newton method is quadratically convergent. We will also demonstrate the numerical performance of the method by numerical examples.

Throughout this paper, the following notation will be used:

- $$T = \begin{bmatrix} t_{1,1} & \cdots & t_{1,n} \\ \vdots & \ldots & \vdots \\ t_{n,1} & \cdots & t_{n,n} \end{bmatrix}.$$
- $A \geq 0$ ($A > 0$) means that A is nonnegative (positive).
- $$\mathcal{K} = \{A : \ A \in \mathbb{R}^{n\times n}, \ A \geq 0\}, \quad (z)_+ = \max\{0, z\}.$$
- $\Pi_{\mathcal{K}}(X)$ denotes the metric projection of X onto $\mathcal{K}$, i.e.,

$$\Pi_{\mathcal{K}}(X) = \begin{bmatrix} (x_{1,1})_+ & \cdots & (x_{1,n})_+ \\ \vdots & \ldots & \vdots \\ (x_{n,1})_+ & \cdots & (x_{n,n})_+ \end{bmatrix} \quad \forall\ X = \begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & \ldots & \vdots \\ x_{n,1} & \cdots & x_{n,n} \end{bmatrix} \in \mathbb{R}^{n\times n}.$$

2. Newton's method. In this section we consider a Newton-type method for computing the solution of problem (1).

Let

$$f(M) := \frac{1}{2}\|M - T\|_F^2, \quad \mathcal{A}(M) := \begin{bmatrix} M\mathbf{e} \\ \begin{bmatrix} I_{n-1} & 0 \end{bmatrix} M^T\mathbf{e} \\ \mathbf{e}_1^T M \mathbf{e}_1 \end{bmatrix}, \quad b := \begin{bmatrix} \mathbf{e} \\ \begin{bmatrix} I_{n-1} & 0 \end{bmatrix} \mathbf{e} \\ \mathbf{e}_1^T T \mathbf{e}_1 \end{bmatrix};$$

then problem (1) is equivalent to

$$\text{(2)} \qquad \begin{cases} \min & f(M) \\ \text{s.t.} & \mathcal{A}(M) = b, \\ & M \in \mathcal{K}. \end{cases}$$

The dual problem [16] of (2) is

$$\begin{cases} \sup & -\theta(x) \\ \text{s.t.} & x \in \mathbb{R}^{2n}, \end{cases} \tag{3}$$

where

$$\theta(x) = \frac{1}{2}\|\Pi_{\mathcal{K}}(T + \mathcal{A}^\star(x))\|_F^2 - x^T b - \frac{1}{2}\|T\|_F^2,$$

and $\mathcal{A}^\star$ is the adjoint of A and is defined by

$$\begin{aligned} \mathcal{A}^\star(x) &= \begin{bmatrix} I_n & 0 \end{bmatrix} x\mathbf{e}^T + \mathbf{e}x^T \begin{bmatrix} 0_{n\times(n-1)} & 0 \\ I_{n-1} & 0 \\ 0 & 0 \end{bmatrix} + \mathbf{e}_1 \begin{bmatrix} 0 & 1 \end{bmatrix} x\mathbf{e}_1^T \\ &= \begin{bmatrix} x_1 + x_{n+1} + x_{2n} & x_1 + x_{n+2} & \cdots & x_1 + x_{2n-1} & x_1 \\ x_2 + x_{n+1} & x_2 + x_{n+2} & \cdots & x_2 + x_{2n-1} & x_2 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ x_n + x_{n+1} & x_n + x_{n+2} & \cdots & x_n + x_{2n-1} & x_n \end{bmatrix} \\ &\forall x = \begin{bmatrix} x_1 \\ \vdots \\ x_{2n} \end{bmatrix} \in \mathbb{R}^{2n}. \end{aligned}$$

The relation between the values of (2) at its minimum and of the dual (3) at its maximum is stated in the following theorem.

THEOREM 2.1. [illegible] $M \in \mathbb{R}^{n\times n}$ [illegible] $\mathcal{K}$ [illegible] $\mathcal{A}(M) = b$ [illegible]

$$0 < \mathbf{e}_1^T T \mathbf{e}_1 < 1. \tag{4}$$

[illegible] (4)

(i) [illegible] (2) [illegible] $M^\star$.

(ii) [illegible] (3) [illegible] $x^\star$ [illegible]

$$M^\star = \Pi_{\mathcal{K}}(T + \mathcal{A}^\star(x^\star)). \tag{5}$$

[illegible]. If M is in the topological interior of $\mathcal{K}$ and $\mathcal{A}(M) = b$, then (4) follows directly from the properties that

$$\mathbf{e}_1^T T \mathbf{e}_1 = \mathbf{e}_1^T M \mathbf{e}_1 > 0, \quad \mathbf{e}_1^T M \mathbf{e} = 1,$$

and all entries of $\mathbf{e}_1^T M$ are positive. Conversely, if (4) holds, then it is clear that the matrix M defined by

$$M := \frac{1}{n-1} \begin{bmatrix} r_0 & r & \cdots & r & r \\ r & r_0 & \ddots & & r \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ r & & \ddots & \ddots & r \\ r & r & \cdots & r & r_0 \end{bmatrix}, \tag{6}$$

$$\text{with } r_0 = (n-1)\,\mathbf{e}_1^T T \mathbf{e}_1 > 0, \quad r = 1 - \mathbf{e}_1^T T \mathbf{e}_1 > 0,$$

satisfies that M is in the topological interior of $\mathcal{K}$ and $\mathcal{A}(M) = b$. Hence Theorem 2.1 follows.

Under the condition (4), parts (i) and (ii) of the theorem are now well known; see [12, 16, 21]. □

. . .. 1. In [12], the condition ensuring that there exists a matrix $M \in \mathbb{R}^{n\times n}$ in the topological interior of $\mathcal{K}$ such that $\mathcal{A}(M) = b$ is called the . . . . . for (2). Hence, we can regard (4) as the Slater condition for (2).

According to Theorem 2.1, once we can compute an optimal solution $x^\star$ of the dual problem (3), then we can obtain the optimal solution $M^\star$ of problem (2) by using (5).

Define

$$F(x) := \mathcal{A}(\Pi_{\mathcal{K}}(T + \mathcal{A}^\star(x))) - b \tag{7}$$

$$= \begin{bmatrix} (t_{1,1} + x_1 + x_{n+1} + x_{2n})_+ + \sum_{i=2}^{n-1}(t_{1,i} + x_1 + x_{n+i})_+ + (t_{1,n} + x_1)_+ \\ \sum_{i=1}^{n-1}(t_{2,i} + x_2 + x_{n+i})_+ + (t_{2,n} + x_2)_+ \\ \vdots \\ \sum_{i=1}^{n-1}(t_{n,i} + x_n + x_{n+i})_+ + (t_{n,n} + x_n)_+ \\ (t_{1,1} + x_1 + x_{n+1} + x_{2n})_+ + \sum_{j=2}^{n}(t_{j,1} + x_j + x_{n+1})_+ \\ \sum_{j=1}^{n}(t_{j,2} + x_j + x_{n+2})_+ \\ \vdots \\ \sum_{j=1}^{n}(t_{j,n-1} + x_j + x_{2n-1})_+ \\ (t_{1,1} + x_1 + x_{n+1} + x_{2n})_+ \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \\ 1 \\ \vdots \\ 1 \\ t_{11} \end{bmatrix}$$

for any

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_{2n} \end{bmatrix} \in \mathbb{R}^{2n}.$$

It is easy to know that the function $\theta(x)$ is continuously differentiable and that its gradient $\nabla\theta(x) = F(x)$ is globally Lipschitz continuous. Thus, both gradient-type methods and quasi-Newton methods can be directly employed to solve (3). However, since $\theta(x)$ is not twice continuously differentiable, the convergence rates of these methods are at most linear.

Since $\theta(x)$ is convex and differentiable, at solution $x^\star$ of (3),

$$\nabla\theta(x^\star) = 0; \qquad \text{i.e., } F(x^\star) = 0.$$

This indicates that we can obtain a solution of (3) by solving the equation $F(x) = 0$. $F(x)$ is globally Lipschitz continuous. According to Rademacher's theorem [22, Chapter 9.J], $F(x)$ is Fréchet differentiable almost everywhere. Let Ω_{F} be the set of points at which F is Fréchet differentiable. Denote the Jacobian of $F(x)$ at $x \in \Omega_{\mathrm{F}}$ by $F'(x)$. The generalized Jacobian $\partial F(x)$ of F at $x \in \mathbb{R}^{2n}$ in the sense of Clarke [6] is defined by

$$\partial F(x) := \mathrm{conv}\{\partial_B F(x)\},$$

where "conv" denotes the convex hull and

$$\partial_B F(x) := \left\{ V \in \mathbb{R}^{2n\times 2n} : \ V \text{ is an accumulation point of } F'(x^{(k)}),\ x^{(k)} \to x,\ x^{(k)} \in \Omega_{\mathrm{F}} \right\}.$$

The nonsmooth Newton method for solving equation

$$F(x) = 0 \tag{8}$$

is given by

$$x^{(k+1)} = x^{(k)} - V_k^{-1} F(x^{(k)}), \quad V_k \in \partial F(x^{(k)}). \tag{9}$$

The following result was established in [20].

THEOREM 2.2 (see [20]). [illegible] $F(x) = 0$ [illegible] $V \in \partial F(x^\star)$ [illegible] F [illegible] $x^\star$ [illegible] F [illegible] $x^\star$ [illegible] $V \in \partial F(x^\star + \delta x)$ [illegible] $\delta x \to 0$

$$F(x^\star + \delta x) - F(x^\star) - V(\delta x) = o(\|\delta x\|_F),$$

[illegible] (9) [illegible] $x^\star$ [illegible] $x^{(0)}$ [illegible] $x^\star$ [illegible] F [illegible] $x^\star$ [illegible] F [illegible] $x^\star$

$$F(x^\star + \delta x) - F(x^\star) - V(\delta x) = o(\|\delta x\|_F^2) \quad \forall\, V \in \partial F(x^\star + \delta x),\ \delta x \to 0,$$

[illegible]

Motivated by Theorem 2.2, in the following we discuss the strong semismoothness of F and the nonsingularity of all $V \in \partial F(x^\star)$ at a solution $x^\star$ of $F(x) = 0$.

Since

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_{2n} \end{bmatrix} \in \Omega_F,$$

i.e., $F'(x)$ exists if and only if

$$\begin{cases} t_{11} + x_1 + x_{n+1} + x_{2n} \neq 0, \\ t_{1,j} + x_1 + x_{n+j} \neq 0, \quad j = 2, \ldots, n-1, \\ t_{i,j} + x_i + x_{n+j} \neq 0, \quad i = 2, \ldots, n,\ j = 1, \ldots, n-1, \\ t_{i,n} + x_i \neq 0, \quad i = 1, \ldots, n, \end{cases}$$

in the case that the inequalities above hold,

$$\begin{aligned} a_{1,1} :&= \frac{\partial (t_{11} + x_1 + x_{n+1} + x_{2n})_+}{\partial x_1} = \frac{\partial (t_{11} + x_1 + x_{n+1} + x_{2n})_+}{\partial x_{n+1}} \\ &= \frac{\partial (t_{11} + x_1 + x_{n+1} + x_{2n})_+}{\partial x_{2n}} = \begin{cases} 1 & \text{if } t_{11} + x_1 + x_{n+1} + x_{2n} > 0, \\ 0 & \text{if } t_{11} + x_1 + x_{n+1} + x_{2n} < 0, \end{cases} \\ a_{1,j} :&= \frac{\partial (t_{1,j} + x_1 + x_{n+j})_+}{\partial x_1} = \frac{\partial (t_{1,j} + x_1 + x_{n+j})_+}{\partial x_{n+j}} \\ &= \begin{cases} 1 & \text{if } t_{1,j} + x_1 + x_{n+j} > 0, \\ 0 & \text{if } t_{1,j} + x_1 + x_{n+j} < 0, \end{cases} \quad j = 2, \ldots, n-1, \\ a_{i,j} :&= \frac{\partial (t_{i,j} + x_i + x_{n+j})_+}{\partial x_i} = \frac{\partial (t_{i,j} + x_i + x_{n+j})_+}{\partial x_{n+j}} \\ &= \begin{cases} 1 & \text{if } t_{i,j} + x_i + x_{n+j} > 0, \\ 0 & \text{if } t_{i,j} + x_i + x_{n+j} < 0, \end{cases} \quad \begin{aligned} &i = 2, \ldots, n, \\ &j = 1, \ldots, n-1, \end{aligned} \\ a_{i,n} :&= \frac{\partial (t_{i,n} + x_i)_+}{x_i} = \begin{cases} 1 & \text{if } t_{i,n} + x_i > 0, \\ 0 & \text{if } t_{i,n} + x_i < 0, \end{cases} \quad i = 1, \ldots, n, \end{aligned}$$

and

$$F'(x) = \left[\begin{array}{cccc|cccc|c} \sum_{i=1}^n a_{1,i} & & & & a_{1,1} & a_{1,2} & \cdots & a_{1,n-1} & a_{1,1} \\ & \sum_{i=1}^n a_{2,i} & & & a_{2,1} & a_{2,2} & \cdots & a_{2,n-1} & 0 \\ & & \ddots & & \vdots & \vdots & \cdots & \vdots & \vdots \\ & & & \sum_{i=1}^n a_{n,i} & a_{n,1} & a_{n,2} & \cdots & a_{n,n-1} & 0 \\ \hline a_{1,1} & a_{2,1} & \cdots & a_{n,1} & \sum_{i=1}^n a_{i,1} & & & & a_{1,1} \\ a_{1,2} & a_{2,2} & \cdots & a_{n,2} & & \sum_{i=1}^n a_{i,2} & & & 0 \\ \vdots & \vdots & \cdots & \vdots & & & \ddots & & \vdots \\ a_{1,n-1} & a_{2,n-1} & \cdots & a_{n,n-1} & & & & \sum_{i=1}^n a_{i,n-1} & 0 \\ \hline a_{1,1} & 0 & \cdots & 0 & a_{1,1} & 0 & \cdots & 0 & a_{1,1} \end{array}\right].$$

Thus, for any

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_{2n} \end{bmatrix} \in \mathbb{R}^{2n},$$

(10)

$$V \in \partial_B F(x) \Leftrightarrow V = \left[\begin{array}{cccc|cccc|c} \sum_{i=1}^n b_{1,i} & & & & b_{1,1} & b_{1,2} & \cdots & b_{1,n-1} & b_{1,1} \\ & \sum_{i=1}^n b_{2,i} & & & b_{2,1} & b_{2,2} & \cdots & b_{2,n-1} & 0 \\ & & \ddots & & \vdots & \vdots & \cdots & \vdots & \vdots \\ & & & \sum_{i=1}^n b_{n,i} & b_{n,1} & b_{n,2} & \cdots & b_{n,n-1} & 0 \\ \hline b_{1,1} & b_{2,1} & \cdots & b_{n,1} & \sum_{i=1}^n b_{i,1} & & & & b_{1,1} \\ b_{1,2} & b_{2,2} & \cdots & b_{n,2} & & \sum_{i=1}^n b_{i,2} & & & 0 \\ \vdots & \vdots & \cdots & \vdots & & & \ddots & & \vdots \\ b_{1,n-1} & b_{2,n-1} & \cdots & b_{n,n-1} & & & & \sum_{i=1}^n b_{i,n-1} & 0 \\ \hline b_{1,1} & 0 & \cdots & 0 & b_{1,1} & 0 & \cdots & 0 & b_{1,1} \end{array}\right],$$

where

$$\left\{\begin{array}{ll} b_{1,1} = 1 & \text{if } t_{11} + x_1 + x_{n+1} + x_{2n} > 0, \\ b_{1,1} \in \{0,1\} & \text{if } t_{11} + x_1 + x_{n+1} + x_{2n} = 0, \\ b_{1,1} = 0 & \text{if } t_{11} + x_1 + x_{n+1} + x_{2n} < 0, \end{array}\right.$$

$$\left\{\begin{array}{ll} b_{1,j} = 1 & \text{if } t_{1,j} + x_1 + x_{n+j} > 0, \\ b_{1,j} \in \{0,1\} & \text{if } t_{1,j} + x_1 + x_{n+j} = 0, \quad j = 2, \ldots, n-1, \\ b_{1,j} = 0 & \text{if } t_{1,j} + x_1 + x_{n+j} < 0, \end{array}\right.$$

(11)

$$\left\{\begin{array}{ll} b_{i,j} = 1 & \text{if } t_{i,j} + x_i + x_{n+j} > 0, \\ b_{i,j} \in \{0,1\} & \text{if } t_{i,j} + x_i + x_{n+j} = 0, \\ b_{i,j} = 0 & \text{if } t_{i,j} + x_i + x_{n+j} < 0, \end{array}\right. \quad \begin{array}{l} i = 2, \ldots, n, \\ j = 1, \ldots, n-1, \end{array}$$

$$\left\{\begin{array}{ll} b_{i,n} = 1 & \text{if } t_{i,n} + x_i > 0, \\ b_{i,n} \in \{0,1\} & \text{if } t_{i,n} + x_i = 0, \quad i = 1, \ldots, n. \\ b_{i,n} = 0 & \text{if } t_{i,n} + x_i < 0, \end{array}\right.$$

As a result, we obtain the following result.

THEOREM 2.3. $V \in \partial F(x)$ [illegible] V [illegible]

(12) $V =$

$$\left[\begin{array}{cccc|cccc|c} \sum_{i=1}^n v_{1,i} & & & & v_{1,1} & v_{1,2} & \cdots & v_{1,n-1} & v_{1,1} \\ & \sum_{i=1}^n v_{2,i} & & & v_{2,1} & v_{2,2} & \cdots & v_{2,n-1} & 0 \\ & & \ddots & & \vdots & \vdots & \cdots & \vdots & \vdots \\ & & & \sum_{i=1}^n v_{n,i} & v_{n,1} & v_{n,2} & \cdots & v_{n,n-1} & 0 \\ \hline v_{1,1} & v_{2,1} & \cdots & v_{n,1} & \sum_{i=1}^n v_{i,1} & & & & v_{1,1} \\ v_{1,2} & v_{2,2} & \cdots & v_{n,2} & & \sum_{i=1}^n v_{i,2} & & & 0 \\ \vdots & \vdots & \cdots & \vdots & & & \ddots & & \vdots \\ v_{1,n-1} & v_{2,n-1} & \cdots & v_{n,n-1} & & & & \sum_{i=1}^n v_{i,n-1} & 0 \\ \hline v_{1,1} & 0 & \cdots & 0 & v_{1,1} & 0 & \cdots & 0 & v_{1,1} \end{array}\right],$$

[illegible]

$$\left\{\begin{array}{ll} v_{1,1} = 1 & \text{if } t_{11} + x_1 + x_{n+1} + x_{2n} > 0, \\ v_{1,1} \in [0,\ 1] & \text{if } t_{11} + x_1 + x_{n+1} + x_{2n} = 0, \\ v_{1,1} = 0 & \text{if } t_{11} + x_1 + x_{n+1} + x_{2n} < 0, \end{array}\right.$$

$$\left\{\begin{array}{ll} v_{1,j} = 1 & \text{if } t_{1,j} + x_1 + x_{n+j} > 0, \\ v_{1,j} \in [0,\ 1] & \text{if } t_{1,j} + x_1 + x_{n+j} = 0, \quad j = 2, \ldots, n-1, \\ v_{1,j} = 0 & \text{if } t_{1,j} + x_1 + x_{n+j} < 0, \end{array}\right.$$

$$\left\{\begin{array}{ll} v_{i,j} = 1 & \text{if } t_{i,j} + x_i + x_{n+j} > 0, \\ v_{i,j} \in [0,\ 1] & \text{if } t_{i,j} + x_i + x_{n+j} = 0, \\ v_{i,j} = 0 & \text{if } t_{i,j} + x_i + x_{n+j} < 0, \end{array}\right. \quad \begin{array}{l} i = 2, \ldots, n, \\ j = 1, \ldots, n-1, \end{array} \tag{13}$$

$$\left\{\begin{array}{ll} v_{i,n} = 1 & \text{if } t_{i,n} + x_i > 0, \\ v_{i,n} \in [0,\ 1] & \text{if } t_{i,n} + x_i = 0, \quad i = 1, \ldots, n. \\ v_{i,n} = 0 & \text{if } t_{i,n} + x_i < 0, \end{array}\right.$$

We are now ready to present our results on the strong semismoothness of F and the nonsingularity of all $V \in \partial F(x)$.

THEOREM 2.4. [illegible] $x \in \mathbb{R}^{2n}$ $F(x)$ [illegible]

$$F(x + \delta x) - F(x) - V\delta x = 0 \quad \forall\ V \in \partial F(x + \delta x),\ \delta x \to 0. \tag{14}$$

[illegible] F [illegible] $x \in \mathbb{R}^{2n}$ [illegible]. A simple calculation yields that

$$\lim_{t \to 0^+} \frac{F(x + th) - F(x)}{t}$$

exists for any $x, h \in \mathbb{R}^{2n}$, and so $F(x)$ is directionally differentiable at any point $x \in \mathbb{R}^{2n}$. In addition, it can be verified using (10) and (11) that

$$F(x + \delta x) - F(x) - V\delta x = 0 \quad \forall\ V \in \partial_B F(x + \delta x),\ \delta x \to 0.$$

Since any $V \in \partial F(x + \delta x)$ is just a convex combination of elements in $\partial_B F(x + \delta x)$, (14) holds. □

THEOREM 2.5. [illegible]

$$x = \begin{bmatrix} x_1 \\ \\ x_{2n} \end{bmatrix} \in \mathbb{R}^{2n},$$

$$M := \begin{bmatrix} m_{1,1} & \cdots & m_{1,n} \\ & \cdots & \\ m_{n,1} & \cdots & m_{n,n} \end{bmatrix} = \Pi_{\mathcal{K}}(T + \mathcal{A}^{\star}(x))$$

$$= \Pi_{\mathcal{K}} \left(\begin{bmatrix} t_{1,1} + x_1 + x_{n+1} + x_{2n} & t_{1,2} + x_1 + x_{n+2} & \cdots & t_{1,n-1} + x_1 + x_{2n-1} & t_{1,n} + x_1 \\ t_{2,1} + x_2 + x_{n+1} & t_{2,2} + x_2 + x_{n+2} & \cdots & t_{2,n-1} + x_2 + x_{2n-1} & t_{2,n} + x_2 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ t_{n,1} + x_n + x_{n+1} & t_{n,2} + x_n + x_{n+2} & \cdots & t_{n,n-1} + x_n + x_{2n-1} & t_{n,n} + x_n \end{bmatrix} \right)$$

$N_M =$

$$\left[\begin{array}{cccc|cccc|c} \sum_{i=1}^n m_{1,i} & & & & m_{1,1} & m_{1,2} & \cdots & m_{1,n-1} & m_{1,1} \\ & \sum_{i=1}^n m_{2,i} & & & m_{2,1} & m_{2,2} & \cdots & m_{2,n-1} & 0 \\ & & \ddots & & \vdots & \vdots & \cdots & \vdots & \vdots \\ & & & \sum_{i=1}^n m_{n,i} & m_{n,1} & m_{n,2} & \cdots & m_{n,n-1} & 0 \\ \hline m_{1,1} & m_{2,1} & \cdots & m_{n,1} & \sum_{i=1}^n m_{i,1} & & & & m_{1,1} \\ m_{1,2} & m_{2,2} & \cdots & m_{n,2} & & \sum_{i=1}^n m_{i,2} & & & 0 \\ \vdots & \vdots & \cdots & \vdots & & & \ddots & & \vdots \\ m_{1,n-1} & m_{2,n-1} & \cdots & m_{n,n-1} & & & & \sum_{i=1}^n m_{i,n-1} & 0 \\ \hline m_{1,1} & 0 & \cdots & 0 & m_{1,1} & 0 & \cdots & 0 & m_{1,1} \end{array}\right].$$

(i) N_M

(ii) $V \in \partial F(x)$ N_M

(i) Since

$$m_{i,j} \geq 0, \quad i, j = 1, \ldots, n,$$

for any

$$h = \begin{bmatrix} h_1 \\ \vdots \\ h_{2n} \end{bmatrix} \in \mathbb{R}^{2n}$$

we find

$$h^T N_M h = m_{1,1} (h_1 + h_{n+1} + h_{2n})^2 + \sum_{i=2}^{n} m_{i,1}(h_i + h_{n+1})^2 + \sum_{j=2}^{n-1} \sum_{i=1}^{n} m_{i,j}(h_i + h_{n+j})^2 + \sum_{i=1}^{n} m_{i,n} h_i^2 \geq 0$$

and N_M is symmetric, so N_M is symmetric and positive semidefinite.

(ii) Among all $V \in \partial F(x)$, we consider the following one:

(15) $$V_{min} =$$

$$\left[\begin{array}{cccc|ccccc}
\sum_{i=1}^{n} v_{1,i}^{(min)} & & & & v_{1,1}^{(min)} & v_{1,2}^{(min)} & \cdots & v_{1,n-1}^{(min)} & v_{1,1}^{(min)} \\
& \sum_{i=1}^{n} v_{2,i}^{(min)} & & & v_{2,1}^{(min)} & v_{2,2}^{(min)} & \cdots & v_{2,n-1}^{(min)} & 0 \\
& & \ddots & & \vdots & \vdots & \cdots & \vdots & \vdots \\
& & & \sum_{i=1}^{n} v_{n,i}^{(min)} & v_{n,1}^{(min)} & v_{n,2}^{(min)} & \cdots & v_{n,n-1}^{(min)} & 0 \\
\hline
v_{1,1}^{(min)} & v_{2,1}^{(min)} & \cdots & v_{n,1}^{(min)} & \sum_{i=1}^{n} v_{i,1}^{(min)} & & & & v_{1,1}^{(min)} \\
v_{1,2}^{(min)} & v_{2,2}^{(min)} & \cdots & v_{n,2}^{(min)} & & \sum_{i=1}^{n} v_{i,2}^{(min)} & & & 0 \\
\vdots & \vdots & \cdots & \vdots & & & \ddots & & \vdots \\
v_{1,n-1}^{(min)} & v_{2,n-1}^{(min)} & \cdots & v_{n,n-1}^{(min)} & & & & \sum_{i=1}^{n} v_{i,n-1}^{(min)} & 0 \\
\hline
v_{1,1}^{(min)} & 0 & \cdots & 0 & v_{1,1}^{(min)} & 0 & \cdots & 0 & v_{1,1}^{(min)}
\end{array}\right],$$

where

$$\left\{\begin{array}{l} v_{1,1}^{(min)} = 1 \quad \text{if } m_{1,1} = (t_{11} + x_1 + x_{n+1} + x_{2n})_+ > 0, \\ v_{1,1}^{(min)} = 0 \quad \text{if } m_{1,1} = (t_{11} + x_1 + x_{n+1} + x_{2n})_+ = 0, \end{array}\right.$$

$$\left\{\begin{array}{l} v_{1,j}^{(min)} = 1 \quad \text{if } m_{1,j} = (t_{1,j} + x_1 + x_{n+j})_+ > 0, \\ v_{1,j}^{(min)} = 0 \quad \text{if } m_{1,j} = (t_{1,j} + x_1 + x_{n+j})_+ = 0, \end{array}\right. \qquad j = 2, \ldots, n-1,$$

(16)
$$\left\{\begin{array}{l} v_{i,j}^{(min)} = 1 \quad \text{if } m_{i,j} = (t_{i,j} + x_i + x_{n+j})_+ > 0, \\ v_{i,j}^{(min)} = 0 \quad \text{if } m_{i,j} = (t_{i,j} + x_i + x_{n+j})_+ = 0, \end{array}\right. \qquad \begin{array}{l} i = 2, \ldots, n, \\ j = 1, \ldots, n-1, \end{array}$$

$$\left\{\begin{array}{l} v_{i,n}^{(min)} = 1 \quad \text{if } m_{i,n} = (t_{i,n} + x_i)_+ > 0, \\ v_{i,n}^{(min)} = 0 \quad \text{if } m_{i,n} = (t_{i,n} + x_i)_+ = 0, \end{array}\right. \qquad i = 1, \ldots, n,$$

V_{min} and $V - V_{min}$ are symmetric and positive semidefinite since all $V \in \partial F(x)$ are given by (12) and (13),

$$v_{i,j}^{(min)} \geq 0, \qquad v_{i,j} - v_{i,j}^{(min)} \geq 0,$$

and for any

$$h = \begin{bmatrix} h_1 \\ \vdots \\ h_{2n} \end{bmatrix} \in \mathbb{R}^{2n},$$

we find

$$h^T V_{min} h = v_{1,1}^{(min)} (h_1 + h_{n+1} + h_{2n})^2 + \sum_{i=2}^{n} v_{i,1}^{(min)} (h_i + h_{n+1})^2 + \sum_{j=2}^{n-1} \sum_{i=1}^{n} v_{i,j}^{(min)} (h_i + h_{n+j})^2 + \sum_{i=1}^{n} v_{i,n}^{(min)} h_i^2 \geq 0,$$

$$h^T (V - V_{min}) h = (v_{1,1} - v_{1,1}^{(min)})(h_1 + h_{n+1} + h_{2n})^2 + \sum_{i=2}^{n} (v_{i,1} - v_{i,1}^{(min)})(h_i + h_{n+1})^2 + \sum_{j=2}^{n-1} \sum_{i=1}^{n} (v_{i,j} - v_{i,j}^{(min)})(h_i + h_{n+j})^2 + \sum_{i=1}^{n} (v_{i,n} - v_{i,n}^{(min)}) h_i^2 \geq 0.$$

Thus, all $V \in \partial F(x)$ are nonsingular if and only if V_{min} is positive definite.

Recall that $v_{i,j}^{(min)} > 0$ if and only if $m_{i,j} > 0$, $i, j = 1, \ldots, n$, and for any

$$h = \begin{bmatrix} h_1 \\ \vdots \\ h_{2n} \end{bmatrix} \in \mathbb{R}^{2n},$$

we have

$$\begin{aligned} h^T V_{min} h = v_{1,1}^{(min)} (h_1 + h_{n+1} + h_{2n})^2 + \sum_{i=2}^{n} v_{i,1}^{(min)} (h_i + h_{n+1})^2 \\ + \sum_{j=2}^{n-1} \sum_{i=1}^{n} v_{i,j}^{(min)} (h_i + h_{n+j})^2 + \sum_{i=1}^{n} v_{i,n}^{(min)} h_i^2, \\ h^T N_M h = m_{1,1} (h_1 + h_{n+1} + h_{2n})^2 + \sum_{i=2}^{n} m_{i,1} (h_i + h_{n+1})^2 \\ + \sum_{j=2}^{n-1} \sum_{i=1}^{n} m_{i,j} (h_i + h_{n+j})^2 + \sum_{i=1}^{n} m_{i,n} h_i^2, \end{aligned}$$

so that we get that $h^T V_{min} h > 0$ if and only if $h^T N_M h > 0$. This implies that V_{min} is positive definite if and only if N_M is positive definite. Hence, part (ii) is proved. □

2. The positive semidefiniteness of N_M can be proved[1] alternatively as follows: Let

$$\mathcal{N}_1 = \left[\begin{array}{cccc|cccc|c} \sum_{i=2}^n m_{1,i} & & & & 0 & m_{1,2} & \cdots & m_{1,n-1} & 0 \\ & \sum_{i=1}^n m_{2,i} & & & m_{2,1} & m_{2,2} & \cdots & m_{2,n-1} & 0 \\ & & \ddots & & \vdots & \vdots & \cdots & \vdots & \vdots \\ & & & \sum_{i=1}^n m_{n,i} & m_{n,1} & m_{n,2} & \cdots & m_{n,n-1} & 0 \\ \hline 0 & m_{2,1} & \cdots & m_{n,1} & \sum_{i=2}^n m_{i,1} & & & & 0 \\ m_{1,2} & m_{2,2} & \cdots & m_{n,2} & & \sum_{i=1}^n m_{i,2} & & & 0 \\ \vdots & \vdots & \cdots & \vdots & & & \ddots & & \vdots \\ m_{1,n-1} & m_{2,n-1} & \cdots & m_{n,n-1} & & & & \sum_{i=1}^n m_{i,n-1} & 0 \\ \hline 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \end{array}\right]$$

and

$$\mathcal{N}_2 = \left[\begin{array}{cccc|cccc|c} m_{1,1} & & & & m_{1,1} & 0 & \cdots & 0 & m_{1,1} \\ & 0 & & & 0 & 0 & \cdots & 0 & 0 \\ & & \ddots & & \vdots & \vdots & \cdots & \vdots & \vdots \\ & & & 0 & 0 & 0 & \cdots & 0 & 0 \\ \hline m_{1,1} & 0 & \cdots & 0 & m_{1,1} & & & & m_{1,1} \\ 0 & 0 & \cdots & 0 & & 0 & & & 0 \\ \vdots & \vdots & \cdots & \vdots & & & \ddots & & \vdots \\ 0 & 0 & \cdots & 0 & & & & 0 & 0 \\ \hline m_{1,1} & 0 & \cdots & 0 & m_{1,1} & 0 & \cdots & 0 & m_{1,1} \end{array}\right].$$

[1]This alternative proof was suggested by an anonymous referee.

Then

$$N_M = \mathcal{N}_1 + \mathcal{N}_2.$$

Obviously, $\mathcal{N}_2$ is positive semidefinite. Furthermore, $\mathcal{N}_1$ is nonnegative, symmetric, and weakly diagonally dominant, so the well-known Gershgorin theorem [13] gives that all eigenvalues of $\mathcal{N}_1$ are nonnegative. Thus, $\mathcal{N}_1$ is positive semidefinite. Hence, $N_M = \mathcal{N}_1 + \mathcal{N}_2$ is also positive semidefinite.

If $x = x^\star$ with $F(x^\star) = 0$, then Theorem 2.5(ii) can be simplified significantly, as shown in the next result.

THEOREM 2.6. $M^\star$ (2) $x^\star \in \mathbb{R}^{2n}$ $F(x^\star) = 0$

$$M^\star =: \begin{bmatrix} t_{1,1} & m^\star_{1,2} & \cdots & m^\star_{1,n-1} & m^\star_{1,n} \\ m^\star_{2,1} & m^\star_{2,2} & \cdots & m^\star_{2,n-1} & m^\star_{2,n} \\ & & \cdots & & \\ m^\star_{n,1} & m^\star_{n,2} & \cdots & m^\star_{n,n-1} & m^\star_{n,n} \end{bmatrix}, \tag{17}$$

$$L^\star := \begin{bmatrix} \frac{1}{\sqrt{1-t_{1,1}}} & \\ & I \end{bmatrix} \begin{bmatrix} 0 & m^\star_{1,2} & \cdots & m^\star_{1,n-1} \\ m^\star_{2,1} & m^\star_{2,2} & \cdots & m^\star_{2,n-1} \\ & & \cdots & \\ m^\star_{n,1} & m^\star_{n,2} & \cdots & m^\star_{n,n-1} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{1-t_{1,1}}} & \\ & I \end{bmatrix}. \tag{18}$$

(i)

$$\|L^\star\|_2 \le 1. \tag{19}$$

(ii) $V \in \partial F(x^\star)$

$$\|L^\star\|_2 < 1. \tag{20}$$

We have from Theorem 2.5(i) that $N_{M^\star}$ is symmetric and positive semidefinite. Now $0 < t_{1,1} < 1$, and $M^\star$ satisfies that

$$\begin{cases} 0 < m^\star_{1,1} = t_{1,1} < 1, \quad \sum_{i=2}^n m^\star_{1,i} = 1 - t_{1,1}, \quad \sum_{i=2}^n m^\star_{i,1} = 1 - t_{1,1}, \\ \sum_{i=1}^n m^\star_{j,i} = 1, \quad j = 2, \ldots, n, \\ \sum_{i=1}^n m^\star_{i,j} = 1, \quad j = 2, \ldots, n-1, \end{cases} \tag{21}$$

and thus we obtain, by using the positive semidefiniteness of $N_{M^\star}$, that the matrix

$$\mathcal{N}_{M^\star} := \left[\begin{array}{cccc|cccc} 1-t_{1,1} & & & & 0 & m^\star_{1,2} & \cdots & m^\star_{1,n-1} \\ & 1 & & & m^\star_{2,1} & m^\star_{2,2} & \cdots & m^\star_{2,n-1} \\ & & \ddots & & \vdots & \vdots & \cdots & \vdots \\ & & & 1 & m^\star_{n,1} & m^\star_{n,2} & \cdots & m^\star_{n,n-1} \\ \hline 0 & m^\star_{2,1} & \cdots & m^\star_{n,1} & 1-t_{1,1} & & & \\ m^\star_{1,2} & m^\star_{2,2} & \cdots & m^\star_{n,2} & & 1 & & \\ \vdots & \vdots & \cdots & \vdots & & & \ddots & \\ m^\star_{1,n-1} & m^\star_{2,n-1} & \cdots & m^\star_{n,n-1} & & & & 1 \end{array}\right] \tag{22}$$

is positive semidefinite. Equivalently, (19) holds.

(ii) By Theorem 2.5(ii) we know that all $V \in \partial F(x^\star)$ are nonsingular if and only if $N_{M^\star}$ is positive definite, which is equivalent to saying that the matrix $\mathcal{N}_{M^\star}$ defined by (22) is positive definite. Therefore, part (ii) follows directly from the property that $\mathcal{N}_{M^\star}$ is positive definite if and only if (20) holds. □

Theorem 2.6 is very pleasant because it indicates that for almost all $T \in \mathbb{R}^{n\times n}$, all $V \in \partial F(x^\star)$ are nonsingular for the solution $x^\star$ of the equation $F(x) = 0$.

The following corollary contains two important sufficient conditions ensuring that all $V \in \partial F(x^\star)$ are nonsingular for the solution $x^\star$ of the equation $F(x) = 0$.

COROLLARY 2.7. [illegible] 2.6 [illegible] $M^\star e_i > 0$ [illegible] $1 \le i \le n$ [illegible] $e_j^T M^\star > 0$ [illegible] $1 \le j \le n$ [illegible] $V \in \partial F(x^\star)$ [illegible] e_i [illegible] e_j [illegible] i [illegible] j [illegible] I_n [illegible]. By Theorem 2.6 and its proof we need to show only that $\mathcal{N}_{M^\star}$ defined by (22) is positive definite, provided $M^\star e_i > 0$ for some $1 \le i \le n$ (or $e_j^T M^\star > 0$ for some $1 \le j \le n$).

In the following we assume only that $M^\star e_i > 0$ for some $1 \le i \le n$ because the case that $e_j^T M^\star > 0$ for some $1 \le j \le n$ can be discussed similarly.

First, we have

$$0 < t_{1,1} < 1, \quad 0 \le m^\star_{1,j} < 1, \quad 0 \le m^\star_{j,1} < 1, \quad j = 2, \dots, n,$$

and for any

$$h = \begin{bmatrix} h_1 \\ \vdots \\ h_{2n-1} \end{bmatrix} \in \mathbb{R}^{2n-1},$$

we find

$$h^T \mathcal{N}_{M^\star} h = \sum_{i=2}^{n} m^\star_{i,1}(h_i + h_{n+1})^2 + \sum_{j=2}^{n-1}\sum_{i=1}^{n} m^\star_{i,j}(h_i + h_{n+j})^2 + \sum_{i=1}^{n} m^\star_{i,n} h_i^2. \tag{23}$$

Next, we show by considering three different cases that $h^T \mathcal{N}_{M^\star} h = 0$ only if $h = 0$, as follows.

[illegible] 1: $m^\star_{2,1} \ne 0, \dots, m^\star_{n,1} \ne 0$. In this case

$$\begin{aligned}
& h^T \mathcal{N}_{M^\star} h = 0 \\
\Longrightarrow & \begin{cases} h_2 = \cdots = h_n = -h_{n+1}, \\ h^T \mathcal{N}_{M^\star} h = \sum_{j=2}^{n-1}(h_{n+1} - h_{n+j})^2 \sum_{i=2}^{n} m^\star_{i,j} + h_{n+1}^2 \sum_{i=2}^{n} m^\star_{i,n} \\ \qquad\qquad + \sum_{j=2}^{n-1} m^\star_{1,j}(h_1 + h_{n+j})^2 + m^\star_{1,n} h_1^2 = 0 \end{cases} \\
\Longrightarrow & \begin{cases} h_2 = \cdots = h_n = -h_{n+1} = \cdots = -h_{2n-1} = 0, \\ h^T \mathcal{N}_{M^\star} h = h_1^2 \sum_{j=2}^{n} m^\star_{1,j} = 0 \end{cases} \\
& \qquad \left(\text{since } 0 < \sum_{i=2}^{n} m^\star_{i,j} = 1 - m^\star_{1,j} \le 1,\ j = 2, \dots, n \right) \\
\Longrightarrow & h_1 = \cdots = h_{2n-1} = 0 \quad \left(\text{since } 0 < \sum_{j=2}^{n} m^\star_{1,j} = 1 - t_{1,1} < 1 \right) \\
\Longrightarrow & h = 0.
\end{aligned}$$

• Case 2: $m^\star_{1,k} \neq 0, \ldots, m^\star_{n,k} \neq 0$ for some k with $2 \le k \le n-1$. In this case, we have

$$
\begin{aligned}
& h^T \mathcal{N}_{M^\star} h = 0 \\
\Longrightarrow & \begin{cases} h_1 = \cdots = h_n = -h_{n+k}, \\ h^T \mathcal{N}_{M^\star} h = (h_{n+k} - h_{n+1})^2 \sum_{i=2}^n m^\star_{i,1} + \sum_{j=2}^{k-1} (h_{n+k} - h_{n+j})^2 \sum_{i=1}^n m^\star_{i,j} \\ \qquad\qquad + \sum_{j=k+1}^{n-1} (h_{n+k} - h_{n+j})^2 \sum_{i=1}^n m^\star_{i,j} + h_{n+k}^2 \sum_{i=1}^n m^\star_{i,n} \end{cases} \\
\Longrightarrow & h_1 = \cdots = h_{2n-1} = 0 \\
& \Bigg(\text{since } 0 < \sum_{i=2}^n m^\star_{i,1} = 1 - t_{1,1} \neq 0,\ \sum_{i=1}^n m^\star_{i,n} = 1,\ \sum_{i=1}^n m^\star_{i,j} = 1, \\
& \qquad j = 2, \ldots, k-1, k+1, \ldots, n-1 \Bigg) \\
\Longrightarrow & h = 0.
\end{aligned}
$$

• Case 3: $m^\star_{1,n} \neq 0, \ldots, m^\star_{n,n} \neq 0$. In this case, we have

$$
\begin{aligned}
& h^T \mathcal{N}_{M^\star} h = 0 \\
\Longrightarrow & \begin{cases} h_1 = \cdots = h_n = 0, \\ h^T \mathcal{N}_{M^\star} h = h_{n+1}^2 \sum_{i=2}^n m^\star_{i,1} + \sum_{j=2}^{n-1} h_{n+j}^2 \sum_{i=1}^n m^\star_{i,j} \end{cases} \\
\Longrightarrow & h_1 = \cdots = h_n = 0,\ h_{n+1} = \cdots = h_{2n-1} = 0 \\
& \left(\text{since } 0 < \sum_{i=2}^n m^\star_{i,1} = 1 - t_{1,1} \neq 0,\ \sum_{i=1}^n m^\star_{i,j} = 1,\ j = 2, \ldots, n-1 \right) \\
\Longrightarrow & h = 0.
\end{aligned}
$$

Now we have shown that $h^T \mathcal{N}_{M^\star} h = 0$ only if $h = 0$. This means that $\mathcal{N}_{M^\star}$ is positive definite. □

Remark 3. Corollary 2.7 can be proved[2] alternatively as follows: Consider nonnegative matrix

$$
Z = \begin{bmatrix} D & R \\ R^T & D \end{bmatrix},
$$

where $D = \left[\begin{smallmatrix} t_{1,1} & 0 \\ 0 & 0 \end{smallmatrix}\right] \in \mathbb{R}^{n \times n}$ and R is obtained from the matrix $M^\star$ by replacing its $(1,1)$ entry with 0. Then Z is symmetric doubly stochastic with the largest eigenvalue equal to 1. If $M^\star$ has a nonzero row or a nonzero column, then Z is irreducible. Thus, up to a multiple,

$$
\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \in \mathbb{R}^{2n}
$$

is its only eigenvector corresponding to the largest eigenvalue 1. Now, let us remove the last row and the last column of Z to get the matrix $\tilde{Z} \in \mathbb{R}^{(2n-1)\times(2n-1)}$; then

[2]This alternative proof was also provided by an anonymous referee.

there is no positive vector $v \in \mathbb{R}^{2n-1}$ such that $\tilde{v} = v$. In other words, the largest eigenvalue of $\tilde{Z}$ is less than 1. Hence, $I_{2n-1} - \tilde{Z}$ is positive definite, and so is

$$\mathcal{N}_{M^\star} = (-I_n \oplus I_{n-1})(I_{2n-1} - \tilde{Z})(-I_n \oplus I_{n+1}).$$

An important consequence of Theorems 2.4 and 2.6 is on the convergence of Newton's method (9).

THEOREM 2.8. [illegible] $x^\star \in \mathbb{R}^{2n}$ [illegible] $F(x) = 0$ [illegible] (20) [illegible] (9) [illegible] $x^{(0)}$ [illegible] $x^\star$

3. Numerical algorithm. In our numerical implementation we use the following globalized version of Newton's method for solving the dual problem (3). Recall that $\nabla\theta(x) = F(x)$ for any $x \in \mathbb{R}^{2n}$.

ALGORITHM 1 (nonsmooth Newton's method).

Step 0. [illegible] $x^{(0)} \in \mathbb{R}^{2n}$ $\eta \in (0,1)$ $\rho, \delta \in (0, 1/2)$ $k := 0$

Step 1 (Newton's iteration). [illegible] $V_{min}^{(k)}$ [illegible] (15) [illegible] (16) [illegible] x [illegible] $x^{(k)}$ [illegible] [11, Algorithm 10.2.1] [illegible] $\Delta x^{(k)} \in \mathbb{R}^{2n}$ [illegible]

$$F(x^{(k)}) + V_{min}^{(k)} \Delta x = 0 \tag{24}$$

[illegible]

$$\|F(x^{(k)}) + V_{min}^{(k)} \Delta x^{(k)}\|_F \le \min\{\eta, \|F(x^{(k)}\|_F\} \|F(x^{(k)})\|_F \tag{25}$$

[illegible] $V_{min}^{(k)}$ [illegible] [illegible] [illegible] (25) [illegible]

$$(\Delta x^{(k)})^T F(x^{(k)}) \le -\min\{\eta, \|F(x^{(k)}\|_F\} (\Delta x^{(k)})^T \Delta x^{(k)} \tag{26}$$

[illegible] $V_{min}^{(k)}$ [illegible]

$$\Delta x^{(k)} = -F(x^{(k)}).$$

Step 2 (line search in the descent direction $\Delta x^{(k)}$ of $\theta(x)$ at $x^{(k)}$). [illegible] s_k [illegible] [illegible] s [illegible]

$$\theta(x^{(k)} + \rho^s \Delta x^{(k)}) - \theta(x^{(k)}) \le \delta \rho^s (\Delta x^{(k)})^T F(x^{(k)}).$$

[illegible]

$$x^{(k+1)} := x^{(k)} + \rho^{s_k} \Delta x^{(k)}.$$

Step 3. [illegible] k [illegible] $k+1$ [illegible] 1

In Algorithm 1, we choose the starting point $x^{(0)}$ as the solution of the following simplified version of (2):

$$\begin{aligned} \min \quad & \tfrac{1}{2}\|M - T\|_F^2 \\ \text{s.t.} \quad & \mathcal{A}(M) = b. \end{aligned} \tag{27}$$

This simplified problem has been studied in [10]. As in section 2, by the dual approach, we know that the unique solution M^0 to problem (27) is given by

$$M^0 = T + \mathcal{A}^*(x^{(0)}), \tag{28}$$

where $x^{(0)} \in R^{2n}$ is a solution of

$$\mathcal{A}\left(T + \mathcal{A}^*(x)\right) = b. \tag{29}$$

$x^{(0)}$ can be obtained by applying the CG method to

$$\begin{bmatrix} n & & & & 1 & \cdots & 1 & 1 \\ & n & & & 1 & \cdots & 1 & 0 \\ & & \ddots & & \vdots & \cdots & \vdots & \vdots \\ & & & n & 1 & \cdots & 1 & 0 \\ 1 & 1 & \cdots & 1 & n & & & 1 \\ \vdots & \vdots & \cdots & \vdots & & \ddots & & \vdots \\ 1 & 1 & \cdots & 1 & & & n & \\ 1 & 1 & \cdots & 1 & 1 & & & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ x_{n+1} \\ \vdots \\ x_{2n-1} \\ x_{2n} \end{bmatrix} = \begin{bmatrix} 1 - \sum_{i=1}^n t_{1,i} \\ 1 - \sum_{i=1}^n t_{2,i} \\ \vdots \\ 1 - \sum_{i=1}^n t_{n,i} \\ 1 - \sum_{i=1}^n t_{i,1} \\ \vdots \\ 1 - \sum_{i=1}^n t_{i,n-1} \\ 0 \end{bmatrix}. \tag{30}$$

THEOREM 3.1. [illegible] (20) [illegible] $\{x^{(k)}\}$ [illegible] 1 [illegible] $x^\star$ [illegible] $F(x) = 0$ [illegible]. Since for any $k \geq 0$, $\Delta x^{(k)}$ is always a descent direction of $\theta(x)$ at $x = x^{(k)}$, and since $\theta(x)$ is convex, we know that $\{x^{(k)}\}$ is bounded. Thus, we obtain by using Theorem 6.3.3 in [7], that

$$\lim_{k \to \infty} \nabla\theta(x^{(k)}) = 0,$$

which, in return, together with the convexity of $\theta(x)$ and the boundedness of $\{x^{(k)}\}$, yields that $x^{(k)} \to x^\star$ for some $x^\star$ satisfying $F(x^\star) = 0$.

Note that (20) holds, by Theorem 2.6(ii), and all $V \in \partial F(x^\star)$ are nonsingular. Since $x^{(k)} \to x^\star$, by Proposition 3.1 in [20], for all k sufficiently large, $V_{min}^{(k)}$ is positive definite and $\{\|V_{min}^{(k)}\|_F\}$ and $\{\|(V_{min}^{(k)})^{-1}\|_F\}$ are uniformly bounded. Thus, for all k sufficiently large, $\Delta x^{(k)}$ can satisfy (25) and (26), and moreover, (14) and that $F(x^\star) = 0$ yields

$$F(x^{(k)}) - F(x^\star) - V_{min}^{(k)}(x^{(k)} - x^\star) = 0, \quad \text{i.e., } F(x^{(k)}) = V_{min}^{(k)}(x^{(k)} - x^\star).$$

Hence, for all k sufficiently large,

$$\begin{aligned} \|x^{(k)} + \Delta x^{(k)} - x^\star\|_F &= \|(V_{min}^{(k)})^{-1}(F(x^{(k)}) + V_{min}^{(k)}\Delta x^{(k)})\|_F \\ &\leq \|(V_{min}^{(k)})^{-1}\|_F \|F(x^{(k)}) + V_{min}^{(k)}\Delta x^{(k)}\|_F \\ &\leq \|(V_{min}^{(k)})^{-1}\|_F \min\{\eta, \|F(x^{(k)})\|_F\}\|F(x^{(k)})\|_F \\ &\leq \|(V_{min}^{(k)})^{-1}\|_F \|F(x^{(k)}\|_F^2 \\ &\leq \|(V_{min}^{(k)})^{-1}\|_F \|V_{min}^{(k)}\|_F^2 \|x^{(k)} - x^\star\|_F^2 \\ &= O(\|x^{(k)} - x^\star\|_F^2). \end{aligned}$$

Then, for all k sufficiently large, $s_k = 0$, $\rho^{s_k} = 1$, and

$$x^{(k+1)} = x^{(k)} + \Delta x^{(k)}.$$

Therefore, the quadratic convergence follows. □

In the rest of this section, we report our numerical results for solving (1) by Algorithm 1. All the tests are implemented in MATLAB 7.0.1 running on a P4 PC with a 2.40GHz CPU. We also compare the performance of our method with that of the alternating projection method proposed in [10].

In our experiments, we tested the following two classes of problems.

1. Let M be given by (6). Set

$$T := M + \tau R,$$

where R is a random $n \times n$ real matrix with entries in $[-1.0, 1.0]$ and $\tau \in \mathbb{R}$ is a perturbed parameter. Here, we set $t_{1,1} = 0.5 < 1$ to ensure that (4) holds. We report our numerical results for $n = 500, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500, 5000$, and $\tau = 0.1, 1.0, 10.0$.

2. The matrix T is generated randomly with entries uniformly distributed between -10.0 and 10.0, but we set $t_{1,1} = 0.5 < 1$. We give our numerical results for $n = 500, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500, 5000$.

To demonstrate the performance of Algorithm 1, the linear systems (24) and (30) are solved with provision for lower (inexact) and higher (approximately exact) accuracy requirements.[3] To do this, in our numerical experiments we set the parameters used in our algorithm as either

(a) $\texttt{Tol} = 10^{-6},\ \eta = 10^{-6},\ \rho = 0.5,$ and $\delta = 10^{-4}$, or

(b) $\texttt{Tol} = 10^{-10},\ \eta = 10^{-15},\ \rho = 0.5,$ and $\delta = 10^{-4}$.

Here, Tol is the required tolerance used in the stopping criterion defined by

$$\|\nabla\theta(x^{(k)})\|_F = \|F(x^{(k)}\|_F \le \texttt{Tol}.$$

Our numerical results are given in Tables 1–4, where Time, Iter., Res0., Res*., and Err*. stand for the CPU times required for convergence, the number of iterations, the residuals $\|\nabla\theta(\cdot)\|_F$ at the starting point $x^{(0)}$ and the final iterate of Algorithm 1, and the error

$$\left\| \begin{bmatrix} M\mathbf{e} \\ M^T\mathbf{e} \\ \mathbf{e}_1^T M\mathbf{e}_1 \end{bmatrix} - \begin{bmatrix} \mathbf{e} \\ \mathbf{e} \\ \mathbf{e}_1^T T\mathbf{e}_1 \end{bmatrix} \right\|_F$$

at the computed solution M^*, respectively.

In our experiments, the quadratic convergence of Algorithm 1 is observed. From Tables 1–4, we note that if we solve the linear system (24) with a lower accuracy, it needs less CPU time, but we can obtain a coarser solution. Conversely, if we solve the linear system (24) with a higher accuracy, we can obtain a relatively more precise solution, but it needs relatively more CPU time. Finally, in our experiments, the largest numerical examples contain $25,000,000$ unknowns in the primal problem (1) and $10,000$ unknowns in the dual problem (3). This shows that Algorithm 1 is very efficient for large scale problems.

Next, we compare the performance of our Algorithm 1 with that of the alternating projection method in [10]. For the purpose of comparison, we set the stopping tolerance for both algorithms at

$$\left\| \begin{bmatrix} M\mathbf{e} \\ M^T\mathbf{e} \\ \mathbf{e}_1^T M\mathbf{e}_1 \end{bmatrix} - \begin{bmatrix} \mathbf{e} \\ \mathbf{e} \\ \mathbf{e}_1^T T\mathbf{e}_1 \end{bmatrix} \right\|_F \le \texttt{Tol}.$$

[3] As an anonymous referee pointed out, the linear system (30) can also be solved in linear time by using the direct methods exploiting the structure of the coefficient matrix of (30) [4, 5, 8, 18].

TABLE 1
Numerical results of Example 1(a).

$\texttt{Tol} = 10^{-6}$, $\eta = 10^{-6}$, $\rho = 0.5$, and $\delta = 10^{-4}$						
τ	n	Time	Iter.	Res0.	Res*.	Err*.
0.1	500	5.6 s	6	3.9×10^{2}	7.8×10^{-8}	7.8×10^{-8}
	1,000	26.2 s	7	1.1×10^{3}	7.2×10^{-13}	7.2×10^{-13}
	1,500	1 m 01 s	7	2.0×10^{3}	4.6×10^{-12}	4.6×10^{-12}
	2,000	1 m 55 s	7	3.1×10^{3}	1.1×10^{-7}	1.1×10^{-7}
	2,500	3 m 34 s	8	4.4×10^{3}	1.3×10^{-13}	1.3×10^{-13}
	3,000	5 m 15 s	8	5.8×10^{3}	2.8×10^{-13}	2.8×10^{-13}
	3,500	7 m 20 s	8	7.3×10^{3}	6.4×10^{-13}	6.4×10^{-13}
	4,000	10 m 07 s	8	8.9×10^{3}	2.0×10^{-12}	2.0×10^{-12}
	4,500	13 m 18 s	8	1.1×10^{4}	3.8×10^{-12}	3.8×10^{-12}
	5,000	16 m 52 s	9	1.2×10^{4}	1.2×10^{-8}	1.2×10^{-8}
1.0	500	8.5 s	8	3.9×10^{3}	1.3×10^{-11}	1.3×10^{-11}
	1,000	37.3 s	9	1.1×10^{4}	8.0×10^{-12}	8.0×10^{-12}
	1,500	1 m 27 s	9	2.1×10^{4}	8.3×10^{-13}	8.3×10^{-13}
	2,000	2 m 39 s	9	3.2×10^{4}	2.7×10^{-11}	2.7×10^{-11}
	2,500	4 m 15 s	9	4.4×10^{4}	2.0×10^{-11}	2.0×10^{-11}
	3,000	6 m 16 s	9	5.8×10^{4}	9.3×10^{-11}	9.3×10^{-11}
	3,500	8 m 50 s	9	7.3×10^{4}	8.5×10^{-11}	8.5×10^{-11}
	4,000	11 m 52 s	9	8.9×10^{4}	4.4×10^{-7}	4.4×10^{-7}
	4,500	17 m 16 s	10	1.1×10^{5}	1.5×10^{-12}	1.5×10^{-12}
	5,000	23 m 34 s	10	1.2×10^{5}	6.2×10^{-12}	6.2×10^{-12}
10.0	500	12.5 s	10	3.9×10^{4}	1.6×10^{-11}	1.6×10^{-11}
	1,000	46.4 s	10	1.1×10^{5}	1.2×10^{-10}	1.2×10^{-10}
	1,500	1 m 44 s	10	2.1×10^{5}	5.0×10^{-10}	5.0×10^{-10}
	2,000	3 m 11 s	10	3.2×10^{5}	2.1×10^{-9}	2.3×10^{-9}
	2,500	5 m 38 s	11	4.4×10^{5}	2.4×10^{-12}	2.4×10^{-12}
	3,000	8 m 11 s	11	5.8×10^{5}	1.7×10^{-12}	1.7×10^{-12}
	3,500	11 m 23 s	11	7.3×10^{5}	3.7×10^{-11}	3.7×10^{-11}
	4,000	15 m 24 s	11	8.9×10^{5}	2.6×10^{-11}	2.6×10^{-11}
	4,500	19 m 46 s	11	1.1×10^{6}	1.3×10^{-9}	1.3×10^{-9}
	5,000	27 m 14 s	11	1.2×10^{6}	2.5×10^{-10}	2.5×10^{-10}

Here, we choose $\texttt{Tol}$ to be different values, e.g., $\texttt{Tol} = 10^{-6}, 10^{-10}$, etc. Consequently, in Algorithm 1 (24) is solved with varying accuracies; see $\eta = 10^{-6}, 10^{-15}$, etc. The values of the remaining parameters used in Algorithm 1 are set as above. Tables 5–6 list the numerical results for Example 1 with varying n, $\texttt{Tol}$, and η, where $\texttt{Dist}$ is the distance between T and the computed closest matrix M in the Frobenius norm. Here, we report the numerical results only for $\tau = 0.1$ and $n = 500, 1000, 1500, 2000$, 2500, 3000, and 3500, as the other cases behave similarly.

From Tables 5–6 we observe that Algorithm 1 is much more efficient than the alternating projection method in [10].

4. Conclusions. In this paper we proposed to solve the dual problem (3) by Algorithm 1 in order to obtain the solution of the nearest doubly stochastic matrix problem (1). Under the mild assumptions (4) and (20), we have shown that Algorithm 1 is quadratically convergent. We have also demonstrated its numerical performance by some examples.

In problem (1), only the $(1, 1)$ entry of the matrix M is fixed to be identical to the given matrix T. This is an assumption without loss of generality, since the framework we establish in this paper can be easily applied to the nearest doubly stochastic matrix problem with k prescribed entries. In fact, consider the following problem:

TABLE 2
Numerical results of Example 1(b).

$\texttt{Tol} = 10^{-10}$, $\eta = 10^{-15}$, $\rho = 0.5$, and $\delta = 10^{-4}$						
τ	n	Time	Iter.	Res0.	Res*.	Err*.
0.1	500	13.5 s	6	3.9×10^{2}	1.3×10^{-14}	1.3×10^{-14}
	1,000	1 m 05 s	7	1.1×10^{3}	2.0×10^{-14}	2.0×10^{-14}
	1,500	2 m 30 s	7	2.0×10^{3}	3.0×10^{-14}	3.0×10^{-14}
	2,000	4 m 18 s	7	3.1×10^{3}	3.9×10^{-14}	3.9×10^{-14}
	2,500	5 m 03 s	8	4.4×10^{3}	4.9×10^{-14}	4.9×10^{-14}
	3,000	11 m 53 s	8	5.8×10^{3}	5.8×10^{-14}	5.8×10^{-14}
	3,500	16 m 33 s	8	7.3×10^{3}	6.7×10^{-14}	6.7×10^{-14}
	4,000	24 m 52 s	9	8.9×10^{3}	7.7×10^{-14}	7.7×10^{-14}
	4,500	28 m 07 s	8	1.1×10^{4}	8.4×10^{-14}	8.4×10^{-14}
	5,000	41 m 23 s	9	1.2×10^{4}	9.3×10^{-14}	9.3×10^{-14}
1.0	500	20.6 s	8	3.9×10^{3}	2.8×10^{-14}	2.8×10^{-14}
	1,000	1 m 19 s	8	1.1×10^{4}	8.0×10^{-14}	8.0×10^{-14}
	1,500	3 m 30 s	9	2.0×10^{4}	8.3×10^{-14}	8.3×10^{-14}
	2,000	6 m 28 s	9	3.2×10^{4}	1.1×10^{-13}	1.1×10^{-13}
	2,500	7 m 08 s	10	4.4×10^{4}	1.4×10^{-13}	1.4×10^{-13}
	3,000	15 m 16 s	10	5.8×10^{4}	1.6×10^{-13}	1.6×10^{-13}
	3,500	23 m 24 s	10	7.3×10^{4}	1.9×10^{-13}	1.9×10^{-13}
	4,000	30 m 02 s	10	8.9×10^{4}	2.2×10^{-13}	2.2×10^{-13}
	4,500	38 m 21 s	10	1.1×10^{5}	2.5×10^{-13}	2.5×10^{-13}
	5,000	48 m 54 s	10	1.2×10^{5}	2.7×10^{-13}	2.7×10^{-13}
10.0	500	29.1 s	9	3.9×10^{4}	3.3×10^{-11}	3.3×10^{-11}
	1,000	2 m 04 s	10	1.1×10^{5}	3.2×10^{-13}	3.2×10^{-13}
	1,500	4 m 29 s	10	2.1×10^{5}	6.5×10^{-13}	6.5×10^{-13}
	2,000	8 m 00 s	10	3.2×10^{5}	2.4×10^{-12}	2.4×10^{-12}
	2,500	8 m 35 s	11	4.4×10^{5}	7.7×10^{-13}	7.7×10^{-13}
	3,000	12 m 42 s	11	5.8×10^{5}	9.1×10^{-13}	9.1×10^{-13}
	3,500	17 m 02 s	11	7.3×10^{5}	1.1×10^{-12}	1.1×10^{-12}
	4,000	22 m 22 s	11	8.9×10^{5}	1.2×10^{-12}	1.2×10^{-12}
	4,500	29 m 25 s	11	1.1×10^{6}	1.3×10^{-12}	1.3×10^{-12}
	5,000	51 m 57 s	11	1.2×10^{6}	1.5×10^{-12}	1.5×10^{-12}

$$
(31)\qquad \begin{cases} \min & \frac{1}{2}\|M - T\|_F^2 \\ \text{s.t.} & M\mathbf{e} = \mathbf{e}, \quad \mathbf{e}^T M = \mathbf{e}^T, \\ & \mathbf{e}_{i_1}^T M\, \mathbf{e}_{j_1} = \mathbf{e}_{i_1}^T T \mathbf{e}_{j_1}, \\ & \qquad\qquad \vdots \\ & \mathbf{e}_{i_k}^T M\, \mathbf{e}_{j_k} = \mathbf{e}_{i_k}^T T \mathbf{e}_{j_k}, \\ & M \geq 0, \end{cases}
$$

where k, $i_1, \ldots, i_k$, $j_1, \ldots, j_k$ are integers,

$$1 \leq k \leq n, \quad 1 \leq i_1 \leq \cdots \leq i_k \leq n, \quad 1 \leq j_1 \leq \cdots \leq j_k \leq n,$$

$(i_1, j_1), \ldots, (i_k, j_k)$ are distinct, and $\mathbf{e}_j$ denotes the jth column of the n-by-n identity matrix. Let

TABLE 3
Numerical results of Example 2(a).

$\texttt{Tol} = 10^{-6}$, $\eta = 10^{-6}$, $\rho = 0.5$, and $\delta = 10^{-4}$					
n	Time	Iter.	Res0.	Res*.	Err*.
500	13.0 s	10	3.9×10^{4}	4.4×10^{-11}	4.4×10^{-11}
1,000	46.7 s	10	1.1×10^{5}	7.4×10^{-11}	7.4×10^{-11}
1,500	1 m 48 s	10	2.1×10^{5}	4.5×10^{-10}	4.5×10^{-10}
2,000	3 m 19 s	10	3.2×10^{5}	7.1×10^{-10}	7.1×10^{-10}
2,500	5 m 20 s	10	4.4×10^{5}	2.2×10^{-9}	2.2×10^{-9}
3,000	8 m 39 s	10	5.8×10^{5}	2.1×10^{-11}	2.1×10^{-11}
3,500	12 m 06 s	11	7.3×10^{5}	3.9×10^{-11}	3.9×10^{-11}
4,000	15 m 54 s	11	8.9×10^{5}	8.0×10^{-11}	8.0×10^{-11}
4,500	21 m 15 s	11	1.1×10^{6}	5.0×10^{-11}	5.0×10^{-11}
5,000	27 m 12 s	11	1.2×10^{6}	1.1×10^{-10}	1.1×10^{-10}

TABLE 4
Numerical results of Example 2(b).

$\texttt{Tol} = 10^{-10}$, $\eta = 10^{-15}$, $\rho = 0.5$, and $\delta = 10^{-4}$					
n	Time	Iter.	Res0.	Res*.	Err*.
500	29.7 s	9	3.9×10^{4}	3.9×10^{-12}	3.9×10^{-12}
1,000	2 m 03 s	10	1.1×10^{5}	3.2×10^{-13}	3.2×10^{-13}
1,500	4 m 25 s	10	2.1×10^{5}	7.8×10^{-13}	7.8×10^{-13}
2,000	8 m 22 s	11	3.2×10^{5}	6.0×10^{-13}	6.0×10^{-13}
2,500	14 m 26 s	11	4.4×10^{5}	7.5×10^{-13}	7.5×10^{-13}
3,000	19 m 57 s	11	5.8×10^{5}	9.2×10^{-13}	9.2×10^{-13}
3,500	26 m 45 s	11	7.3×10^{5}	1.0×10^{-12}	1.0×10^{-12}
4,000	33 m 12 s	11	8.9×10^{5}	1.2×10^{-12}	1.2×10^{-12}
4,500	49 m 47 s	12	1.1×10^{6}	1.4×10^{-12}	1.4×10^{-12}
5,000	56 m 41 s	11	1.2×10^{6}	1.5×10^{-12}	1.5×10^{-12}

$$f(M) := \frac{1}{2}\|M - T\|_F^2, \quad \mathcal{B}(M) := \begin{bmatrix} M\mathbf{e} \\ \begin{bmatrix} I_{n-1} & 0 \end{bmatrix} M^T\mathbf{e} \\ \mathbf{e}_{i_1}^T M \mathbf{e}_{j_1} \\ \vdots \\ \mathbf{e}_{i_k}^T M \mathbf{e}_{j_k} \end{bmatrix}, \quad c := \begin{bmatrix} \mathbf{e} \\ \begin{bmatrix} I_{n-1} & 0 \end{bmatrix} \mathbf{e} \\ \mathbf{e}_{i_1}^T T \mathbf{e}_{j_1} \\ \vdots \\ \mathbf{e}_{i_k}^T T \mathbf{e}_{j_k} \end{bmatrix};$$

then problem (31) is equivalent to

$$\begin{cases} \min & f(M) \\ \text{s.t.} & \mathcal{B}(M) = c, \\ & M \in \mathcal{K}. \end{cases} \tag{32}$$

The dual problem of (32) is

$$\begin{cases} \sup & -\theta(x) \\ \text{s.t.} & x \in \mathbb{R}^{2n+k-1}, \end{cases} \tag{33}$$

where

$$\theta(x) = \frac{1}{2}\|\Pi_{\mathcal{K}}(T + \mathcal{B}^\star(x))\|_F^2 - x^T c - \frac{1}{2}\|T\|_F^2,$$

and $\mathcal{B}^\star$ is the adjoint of $\mathcal{B}$. Hence, we can extend the results that we derived for problem (1) to problem (31) and apply a Newton-type method to solve the dual problem (33) and then obtain the desired solution of the problem (31).

TABLE 5
Numerical results of Example 1(a).

$\texttt{Tol} = 10^{-6}$				
	n	Time	Iter.	Dist
Algorithm 1	500	6.7 s	6	28.028746
	1,000	26.2 s	7	56.855379
	1,500	1 m 05 s	7	85.690588
	2,000	2 m 01 s	7	114.556635
$\eta = 10^{-6}$	2,500	3 m 08 s	7	143.421146
	3,000	5 m 16 s	8	172.273970
	3,500	7 m 24 s	8	201.138311
Algorithm 1	500	7.0 s	6	28.028746
	1,000	31.7 s	7	56.855379
	1,500	1 m 17 s	7	85.690588
	2,000	2 m 25 s	7	114.556635
$\eta = 10^{-10}$	2,500	3 m 42 s	7	143.421146
	3,000	6 m 21 s	8	172.273970
	3,500	8 m 40 s	8	201.138311
Algorithm 1	500	8.6 s	6	28.028746
	1,000	39.0 s	7	56.855379
	1,500	1 m 32 s	7	85.690588
	2,000	2 m 51 s	7	114.556635
$\eta = 10^{-15}$	2,500	4 m 16 s	7	143.421146
	3,000	7 m 22 s	8	172.273970
	3,500	10 m 10 s	8	201.138311
Method in [10]	500	21.1 s	181	28.028746
	1,000	2 m 53 s	263	56.855379
	1,500	10 m 51 s	321	85.690588
	2,000	25 m 55 s	371	114.556635
	2,500	53 m 19 s	417	143.421146
	3,000	1 h 59 m 01 s	457	172.273970
	3,500	2 h 58 m 54 s	495	201.138311

TABLE 6
Numerical results of Example 1(b).

$\texttt{Tol} = 10^{-10}$				
	n	Time	Iter.	Dist
Algorithm 1	500	7.3 s	6	28.0045624140
	1,000	33.6 s	7	56.8449830344
	1,500	1 m 18 s	7	85.6644919002
	2,000	2 m 47 s	8	114.5446535050
$\eta = 10^{-10}$	2,500	4 m 27 s	8	143.4251826139
	3,000	6 m 31 s	8	172.3135425545
	3,500	9 m 08 s	8	201.0862903056
Algorithm 1	500	9.0 s	6	28.0045624140
	1,000	40.9 s	7	56.8449830344
	1,500	1 m 35 s	7	85.6644919002
	2,000	3 m 17 s	8	114.5446535050
$\eta = 10^{-15}$	2,500	5 m 13 s	8	143.4251826139
	3,000	7 m 32 s	8	172.3135425545
	3,500	10 m 16 s	8	201.0862903056
Method in [10]	500	35.1 s	294	28.0045624140
	1,000	4 m 36 s	413	56.8449830344
	1,500	16 m 10 s	503	85.6644919002
	2,000	40 m 01 s	581	114.5446535050
	2,500	1 h 30 m 24 s	643	143.4251826139
	3,000	2 h 47 m 51 s	709	172.3135425545
	3,500	4 h 20 m 10 s	770	201.0862903056

Acknowledgment. We are very grateful to Dr. Defeng Sun for his valuable suggestions and comments on the reported work.

REFERENCES

[1] Z. Bai and R. W. Freund, *A partial Padé-via-Lanczos method for reduced-order modeling*, Linear Algebra Appl., 332–334 (2001), pp. 139–164.
[2] J. P. Boyle and R. L. Dykstra, *A method for finding projections onto the intersections of convex sets in Hilbert spaces*, in Advances in Order Restricted Statistical Inference, R. Dykstra, T. Robertson, and F. T. Wright, eds., Lecture Notes in Statist. 37, Springer-Verlag, Berlin, New York, 1986, pp. 28–47.
[3] R. A. Brualdi, *Some applications of doubly stochastic matrices*, Linear Algebra Appl., 107 (1988), pp. 77–100.
[4] S. Chandrasekaran, P. Dewilde, M. Gu, T. Pals, X. Sun, A. J. van der Veen, and D. White, *Some fast algorithms for sequentially semiseparable representations*, SIAM J. Matrix Anal. Appl., 27 (2005), pp. 341–364.
[5] S. Chandrasekaran and M. Gu, *Fast and stable algorithms for banded plus semiseparable systems of linear equations*, SIAM J. Matrix Anal. Appl., 25 (2003), pp. 373–384.
[6] F. H. Clarke, *Optimization and Nonsmooth Analysis*, John Wiley & Sons, New York, 1983.
[7] J. E. Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice–Hall, Englewood Cliffs, NJ, 1983.
[8] Y. Eidelman and I. Gohberg, *A modification of the Dewilde–van der Veen method for inversion of finite structured matrices*, Linear Algebra Appl., 343–344 (2002), pp. 419–450.
[9] R. Feldmann and R. W. Freund, *Efficient linear circuit analysis by Padé approximation via the Lanczos process*, IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, 14 (1995), pp. 639–649.
[10] W. Glunt, T. L. Hayden, and R. Reams, *The nearest doubly stochastic matrix to a real matrix with the same first moment*, Numer. Linear Algebra Appl., 5 (1998), pp. 475–482.
[11] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore and London, 1996.
[12] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex Analysis and Minimization Algorithms*, Springer-Verlag, Berlin, 1993.
[13] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, UK, 1985.
[14] R. N. Khoury, *Closest matrices in the space of generalized doubly stochastic matrices*, J. Math. Anal. Appl., 222 (1998), pp. 562–568.
[15] J. D. Louck, *Doubly stochastic matrices in quantum mechanics*, Found. Phys., 27 (1997), pp. 1085–1104.
[16] J. Malick, *A dual approach to semidefinite least-squares problems*, SIAM J. Matrix Anal. Appl., 26 (2004), pp. 272–284.
[17] M. Marcus, *Some properties and applications of doubly stochastic matrices*, Amer. Math. Monthly, 67 (1960), pp. 215–221.
[18] N. Mastronardi, S. Chandrasekaran, and S. van Huffel, *Fast and stable two-way chasing algorithms for diagonal plus semiseparable systems of linear equations*, Numer. Linear Algebra Appl., 38 (2000), pp. 7–12.
[19] H. Minc, *Nonnegative Matrices*, John Wiley & Sons, New York, 1988.
[20] L. Qi and J. Sun, *A nonsmooth version of Newton's method*, Math. Programming, 58 (1993), pp. 353–367.
[21] R. T. Rockafellar, *Conjugate Duality and Optimization*, CBMS-NSF Reg. Conf. Ser. Appl. Math. 16, SIAM, Philadelphia, 1974.
[22] R. T. Rockafellar and R. J. B. Wets, *Variational Analysis*, Springer, Berlin, 1998.
[23] V. S. Sunder and N. J. Wildberger, *Action of finite hypergroups and examples*, Harmonic Analysis and Hypergroups, Birkhäuser Boston, Boston, MA, 1998, pp. 145–163.

SIAM J. SCI. COMPUT.
Vol. 29, No. 2, pp. 656–673
© 2007 Society for Industrial and Applied Mathematics

A SCHWARZ WAVEFORM MOVING MESH METHOD*

RONALD D. HAYNES† AND ROBERT D. RUSSELL‡

Abstract. An r-refinement (moving mesh) method is considered for solving time dependent partial differential equations (PDEs). The resulting coupled system, consisting of the physical PDE and a moving mesh PDE, is solved by a Schwarz waveform relaxation method. In particular, the computational space-time domain is decomposed into overlapping subdomains and the solution obtained by iteratively solving the system of PDEs on each subdomain. Dirichlet boundary conditions are used to pass solution information between neighboring regions. The efficacy of this approach is demonstrated for some model problems. For problems where the solutions evolve on disparate time scales in different regions of the spatial domain, this approach demonstrates the significant savings in computational time and effort which are possible.

Key words. Schwarz waveform relaxation, moving meshes, domain decomposition, dynamic iteration

AMS subject classifications. 65M55, 65M06, 65M50, 65M20

DOI. 10.1137/050631549

1. Introduction.

The method of lines (MOL), one of the predominant approaches for solving time dependent parabolic partial differential equations (PDEs), involves discretizing spatially and then integrating in time along either fixed or moving lines. The efficacy of the method rests on that of the subsequent initial value problem (IVP) solvers. Traditional IVP solvers integrate using time steps chosen to keep an estimate of the local error below some user specified tolerance. As a consequence, the time step, largely dictated by the behavior of the fastest components locally, is used to integrate all solution components.

This global approach introduces inefficiencies by often requiring a small time step for each unknown, particularly when there are disparate scales in solution components. This occurs when solution scales along lines differ substantially, which is the case, for example, when the solution has moving interfaces or singular behavior.

A number of ways to overcome this limitation by integrating using steps locally suited to the time dynamics have been investigated. The idea appears to have originated in a paper by Rice [42] which considers split or multirate Runge–Kutta methods applied to systems of ordinary differential equations (ODEs). In the independent work of Gander and Stuart [22] and Giladi and Keller [25], the domain decomposition method is applied for solving time dependent PDEs. Splitting the problem spatially allows the solution in different parts of the spatial domain to be evolved according to local time scale. The technique, known as Schwarz waveform relaxation, is able to overcome many of the problems which exist in applying the ODE methods mentioned above to semidiscretized PDEs.

It is well accepted that the efficient solution of complex PDEs generally requires methods which are adaptive in both space and time. In this paper we are interested in

*Received by the editors May 13, 2005; accepted for publication (in revised form) September 20, 2006; published electronically April 6, 2007.

http://www.siam.org/journals/sisc/29-2/63154.html

†Department of Mathematics & Statistics, Acadia University, Wolfville, NS B4P 2R6, Canada (ronald.haynes@acadiau.ca). This author was supported in part by NSERC (Canada) postdoctoral fellowships and discovery grants.

‡Department of Mathematics, Simon Fraser University, Burnaby, BC V5A 1S6, Canada (rdr@cs.sfu.ca). This author was supported in part by NSERC (Canada) under grant OGP-0008781.

a class of spatially adaptive moving mesh PDE methods introduced in [41, 29, 30]. Our purpose is to introduce and explore a natural coupling of domain decomposition, in the Schwarz waveform context, and the spatially adaptive moving mesh PDE methods.

In the next two sections we review the relevant literature for both the ODE and PDE contexts. In section 4 we review the so-called moving mesh PDE methods. Section 5 introduces a Schwarz waveform moving mesh method. In section 6 we provide some numerical results to justify the utility of the technique. We conclude in section 7 with a summary and discussion of ongoing and future work.

2. ODE methods. There have been a number of approaches for solving systems of ODEs using decoupled integration. A decoupled integration method proceeds by identifying groups of components which evolve at similar time scales or for which it is advantageous to use particular integration formulas. Communication between the subsystems is typically handled through either interpolation or some iterative procedure. Gear [24] investigates solving a system of ODEs where the ith component is integrated using a step size function $h_i(t)$. Due to the cost of the required interpolations the method is deemed competitive only if the cost of the function evaluations is high or if f_i depends on few solution components. Hofer [28] proposes a method where the ODE is decoupled into stiff and nonstiff components. Computational efficiency is achieved by the decoupling of the integration and by restricting the number of implicit formulations to a few equations. Assuming the system has been decoupled a priori, Andrus [2] presents a method where both a different integration method and time step are used for the slow and fast subsystems. Skelboe [44, 43] investigates multirate decoupled backward differentiation formulas (BDFs) for general systems of ODEs separated into subsystems. The partitioning requires subsystems having strongly interacting components but weak couplings to the other subsystems. Deuflhard [10] suggests analyzing the Jacobian to detect groups of fast and slow components. More recently, Engstler and Lubich [11, 12] have considered a multirate technique which does dynamic decoupling using extrapolation methods.

One of the earliest techniques for decoupling the integration of systems of ODEs was waveform relaxation, where subsystems are integrated separately and communication between them handled through iteration. The theory of the method is considered in Lelarasmee, Ruehli, and Sangiovanni-Vincentelli [35] and later in Miekkala and Nevanlinna [38], which contributed to the subsequent popularization of the method.

3. PDE methods. There have been several attempts to incorporate a multirate strategy within a solution methodology for PDEs. The MOL, for example, yields a large system of ODEs for which there may be a motivation to decouple the fast and slow components. The partitioning of components can be done dynamically and automatically to adapt to temporally and spatially evolving features in the solution, although this complicates the use of the majority of the above ODE methods.

Not surprisingly the motivation to use different space-time grids in regions of the computational domain has spawned intensive research. Flaherty and Moore [14] develop integrated space-time *hp*-refinement strategies. These methods make decisions about the number of mesh points, time step size, and the order of the time integration method locally in space-time in a unified manner. In this sense these methods may be viewed as a class of multirate methods. These methods are computationally robust; however, the resulting space-time meshes are complicated and require sophisticated data structures for efficient implementation [13].

Domain decomposition methods (e.g., see [7, 40]) attempt to solve a problem over the entire spatial domain by iteratively solving on subdomains. There have been at least three general approaches for applying domain decomposition to parabolic problems in a space-time domain:

1. Discretize in time and solve the resulting elliptic problems with classical domain decomposition [5, 37, 34].
2. Discretize in space and apply waveform relaxation to the system of ODEs [36, 33, 32].
3. Subdivide the spatial domain and iteratively solve a sequence of PDEs defined on each subdomain [15, 18, 25].

The first approach is in some sense the most natural since one utilizes the extensive literature for domain decomposition applied to elliptic problems. This technique suffers from a couple of difficulties. First, since the elliptic problems arise after discretizing in time, we are forced to use the same time step on each subdomain. Second, the computed solution information must be exchanged at the end of each time step. This necessitates that identical time steps be used throughout the spatial domain and makes a natural parallel implementation difficult.

The second approach allows different time steps in different regions of the spatial domains. Information is transmitted between subdomains after solving a subset of ODEs over a time window, not after each step. Unfortunately, for Jacobi, Gauss–Seidel, and SOR waveform relaxation methods, in the case where the ODEs result from a spatial discretization of a PDE, the constants which arise in the error estimates depend negatively on the mesh parameter Δx. Sophisticated multigrid techniques have been proposed to overcome the negative dependence on the mesh parameter for these classical waveform relaxation methods; cf. [32, 36, 46].

The third class of methods, which have become known as Schwarz waveform relaxation methods, overcome the poor convergence of waveform relaxation methods. These methods were developed by Gander and colleagues [17, 22, 18, 20, 15] and independently by Giladi and Keller in [25]. The methods allow different numerical treatments (time step and integration formula) on different subdomains and have convergence independent of the mesh parameter, without the added complication of the multigrid framework. Since these methods provide the basis for the methods developed in the next section, we describe them in more detail below.

Consider a general parabolic problem

$$u_t = \mathcal{L}(u, x, t) \qquad \text{in } \Omega,$$

subject to appropriate initial and boundary conditions on $\partial\Omega$. In the case of two overlapping subdomains Ω_0 and Ω_1, the classical Schwarz waveform relaxation method can be succinctly written as follows: for $j = 0, 1$,

$$\frac{\partial u_j^{(k+1)}}{\partial t} = \mathcal{L}(u_j^{(k+1)}, x, t), \qquad (x,t) \in \Omega_j,$$

$$u_j^{(k+1)}(x,t) = \begin{cases} u_{1-j}^{(k)}(x,t), & (x,t) \in \Gamma_j = \partial\Omega_j \cap \Omega_{1-j}, \\ \text{given boundary condition} & x, t \in \partial\Omega_j - \Gamma_j. \end{cases}$$

There have been many theoretical advances towards understanding the convergence of Schwarz waveform relaxation. In [22] linear and superlinear convergence results are obtained for the one-dimensional heat equation. Convergence is shown to be independent of the mesh parameter (and hence robust with respect to mesh refinement), and the convergence rate improves by increasing the size of the overlap. Giladi

and Keller [25] prove superlinear convergence for constant coefficient convection diffusion equations. Gander [17, 18] extends these results to the one-dimensional, variable coefficient reaction diffusion case. Gander, Halpern, and Nataf [20] apply overlapping Schwarz waveform methods to the wave equation and a constant coefficient, linear convection reaction diffusion equation. The effect of the Dirichlet transmission conditions at the subdomain boundaries is studied and found to slow convergence of the algorithms. Optimal transmission conditions are then derived which lead to nonoverlapping Schwarz methods which converge in a finite number of steps [16]. Multidimensional extensions of many of these results are available [23, 19]. In a recent paper [15], Gander and Rohde provide results for one-dimensional convection-dominated conservation laws.

In the next section we describe the moving mesh PDE method and in section 5 we propose a new moving mesh Schwarz waveform method which inherits the favorable properties of the basic method and the spatial mesh resolution abilities of moving mesh PDEs.

4. Moving mesh methods. Adaptive mesh methods for PDEs typically fall into one or more of the following broad categories:

- r-refinement: moving a fixed number of mesh points to difficult regions of the physical domain,
- p-refinement: varying the order of the numerical method to adapt to local solution smoothness,
- h-refinement: mesh refinement and derefinement, depending upon the local level of resolution.

These methods may be applied in either a static or dynamic fashion. Static methods involve refining/coarsening or redistributing mesh points at fixed times during a calculation. Dynamic (or moving mesh) methods solve for the solution and mesh simultaneously. The latter, some of whose advantageous features will be explained subsequently, are the ones considered here.

To be specific, consider the solution of a PDE of the form

$$u_t = \mathcal{L}(u), \qquad 0 < x < 1, \quad t > 0,$$

subject to appropriate initial and boundary conditions, where again $\mathcal{L}$ denotes a spatial differential operator in the physical coordinate x. Our goal is basically to find, for fixed t, a one-to-one coordinate transformation

$$x = x(\xi, t) : [0,1] \to [0,1] \quad \text{with } x(0,t) = 0,\ x(1,t) = 1$$

such that $u(x(\xi,t),t)$ is sufficiently smooth that a simple mesh can be used to resolve solution features in the computational domain $\xi \in [0,1]$. Typically a uniform mesh,

$$\xi_i = \frac{i}{N}, \quad i = 0, 1, \ldots, N,$$

suffices, and the mesh in the physical coordinate x is then specified from the mesh transformation by $x_i(t) = x(\xi_i, t),\ i = 0, 1, \ldots, N$.

A standard way to perform mesh adaptation in space is to use the equidistribution principle (EP) [9]: Given some measure $M(t,x,u)$ of the error in the numerical solution, equidistribution requires that the mesh points satisfy

$$\int_{x_{i-1}}^{x_i} M(t,\tilde{x},u)\, d\tilde{x} \equiv \frac{1}{N} \int_0^1 M(t,\tilde{x},u) d\tilde{x}$$

on each subinterval $[x_{i-1}(t), x_i(t)]$ or

$$\int_0^{x(\xi_i,t)} M(t,\tilde{x},u)\,d\tilde{x} \equiv \xi_i \int_0^1 M(t,\tilde{x},u)d\tilde{x} \quad \text{for } i = 1,\ldots,N.$$

The continuous generalization of this is that

$$\int_0^{x(\xi,t)} M(t,\tilde{x},u)\,d\tilde{x} = \xi\theta(t), \tag{4.1}$$

where $\theta(t) \equiv \int_0^1 M(t,\tilde{x},u)\,d\tilde{x}$ (e.g., see [29]). It follows directly from (4.1) that

$$\frac{\partial}{\partial\xi}\left\{M(t,x(\xi,t),u)\frac{\partial}{\partial\xi}x(\xi,t)\right\} = 0. \tag{4.2}$$

Note that (4.2) does not explicitly involve the node speed $\dot{x}$. This is generally introduced by relaxing the equation to require equidistribution at time $t+\tau$. In [41, 29, 30] a number of parabolic moving mesh PDEs (MMPDEs) are developed using somewhat subtle simplifying assumptions and their correspondence to various moving mesh methods (previously motivated only by heuristic arguments) is shown. One particularly useful MMPDE is

$$\frac{\partial}{\partial\xi}\left(M(t,x(\xi,t),u)\frac{\partial\dot{x}}{\partial\xi}\right) = -\frac{1}{\tau}\frac{\partial}{\partial\xi}\left(M(t,x(\xi,t),u)\frac{\partial x}{\partial\xi}\right). \tag{MMPDE4}$$

The relaxation parameter τ is chosen in practice so that the mesh evolves at a rate commensurate with that of the solution $u(x,t)$.

A simple popular choice for $M(t,x,u)$ is the arclength-like monitor function [3, 45]

$$M(x,u,t) = \sqrt{1+\frac{1}{\alpha}|u_x|^2}. \tag{4.3}$$

This choice is based on the premise that we expect the error in the numerical solution to be largest in regions where the solution has large gradients. The choice of monitor function is often problem class dependent; e.g., monitor functions which preserve the scaling invariance of the physical PDE [4] are seen to work well for reaction diffusion problems having blow-up solutions (as opposed to the arclength monitor function, for instance, which ultimately fails to resolve the solution near the singularity in time).

As outlined in the previous section, using a moving mesh method to solve a PDE requires solving a coupled system of PDEs for the unknown mesh transformation and solution. Using the mesh transformation $x = x(\xi,t)$ to rewrite the physical PDE in quasi-Lagrangian form, we get

$$\dot{u} - u_x\dot{x} = \mathcal{L}u, \tag{4.4}$$

where the total time derivative $\dot{u}$ is given as

$$\dot{u} = u_t + u_x\dot{x}.$$

Equation (4.4) and a moving mesh equation are solved simultaneously for the mesh $x(\xi,t)$ and corresponding solution $u(x(\xi,t),t)$. Specifically, we solve the coupled PDE system

$$\begin{aligned} \dot{u} - u_x\dot{x} &= \mathcal{L}(u), \\ \left(M(x,u,t)\dot{x}_\xi\right)_\xi &= -\frac{1}{\tau}\left(M(x,u,t)x_\xi\right)_\xi \end{aligned} \tag{4.5}$$

using the moving MOL approach. Discretizing spatially with centered differences we obtain the semidiscrete approximation

(4.6)
$$\dot{u}_i - \left(\frac{u_{i+1}-u_{i-1}}{x_{i+1}-x_{i-1}}\right)\dot{x}_i = f_i,$$
$$\frac{M_{i+1}+M_i}{2(1/N)^2}\left(\dot{x}_{i+1}-\dot{x}_i\right) - \frac{M_i+M_{i-1}}{2(1/N)^2}\left(\dot{x}_i-\dot{x}_{i-1}\right) =$$
$$-\frac{1}{\tau}\left\{\frac{M_{i+1}+M_i}{2(1/N)^2}\left(x_{i+1}-x_i\right) - \frac{M_i+M_{i-1}}{2(1/N)^2}\left(x_i-x_{i-1}\right)\right\}$$

for $i = 1, \ldots, N-1$. The right-hand side of the semidiscrete physical PDE, f_i, denotes the discrete approximation to the spatial operator $\mathcal{L}$ at $\xi = \xi_i$. Centered differences are used to discretize $\mathcal{L}$. The quantity M_i denotes a centered difference approximation to the monitor function at x_i. In actual calculations, M_i is replaced by a smoothed value

$$\tilde{M}_i = \sqrt{\sum_{k=i-p}^{i+p}(M_k)^2\left(\frac{\gamma}{\gamma+1}\right)^{|k-i|} \Bigg/ \sum_{k=i-p}^{i+p}\left(\frac{\gamma}{\gamma+1}\right)^{|k-i|}},$$

where $\gamma = 2$ and $p = 2$ [29].

Initial and boundary conditions for the physical PDE come from the problem description. On a fixed interval we specify $\dot{x}_0 = \dot{x}_N = 0$ as boundary conditions for the mesh. If the initial solution $u(x,0)$ is smooth, then an initial uniform mesh for $x(\xi,0)$ is normally sufficient. If the initial solution has features which are not resolved on a uniform mesh for the chosen value of N, then an equidistributed initial mesh is required. One way used in [31] to compute an initial equidistributed mesh is to solve the PDE

$$u_t = u_0(x), \quad u(x,0) = 0,$$

coupled with a moving mesh equation. Solving this system over the time interval $0 \le t \le 1$ with a uniform initial mesh will yield the solution $u(x,1) = u_0(x)$, and the resulting mesh will approximately equidistribute the initial solution. To avoid propagating any errors from this calculation, the initial solution is reevaluated at the new initial mesh.

The semidiscrete approximation (4.6) describes a system of linearly implicit ODEs which may be written in the form

$$V(y)y' = g(y). \tag{4.7}$$

The vector y denotes the unknowns ordered as $(u_0, x_0, u_1, x_1, \ldots, u_N, x_N)^T$, and $V(y)$ is a banded matrix which depends on the unknown solution and mesh.

A typical implementation (cf. [31]) for solving (4.5) involves discretizing spatially and solving (4.7) with a stiff ODE solver such as DASSL [39]. In the next section we propose a Schwarz waveform moving mesh method to solve (4.5) and describe its implementation.

5. Schwarz waveform moving mesh method. In this section we propose a new Schwarz waveform moving mesh (or simply "moving Schwarz") method. The basic idea is to apply Schwarz waveform relaxation, as described in section 3, to the coupled system of physical and moving mesh PDEs (4.5). Specifically, we first decompose $\Omega = [0,1]$ into D nonoverlapping fixed subdomains $\tilde{\Omega}_j$, $j = 1, \ldots, D$. Each subdomain $\tilde{\Omega}_j$ is enlarged by an overlap region consisting of R mesh points, giving overlapping domains $\Omega_1, \Omega_2, \ldots, \Omega_D$. The physical PDE is then discretized along with a moving mesh equation on each subdomain. We fix the mesh points on the boundary of $\tilde{\Omega}_j$, $j = 1, \ldots, D$, but allow the mesh points to move within and on the outer boundary of the overlap regions. Algorithmically, there is no reason to keep the boundaries fixed. However, there is a tendency for moving mesh methods to emphasize the most pathological regions of the solution. Allowing the subdomain boundaries to move could potentially negate the effort to isolate difficult regions within individual subdomains. We will return to this point in the next section.

The convergence of overlapping classical Schwarz waveform relaxation is known to depend on the width of the overlap region. In this implementation we have found that having the width of the overlap in the computational domain of moderate size works well, even for somewhat pathological solution behavior in the physical domain where the width of the overlap in the physical variables varies (but the number of mesh points in this region is fixed). A priori determining an optimal width of the overlap region in the computational domain would be difficult. As in the fixed mesh case, the rate of convergence of the Schwarz iteration is improved as the size of the overlap is increased, with the faster convergence being offset by the increased computational cost. Things are further complicated by the desire to isolate difficult regions of the solution from regions where there is little activity. As the overlap is increased more subdomains become "active."

Let x_j and ξ_j denote the physical and computational meshes on each subdomain Ω_j, and denote the solution on each subdomain by u_j. The moving Schwarz method requires solving

$$\dot{u}_j^k - \frac{\partial u_j^k}{\partial x}\dot{x}_j^k = \mathcal{L}(u_j^k),$$
$$\left(M(x_j^k, u_j^k, t)\frac{\partial \dot{x}_j^k}{\partial \xi}\right)_\xi = -\frac{1}{\tau}\left(M(x_j^k, u_j^k, t)\frac{\partial x_j^k}{\partial \xi}\right)_\xi,$$

on Ω_j with $j = 1, \ldots, D$, for $k = 1, 2, \ldots$.

The boundary values for u_j^k and x_j^k come from the values $u_{j-1}^{k-1}, x_{j-1}^{k-1}$ and $u_{j+1}^{k-1}, x_{j+1}^{k-1}$ on the left and right boundaries of Ω_j, respectively, from the previous iteration. Each Schwarz waveform iteration requires the solution of D problems with (known) moving boundaries. This is illustrated in Figure 5.1 for $D = 3$.

On each subdomain the coupled system of PDEs is discretized in space as described in section 4 and the resulting ODEs are integrated over successive time windows. The time dependent boundary conditions for u_j and x_j on these time windows are obtained by cubic spline interpolation of the data from the previous iteration. Interpolation is required since in general the data from iteration $k-1$ is not computed on the same sequence of time steps as iteration k.

For efficiency, the algorithm is designed with an adaptive time windowing strategy. The algorithm begins by solving the sequence of moving boundary problems on a time window $[0, T]$. If u_j or x_j fails to reach $t = T$ in the maximum allowable time

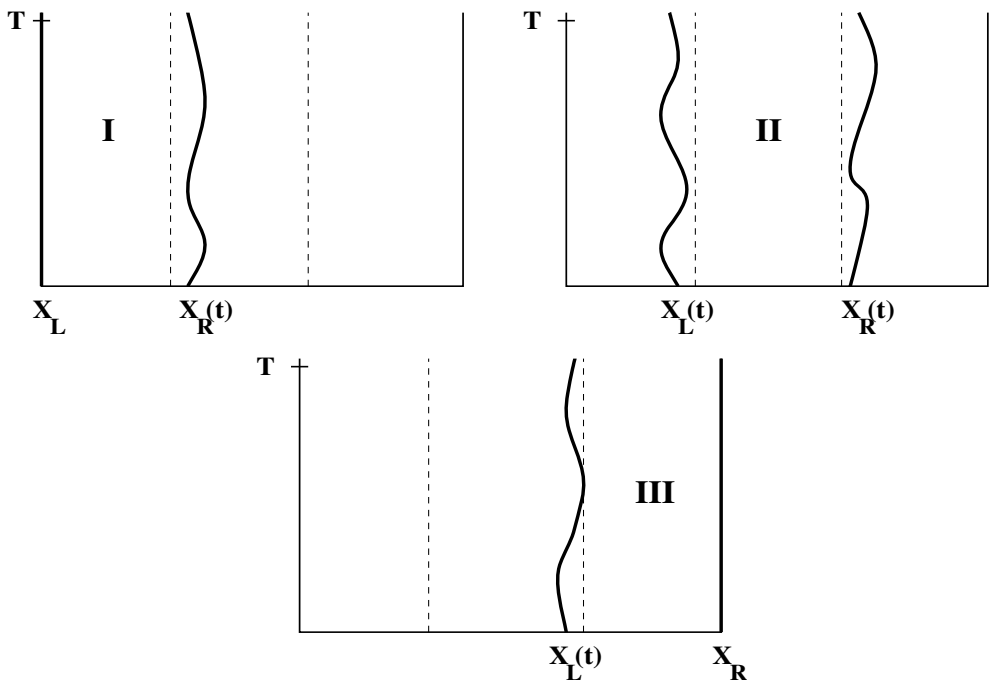


FIG. 5.1. *Sequence of moving boundary problems solved during one iteration of the Schwarz waveform moving mesh method over a time window* $[0, T]$.

steps, the iteration is restarted with a smaller time window. The window is also reduced if the Schwarz iteration fails to converge in 6 iterations and enlarged if the iteration converges in fewer than 4 iterations. These values, chosen on the basis of significant experimentation [27], provide a balance between efficiency and storage. The latter can be significant even in the one-dimensional case because in the absence of a guarantee that the maximum principle holds, we conservatively require successive approximations over subdomains to agree to within some user specified tolerance δ before the waveform iteration is terminated. A tolerance of $\delta = 1e-4$ is used in the experiments with local error control tolerance (for the time integration) of $1e-6$.

As a final note, the code provides each subdomain with an initial equidistributed mesh. That is, we solve $u_t = u_0(x)$ for $x \in \tilde{\Omega}_j$ and a moving mesh PDE subject to $u(x,0) = 0$. This provides a grid which equidistributes the initial solution to the PDE over each subdomain.

6. Numerical results. To provide some initial testing of the moving Schwarz method we first solve a straightforward test problem.

1: I Our first example is a standard test problem for adaptive methods, Burgers' equation. Specifically, we solve the one-dimensional viscous Burgers' equation

$$u_t = \epsilon u_{xx} - \frac{1}{2}(u^2)_x,$$

$$u(0,t) = 1, \quad u(1,t) = 0,$$

$$u(x,0) = c - \frac{1}{2}\tanh\left(\frac{(x-x_0)}{4\epsilon}\right),$$

where $c = 1/2$, $x_0 = 1/4$ or $1/10$, and $\epsilon \ll 1$. The solution is a traveling front of thickness $O(\epsilon)$ which moves to the right from x_0 at speed c. The initial and boundary conditions are not compatible at $t = 0$ and $x = 0, 1$, but the discrepancy is small for small ϵ. This does not pose any difficulties with the numerics.

We take $\epsilon = 1e - 4$ and use 40 mesh points in each of the three equal sized subdomains. The top row of plots in Figure 6.1 illustrates the computed solution at times $t = 0.25, 0.45$, and 1.7. The bottom plots show the corresponding pointwise error $|\tilde{u}(x_i, t^*) - u(x_i, t^*)|$, where $\tilde{u}$ is the numerical approximation to the exact solution u obtained using the Schwarz waveform moving mesh method. As expected we see a sharp front moving from left to right. The mesh trajectories as functions of time are displayed in Figure 6.2.

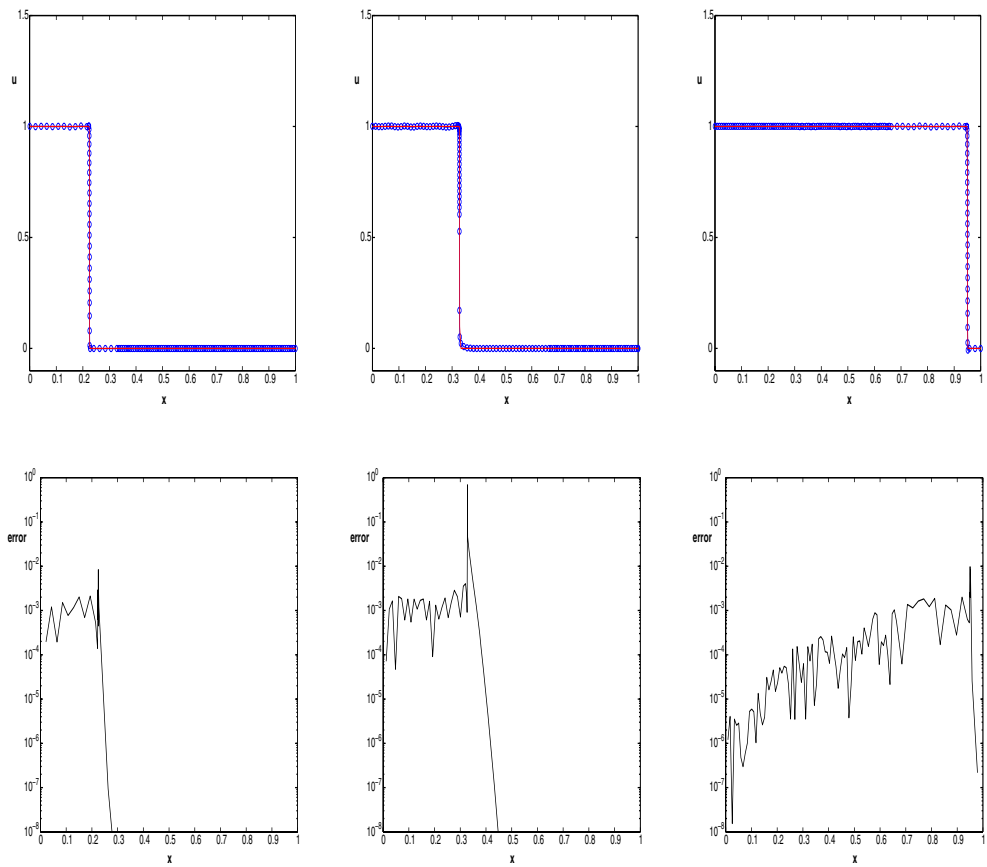


FIG. 6.1. *Solutions and pointwise errors for Burgers' equation with the Schwarz waveform moving mesh method at* $t = 0.25$ *(left),* $t = 0.45$ *(middle), and* $t = 1.7$ *(right).*

At $t = 0.25$ and $t = 1.7$, the actual errors in the solution are on the order of the tolerance controlling the convergence of the Schwarz waveform iteration. At $t = 0.45$, however, the front encounters the boundary between the first and second subdomains, and the level of communication between the solutions in these adjacent subdomains is inadequate to preserve the accuracy. A possible fix may be provided by some of the more sophisticated Schwarz waveform methods which use "higher-order" transmission conditions at the boundary [20, 21].

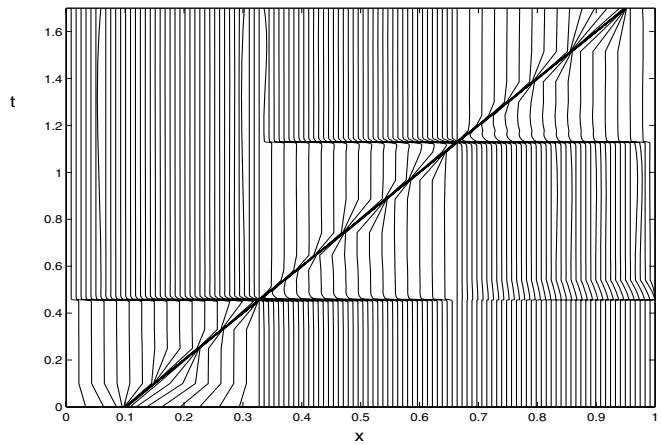


FIG. 6.2. *Mesh trajectories for Burgers' equation with* $\epsilon = 1e-4$ *and* 40 *points per domain using the Schwarz waveform moving mesh method.*

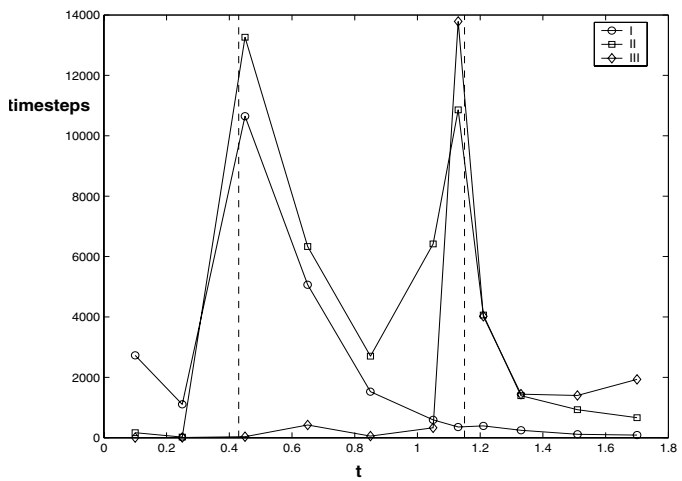


FIG. 6.3. *Number of time steps (per time window) in each subdomain.*

Both the tremendous potential and current difficulties with our basic Schwarz waveform moving mesh method are highlighted in Figure 6.3. In this plot we display the number of time steps per time window taken by implicit Euler in each subdomain, labeled I, II, and III in the legend of the plot. The dotted vertical lines specify the time at which the layer crosses the subdomain boundaries. The data shows that while the layer is completely contained in subdomain I ($0 \leq t \leq 0.4$), the work involved to integrate over subdomains II and III is negligible. As the front approaches the subdomain boundaries, the number of time steps in the adjoining domains increases dramatically, indicating the standard difficulty experienced by the classical Schwarz waveform iteration.

Another indicator of the performance of the classical Schwarz method is illustrated in Figure 6.4. The maximum number of time steps per time window is fixed, so the

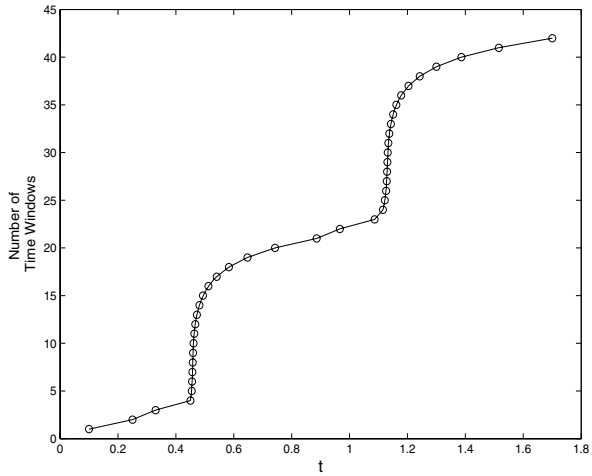


FIG. 6.4. *Number of time windows used up to time t for the Schwarz waveform moving mesh method applied to Burgers' equation with* $\epsilon = 1e-4$.

size of the time window varies over the duration of the computation. There is a dramatic decrease in the size of the time window as the moving front moves from one subregion to the next. The time window is increased or decreased based on the convergence of the Schwarz iteration. If the iteration has not converged within a user specified number of iterations, the length of the time window is decreased. We see in Figure 6.4 that the length of the time window is shortest when the front moves from one subdomain to another.

[illegible] 2: [illegible] II We now solve Burgers' equation subject to the initial condition

$$u(x,0) = \sin(2\pi x) + \frac{1}{2}\sin(\pi x)$$

and boundary conditions $u(0,t) = u(1,t) = 0$, using three overlapping subdomains. The solution develops a sharp front and moves to the right with diminishing amplitude. Mesh trajectories for this initial condition are shown in Figure 6.5. In this case, the solution evolves to a front in subdomain II. This results in a nearly uniform mesh in subdomain I for the entire computation. The grids in subdomains II and III, however, react to the evolving solution and carry the front to the boundary at $x = 1$. As with the previous example, the convergence of the Schwarz iteration deteriorates as the front moves from subdomain II to subdomain III.

[illegible] 3: [illegible] The next problem is

$$u_t = \epsilon u_{xx} + f(x,t),$$

where $f(x,t)$ is defined such that the solution has two spikes and is given by

$$u(x,t) = \frac{1}{\epsilon}\left(\max(0,\tanh(t-t_0))e^{-(x-x_0)^2/\epsilon} + \max(0,\tanh(t-t_1))e^{-(x-x_1)^2/\epsilon}\right).$$

These two spikes are centered at fixed locations x_0 and x_1 and begin to evolve at different times t_0 and t_1. We choose x_0 and x_1 to keep the spikes well separated in the spatial domain and choose $t_1 > t_0$. This allows the first spike to grow to its

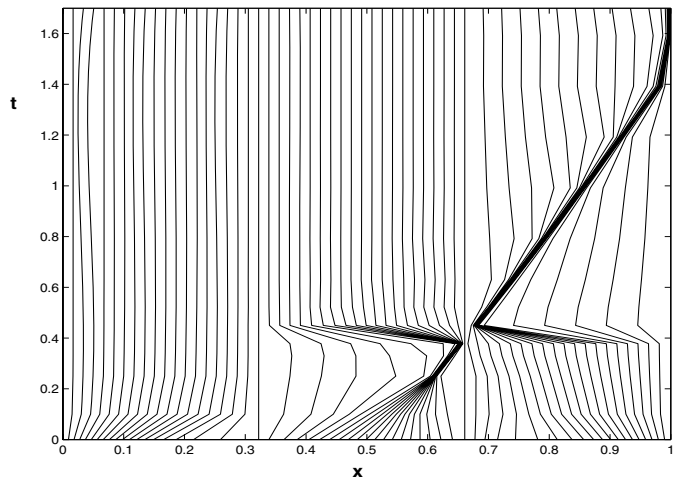


FIG. 6.5. *Mesh trajectories for Burgers' equation with* $\epsilon = 1e-3$ *and* 20 *points per domain using the Schwarz waveform moving mesh method.*

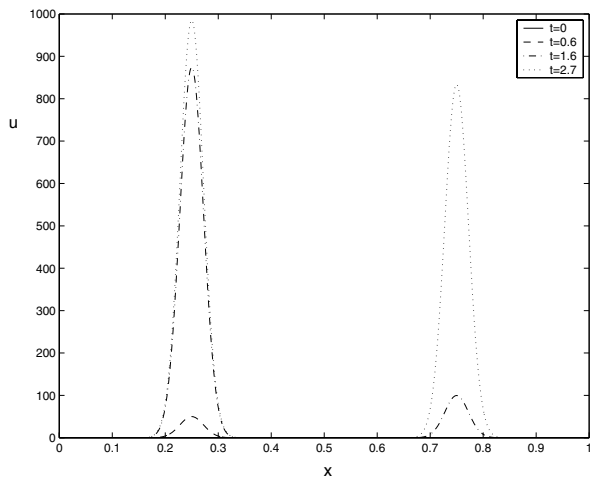


FIG. 6.6. *Exact solutions of the two-spike problem at* $t = 0, 0.6, 1.6,$ *and* 2.7.

maximum height before the second spike appears. All computations are done with $\epsilon = 1e-3$. Figure 6.6 illustrates the exact solution of the two-spike problem for times $t = 0.6$, 1.6, and 2.7. (At $t = 0$, $u(x, 0) \equiv 0$.) At $t = 0.6$ the spike centered at $x_0 = 1/4$ has appeared and continues to grow. The second spike centered at $x_1 = 3/4$ emerges at $t = 3/2$. At $t = 2.7$, both spikes have nearly reached their maximum height. This problem is chosen to show the difficulties which can occur for moving mesh methods when the solution to the PDE behaves badly in several regions, and how the Schwarz waveform moving mesh method can be used to ameliorate the situation. Solving the problem with 80 mesh points on one domain, the final solution at $t = 2.7$ and the mesh trajectories on $[0, 1]$ are shown in Figure 6.7. The computed solution is illustrated by a line with open circles indicating the location of the mesh points, and the exact solution is depicted with a thick solid line. The moving mesh method does indeed

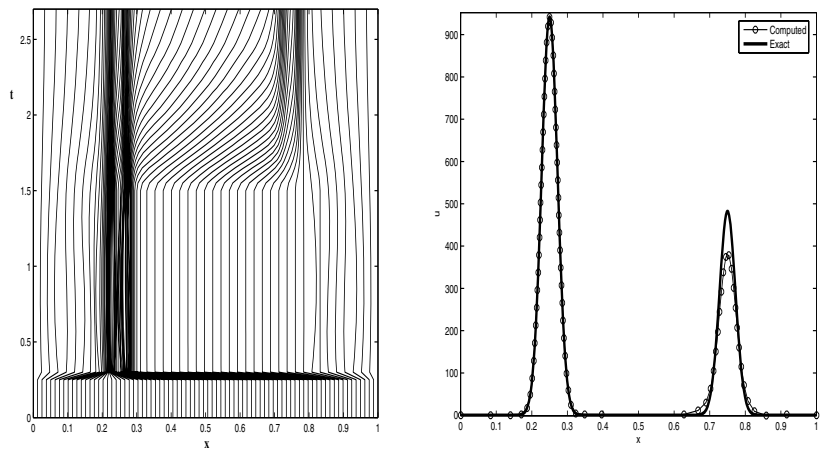


FIG. 6.7. *Mesh trajectories (left) and solutions (right) of the two-spike problem with one domain using the arclength monitor function.*

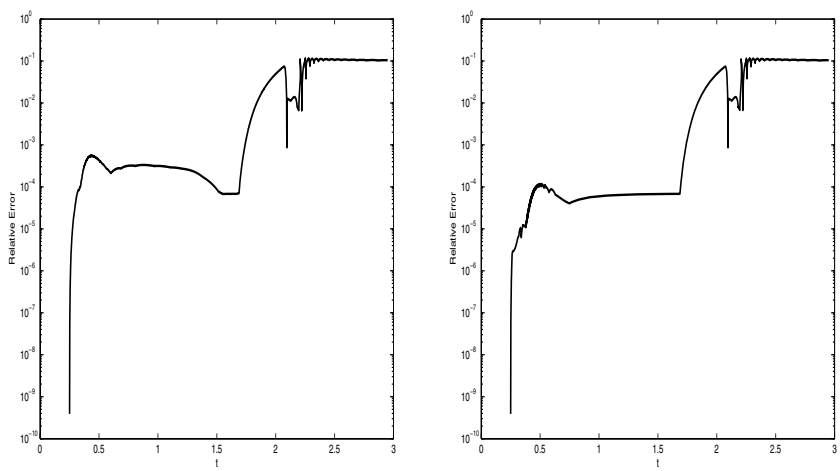


FIG. 6.8. *Relative error in the computed solution on one domain using the arclength monitor function and backward Euler (left) and variable order BDF formulation (right).*

capture both spikes; however, a loss of accuracy in the second spike is evident by the end of the computation. In fact, substantial errors arise in the computed solution for $t \geq 3/2$, corresponding to the birth of the second spike. Indeed, a moving mesh method on one domain has great difficulty achieving sufficient resolution of the second spike. The mesh trajectories indicate a relatively quick movement of mesh points from the region of the left spike towards the emerging right spike for $t > 3/2$.

Figure 6.8 illustrates the relative error in the computed solution for the two-spike problem on one domain. The plot on the left was obtained using implicit Euler to integrate forward in time, while the plot on the right was obtained using a variable order BDF implementation. The results are nearly identical. The loss of accuracy in the computed solutions arises with the birth of the second spike at $t = 1.5$. In fact, a

relatively large number of mesh points is vital to allow resolution of the second spike.

The loss of accuracy which occurs as the second spike grows is due, in part, to the time required for the mesh to adapt to new features in the solution. This may be controlled by the moving mesh parameter τ in MMPDE4. Decreasing the value of τ facilitates a quicker mesh movement and hence a quicker return to an equidistributed grid. However, this improves the accuracy of the computed solution at the expense of necessitating much smaller steps in the time integration. The situation may be improved by controlling the proportion of points outside the initial region of difficulty by an appropriate choice of α in (4.3).

This problem is ideal for a two-domain simulation with the Schwarz waveform moving mesh method. The computed and exact solutions and resulting mesh trajectories are shown in Figure 6.9.

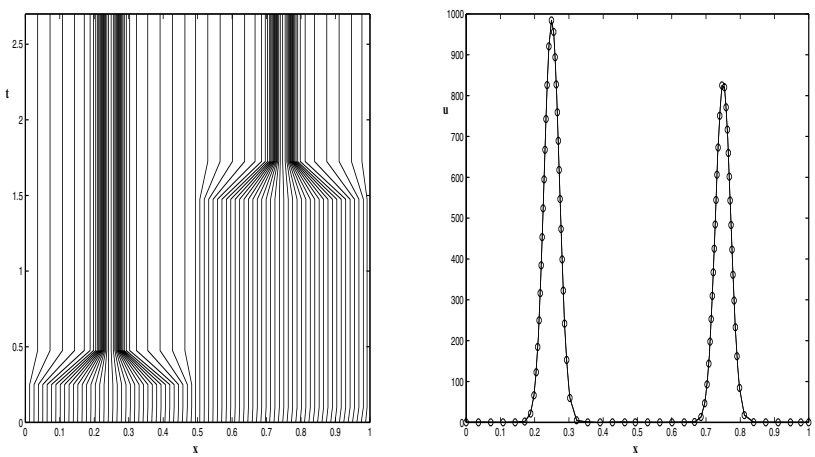


FIG. 6.9. *Mesh trajectories (left), computed and exact solutions (right) for the two-spike problem on two subdomains.*

Compared to the one-domain calculation, we see improved accuracy between the spikes and in regions of high curvature. As expected, the mesh remains uniform in each subdomain until the spike in that region is activated. At that point the mesh in the subdomain adapts well to the evolving features of the solution using a simple arclength monitor function and moderate values of τ. In fact, comparable accuracy to that for the one-domain calculation is possible with many fewer mesh points.

Figure 6.10 compares the time steps taken by the ODE solver (backward Euler) for the one-domain (left) and two-subdomain (right) simulations. In the two-subdomain case we illustrate the time steps for subdomain I (solid) and subdomain II (dashed) corresponding to the first waveform iteration. The time steps chosen for the one-domain calculation are quite large until $t = 1/4$, when the left spike begins to grow. The time steps remain relatively steady at 10^{-4} until the second spike emerges at $t = 3/2$. Immediately the time steps are reduced by an order of magnitude as the mesh points race to adjust to the emerging features in the solution.

The time steps chosen in each subdomain by the Schwarz waveform moving mesh method are controlled primarily by the local features of the solution. During the solution on subdomain I the time steps reduce to approximately 10^{-4} at $t = 1/4$. This time corresponds to the emergence of the first spike. The time steps persist at

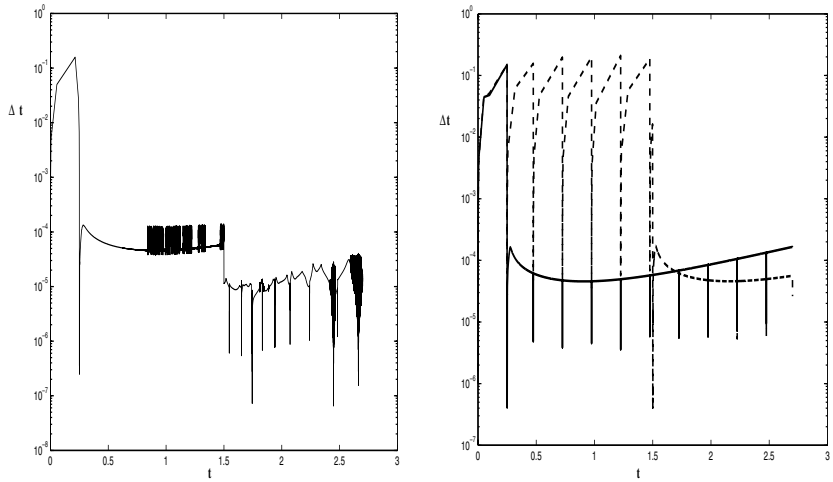


FIG. 6.10. *Time steps for the one-domain (left) and two-subdomain (right) solution of the two-spike problem.*

this level for the duration of the simulation. Little work is required to integrate the solution and mesh components in subdomain II until $t = 3/2$. At that time, the size of the acceptable time steps decreases to a level commensurate with that on subdomain I. It is important to note that the time steps in subdomain I are not affected by the development of the second spike. No mesh points move from one subdomain to another. This keeps the time steps an order of magnitude larger than for the one-domain calculation. This more than compensates for the cost of the iteration required in the Schwarz waveform moving mesh method, resulting in approximately half the run time.

7. Conclusions. The moving method of lines has proved to be an effective tool for solving a number of time dependent PDEs [31]. In this paper we propose a modification of the basic procedure—a Schwarz waveform moving mesh method. The moving mesh PDE and physical PDE are solved by a domain decomposition algorithm. The computational space-time domain is decomposed into a number of overlapping subdomains. The solution is computed iteratively by solving the system of PDEs on each subdomain with information exchange achieved through Dirichlet conditions at the subdomain boundaries. Problems with localized fast-changing solution components are shown to benefit from this approach.

Traditionally, the solution of the resulting system of ODEs in time has been obtained using standard implementations of stiff ODE system solvers with a fully implicit time stepping package, such as in [31] where DASSL [39] is used.

In the simple implementation of the Schwarz waveform method investigated here, we solve (4.7) using a backward Euler method,

$$V(y_n)(y_n - y_{n-1}) - h_n g(y_n) = 0,$$

and a crude time-stepping procedure where the local error is controlled by step doubling/halving. The nonlinear algebraic systems are solved with a modified Newton method, using the strategies of Gustafsson and Söderlind [26] and Alexander [1] to control factorizations and evaluations of the Jacobian matrix. The Jacobian is evaluated using finite differences adapted to sparse matrices [8].

There are many important aspects to consider when investigating the feasibility of the Schwarz waveform method. Classical Schwarz iteration is known to suffer from slow convergence, particularly when the solution components have substantially different time scales. The moving mesh method is specifically designed so that solution components are closer to having the same time scale than for the standard MOL approach. It remains an issue, although we have seen marked improvement in numerical results. The slow convergence can be alleviated through the use of higher-order transmission conditions at the subdomain boundaries [20]. Furthermore, to avoid fast-changing solution components crossing subdomain boundaries, a coarse grain adaptivity may be employed as well, where one dynamically determines (spatially) the number and placement of subdomains. As a front approaches a subdomain boundary, for example, domains may be collapsed or merged until the front has passed. In our implementation, the Schwarz iteration proceeds until the solutions on all subdomains agree to within some specified tolerance. This requires storage of the solution at all points in the computational space. When a maximum principle holds, it is possible to save substantially on storage by keeping boundary data only to test for convergence. Further study is certainly required to understand the role of overlap for Schwarz waveform moving mesh methods.

The issues of efficiency are even more important in higher spatial dimensions, where fast solution behavior locally comes at a much higher cost if adaptivity is not done properly or system solutions not done efficiently; cf. [47]. One major reason that moving mesh methods are ideally suited to handle certain problems with singularities is that, by choosing a suitable monitor function to incorporate scaling as proposed by Beckett and Mackenzie [3], a proper balance between adaptivity done in the inner and outer solution regions is assured. Unfortunately, for problems with multiple scales or multiple regions with fast scale behavior, using moving mesh methods alone for adaptivity can be problematic in determining the relative importance of the regions of fast solution activity. While a simple implementation of the Schwarz waveform method is not a panacea in and of itself, it can be the ideal tool for overcoming many of these difficulties. A simple Schwarz waveform relaxation has been applied directly for the mesh equation for a two-dimensional problem in [6]. We intend to investigate a more sophisticated implementation of the Schwarz waveform method, incorporating the features discussed here, in future work. An ultimate test of the method would entail a careful comparison with the other approaches of section 3.

REFERENCES

[1] R. Alexander, *The modified Newton method in the solution of stiff ordinary differential equations*, Math. Comp., 57 (1991), pp. 673–701.

[2] J. F. Andrus, *Numerical solution of systems of ordinary differential equations separated into subsystems*, SIAM J. Numer. Anal., 16 (1979), pp. 605–611.

[3] G. Beckett and J. A. Mackenzie, *Uniformly convergent high order finite element solutions of a singularly perturbed reaction-diffusion equation using mesh equidistribution*, Appl. Numer. Math., 39 (2001), pp. 31–45.

[4] C. J. Budd, W. Huang, and R. D. Russell, *Moving mesh methods for problems with blow-up*, SIAM J. Sci. Comput., 17 (1996), pp. 305–327.

[5] X. Cai, *Additive Schwarz algorithms for parabolic convection-diffusion equations*, Numer. Math., 60 (1991), pp. 41–61.

[6] W. Cao, W. Huang, and R. D. Russell, *A moving mesh method in multiblock domains with application to a combustion problem*, Numer. Methods Partial Differential Equations, 15 (1999), pp. 449–467.

[7] T. F. Chan and T. P. Mathew, *Domain decomposition algorithms*, in Acta Numerica, 1994, Acta Numer., Cambridge University Press, Cambridge, UK, 1994, pp. 61–143.
[8] A. Curtis, M. Powell, and J. Reid, *On the estimation of sparse Jacobian matrices*, IMA J. Appl. Math., 13 (1974), pp. 117–119.
[9] C. de Boor, *Good approximation by splines with variable knots.* II, in Conference on the Numerical Solution of Differential Equations (Dundee, 1973), Lecture Notes in Math. 363, Springer, Berlin, 1974, pp. 12–20.
[10] P. Deuflhard and J. Heroth, *Dynamic dimension reduction in ODE models*, in Scientific Computing in Chemical Engineering, F. Keil, W. Mackens, H. Voß, and J. Werther, eds., Springer-Verlag, Berlin, 1996, pp. 29–43.
[11] C. Engstler and C. Lubich, *Multirate extrapolation methods for differential equations with different time scales*, Computing, 58 (1997), pp. 173–185.
[12] C. Engstler and C. Lubich, *MUR8: A multirate extension of the eighth-order Dormand-Prince method*, Appl. Numer. Math., 25 (1997), pp. 185–192.
[13] J. Flaherty, R. Loy, M. Shephard, B. Szymanski, J. Teresco, and L. Ziantz, *Adaptive local refinement with octree load-balancing for the parallel solution of three-dimensional conservation laws*, J. Parallel Distrib. Comput., 47 (1997), pp. 139–152.
[14] J. E. Flaherty and P. K. Moore, *Integrated space-time adaptive hp-refinement methods for parabolic systems*, Appl. Numer. Math., 16 (1995), pp. 317–341.
[15] M. J. Gander and C. Rohde, *Overlapping Schwarz waveform relaxation for convection-dominated nonlinear conservation laws*, SIAM J. Sci. Comput., 27 (2005), pp. 415–439.
[16] M. J. Gander, *Optimized Schwarz methods*, SIAM J. Numer. Anal., 44 (2006), pp. 699–731.
[17] M. J. Gander, *Overlapping Schwarz waveform relaxation for parabolic problems*, in Tenth International Conference on Domain Decomposition Methods, Contemp. Math. 218, J. Mandel, C. Farhat, and X.-C. Cai, eds., AMS, Providence, RI, 1998, pp. 425–431.
[18] M. J. Gander, *A waveform relaxation algorithm with overlapping splitting for reaction diffusion equations*, Numer. Linear Algebra Appl., (1998), pp. 125–145.
[19] M. J. Gander and D. Daoud, *Overlapping Schwarz waveform relaxation for convection reaction diffusion problems*, in Thirteenth International Conference on Domain Decomposition Methods, N. Debit, M. Garbey, R. Hoppe, J. Périaux, D. Keyes, and Y. Kuznetsov, eds., Domain Decomposition Press, Bergen, Norway, 2002, pp. 227–233.
[20] M. J. Gander, L. Halpern, and F. Nataf, *Optimal convergence for overlapping and non-overlapping Schwarz waveform relaxation*, in Eleventh International Conference on Domain Decomposition Methods, C.-H. Lai, P. Bjørstad, M. Cross, and O. Widlund, eds., ddm.org, Augsburg, Germany, 1999, pp. 27–36.
[21] M. J. Gander, L. Halpern, and F. Nataf, *Optimized Schwarz methods*, in Twelfth International Conference on Domain Decomposition Methods (Chiba, Japan), T. Chan, T. Kako, H. Kawarada, and O. Pironneau, eds., Domain Decomposition Press, Bergen, Norway, 2001, pp. 15–28.
[22] M. J. Gander and A. M. Stuart, *Space-time continuous analysis of waveform relaxation for the heat equation*, SIAM J. Sci. Comput., 19 (1998), pp. 2014–2031.
[23] M. J. Gander and H. Zhao, *Overlapping Schwarz waveform relaxation for the heat equation in n dimensions*, BIT, 42 (2002), pp. 779–795.
[24] C. W. Gear and D. R. Wells, *Multirate linear multistep methods*, BIT, 24 (1984), pp. 484–502.
[25] E. Giladi and H. B. Keller, *Space-time domain decomposition for parabolic problems*, Numer. Math., 93 (2002), pp. 279–313.
[26] K. Gustafsson and G. Söderlind, *Control strategies for the iterative solution of nonlinear equations in ODE solvers*, SIAM J. Sci. Comput., 18 (1997), pp. 23–40.
[27] R. D. Haynes, *The Numerical Solution of Differential Equations: Grid Selection for Boundary Value Problems and Adaptive Time Integration Strategies*, Ph.D. thesis, Simon Fraser University, Burnaby, BC, 2003.
[28] E. Hofer, *A partially implicit method for large stiff systems of ODEs with only few equations introducing small time-constants*, SIAM J. Numer. Anal., 13 (1976), pp. 645–663.
[29] W. Huang, Y. Ren, and R. D. Russell, *Moving mesh methods based on moving mesh partial differential equations*, J. Comput. Phys., 113 (1994), pp. 279–290.
[30] W. Huang, Y. Ren, and R. D. Russell, *Moving mesh partial differential equations (MMPDEs) based on the equidistribution principle*, SIAM J. Numer. Anal., 31 (1994), pp. 709–730.
[31] W. Huang and R. D. Russell, *A moving collocation method for solving time dependent partial differential equations*, Appl. Numer. Math., 20 (1996), pp. 101–116.
[32] J. Janssen and S. Vandewalle, *Multigrid waveform relaxation on spatial finite element meshes: The continuous-time case*, SIAM J. Numer. Anal., 33 (1996), pp. 456–474.

[33] R. Jeltsch and B. Pohl, *Waveform relaxation with overlapping splittings*, SIAM J. Sci. Comput., 16 (1995), pp. 40–49.

[34] Y. Kuznetsov, *Domain decomposition methods for unsteady convection-diffusion problems*, in Computing Methods in Applied Sciences and Engineering (Paris, 1990), SIAM, Philadelphia, 1990, pp. 211–227.

[35] E. Lelarasmee, A. Ruehli, and A. Sangiovanni-Vincentelli, *The waveform relaxation method for time-domain analysis of large scale integrated circuits*, IEEE Trans. CAD of Integrated Circuits Syst., 1 (1982), pp. 131–145.

[36] C. Lubich and A. Ostermann, *Multigrid dynamic iteration for parabolic equations*, BIT, 27 (1987), pp. 216–234.

[37] G. A. Meurant, *A domain decomposition method for parabolic problems*, Appl. Numer. Math., 8 (1991), pp. 427–441.

[38] U. Miekkala and O. Nevanlinna, *Convergence of dynamic iteration methods for initial value problems*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 459–482.

[39] L. R. Petzold, *A description of DASSL: A differential/algebraic system solver*, in Scientific Computing (Montreal, 1982), IMACS Trans. Sci. Comput. I, IMACS, New Brunswick, NJ, 1983, pp. 65–68.

[40] A. Quarteroni and A. Valli, *Domain Decomposition Methods for Partial Differential Equations*, Numer. Math. Sci. Comput., Oxford Science Publications, The Clarendon Press, Oxford University Press, New York, 1999.

[41] Y. Ren and R. D. Russell, *Moving mesh techniques based upon equidistribution, and their stability*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 1265–1286.

[42] J. R. Rice, *Split Runge-Kutta method for simultaneous equations*, J. Res. Nat. Bur. Standards Sect. B, 64B (1960), pp. 151–170.

[43] S. Skelboe, *Stability properties of backward differentiation multirate formulas. Recent theoretical results in numerical ordinary differential equations*, Appl. Numer. Math., 5 (1989), pp. 151–160.

[44] S. Skelboe and P. U. Andersen, *Stability properties of backward Euler multirate formulas*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 1000–1009.

[45] J. M. Stockie, J. A. Mackenzie, and R. D. Russell, *A moving mesh method for one-dimensional hyperbolic conservation laws*, SIAM J. Sci. Comput., 22 (2001), pp. 1791–1813.

[46] S. Ta'asan and H. Zhang, *On the multigrid waveform relaxation method*, SIAM J. Sci. Comput., 16 (1995), pp. 1092–1104.

[47] A. Toselli and O. Widlund, *Domain Decomposition Methods—Algorithms and Theory*, Springer Ser. Comput. Math. 34, Springer-Verlag, Berlin, 2005.

SIAM J. SCI. COMPUT.
Vol. 29, No. 2, pp. 674–709
© 2007 Society for Industrial and Applied Mathematics

MUSIC-TYPE ELECTROMAGNETIC IMAGING OF A COLLECTION OF SMALL THREE-DIMENSIONAL INCLUSIONS*

HABIB AMMARI†, EKATERINA IAKOVLEVA‡, DOMINIQUE LESSELIER‡, AND GAËLE PERRUSSON‡

Abstract. In this paper we consider the localization of a collection of small, three-dimensional bounded homogeneous inclusions via time-harmonic electromagnetic means, typically using arrays of electric or magnetic dipole transmitters and receivers with given polarization(s) at some distance from the collection, possibly also lying in the far field. The inclusions, somewhat apart or closely spaced, are buried within a homogeneous medium, and are of arbitrary contrast of permittivity, conductivity, and permeability vis-à-vis this embedding medium. The problem is formulated as an inverse scattering problem for the full Maxwell equations and it involves a robust asymptotic modeling of the multistatic response matrix. No specific application is studied at this stage, but characterization of obstacles in subsoils, nondestructive evaluation of man-made structures, and medical imaging are primary fields of application envisaged. The proposed approach uses a MUSIC (multiple signal classification)-type algorithm, and it yields fast numbering, accurate localization, and estimates of the electromagnetic and geometric parameters (polarization tensors) of the inclusions. The mathematical machinery is detailed first, some specific attention being given to triaxial ellipsoidal inclusions and degenerate spherical shapes (for the latter known results are retrieved). Then, the viability of this algorithm—which would be easily extended to planarly layered environments by introduction of their Green's functions—is documented by a variety of numerical results from synthetic, noiseless, and severely noisy field data.

Key words. electromagnetic inclusions, Maxwell's equations, imaging, inverse problems, MUSIC algorithm

AMS subject classifications. 35R30, 78A46

DOI. 10.1137/050640655

1. Introduction.

The location of a collection of small three-dimensional bounded homogeneous inclusions buried within a homogeneous medium, with arbitrary contrasts of permittivity, conductivity, and permeability vis-à-vis this embedding medium, is considered herein by time-harmonic electromagnetic means.

The applications envisaged are found in military and civil engineering, and in environmental characterization, e.g., the identification of obstacles in subsoils, nondestructive testing of man-made structures, medical imaging, etc. But the proposed framework is of broad applicability and it might be tailored to many cases of practical interest, and examples given herein in what could be the microwave scattering regime do not presume of any specific situation investigated later on.

The proposed approach uses a MUSIC (multiple signal classification)-type algorithm to accurately image these small electromagnetic inclusions. The latter are of unknown finite number and they are assumed of small size with respect to the wavelength at the frequency of operation. Usually they are located somewhat far from one another (say, at distance of the order of the wavelength or more), but closer spacings

*Received by the editors September 19, 2005; accepted for publication (in revised form) September 26, 2006; published electronically April 6, 2007.
http://www.siam.org/journals/sisc/29-2/64065.html

†Centre de Mathématiques Appliquées, CNRS UMR 7641 and Ecole Polytechnique, 91128 Palaiseau Cedex, France (habib.ammari@polytechnique.fr).

‡Département de Recherche en Électromagnétisme, Laboratoire des Signaux et Systèmes (CNRS-SUPÉLEC-UPS), Plateau de Moulon, 91192 Gif-sur-Yvette, France (iakovleva@lss.supelec.fr, lesselier@lss.supelec.fr, perrusson@lss.supelec.fr).

(at a fraction of the wavelength) are also modeled. The analysis is developed for arbitrarily shaped inclusions, though specific attention is given to the case of triaxial ellipsoidal shapes and their degenerate counterparts whenever it leads to useful complementary results. Notice that any three-dimensional inclusion can be represented in a canonical way by an ellipsoid with the same polarization tensor. See [7].

The collection is illuminated by an array (typically linear or planar) of electric and/or magnetic dipoles with any given polarization(s), located at some, not necessarily large, distance from it, extensions to far-field data being considered as well in straightforward fashion. The resulting electric and/or magnetic field is collected by another array, possibly but not necessarily the same as the transmitter one—in effect, one is always dealing with strongly aspect-limited data, and no full coverage is assumed. This procedure yields the multistatic data response (MSR) matrix that is characteristic of the collection of inclusions for given sets (e.g., arrays) of transmitters and receivers at the (single) frequency of operation.

Extending our previous works in two-dimensional scalar scattering situations [3, 4, 5] to full three-dimensional vector electromagnetics, we show how the eigenvalue structure of the MSR matrix can be employed within the framework of the MUSIC method in order to retrieve either dielectric inclusions or magnetic inclusions, or combinations thereof, with the limiting perfectly electric conductor (PEC) or perfectly magnetic conductor (PMC) encompassed as well.

In practice this means carrying out two steps: first, the calculation of singular values (and the corresponding eigenvectors), the number of nonzero values depending upon the number and the electromagnetic nature of the inclusions and upon the transmitters' and/or receivers' geometrical arrangement and polarization; second, the orthogonal projection of a properly built vector propagator onto the null space of the MSR matrix, coincidence with an inclusion being then associated to a peak of the inverse norm of the projection, simple forms of the indicator being used if the electromagnetic nature of the inclusions is known from prior information and is either purely dielectric or purely magnetic (e.g., all inclusions are known to be nonmagnetic dielectric ones), a slightly more complex form being used in the other situations. It is worth emphasizing that the number of nonzero singular values is computed without any knowledge of what kind of inclusion we have. Further, as will be shown in section 5, it allows us to characterize the electromagnetic nature of the inclusion.

In the case of a single triaxial ellipsoidal inclusion with arbitrary semiaxes and location with respect to the transmitter and receiver arrays, the single vectors and values are exhibited in closed form. Notice that this also enables us to retrieve as a particular case the results described earlier by Chambers and Berryman [19] for a lossless dielectric or PEC sphere with symmetric position with respect to a common transmitter/receiver array.

Numerical examples are given in this paper in order to illustrate the main features of our approach for both asymptotically exact and severely noisy data calculated for arbitrary electromagnetic and geometric configurations. A systematic discussion, however, is not attempted, being left to further studies in specific situations of practical applicability.

To put the present work in proper context, we will now emphasize some significant differences with prior work. Nearly all of the existing algorithms to solve three-dimensional inverse problems for Maxwell's equations are iterative and are then based on regularization techniques. See, for example, [28, 1, 41, 42, 23] and the many references therein. Recently, noniterative methods such as linear sampling methods and factorization methods have been developed which avoid the issue of nonlinearity

[33, 24, 30, 31, 39] but not ill-posedness. In order to handle this challenging issue it is generally advisable to incorporate all a priori knowledge about the unknown parameter and to try to determine very specific features. In this context, the method of asymptotic expansions of small volume inclusions provides a useful framework to accurately reconstruct the location and geometric features of the inclusions in a stable way, even for moderately noisy data [7]. Among the direct reconstruction methods that have been developed during the last years, MUSIC-type algorithms seem to be particularly accurate and stable [27]. In [3, 4, 5], we have designed robust numerical methods of MUSIC type for efficiently determining the locations and/or shapes of a collection of small inclusions in two-dimensional scalar scattering situations. The paper [2] points out the strong relation between MUSIC-type algorithms and linear sampling methods (cf. also [21, 32]). To the best of our knowledge, the present paper is the first attempt to design and analyze a MUSIC-type algorithm for imaging a collection of small volumetric inclusions of arbitrary shape and electromagnetic nature for fully general three-dimensional geometrical configurations and illuminations. As such, our analysis encompasses the DORT method (DORT stands for "decomposition of time-reversal operator" in French) [36] and three-dimensional time-reversal analysis [27, 29].

Let us finally emphasize again that our MUSIC-type method makes use of asymptotic expansions of the scattered electric and/or magnetic fields. Starting from application of Green's theorems to Maxwell's equations satisfied by the electromagnetic fields, we formally derive asymptotic formulations of the scattered electric and/or magnetic fields as series expansions in terms of the common order of magnitude of the diameters of the inclusions, here limiting ourselves for simplicity to the leading-order terms. Such formulations involve the polarization tensors associated to the inclusions. We refer to the closely related works [8, 6, 10, 11, 12, 15, 16, 17, 18, 35, 40] and to the monograph [7] for a systematic exposition of recent developments on the reconstruction of small inclusions.

2. Asymptotic expansions of the electric and magnetic fields.

2.1. Problem formulation. We suppose that there is a finite number m of electromagnetic inclusions in $\mathbb{R}^3$, each of the form $z_j + \alpha B_j$, where $B_j \subset \mathbb{R}^3$ is a bounded, smooth ($\mathcal{C}^\infty$) domain containing the origin. This regularity assumption could be considerably weakened. The total collection of inclusions thus takes the form $\mathcal{I}_\alpha = \bigcup_{j=1}^m z_j + \alpha B_j$. The points $z_j \in \mathbb{R}^3$, $j = 1, \ldots, m$, that determine the location of the inclusions are assumed to satisfy

$$0 < d_0 \leq |z_j - z_l| \quad \forall\, j \neq l,$$

where d_0 is a fixed positive constant. We also assume that α, the common order of magnitude of the diameters of the inclusions, is sufficiently small compared to d_0 that these inclusions are disjoint.

Let μ_0 and ε_0 denote the magnetic permeability and the electric permittivity of the free space; we shall assume that $\mu_0, \Im m(\varepsilon_0)$, and $\Re e(\varepsilon_0)$ are positive constants. Let μ_j and ε_j denote the permeability and the complex permittivity of the jth inclusion, $z_j + \alpha B_j$; these are also assumed to be constants. Using this notation, we introduce the piecewise constant magnetic permeability

$$\mu_\alpha(\mathbf{r}) := \begin{cases} \mu_j & \text{in } z_j + \alpha B_j, \qquad j = 1, \ldots, m, \\ \mu_0 & \text{in } \mathbb{R}^3 \backslash \overline{\mathcal{I}_\alpha}. \end{cases}$$

The piecewise constant electric permittivity $\varepsilon_\alpha(x)$ is defined analogously by

$$\varepsilon_\alpha(\mathbf{r}) := \begin{cases} \varepsilon_j & \text{in } z_j + \alpha B_j, \qquad j = 1, \dots, m, \\ \varepsilon_0 & \text{in } \mathbb{R}^3 \backslash \overline{\mathcal{I}_\alpha}. \end{cases}$$

Let $k = \omega\sqrt{\varepsilon_0 \mu_0} > 0$ be the wavenumber, where $\omega > 0$ is a given frequency. Positive real and imaginary parts are chosen for the square root. The conductivities of the free space and the electromagnetic inclusions are given by $\sigma_j = \omega\, \Im m(\varepsilon_j)$, $j = 0, \dots, m$. For convenience we introduce the following notation. For a vector $\mathbf{r} = (\mathbf{r}_x, \mathbf{r}_y, \mathbf{r}_z)$ in $\mathbb{R}^3$, we denote by $r = |\mathbf{r}|$ its magnitude and by $\hat{\mathbf{r}} = \mathbf{r}/r$ the dimensionless vector with a unit magnitude pointing in the direction of $\mathbf{r}$.

2.2. Lippman–Schwinger integral representation formula. We introduce the scalar function

$$g(\mathbf{r}, \mathbf{r}') = \frac{e^{ik|\mathbf{r}-\mathbf{r}'|}}{4\pi|\mathbf{r}-\mathbf{r}'|},$$

which is the outgoing fundamental solution for the Helmholtz operator $\Delta + k^2$. Using this scalar function we define the Green's function $\mathbf{G}$ by

$$\mathbf{G}(\mathbf{r}, \mathbf{r}') = \left[\mathbf{I}_3 + \frac{\nabla\nabla}{k^2}\right] g(\mathbf{r}, \mathbf{r}'),$$

where $\mathbf{I}_3$ is the 3×3 identity matrix. The function $\mathbf{G}(\mathbf{r}, \mathbf{r}')$ is the solution of

$$\begin{cases} \nabla\times\nabla\times\mathbf{G}(\mathbf{r}, \mathbf{r}') - k^2\mathbf{G}(\mathbf{r}, \mathbf{r}') = \mathbf{I}_3\delta(\mathbf{r} - \mathbf{r}') \quad \text{in } \mathbb{R}^3, \\ \lim\limits_{r\to\infty} r\left[\nabla\times\mathbf{G}(\mathbf{r}, \mathbf{r}') - ik\hat{\mathbf{r}} \times \mathbf{G}(\mathbf{r}, \mathbf{r}')\right] = 0. \end{cases}$$

It also satisfies

$$\mathbf{G}(\mathbf{r}', \mathbf{r}) = \left[\mathbf{I}_3 + \frac{\nabla'\nabla'}{k^2}\right] g(\mathbf{r}', \mathbf{r}), \tag{1}$$

$$\left[\mathbf{G}(\mathbf{r}', \mathbf{r})\right]^T = \mathbf{G}(\mathbf{r}, \mathbf{r}'), \tag{2}$$

$$\left[\nabla'\times\mathbf{G}(\mathbf{r}', \mathbf{r})\right]^T = \nabla\times\mathbf{G}(\mathbf{r}, \mathbf{r}'). \tag{3}$$

Here ∇' denotes the differentiation with respect to the variable $\mathbf{r}'$ and T the transpose.

Suppose that the collection of inclusions $\mathcal{I}_\alpha$ is within an open subset Ω of $\mathbb{R}^3$. Let $\mathbf{r}_n \in \mathbb{R}^3 \setminus \overline{\Omega}$ be the position of the electric dipole. Then the incident electric field $\mathbf{E}_0^{(n)}$ in a homogeneous medium due to a point current density at the position $\mathbf{r}_n$ and directed in the $\overrightarrow{\beta}$ direction, $\mathbf{J}_0^{(n)}(\mathbf{r}) = \overrightarrow{\beta}\, Il\, \delta(\mathbf{r} - \mathbf{r}_n)$, is given by

$$\mathbf{E}_0^{(n)}(\mathbf{r}) = i\omega\mu_0 \left[\mathbf{I}_3 + \frac{\nabla\nabla}{k^2}\right] \cdot \overrightarrow{\beta}\, Il\, g(\mathbf{r}, \mathbf{r}_n) = i\omega\mu_0\, Il\, \mathbf{G}(\mathbf{r}, \mathbf{r}_n) \cdot \overrightarrow{\beta},$$

where the constant Il is the current moment. The magnetic field is given by

$$\mathbf{H}_0^{(n)}(\mathbf{r}) = \nabla\times Il\, \mathbf{G}(\mathbf{r}, \mathbf{r}_n) \cdot \overrightarrow{\beta} = Il\, \nabla g(\mathbf{r}, \mathbf{r}_n) \times \overrightarrow{\beta}.$$

The electromagnetic fields $(\mathbf{E}_0^{(n)}, \mathbf{H}_0^{(n)})$ are the solutions of the Maxwell's equations

$$\begin{cases} \nabla\times\mathbf{E}_0^{(n)} = \ i\omega\mu_0\mathbf{H}_0^{(n)} \quad \text{in } \mathbb{R}^3, \\ \nabla\times\mathbf{H}_0^{(n)} = \ -i\omega\varepsilon_0\mathbf{E}_0^{(n)} + \mathbf{J}_0^{(n)} \quad \text{in } \mathbb{R}^3, \\ \lim\limits_{r\to\infty} r\left[\nabla\times\mathbf{E}_0^{(n)} - ik\hat{\mathbf{r}}\times\mathbf{E}_0^{(n)}\right] = 0, \ \text{ and } \ \lim\limits_{r\to\infty} r\left[\nabla\times\mathbf{H}_0^{(n)} - ik\hat{\mathbf{r}}\times\mathbf{H}_0^{(n)}\right] = 0. \end{cases}$$

Let $(\mathbf{E}_\alpha^{(n)}, \mathbf{H}_\alpha^{(n)})$ denote the time-harmonic electromagnetic fields in the presence of the electromagnetic inclusions $\mathcal{I}_\alpha$. These time-harmonic fields are the solutions of the following Maxwell's equations:

(4)
$$\begin{cases} \nabla\times\mathbf{E}_\alpha^{(n)} = \ i\omega\mu_\alpha\mathbf{H}_\alpha^{(n)} \quad \text{in } \mathbb{R}^3, \\ \nabla\times\mathbf{H}_\alpha^{(n)} = \ -i\omega\varepsilon_\alpha\mathbf{E}_\alpha^{(n)} + \mathbf{J}_0^{(n)} \quad \text{in } \mathbb{R}^3, \\ \nu\times\mathbf{E}_\alpha^{(n)} \text{ and } \nu\times\mathbf{H}_\alpha^{(n)} \text{ are continuous across } \partial(z_j+\alpha B_j), \\ \nu\cdot\left(\varepsilon_\alpha\mathbf{E}_\alpha^{(n)}\right)^+ - \nu\cdot\left(\varepsilon_\alpha\mathbf{E}_\alpha^{(n)}\right)^- = 0 \ \text{ on } \ \partial(z_j+\alpha B_j), \\ \nu\cdot\left(\mu_\alpha\mathbf{H}_\alpha^{(n)}\right)^+ - \nu\cdot\left(\mu_\alpha\mathbf{H}_\alpha^{(n)}\right)^- = 0 \ \text{ on } \ \partial(z_j+\alpha B_j), \\ \lim\limits_{r\to\infty} r\left[\nabla\times\mathbf{E}_\alpha^{(n)} - ik\hat{\mathbf{r}}\times\mathbf{E}_\alpha^{(n)}\right] = 0, \ \text{ and } \ \lim\limits_{r\to\infty} r\left[\nabla\times\mathbf{H}_\alpha^{(n)} - ik\hat{\mathbf{r}}\times\mathbf{H}_\alpha^{(n)}\right] = 0. \end{cases}$$

Here ν denotes the outward unit normal to $\partial(z_j+\alpha B_j)$; superscript $+$ and $-$ indicate the limiting values as we approach $\partial(z_j+\alpha B_j)$ from outside $z_j+\alpha B_j$, and from inside $z_j+\alpha B_j$, respectively.

From (4), it follows that the electric field satisfies the following:

(5)
$$\begin{cases} \nabla\times\mu_\alpha^{-1}\nabla\times\mathbf{E}_\alpha^{(n)} - \omega^2\varepsilon_\alpha\mathbf{E}_\alpha^{(n)} = i\omega\mathbf{J}_0^{(n)}, \\ \nu\times\mathbf{H}_\alpha^{(n)} \ \text{ is continuous across } \ \partial(z_j+\alpha B_j), \\ \nu\cdot\left(\varepsilon_\alpha\mathbf{E}_\alpha^{(n)}\right)^+ - \nu\cdot\left(\varepsilon_\alpha\mathbf{E}_\alpha^{(n)}\right)^- = 0 \ \text{ on } \ \partial(z_j+\alpha B_j), \\ \lim\limits_{r\to\infty} r\left[\nabla\times\mathbf{E}_\alpha^{(n)} - ik\hat{\mathbf{r}}\times\mathbf{E}_\alpha^{(n)}\right] = 0. \end{cases}$$

The following Lippman–Schwinger integral representation formulae hold:

(6)
$$\mathbf{E}_\alpha^{(n)}(\mathbf{r}) - \mathbf{E}_0^{(n)}(\mathbf{r}) = \sum_{j=1}^m \int_{z_j+\alpha B_j} d\mathbf{r}' \Big[-i\omega\,(\mu_j-\mu_0)\,\nabla'\times\mathbf{G}(\mathbf{r},\mathbf{r}')\cdot\mathbf{H}_\alpha^{(n)}(\mathbf{r}') + \omega^2\mu_0\,(\varepsilon_j-\varepsilon_0)\,\mathbf{G}(\mathbf{r},\mathbf{r}')\cdot\mathbf{E}_\alpha^{(n)}(\mathbf{r}')\Big]$$

and

$$\mathbf{H}_\alpha^{(n)}(\mathbf{r}) - \mathbf{H}_0^{(n)}(\mathbf{r}) = \sum_{j=1}^m \int_{z_j+\alpha B_j} d\mathbf{r}' \Big[i\omega\,(\varepsilon_j-\varepsilon_0)\,\nabla'\times\mathbf{G}(\mathbf{r},\mathbf{r}')\cdot\mathbf{E}_\alpha^{(n)}(\mathbf{r}') + \omega^2\varepsilon_0\,(\mu_j-\mu_0)\,\mathbf{G}(\mathbf{r},\mathbf{r}')\cdot\mathbf{H}_\alpha^{(n)}(\mathbf{r}')\Big].$$

2.3. Polarization tensors. Let $\{q_j\}$ denote either the set $\{\mu_j\}$ or $\{\varepsilon_j\}$ for $1 \le j \le m$. Fix j and let q be given by

$$q(\mathbf{r}) := \begin{cases} q_0, & \mathbf{r} \in \mathbb{R}^3 \setminus \overline{B}_j, \\ q_j, & \mathbf{r} \in B_j. \end{cases}$$

By v^q, we denote the solution of

$$\begin{cases} \nabla \cdot q(\mathbf{r}) \nabla v^q(\mathbf{r}) = 0 \quad \text{in } \mathbb{R}^3, \\ v^q(\mathbf{r}) - \mathbf{r} \to 0 \quad \text{as } r \to \infty. \end{cases} \tag{7}$$

It is obvious that the function v^q depends only on the ratio q_j/q_0.

Define the polarization tensor $\mathbf{M}(q_j/q_0; B_j) \in \mathbb{R}^{3\times 3}$ associated with the inclusion B_j and the contrast q_j/q_0 by

$$\mathbf{M}\left(\frac{q_j}{q_0}; B_j\right) := \int_{B_j} \nabla v^q(\mathbf{r})\, d\mathbf{r}. \tag{8}$$

Introduce the vectors φ and ψ defined in $\mathbb{R}^3 \setminus \overline{B}_j$ and satisfying

$$\begin{cases} \Delta \varphi = 0 \quad \text{in } \mathbb{R}^3 \setminus \overline{B}_j, \\ \partial_\nu \varphi = -\nu \quad \text{on } \partial B_j, \\ \lim_{r\to+\infty} \varphi(\mathbf{r}) = 0, \end{cases}$$

and

$$\begin{cases} \Delta \psi = 0 \quad \text{in } \mathbb{R}^3 \setminus \overline{B}_j, \\ \psi = -\mathbf{r} + \mathbf{c} \quad \text{on } \partial B_j, \\ \lim_{r\to+\infty} \psi(\mathbf{r}) = 0, \end{cases}$$

where the constant vector $\mathbf{c}$ is to be chosen such that $\psi(\mathbf{r}) \to 0$ as $r \to +\infty$.

The tensor

$$\mathbf{W}(B_j) := \int_{\mathbb{R}^3\setminus\overline{B}_j} \nabla\varphi(\mathbf{r})^T \cdot \nabla\varphi(\mathbf{r})\, d\mathbf{r}$$

is the so-called virtual mass, and the tensor

$$\mathbf{Q}(B_j) := \int_{\mathbb{R}^3\setminus\overline{B}_j} \nabla\psi(\mathbf{r})^T \cdot \nabla\psi(\mathbf{r})\, d\mathbf{r}$$

is the so-called Pólya–Szegö tensor. See [38, 34, 26, 7]. Moreover, according to [18, 10], the following convergence results hold as q_j/q_0 tends towards zero or infinity:

$$\begin{aligned} &\int_{\partial B_j} \left((\partial_\nu v^q)^+\right)^T \mathbf{r}\, d\sigma \to |B_j|\mathbf{I}_3 + \mathbf{Q}(B_j) \quad \text{as } \frac{q_j}{q_0} \to +\infty, \\ &\int_{\partial B_j} \left((\partial_\nu v^q)^-\right)^T \mathbf{r}\, d\sigma \to |B_j|\mathbf{I}_3 + \mathbf{W}(B_j) \quad \text{as } \frac{q_j}{q_0} \to 0. \end{aligned} \tag{9}$$

Here ν is the outward unit normal to ∂B_j and $(\partial_\nu u)^\pm(\mathbf{r}) = \lim_{t\to 0^+} \nabla u(\mathbf{r} \pm t\nu(\mathbf{r}))$ if the limit exists.

2.4. Asymptotic formula. For any $\mathbf{r}$ away from $\mathbf{r}_n$ and $z_j, j = 1, \dots, m$, we can show that

$$\mathbf{E}_\alpha^{(n)}(\mathbf{r}) - \mathbf{E}_0^{(n)}(\mathbf{r}) = \alpha^3 \sum_{j=1}^{m} \left[-i\omega\ (\mu_j - \mu_0)\, \nabla' \times \mathbf{G}(\mathbf{r}, z_j) \cdot \mathbf{M}\left(\frac{\mu_j}{\mu_0}; B_j\right) \mathbf{H}_0^{(n)}(z_j) \right. \\ \left. + \omega^2 \mu_0\, (\varepsilon_j - \varepsilon_0)\, \mathbf{G}(\mathbf{r}, z_j) \cdot \mathbf{M}\left(\frac{\varepsilon_j}{\varepsilon_0}; B_j\right) \mathbf{E}_0^{(n)}(z_j) \right] + O(\alpha^4). \tag{10}$$

See Appendix A. If the inclusions B_j are symmetric about their centers, then we can show that the correction of order four in (10) is zero. The remainder in this case is in fact $O(\alpha^5)$. See [14].

Note that if the inclusion B_j is a ball, then its polarization tensor $\mathbf{M}(q_j/q_0; B_j)$ has the following explicit form [18]:

$$\mathbf{M}\left(\frac{q_j}{q_0}; B_j\right) = \frac{3q_0}{2q_0 + q_j} |B_j| \mathbf{I}_3, \quad q(\mathbf{r}) = \varepsilon(\mathbf{r}) \ \text{ or } \ \mu(\mathbf{r}),$$

and therefore (10) becomes

$$\mathbf{E}_\alpha^{(n)}(\mathbf{r}) - \mathbf{E}_0^{(n)}(\mathbf{r}) = \alpha^3 \sum_{j=1}^{m} \left[-i\omega\, \mu_0 \frac{3(\mu_j - \mu_0)}{2\mu_0 + \mu_j} |B_j|\, \nabla' \times \mathbf{G}(\mathbf{r}, z_j) \cdot \mathbf{H}_0^{(n)}(z_j) \right. \\ \left. + k^2 \frac{3(\varepsilon_j - \varepsilon_0)}{2\varepsilon_0 + \varepsilon_j} |B_j|\, \mathbf{G}(\mathbf{r}, z_j) \cdot \mathbf{E}_0^{(n)}(z_j) \right] + O(\alpha^4). \tag{11}$$

Formula (11) would easily extend to the case of triaxial ellipsoidal inclusions, using their standard tensors which, for inclusion B_j, read in the ellipsoidal coordinate system attached to it as

$$\mathbf{M}\left(\frac{q_j}{q_0}; B_j\right) = \begin{bmatrix} m_{j1} & 0 & 0 \\ 0 & m_{j2} & 0 \\ 0 & 0 & m_{j3} \end{bmatrix} |B_j|,$$

where the coefficients $m_{jl}, l = 1, 2, 3$, are equal to $q_0/(q_0 + L_l(q_j - q_0))$, the L_l being the well-known depolarization factors of the ellipsoid proportional to harmonic elliptic integrals of degree 1. See [34, 26, 9]. We also note that from (9) it follows that

$$\frac{\varepsilon_j}{\varepsilon_0} \mathbf{M}\left(\frac{\varepsilon_j}{\varepsilon_0}; B_j\right) \to \mathbf{Q}(B_j) \quad \text{as} \quad \left|\frac{\varepsilon_j}{\varepsilon_0}\right| \to +\infty$$

and

$$\mathbf{M}\left(\frac{\mu_j}{\mu_0}; B_j\right) \to \mathbf{W}(B_j) \quad \text{as } \frac{\mu_j}{\mu_0} \to 0,$$

which (formally) extends the validity of (10) and its specialization (11) to PEC, results being likewise obtained for PMC by duality. See [18].

2.5. Far-field pattern. Since for fixed $\mathbf{r}'$ the far-field expansions

$$g(\mathbf{r}, \mathbf{r}') = \frac{e^{ik|\mathbf{r} - \mathbf{r}'|}}{4\pi|\mathbf{r} - \mathbf{r}'|} = \frac{e^{ikr}}{4\pi r} e^{-ik\hat{\mathbf{r}} \cdot \mathbf{r}'} + O\left(\frac{1}{r^2}\right), \\ \nabla' g(\mathbf{r}, \mathbf{r}') = -ik \frac{e^{ikr}}{4\pi r} e^{-ik\hat{\mathbf{r}} \cdot \mathbf{r}'} \hat{\mathbf{r}} + O\left(\frac{1}{r^2}\right) \tag{12}$$

hold, then, using the vector identity $\nabla\times(\psi\mathbf{A}) = \nabla\psi\times\mathbf{A} + \psi\nabla\times\mathbf{A}$, we obtain that the following expansions hold uniformly in $\hat{\mathbf{r}}$ for any vector $\mathbf{a}$ and any fixed $\mathbf{r}'$:

(13)
$$\begin{aligned}\mathbf{G}(\mathbf{r},\mathbf{r}')\cdot\mathbf{a} &= -\frac{e^{ikr}}{4\pi r}e^{-ik\hat{\mathbf{r}}\cdot\mathbf{r}'}\hat{\mathbf{r}}\times(\hat{\mathbf{r}}\times\mathbf{a}) + O\left(\frac{1}{r^2}\right),\\ \nabla'\times\mathbf{G}(\mathbf{r},\mathbf{r}')\cdot\mathbf{a} &= \nabla' g(\mathbf{r},\mathbf{r}')\times\mathbf{a} = -ik\frac{e^{ikr}}{4\pi r}e^{-ik\hat{\mathbf{r}}\cdot\mathbf{r}'}\hat{\mathbf{r}}\times\mathbf{a} + O\left(\frac{1}{r^2}\right) \quad \text{as } r\to+\infty.\end{aligned}$$

Note that $-\hat{\mathbf{r}}\times(\hat{\mathbf{r}}\times\mathbf{a}) = \Delta(\mathbf{r})\cdot\mathbf{a}$, where $\Delta(\mathbf{r}) = \mathbf{I_3} - \hat{\mathbf{r}}\hat{\mathbf{r}}^T$. Moreover, $\Delta(\mathbf{r}) = \Delta^T(\mathbf{r}), \Delta^2(\mathbf{r}) = \Delta(\mathbf{r})$, and $\hat{\mathbf{r}}\times\mathbf{a} = \Lambda(\mathbf{r})\cdot\mathbf{a}$, where

$$\Lambda(\mathbf{r}) = \begin{bmatrix} 0 & -\hat{\mathbf{r}}_z & \hat{\mathbf{r}}_y \\ \hat{\mathbf{r}}_z & 0 & -\hat{\mathbf{r}}_x \\ -\hat{\mathbf{r}}_y & \hat{\mathbf{r}}_x & 0 \end{bmatrix}.$$

Here $(\hat{\mathbf{r}}_x,\hat{\mathbf{r}}_y,\hat{\mathbf{r}}_z) = (\mathbf{r}_x,\mathbf{r}_y,\mathbf{r}_z)/r$. It is easy to see that $\Lambda(\mathbf{r}) = -\Lambda^T(\mathbf{r})$, and $\Lambda(\mathbf{r})\Lambda^T(\mathbf{r}) = \Delta(\mathbf{r})$.

With this notation, since

$$\begin{aligned}&\nabla g(\mathbf{r},\mathbf{r}')\times\mathbf{a} = -\nabla' g(\mathbf{r},\mathbf{r}')\times\mathbf{a},\\ &\nabla g(\mathbf{r},\mathbf{r}')\times\mathbf{a} = ik\frac{e^{ikr}}{4\pi r}e^{-ik\hat{\mathbf{r}}\cdot\mathbf{r}'}\Lambda(\mathbf{r})\cdot\mathbf{a} + O\left(\frac{1}{r^2}\right) \quad \text{as } r\to+\infty,\end{aligned}$$

the far-field expansions in (13) become

(14)
$$\begin{aligned}&\mathbf{G}(\mathbf{r},\mathbf{r}') = \frac{e^{ikr}}{4\pi r}e^{-ik\hat{\mathbf{r}}\cdot\mathbf{r}'}\Delta(\mathbf{r}) + O\left(\frac{1}{r^2}\right),\\ &\nabla'\times\mathbf{G}(\mathbf{r},\mathbf{r}') = -ik\frac{e^{ikr}}{4\pi r}e^{-ik\hat{\mathbf{r}}\cdot\mathbf{r}'}\Lambda(\mathbf{r}) + O\left(\frac{1}{r^2}\right) \quad \text{as } r\to+\infty.\end{aligned}$$

Let $A_\alpha(\hat{\mathbf{r}},\hat{\mathbf{r}}_n)$ be the scattering amplitude of the electromagnetic inclusions $\mathcal{I}_\alpha$. The function $A_\alpha(\hat{\mathbf{r}},\hat{\mathbf{r}}_n)$ is defined to be the function which satisfies

$$\mathbf{E}_\alpha^{(n)}(\mathbf{r}) = \mathbf{E}_0^{(n)}(\mathbf{r}) + A_\alpha(\hat{\mathbf{r}},\hat{\mathbf{r}}_n)\frac{e^{ikr}}{4\pi r} + O\left(\frac{1}{r^2}\right)$$

as $r\to+\infty$, uniformly with respect to $\hat{\mathbf{r}}$ and $\hat{\mathbf{r}}_n$ on the unit sphere in $\mathbb{R}^3$, where

$$\mathbf{E}_0^{(n)}(\mathbf{r}) = -i\omega\mu_0(Il)_n\,\hat{\mathbf{r}}_n\times(\hat{\mathbf{r}}_n\times\overrightarrow{\beta})\,e^{-ik\hat{\mathbf{r}}_n\cdot\mathbf{r}}$$

is the incident field and $\mathbf{E}_\alpha^{(n)}$ is the unique solution of

(15)
$$\begin{cases}\nabla\times\dfrac{1}{\mu_\alpha}\nabla\times\mathbf{E}_\alpha^{(n)}(\mathbf{r}) - \omega^2\varepsilon_\alpha\mathbf{E}_\alpha^{(n)}(\mathbf{r}) = 0 \quad \text{in } \mathbb{R}^3,\\ \lim\limits_{r\to+\infty} r\left[\nabla\times(\mathbf{E}_\alpha^{(n)}-\mathbf{E}_0^{(n)})(\mathbf{r}) - ik\hat{\mathbf{r}}\times(\mathbf{E}_\alpha^{(n)}-\mathbf{E}_0^{(n)})(\mathbf{r})\right] = 0.\end{cases}$$

It is well known (e.g., from [25, 37]) that there exists a unique solution to (15).

By using (14), it follows from (10) that the leading-order term in the asymptotic expansion of $A_\alpha(\hat{\mathbf{r}}, \hat{\mathbf{r}}_n)$ is given by

$$A_\alpha(\hat{\mathbf{r}}, \hat{\mathbf{r}}_n) = k^2 i\omega\mu_0\, \alpha^3 \sum_{j=1}^{m} e^{-ik(\hat{\mathbf{r}}+\hat{\mathbf{r}}_n)\cdot z_j} \left[-\left(\frac{\mu_j}{\mu_0} - 1\right) \Lambda(\mathbf{r}) \cdot \mathbf{M}\left(\frac{\mu_j}{\mu_0}; B_j\right) \Lambda(\mathbf{r}_n) \right.$$
$$\left. + \left(\frac{\varepsilon_j}{\varepsilon_0} - 1\right) \Delta(\mathbf{r}) \cdot \mathbf{M}\left(\frac{\varepsilon_j}{\varepsilon_0}; B_j\right) \Delta(\mathbf{r}_n) \right] \cdot \overrightarrow{\beta}\,(Il)_n + O(\alpha^4), \tag{16}$$

where $\overrightarrow{\beta}$ is the incidence direction. See [13] for a proof. In the special case where all the inclusions are small spheres we get from (11), as shown in [20], that

$$\begin{aligned} &A_\alpha(\hat{\mathbf{r}}, \hat{\mathbf{r}}_n) \\ &= i\omega\mu_0 k^2 \alpha^3 \sum_{j=1}^{m} e^{-ik(\hat{\mathbf{r}}+\hat{\mathbf{r}}_n)\cdot z_j} \left[-\frac{3\,(\mu_j - \mu_0)}{2\mu_0 + \mu_j} |B_j|\, \Lambda(\mathbf{r})\Lambda(\mathbf{r}_n) + \frac{3(\varepsilon_j - \varepsilon_0)}{2\varepsilon_0 + \varepsilon_j} |B_j|\, \Delta(\mathbf{r})\Delta(\mathbf{r}_n) \right] \\ &\quad \cdot \overrightarrow{\beta}\,(Il)_n \\ &= i\omega\mu_0 k^2 \alpha^3 \sum_{j=1}^{m} e^{-ik(\hat{\mathbf{r}}+\hat{\mathbf{r}}_n)\cdot z_j} \left[\frac{3\,(\mu_j - \mu_0)}{2\mu_0 + \mu_j} |B_j|\, \Delta(\mathbf{r}, \mathbf{r}_n) + \frac{3(\varepsilon_j - \varepsilon_0)}{2\varepsilon_0 + \varepsilon_j} |B_j|\, \Delta(\mathbf{r})\Delta(\mathbf{r}_n) \right] \\ &\quad \cdot \overrightarrow{\beta}\,(Il)_n, \end{aligned}$$

where

$$\Delta(\mathbf{r}, \mathbf{r}_n) = \hat{\mathbf{r}} \cdot \hat{\mathbf{r}}_n \mathbf{I_3} - \hat{\mathbf{r}}_n \hat{\mathbf{r}}^T.$$

Here we have used the fact that

$$\Lambda(\mathbf{r})\Lambda(\mathbf{r}_n) \cdot \mathbf{a} = \hat{\mathbf{r}} \times (\hat{\mathbf{r}}_n \times \mathbf{a}) = -\Delta(\mathbf{r}, \mathbf{r}_n) \cdot \mathbf{a}.$$

3. The multistatic response matrix.

3.1. The multistatic response matrix for transmitter and receiver arrays. Let $\{\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z\}$ form an orthonormal basis of $\mathbb{R}^3$. Let $\{\mathbf{r}_1, \ldots, \mathbf{r}_{n_t}\} \in \mathbb{R}^3 \setminus \overline{\Omega}$ be the positions of the transmitters and $\overrightarrow{\beta}_1, \ldots, \overrightarrow{\beta}_{d_t}$ be the unit directions of incident fields due to a point current density at the position $\mathbf{r}_n$. Let $\{\mathbf{R}_1, \ldots, \mathbf{R}_{n_r}\} \in \mathbb{R}^3 \setminus \overline{\Omega}$ be the positions of the receivers and $\overrightarrow{\gamma}_1, \ldots, \overrightarrow{\gamma}_{d_r}$ be the observation directions of the fields at the position $\mathbf{R}_n$.

Recall that the incident electric and magnetic fields are given by

$$\begin{aligned} \mathbf{E}_0^{(n)}(\mathbf{r}) &= i\omega\mu_0\, \mathbf{G}(\mathbf{r}, \mathbf{r}_n) \cdot \overrightarrow{\beta}_d\, (Il)_n^d, \\ \mathbf{H}_0^{(n)}(\mathbf{r}) &= \nabla\times\, \mathbf{G}(\mathbf{r}, \mathbf{r}_n) \cdot \overrightarrow{\beta}_d\, (Il)_n^d, \quad n = 1, \ldots, n_t,\ d = 1, \ldots, d_t, \end{aligned}$$

where $(Il)_n^d$ are current moments.

The voltages induced on the pth receiving element can be expressed as

$$\begin{aligned} V_{p,n}^{1,d} &= \overrightarrow{\gamma}_1 \cdot \left(\mathbf{E}_\alpha^{(n)}(\mathbf{R}_p) - \mathbf{E}_0^{(n)}(\mathbf{R}_p) \right), \\ &\vdots \\ V_{p,n}^{d_r,d} &= \overrightarrow{\gamma}_{d_r} \cdot \left(\mathbf{E}_\alpha^{(n)}(\mathbf{R}_p) - \mathbf{E}_0^{(n)}(\mathbf{R}_p) \right), \end{aligned} \tag{17}$$

where the perturbation $\mathbf{E}_\alpha^{(n)}(\mathbf{R}_p) - \mathbf{E}_0^{(n)}(\mathbf{R}_p)$ is given by

$$\begin{aligned}\mathbf{E}_\alpha^{(n)}(\mathbf{R}_p) - \mathbf{E}_0^{(n)}(\mathbf{R}_p) = i\omega\mu_0\,\alpha^3 \sum_{j=1}^{m} \Bigg[&- \left(\frac{\mu_j}{\mu_0} - 1\right)\nabla' \times \mathbf{G}(\mathbf{R}_p, z_j) \\ &\cdot \mathbf{M}\left(\frac{\mu_j}{\mu_0}; B_j\right)\nabla \times \mathbf{G}(z_j, \mathbf{r}_n) + k^2\left(\frac{\varepsilon_j}{\varepsilon_0} - 1\right)\mathbf{G}(\mathbf{R}_p, z_j) \\ &\cdot \mathbf{M}\left(\frac{\varepsilon_j}{\varepsilon_0}; B_j\right)\mathbf{G}(z_j, \mathbf{r}_n)\Bigg] \cdot \overrightarrow{\beta}_d (Il)_n^d \end{aligned} \tag{18}$$

for $d = 1, \ldots, d_t$ and $n = 1, \ldots, n_t$. Here we have neglected the asymptotic term of order α^4 in (10).

Combining (17) and (18) yields

$$\mathbf{V} = \mathbf{A}\, J,$$

where

$$\mathbf{V} = \left[\mathbf{V}_1^1 \ldots \mathbf{V}_1^{d_t}, \ldots, \mathbf{V}_{n_t}^1 \ldots \mathbf{V}_{n_t}^{d_t}\right],$$

$$\mathbf{V}_n^d = \left(V_{1,n}^{1,d}, \ldots, V_{1,n}^{d_r,d}, \ldots, V_{n_r,n}^{1,d}, \ldots, V_{n_r,n}^{d_r,d}\right)^T, \quad d = 1, \ldots, d_t, \;\; n = 1, \ldots, n_t,$$

and

$$J = \operatorname{diag}\left((Il)_1^1, \ldots, (Il)_1^{d_t}, \ldots, (Il)_{n_t}^1, \ldots, (Il)_{n_t}^{d_t}\right).$$

The matrix $\mathbf{A} \in \mathbb{C}^{d_r n_r \times d_t n_t}$ is the so-called MSR matrix.

For $z \in \mathbb{R}^3 \backslash \{\mathbf{R}_1, \ldots, \mathbf{R}_{n_r}, \mathbf{r}_1, \ldots, \mathbf{r}_{n_t}\}$, introduce the matrices $\mathbf{g}_{r,z}^\varepsilon, \mathbf{g}_{r,z}^\mu \in \mathbb{C}^{d_r n_r \times 3}$, $\mathbf{g}_{t,z}^\varepsilon, \mathbf{g}_{t,z}^\mu \in \mathbb{C}^{d_t n_t \times 3}$:

$$\begin{cases} \mathbf{g}_{r,z}^\varepsilon = (\mathbf{G}(z, \mathbf{R}_1) \cdot \overrightarrow{\gamma}_1, \ldots, \mathbf{G}(z, \mathbf{R}_1) \cdot \overrightarrow{\gamma}_{d_r}, \ldots, \mathbf{G}(z, \mathbf{R}_{n_r}) \cdot \overrightarrow{\gamma}_1, \ldots, \\ \qquad \mathbf{G}(z, \mathbf{R}_{n_r}) \cdot \overrightarrow{\gamma}_{d_r})^T, \\ \mathbf{g}_{t,z}^\varepsilon = (\mathbf{G}(z, \mathbf{r}_1) \cdot \overrightarrow{\beta}_1, \ldots, \mathbf{G}(z, \mathbf{r}_1) \cdot \overrightarrow{\beta}_{d_t}, \ldots, \mathbf{G}(z, \mathbf{r}_{n_t}) \cdot \overrightarrow{\beta}_1, \ldots, \mathbf{G}(z, \mathbf{r}_{n_t}) \cdot \overrightarrow{\beta}_{d_t})^T, \\ \mathbf{g}_{r,z}^\mu = (\nabla g(z, \mathbf{R}_1) \times \overrightarrow{\gamma}_1, \ldots, \nabla g(z, \mathbf{R}_1) \times \overrightarrow{\gamma}_{d_r}, \ldots, \nabla g(z, \mathbf{R}_{n_r}) \times \overrightarrow{\gamma}_1, \ldots, \\ \qquad \nabla g(z, \mathbf{R}_{n_r}) \times \overrightarrow{\gamma}_{d_r})^T, \\ \mathbf{g}_{t,z}^\mu = (\nabla g(z, \mathbf{r}_1) \times \overrightarrow{\beta}_1, \ldots, \nabla g(z, \mathbf{r}_1) \times \overrightarrow{\beta}_{d_t}, \ldots, \nabla g(z, \mathbf{r}_{n_t}) \times \overrightarrow{\beta}_1, \ldots, \\ \qquad \nabla g(z, \mathbf{r}_{n_t}) \times \overrightarrow{\beta}_{d_t})^T, \end{cases} \tag{19}$$

and define the matrices $\mathbf{M}_j^\varepsilon, \mathbf{M}_j^\mu \in \mathbb{C}^{3\times 3}$, $\mathbf{M}_j^{\mu,\varepsilon} \in \mathbb{C}^{6\times 6}$ by

$$\begin{aligned} \mathbf{M}_j^\varepsilon &= i\omega\mu_0\,\alpha^3\,k^2\left(\frac{\varepsilon_j}{\varepsilon_0} - 1\right)\mathbf{M}\left(\frac{\varepsilon_j}{\varepsilon_0}; B_j\right), \\ \mathbf{M}_j^\mu &= i\omega\mu_0\,\alpha^3\left(\frac{\mu_j}{\mu_0} - 1\right)\mathbf{M}\left(\frac{\mu_j}{\mu_0}; B_j\right), \\ \mathbf{M}_j^{\mu,\varepsilon} &= \operatorname{diag}(\mathbf{M}_j^\mu, \mathbf{M}_j^\varepsilon), \quad j = 1, \ldots, m. \end{aligned}$$

Throughout this paper we consider separately the following three cases:

- case 1: $\mu_j = \mu_0$ for all $j = 1, \dots, m$,
- case 2: $\varepsilon_j = \varepsilon_0$ for all $j = 1, \dots, m$,
- case 3: $\mu_j \neq \mu_0$ and $\varepsilon_j \neq \varepsilon_0$ at least for one $j \in \{1, \dots, m\}$.

For simplicity we use the notation

$$\mathbf{g}_{p,z} := \begin{cases} \mathbf{g}^{\varepsilon}_{p,z} & \text{in case 1,} \\ \mathbf{g}^{\mu}_{p,z} & \text{in case 2,} \\ [\mathbf{g}^{\mu}_{p,z}, \mathbf{g}^{\varepsilon}_{p,z}] & \text{in case 3,} \end{cases} \tag{20}$$

$$\mathbf{M}_j := \begin{cases} \mathbf{M}^{\varepsilon}_j & \text{in case 1,} \\ \mathbf{M}^{\mu}_j & \text{in case 2,} \\ \mathbf{M}^{\mu,\varepsilon}_j & \text{in case 3,} \end{cases} \tag{21}$$

for $p = r, t$ and $j = 1, \dots, m$, to decompose the MSR matrix $\mathbf{A}$ as follows:

$$\mathbf{A} = \mathbf{g}_{r,z_1,\dots,z_m}\, \mathbf{M}\, \mathbf{g}^T_{t,z_1,\dots,z_m}, \tag{22}$$

where, for $p = r, t$,

$$\mathbf{g}_{p,z_1,\dots,z_m} = [\mathbf{g}_{p,z_1}, \dots, \mathbf{g}_{p,z_m}] \quad \text{and} \quad \mathbf{M} = \operatorname{diag}(\mathbf{M}_1, \dots, \mathbf{M}_m).$$

Note that if $d_t = d_r$, $n_t = n_r = n$, and $\{\mathbf{r}_1, \dots, \mathbf{r}_n\} = \{\mathbf{R}_1, \dots, \mathbf{R}_n\}$, then the matrix $\mathbf{A}$ is symmetric. In writing (22) we have used the fact that for any vector $\mathbf{a} \in \mathbb{R}^3$

$$\nabla \times \mathbf{G}(\mathbf{r}, \mathbf{r}') \cdot \mathbf{a} = \nabla g(\mathbf{r}, \mathbf{r}') \times \mathbf{a}$$

and

$$(\nabla g(\mathbf{r}, \mathbf{r}') \times \mathbf{a})^T = (\nabla \times \mathbf{G}(\mathbf{r}, \mathbf{r}') \cdot \mathbf{a})^T = \mathbf{a} \cdot \nabla' \times \mathbf{G}(\mathbf{r}', \mathbf{r}) - \mathbf{a} \cdot \nabla \times \mathbf{G}(\mathbf{r}', \mathbf{r}).$$

Recall that the matrix $\mathbf{A}^*\mathbf{A}$, where $\mathbf{A}^* = \overline{\mathbf{A}}^T$, is called the time-reversal matrix.

3.2. The multistatic response matrix for the far-field pattern. From (16) the leading-order term in the asymptotic expansion of $A_\alpha(\hat{\mathbf{r}}, \hat{\mathbf{r}}_n)$ for any directions $\overrightarrow{\beta}$, $\overrightarrow{\gamma}$, and $\hat{\mathbf{R}}_p, \hat{\mathbf{r}}_n$ is given, after neglecting the remainder $O(\alpha^4)$, by

$$\begin{aligned} &\overrightarrow{\gamma} \cdot (A_\alpha(\hat{\mathbf{R}}_p, \hat{\mathbf{r}}_n) - A_0(\hat{\mathbf{R}}_p, \hat{\mathbf{r}}_n)) \\ &\quad = k^2 i\omega\mu_0\, \alpha^3 \sum_{j=1}^{m} e^{-ik(\hat{\mathbf{R}}_p + \hat{\mathbf{r}}_n)\cdot z_j}\, \overrightarrow{\gamma} \cdot \left[-\left(\frac{\mu_j}{\mu_0} - 1\right) \Lambda(\mathbf{R}_p) \cdot \mathbf{M} \left(\frac{\mu_j}{\mu_0}; B_j\right) \Lambda(\mathbf{r}_n) \right. \\ &\qquad \left. + \left(\frac{\varepsilon_j}{\varepsilon_0} - 1\right) \Delta(\mathbf{R}_p) \cdot \mathbf{M} \left(\frac{\varepsilon_j}{\varepsilon_0}; B_j\right) \Delta(\mathbf{r}_n) \right] \cdot \overrightarrow{\beta}\, (Il)_n, \end{aligned}$$

where $\hat{\mathbf{R}}_p = \mathbf{R}_p/|\mathbf{R}_p|, \hat{\mathbf{r}}_n = \mathbf{r}_n/|\mathbf{r}_n|$.

In this case, the vectors $\mathbf{g}^{\varepsilon}_{r,z}, \mathbf{g}^{\mu}_{r,z} \in \mathbb{C}^{d_r n_r \times 3}$, $\mathbf{g}^{\varepsilon}_{t,z}, \mathbf{g}^{\mu}_{t,z} \in \mathbb{C}^{d_t n_t \times 3}$, for $z \in \mathbb{R}^3$, are

given by

(23)
$$\begin{cases} \mathbf{g}^{\varepsilon}_{r,z} = \Big(e^{-ik\hat{\mathbf{R}}_1\cdot z}\Delta(\mathbf{R}_1)\cdot\overrightarrow{\gamma}_1,\dots,e^{-ik\hat{\mathbf{R}}_1\cdot z}\Delta(\mathbf{R}_1)\cdot\overrightarrow{\gamma}_{d_r},\dots, \\ \qquad\qquad e^{-ik\hat{\mathbf{R}}_{n_r}\cdot z}\Delta(\mathbf{R}_{n_r})\cdot\overrightarrow{\gamma}_1,\dots,e^{-ik\hat{\mathbf{R}}_{n_r}\cdot z}\Delta(\mathbf{R}_{n_r})\cdot\overrightarrow{\gamma}_{d_r}\Big)^T, \\ \mathbf{g}^{\varepsilon}_{t,z} = \Big(e^{-ik\hat{\mathbf{r}}_1\cdot z}\Delta(\mathbf{r}_1)\cdot\overrightarrow{\beta}_1,\dots,e^{-ik\hat{\mathbf{r}}_1\cdot z}\Delta(\mathbf{r}_1)\cdot\overrightarrow{\beta}_{d_t},\dots, \\ \qquad\qquad e^{-ik\hat{\mathbf{r}}_{n_t}\cdot z}\Delta(\mathbf{r}_{n_t})\cdot\overrightarrow{\beta}_1,\dots,e^{-ik\hat{\mathbf{r}}_{n_t}\cdot z}\Delta(\mathbf{r}_{n_t})\cdot\overrightarrow{\beta}_{d_t}\Big)^T, \\ \mathbf{g}^{\mu}_{r,z} = \Big(e^{-ik\hat{\mathbf{R}}_1\cdot z}\Lambda(\mathbf{R}_1)\cdot\overrightarrow{\gamma}_1,\dots,e^{-ik\hat{\mathbf{R}}_1\cdot z}\Lambda(\mathbf{R}_1)\cdot\overrightarrow{\gamma}_{d_r},\dots, \\ \qquad\qquad e^{-ik\hat{\mathbf{R}}_{n_r}\cdot z}\Lambda(\mathbf{R}_{n_r})\cdot\overrightarrow{\gamma}_1,\dots,e^{-ik\hat{\mathbf{R}}_{n_r}\cdot z}\Lambda(\mathbf{R}_{n_r})\cdot\overrightarrow{\gamma}_{d_r}\Big)^T, \\ \mathbf{g}^{\mu}_{t,z} = \Big(e^{-ik\hat{\mathbf{r}}_1\cdot z}\Lambda(\mathbf{r}_1)\cdot\overrightarrow{\beta}_1,\dots,e^{-ik\hat{\mathbf{r}}_1\cdot z}\Lambda(\mathbf{r}_1)\cdot\overrightarrow{\beta}_{d_t},\dots, \\ \qquad\qquad e^{-ik\hat{\mathbf{r}}_{n_t}\cdot z}\Lambda(\mathbf{r}_{n_t})\cdot\overrightarrow{\beta}_1,\dots,e^{-ik\hat{\mathbf{r}}_{n_t}\cdot z}\Lambda(\mathbf{r}_{n_t})\cdot\overrightarrow{\beta}_{d_t}\Big)^T. \end{cases}$$

The MSR matrix $\mathbf{A}\in\mathbb{C}^{d_rn_r\times d_tn_t}$ admits the decomposition (22), where in the definition (20) the vectors $\mathbf{g}^{\varepsilon}_{p,z},\mathbf{g}^{\mu}_{p,z},p=r,t$, are given by (23).

4. The location search algorithm. Recall that MUSIC is essentially based on characterizing the range of the MSR matrix $\mathbf{A}$, forming projections onto its null spaces, and computing its singular value decomposition. In the next subsection, we give some useful results on the eigenvalue structure of matrices of the form (22).

4.1. Some preliminaries. Let $N,N_1,N_2\in\mathbb{N}$. Suppose $N\le\min\{N_1,N_2\}$. Define the matrix $\mathbf{A}\in\mathbb{C}^{N_1\times N_2}$ as

$$\mathbf{A}=\mathbf{g}_r\,\mathbf{M}\,\mathbf{g}_t^T, \tag{24}$$

where the matrices $\mathbf{g}_r\in\mathbb{C}^{N_1\times N}$, $\mathbf{g}_t\in\mathbb{C}^{N_2\times N}$, and the matrix $\mathbf{M}\in\mathbb{C}^{N\times N}$ is symmetric and nonsingular.

Let the matrix $\mathbf{g}_r$ have rank $k_r\le N$ and let $\mathbf{g}_t$ have rank $k_t\le N$. Then the matrices G_r and $G_t\in\mathbb{C}^{N\times N}$ defined by

$$G_r=\mathbf{g}_r^*\mathbf{g}_r,\quad G_t=\mathbf{g}_t^*\mathbf{g}_t$$

have rank k_r and k_t, respectively. Here, $\mathbf{g}^*=\overline{\mathbf{g}}^T$.

Suppose $k_r=k_t=N$. The matrices G_r and G_t are invertible. The polar decompositions of the matrices $\mathbf{g}_r$ and $\mathbf{g}_t$ are written as

$$\mathbf{g}_r=U_r(G_r)^{\frac12},\quad \mathbf{g}_t=U_t(G_t)^{\frac12},$$

where $(G_r)^{\frac12}$ and $(G_t)^{\frac12}$ are positive definite, and the matrices $U_r\in\mathbb{C}^{N_1\times N}$ and $U_t\in\mathbb{C}^{N_2\times N}$ have orthonormal columns, i.e., $U_r^*U_r=U_t^*U_t=\mathbf{I}_N$. Here $\mathbf{I}_N$ is the identity matrix in $\mathbb{R}^{N\times N}$. Therefore, U_r and U_t are uniquely determined by the formulae

$$U_r=\mathbf{g}_r(G_r)^{-\frac12},\quad U_t=\mathbf{g}_t(G_t)^{-\frac12}. \tag{25}$$

From (24) we have

$$\mathbf{A}=\mathbf{g}_r\,\mathbf{M}\,\mathbf{g}_t^T=U_r\,\underbrace{(G_r)^{\frac12}\,\mathbf{M}\,(\overline{G_t})^{\frac12}}_{\Xi}\,U_t^T=U_r\,\Xi\,U_t^T,$$

with $\Xi \in \mathbb{C}^{N\times N}$. Let the singular value decomposition (SVD) of the matrix Ξ be given by $\Xi = U_\Xi\, \Sigma_\Xi\, W_\Xi^*$. Then $\mathbf{A}$ can be written as

$$\mathbf{A} = U_r U_\Xi\, \Sigma_\Xi\, (\overline{U_t} W_\Xi)^*. \tag{26}$$

Therefore, the nonzero singular values of $\mathbf{A}$ are the diagonal elements of $\Sigma_\Xi = \mathrm{diag}(\sigma_{\Xi,1}, \ldots, \sigma_{\Xi,N})$. The first N left singular vectors of $\mathbf{A}$ are the columns of the matrix $U_r U_\Xi$ and form an orthonormal left basis for the space of $\mathbf{A}$, denoted by $\mathbf{U}_S$; the first N right singular vectors of $\mathbf{A}$ are the columns of the matrix $\overline{U_t} W_\Xi$ and form an orthonormal right basis for the space of $\mathbf{A}$, denoted by $\mathbf{W}_S$. From (25) and (26) it follows that for any vector $\mathbf{a} \in \mathbb{R}^N \setminus \{0\}$ the vector $\mathbf{g}_r \cdot \mathbf{a}$ is in the range, $\mathcal{R}(\mathbf{A}\mathbf{A}^*)$, of $\mathbf{A}\mathbf{A}^*$, and for any vector $\mathbf{b} \in \mathbb{R}^N \setminus \{0\}$ the vector $\overline{\mathbf{g}_t} \cdot \mathbf{b}$ is in the range, $\mathcal{R}(\mathbf{A}^*\mathbf{A})$, of $\mathbf{A}^*\mathbf{A}$.

The orthogonal projection onto the left null space of $\mathbf{A}$ is given by

$$\mathbf{P}_r = \mathbf{I}_{N_1} - \mathbf{U}_S \mathbf{U}_S^* = \mathbf{I}_{N_1} - U_r U_\Xi (U_r U_\Xi)^* = \mathbf{I}_{N_1} - U_r U_r^* = \mathbf{I}_{N_1} - \mathbf{g}_r (G_r)^{-1} \mathbf{g}_r^*,$$

while the orthogonal projection onto the right null space of $\mathbf{A}$ is given by

$$\mathbf{P}_t = \mathbf{I}_{N_2} - \mathbf{W}_S \mathbf{W}_S^* = \mathbf{I}_{N_2} - \overline{U_t} W_\Xi (\overline{U_t} W_\Xi)^* \mathbf{I}_{N_2} - \overline{U_t} {U_t}^T = \mathbf{I}_{N_2} - \overline{\mathbf{g}_t} (\overline{G_t})^{-1} {\mathbf{g}_t}^T.$$

Suppose now that $k_r \le k_t < N$. The matrices G_r and G_t are singular. Suppose throughout this subsection that the ranges of the matrices

$$\mathbf{g}_r = [g_{r,1}, \ldots, g_{r,k_r}, g_{r,k_r+1}, \ldots g_{r,N}] \text{ and } \mathbf{g}_t = [g_{t,1}, \ldots, g_{t,k_t}, g_{t,k_t+1}, \ldots, g_{t,N}]$$

are spanned by the matrices' first k_r and k_t columns denoted by $\mathbf{g}_{r,S} \in \mathbb{C}^{N_1\times k_r}$ and $\mathbf{g}_{t,S} \in \mathbb{C}^{N_2\times k_t}$, respectively. Then

$$g_{r,j} = \sum_{i=1}^{k_r} \alpha_{i,j} g_{r,i}, \quad j = k_r+1, \ldots, N, \quad \text{and} \quad g_{t,l} = \sum_{i=1}^{k_t} \beta_{i,l} g_{t,i}, \quad l = k_t+1, \ldots, N,$$

where $\{\alpha_{i,j}\}$ and $\{\beta_{i,l}\} \in \mathbb{C}$. Now the matrix $\mathbf{A}$, defined by (24), can be rewritten as

$$\mathbf{A} = \mathbf{g}_{r,S}\, \mathbf{M}_S\, \mathbf{g}_{t,S}^T,$$

where $\mathbf{M}_S \in \mathbb{C}^{k_r\times k_t}$. The elements of $\mathbf{M}_S$ depend on the elements of the matrix $\mathbf{M}$ and the coefficients $\{\alpha_{i,j}\}$ and $\{\beta_{i,l}\}$.

Suppose $k_r = k_t = k < N$. The square matrix $\mathbf{M}_S$ is nonsingular but not always symmetric. The calculation of the SVD of $\mathbf{A}$ is similar to the case where $k_r = k_t = k = N$. It can be proved that for any vector $\mathbf{a} = (a_1, \ldots, a_N)^T \in \mathbb{R}^N \setminus \{0\}$ such that $\mathbf{g}_r \cdot \mathbf{a} \neq 0$, the vector $\mathbf{g}_r \cdot \mathbf{a}$ is in the range of $\mathbf{A}\mathbf{A}^*$. Analogously, for any vector $\mathbf{b} = (b_1, \ldots, b_N)^T \in \mathbb{R}^N \setminus \{0\}$ such that $\overline{\mathbf{g}_t} \cdot \mathbf{b} \neq 0$, the vector $\overline{\mathbf{g}_t} \cdot \mathbf{b}$ is in the range of $\mathbf{A}^*\mathbf{A}$. Here

$$\mathbf{g}_r \cdot \mathbf{a} = \mathbf{g}_{r,S} \cdot \mathbf{a}_r, \quad \overline{\mathbf{g}_t} \cdot \mathbf{b} = \overline{\mathbf{g}_{t,S}} \cdot \mathbf{b}_t, \tag{27}$$

where

$$\mathbf{a}_r = \left(a_1 + \sum_{j=k+1}^{N} a_j\, \alpha_{1,j}, \ldots, a_k + \sum_{j=k+1}^{N} a_j\, \alpha_{j,k} \right)^T \in \mathbb{C}^k,$$

$$\mathbf{b}_t = \left(b_1 + \sum_{j=k+1}^{N} b_j\, \beta_{1,j}, \ldots, b_k + \sum_{j=k+1}^{N} b_j\, \beta_{j,k} \right)^T \in \mathbb{C}^k.$$

Suppose $k_r < k_t \leq N$. Then

$$\mathbf{A} = \mathbf{g}_{r,S}\,\mathbf{M}_S\,\mathbf{g}_{t,S}^T = \mathbf{g}_{r,S}\,(\mathbf{g}_{t,S}\,\mathbf{M}_S^T)^T = \mathbf{g}_{r,S}\,\mathbf{f}_{t,S}^T,$$

with $\mathbf{f}_{t,S} = \mathbf{g}_{t,S}{\mathbf{M}_S}^T \in \mathbb{C}^{N_2\times k_r}$. Again, define

$$\mathbf{g}_{rS} = U_{r,S}(\mathbf{g}_{r,S}^*\mathbf{g}_{r,S})^{\frac{1}{2}} = U_{r,S}(G_{r,S})^{\frac{1}{2}}, \quad \mathbf{f}_{t,S} = U_{t,S}\,(F_{t,S})^{\frac{1}{2}},$$

where the matrix

$$F_{t,S} = \overline{\mathbf{M}_S}\,\mathbf{g}_{t,S}^*\,\mathbf{g}_{t,S}{\mathbf{M}_S}^T = \overline{\mathbf{M}_S}\,G_{t,S}{\mathbf{M}_S}^T$$

has rank k_r. Then $U_{r,S}$ and $U_{t,S}$ are uniquely determined by

$$U_{r,S} = \mathbf{g}_{r,S}(G_{r,S})^{-\frac{1}{2}}, \quad U_{t,S} = \mathbf{f}_{t,S}(F_{t,S})^{-\frac{1}{2}}, \tag{28}$$

and the matrix $\mathbf{A}$ becomes

$$\mathbf{A} = U_{r,S}\,\underbrace{(G_{r,S})^{\frac{1}{2}}(F_{t,S})^{\frac{1}{2}}}_{\Xi = U_\Xi\,\Sigma_\Xi\,W_\Xi^*}\,U_{t,S}^T = U_{r,S}U_\Xi\,\Sigma_\Xi\,(\overline{U_{t,S}}\,W_\Xi)^*. \tag{29}$$

Once again, the first N left singular vectors of $\mathbf{A}$ are the columns of the matrix $U_{r,S}U_\Xi$; the first N right singular vectors of $\mathbf{A}$ are the columns of the matrix $\overline{U_{t,S}}W_\Xi$. From (27), (28), and (29) it follows that for any vector $\mathbf{a} \in \mathbb{R}^N \setminus \{0\}$ such that $\mathbf{g}_r \cdot \mathbf{a} \neq 0$, the vector $\mathbf{g}_r \cdot \mathbf{a}$ is in the range of $\mathbf{A}\mathbf{A}^*$, but the vector $\overline{\mathbf{g}_t} \cdot \mathbf{a}$ does not belong to the range of $\mathbf{A}^*\mathbf{A}$.

The orthogonal projection onto the left null space of $\mathbf{A}$ is given by

$$\mathbf{P}_r = \mathbf{I}_{N_1} - U_{r,S}U_\Xi(U_{r,S}U_\Xi)^* = \mathbf{I}_{N_1} - U_{r,S}U_{r,S}^* = \mathbf{I}_{N_1} - \mathbf{g}_{r,S}(G_{r,S})^{-1}\mathbf{g}_{r,S}^*,$$

while the orthogonal projection onto the right null space of $\mathbf{A}$ is given by

$$\begin{aligned}\mathbf{P}_t &= \mathbf{I}_{N_2} - \overline{U_{t,S}}W_\Xi(\overline{U_{t,S}}W_\Xi)^* = \mathbf{I}_{N_2} - \overline{U_{t,S}}{U_{t,S}}^T = \mathbf{I}_{N_2} - \overline{\mathbf{f}_{t,S}}(\overline{F_{t,S}})^{-1}{\mathbf{f}_{t,S}}^T \\ &= \mathbf{I}_{N_2} - \overline{\mathbf{g}_{t,S}}\mathbf{M}_S^*(\mathbf{M}_S\,\overline{G_{t,S}}\mathbf{M}_S^*)^{-1}\,\mathbf{M}_S\mathbf{g}_{t,S}^T.\end{aligned}$$

If $k_t < k_r \leq N$, we consider $\mathbf{A}^T$ instead of $\mathbf{A}$.

Finally, since the matrix $\mathbf{M}$ is nonsingular, the rank of $\mathbf{A}$ (the number of nonzero singular values of $\mathbf{A}$) depends only on the rank of the matrices G_r and G_t. Denote by $\hat{G}_r$ (and $\hat{G}_t$) a matrix of k_r (and k_t) linearly independent rows of G_r (and G_t), respectively. Suppose that the matrix $\hat{G}_r\mathbf{M}\hat{G}_t^T$ has rank $\min\{k_r, k_t\} \leq N$. Then $\operatorname{rank}(\mathbf{A}) = \min\{k_r, k_t\}$.

We have established the following result.

PROPOSITION 4.1. $\mathbf{A} = \mathbf{g}_r\,\mathbf{M}\,\mathbf{g}_t^T \in \mathbb{C}^{N_1\times N_2}$, $\mathbf{g}_r \in \mathbb{C}^{N_1\times N}$ $\mathbf{g}_t \in \mathbb{C}^{N_2\times N}$ $N \leq \min\{N_1, N_2\}$ $\mathbf{M} \in \mathbb{C}^{N\times N}$ $(\mathbf{g}_r) = k_r \leq N$ $(\mathbf{g}_t) = k_t \leq N$ $\hat{\mathbf{g}}_r$ $\hat{\mathbf{g}}_t$ k_r k_t $\mathbf{g}_r$ $\mathbf{g}_t$ $\hat{\mathbf{g}}_r\,\mathbf{M}\,\hat{\mathbf{g}}_t^T$ $\min\{k_r, k_t\}$ $\mathbf{A}$ $\min\{k_r, k_t\} \leq N$

- $k_r = k_t \leq N$ $\mathbf{a}, \mathbf{b} \in \mathbb{R}^N \setminus \{0\}$ $\mathbf{g}_r \cdot \mathbf{a} \neq 0$ $\overline{\mathbf{g}_t} \cdot \mathbf{b} \neq 0$ $\mathbf{g}_r \cdot \mathbf{a} \in \mathcal{R}(\mathbf{A}\mathbf{A}^*)$ $\overline{\mathbf{g}_t} \cdot \mathbf{b} \in \mathcal{R}(\mathbf{A}^*\mathbf{A})$.
- $k_r < k_t \leq N$ $\mathbf{a} \in \mathbb{R}^N \setminus \{0\}$ $\mathbf{g}_r \cdot \mathbf{a} \neq 0$, $\mathbf{g}_r \cdot \mathbf{a} \in \mathcal{R}(\mathbf{A}\mathbf{A}^*)$.
- $k_t < k_r \leq N$ $\mathbf{b} \in \mathbb{R}^N \setminus \{0\}$ $\overline{\mathbf{g}_t} \cdot \mathbf{b} \neq 0$ $\overline{\mathbf{g}_t} \cdot \mathbf{b} \in \mathcal{R}(\mathbf{A}^*\mathbf{A})$

4.2. MUSIC characterization of the multistatic response matrix. Now we return to the decomposition (22) of the MSR matrix $\mathbf{A} \in \mathbb{C}^{d_r n_r \times d_t n_t}$ and remember that we are considering separately the three (electromagnetic) cases.

Let $\min(d_r n_r, d_t n_t) \geq 3$ (cases 1 and 2) and $\min(d_r n_r, d_t n_t) \geq 6$ (case 3). For any $z \in \mathbb{R}^3 \setminus \{\mathbf{r}_1, \ldots, \mathbf{r}_{n_t}, \mathbf{R}_1, \ldots, \mathbf{R}_{n_r}\}$ (or $z \in \mathbb{R}^3$ for the far-field regime) define

$$G_{p,z} = \overline{\mathbf{g}_{p,z}}^T \mathbf{g}_{p,z}, \quad p = r, t,$$

where $\mathbf{g}_{p,z}$ is given by (20) with $\mathbf{g}^{\varepsilon}_{p,z}, \mathbf{g}^{\mu}_{p,z}, p = r, t$, defined by either (19) or (23). The matrix $G_{p,z}$ has rank k_p, where

$$\begin{aligned}
&\text{case 1:} \quad G_{p,z} = G^{\varepsilon}_{p,z} = \overline{\mathbf{g}^{\varepsilon}_{p,z}}^T \mathbf{g}^{\varepsilon}_{p,z}, \quad \operatorname{rank}(G^{\varepsilon}_{p,z}) = k^{\varepsilon}_p = k_p \leq 3,\\
&\text{case 2:} \quad G_{p,z} = G^{\mu}_{p,z} = \overline{\mathbf{g}^{\mu}_{p,z}}^T \mathbf{g}^{\mu}_{p,z}, \quad \operatorname{rank}(G^{\mu}_{p,z}) = k^{\mu}_p = k_p \leq 3,\\
&\text{case 3:} \quad G_{p,z} = G^{\mu,\varepsilon}_{p,z} = \overline{[\mathbf{g}^{\mu}_{p,z}, \mathbf{g}^{\varepsilon}_{p,z}]}^T [\mathbf{g}^{\mu}_{p,z}, \mathbf{g}^{\varepsilon}_{p,z}],\\
&\qquad\quad \operatorname{rank}(G^{\mu,\varepsilon}_{p,z}) = k^{\mu,\varepsilon}_p = k_p \leq 6, \quad p = t, r.
\end{aligned} \tag{30}$$

Moreover, in case 3, we have $k^{\mu,\varepsilon}_p = k^{\mu}_p + k^{\varepsilon}_p$.

For the far-field pattern, it is easy to see that the following proposition holds since $\overline{G_{p,z}}^T G_{p,z}$ is z-independent.

PROPOSITION 4.2. [illegible] $G_{p,z}$ [illegible] (30) [illegible] $z \in \mathbb{R}^3$

Propositions 4.1 and 4.2 together with (22) yield that the maximal rank of $\mathbf{A}$ is $m \min(k_r, k_t)$.

For the near-field regime, the rank k_p depends on the configuration (transmitter and receiver arrays), the polarizations, and the position z. If we suppose that the transmitter and receiver arrays are planar, then by the far-field expansion of the Green's function we can see that the claim in Proposition 4.2 holds for z far enough from the set of points $\{\mathbf{r}_1, \ldots, \mathbf{r}_{n_t}, \mathbf{R}_1, \ldots, \mathbf{R}_{n_r}\}$. See section 5.

The following is a MUSIC characterization of the locations of the electromagnetic inclusions [32, 21] and is valid if the numbers n_r and n_t are sufficiently large. See Appendix B for the statement of some technical assumptions which are necessary to rigorously prove the following proposition.

PROPOSITION 4.3. [illegible] $\min(d_r n_r, d_t n_t) \geq m \min(k_r, k_t)$ [illegible] k_r, k_t [illegible] (30)

- [illegible] $k_r = k_t$ [illegible] $\mathbf{a} \in \mathbb{R}^N \setminus \{0\}$ [illegible] $\mathbf{g}_{r,z} \cdot \mathbf{a} \neq 0$ [illegible] z [illegible]

$$\mathbf{g}_{r,z} \cdot \mathbf{a} \in \mathcal{R}(\mathbf{A}\mathbf{A}^*) \quad \text{for} \quad z \in \{z_1, \ldots, z_m\}.$$

[illegible] $\mathbf{b} \in \mathbb{R}^N \setminus \{0\}$ [illegible] $\overline{\mathbf{g}_{t,z}} \cdot \mathbf{b} \neq 0$ [illegible] z [illegible]

$$\overline{\mathbf{g}_{t,z}} \cdot \mathbf{b} \in \mathcal{R}(\mathbf{A}^*\mathbf{A}) \quad \text{for} \quad z \in \{z_1, \ldots, z_m\}.$$

- [illegible] $k_r < k_t$ [illegible] $\mathbf{a} \in \mathbb{R}^N \setminus \{0\}$ [illegible] $\mathbf{g}_{r,z} \cdot \mathbf{a} \neq 0$ [illegible] z [illegible]

$$\mathbf{g}_{r,z} \cdot \mathbf{a} \in \mathcal{R}(\mathbf{A}\mathbf{A}^*) \quad \text{for} \quad z \in \{z_1, \ldots, z_m\}.$$

• [illegible] $k_t < k_r$ [illegible] $\mathbf{b} \in \mathbb{R}^N \setminus \{0\}$ [illegible] $\overline{\mathbf{g}_{t,z}} \cdot \mathbf{b} \neq 0$ [illegible] z

$$\overline{\mathbf{g}_{t,z}} \cdot \mathbf{b} \in \mathcal{R}(\mathbf{A}^*\mathbf{A}) \quad \text{for} \quad z \in \{z_1, \ldots, z_m\}.$$

[illegible] N [illegible] $\mathbf{a}$ [illegible] $\mathbf{b}$ [illegible] 3 [illegible] 1 [illegible] 2 [illegible] 6 [illegible] 3

Note that, in view of Proposition 4.2, if a vector $\mathbf{a} \in \mathbb{R}^N \setminus \{0\}$ satisfies $\mathbf{g}_{p,0} \cdot \mathbf{a} \neq 0$, then $\mathbf{g}_{p,z} \cdot \mathbf{a} \neq 0$ for any point z (in the searching domain).

4.3. The MUSIC-type algorithm. Let the SVD of the matrix $\mathbf{A}$ be defined by $\mathbf{A} = \mathbf{U}\Sigma\mathbf{W}^*$. From Proposition 4.3, $\mathbf{A}$ has rank $m\,(\min(k_r, k_t))$. Then, as has been said, the first $m\,(\min(k_r, k_t))$ columns of $\mathbf{W}$ provide an orthonormal right basis for the image space of $\mathbf{A}$ which is denoted by $\mathbf{W}_S$, and the rest of the matrix $\mathbf{W}$ provides the basis for the right null space of $\mathbf{A}$, denoted by $\mathbf{W}_N$. Then the best rank approximation for $\mathbf{A}$ is $(\mathbf{W}_S\mathbf{W}_S^*)\mathbf{A}$ with error $E = \mathbf{A} - (\mathbf{W}_S\mathbf{W}_S^*)\mathbf{A}$. See [3, 4].

The orthogonal projections onto the left and right null space of $\mathbf{A}$ are given by $\mathbf{P}_r = \mathbf{I} - (\mathbf{U}_S\mathbf{U}_S^*)$ and $\mathbf{P}_t = \mathbf{I} - (\mathbf{W}_S\mathbf{W}_S^*)$. If $k_r = k_t$, a test point z coincides with one of the positions z_j if and only if $\mathbf{P}_r(\mathbf{g}_{r,z} \cdot \mathbf{a}) = 0$ and $\mathbf{P}_t(\overline{\mathbf{g}_{t,z}} \cdot \mathbf{a}) = 0$; if $k_r < k_t$, then a test point z coincides with one of the positions z_j if and only if $\mathbf{P}_r(\mathbf{g}_{r,z} \cdot \mathbf{a}) = 0$; if $k_t < k_r$, then a test point z coincides with one of the positions z_j if and only if $\mathbf{P}_t(\overline{\mathbf{g}_{t,z}} \cdot \mathbf{a}) = 0$.

Thus we can form an image of $z_j, j = 1, \ldots, m$, by plotting, at each point z, the following quantities:

- if $k_r = k_t$: $W_r(z) = 1/||\mathbf{P}_r(\mathbf{g}_{r,z} \cdot \mathbf{a})||$, or $W_t(z) = 1/||\mathbf{P}_t(\overline{\mathbf{g}_{t,z}} \cdot \mathbf{a})||$, or $W_{r,t}(z) = 1/||\mathbf{P}_r(\mathbf{g}_{r,z} \cdot \mathbf{a}_r) + \mathbf{P}_t(\overline{\mathbf{g}_{t,z}} \cdot \mathbf{a}_t)||$;
- if $k_r < k_t$: $W_r(z) = 1/||\mathbf{P}_r(\mathbf{g}_{r,z} \cdot \mathbf{a})||$;
- if $k_t < k_r$: $W_t(z) = 1/||\mathbf{P}_t(\overline{\mathbf{g}_{t,z}} \cdot \mathbf{a})||$.

We shall emphasize the fact that this algorithm can be used to recover the location of m well-separated electromagnetic inclusions provided that $\min(d_r n_r, d_t n_t) > m\,(\min(k_r, k_t))$. The obtained images do not depend on the shape of the inclusions.

Note also that once the location z_0 of the inclusion is found, we can approximate its polarization tensor. In the simplest case, $k_r = k_t = N$ ($N = 3$ in cases 1 and 2 and $N = 6$ in case 3), $m = 1$, and

$$\mathbf{M} = (G_{r,z_0})^{-1}(\mathbf{g}_{r,z_0})^*\, \mathbf{A}\, \overline{\mathbf{g}_{t,z_0}}\, (\overline{G_{t,z_0}})^{-1}.$$

In the case of multiple inclusions, if $z_1, \ldots, z_m$ are the reconstructed locations, then

$$\mathbf{M} = \operatorname{diag}(\mathbf{M}_1, \ldots, \mathbf{M}_m) = (G_{r,z_1,\ldots,z_m})^{-1}(\mathbf{g}_{r,z_1,\ldots,z_m})^*\, \mathbf{A}\, \overline{\mathbf{g}_{t,z_1,\ldots,z_m}}\, (\overline{G_{t,z_1,\ldots,z_m}})^{-1}.$$

We can represent and visualize the inclusion by means of an ellipsoid with the same polarization tensor which is the only geometric information of real interest that could be reconstructed from the MSR matrix. It is also worth mentioning that the polarization tensor yields information about the size and orientation of the inclusion. However, this information is a mixture of the electromagnetic parameters and the volume. See [9] for details.

5. Analysis of the eigenvalue structure of the multistatic response matrix. Suppose that the array of transmitters is in the plane $z = z_a$ and is symmetric about the ξ and η axes (see Figure 5.1). We make the same assumption for the array of receivers (not necessarily the same plane $z = z_a$ of the transmitters; the array of receivers can be in the plane $x = x_a$ or $y = y_a$).

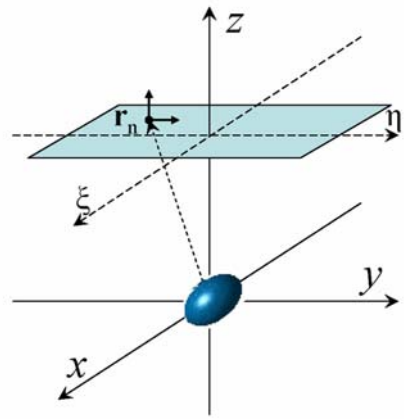


FIG. 5.1. *The planar array of transmitters.*

Suppose that the inclusions are ellipsoids. Then, as already noted,

$$\mathbf{M}^{\varepsilon} = R \begin{bmatrix} m_1^{\varepsilon} & 0 & 0 \\ 0 & m_2^{\varepsilon} & 0 \\ 0 & 0 & m_3^{\varepsilon} \end{bmatrix} R^T, \quad \mathbf{M}^{\mu} = R \begin{bmatrix} m_1^{\mu} & 0 & 0 \\ 0 & m_2^{\mu} & 0 \\ 0 & 0 & m_3^{\mu} \end{bmatrix} R^T,$$

where $m_i^{\varepsilon}, m_i^{\mu} \in \mathbb{C}$, $i = 1, 2, 3$, and $R \in \mathbb{R}^{3\times 3}$ is a rotation matrix. Note that $R = \mathbf{I}_3$ if the ellipsoid is aligned with the coordinate axes. See [34, 26, 9].

5.1. Case of a single dielectric inclusion. We first consider the case of a single dielectric inclusion ($m = 1$ and $\mu_1 = \mu_0$). Suppose that this inclusion is centered at the point z, so it is of the form αB, where B is an ellipsoid. Then

$$\mathbf{A} = \mathbf{g}_{r,z}^{\varepsilon} \mathbf{M}^{\varepsilon} (\mathbf{g}_{t,z}^{\varepsilon})^T,$$

and the positive semidefinite matrix $G_{r,z}^{\varepsilon}$ is given, for the near-field regime, by

$$G_{r,z}^{\varepsilon} = (\overline{\mathbf{g}_{r,z}^{\varepsilon}})^T \mathbf{g}_{r,z}^{\varepsilon} = \sum_{l=1}^{d_r} \sum_{j=1}^{n_r} \overline{\mathbf{G}(z, \mathbf{R}_j)} \, (\overrightarrow{\gamma}_l \, \overrightarrow{\gamma}_l^T) \, \mathbf{G}(\mathbf{R}_j, z),$$

where

$$\mathbf{G}(\mathbf{r}, \mathbf{r}') = \frac{e^{ik|s|}}{4\pi|s|} \left(\mathbf{I}_3 - \hat{s}\hat{s}^T + \frac{i}{k|s|} \left(\mathbf{I}_3 - 3\,\hat{s}\hat{s}^T \right) - \frac{1}{k^2|s|^2} \left(\mathbf{I}_3 - 3\,\hat{s}\hat{s}^T \right) \right), \quad s = \mathbf{r} - \mathbf{r}'.$$

If $|z| \to +\infty$, then using the far-field expansion of the Green's function $\mathbf{G}$ we obtain that $G_{r,z}^{\varepsilon}$ can be approximated by

$$G_{r,\infty}^{\varepsilon} = \sum_{l=1}^{d_r} \sum_{j=1}^{n_r} \Delta(\mathbf{R}_j) \, (\overrightarrow{\gamma}_l \, \overrightarrow{\gamma}_l^T) \, \Delta(\mathbf{R}_j),$$

which does not depend on the point z.

In the limit that the number of array elements becomes large and the spacing between elements small, the inner products can be replaced with integrals over the

aperture of the array [19]. Consider, for example, the far-field expansion of $G^{\varepsilon}_{r,z}$; we have

$$G^{\varepsilon}_{r,\infty} = \sum_{l=1}^{d_r} \int_{S_r} \Delta(\xi,\eta)\,(\overrightarrow{\gamma}_l\,\overrightarrow{\gamma}_l^{T})\,\Delta(\xi,\eta)\,d\xi\,d\eta, \tag{31}$$

where S_r and S_t are the aperture of the arrays of transmitters and receivers, respectively.

Writing $G^{\varepsilon}_{r,\infty} = \sum_{j=1}^{d_r} G^{\varepsilon,j}_{r,\infty}$, we have

$$\overrightarrow{\gamma}_1 = \mathbf{e}_z: \quad G^{\varepsilon,1}_{r,\infty} = \int_{S_r} \frac{1}{r^4} \begin{bmatrix} \xi^2 z_a^2 & \xi\eta z_a^2 & \xi z_a(z_a^2 - r^2) \\ \xi\eta z_a^2 & \eta^2 z_a^2 & \eta z_a(z_a^2 - r^2) \\ \xi z_a(z_a^2 - r^2) & \eta z_a(z_a^2 - r^2) & (z_a^2 - r^2)^2 \end{bmatrix} d\xi d\eta, \tag{32}$$

$$\overrightarrow{\gamma}_2 = \mathbf{e}_y: \quad G^{\varepsilon,2}_{r,\infty} = \int_{S_r} \frac{1}{r^4} \begin{bmatrix} \xi^2\eta^2 & \xi\eta(\eta^2 - r^2) & \xi\eta^2 z_a \\ \xi\eta(\eta^2 - r^2) & (\eta^2 - r^2)^2 & \eta z_a(\eta^2 - r^2) \\ \xi\eta^2 z_a & \eta z_a(\eta^2 - r^2) & \eta^2 z_a^2 \end{bmatrix} d\xi d\eta, \tag{33}$$

$$\overrightarrow{\gamma}_3 = \mathbf{e}_x: \quad G^{\varepsilon,3}_{r,\infty} = \int_{S_r} \frac{1}{r^4} \begin{bmatrix} (\xi^2 - r^2)^2 & \xi\eta(\xi^2 - r^2) & \xi z_a(\xi^2 - r^2) \\ \xi\eta(\xi^2 - r^2) & \xi^2\eta^2 & \xi^2\eta z_a \\ \xi z_a(\xi^2 - r^2) & \xi^2\eta z_a & \xi^2 z_a^2 \end{bmatrix} d\xi d\eta, \tag{34}$$

where $r = \sqrt{\xi^2 + \eta^2 + z_a^2}$. We conclude that for a symmetric array of receivers (symmetric in ξ and η) the matrices $G^{\varepsilon,j}_{r,\infty}$, $j = 1,2,3$, are diagonal with positive elements on the diagonal, which implies that $\operatorname{rank}(G^{\varepsilon}_{r,\infty}) = 3$. Or, equivalently,

$$G^{\varepsilon}_{r,\infty} = \begin{bmatrix} G^{\varepsilon}_{r,11} & 0 & 0 \\ 0 & G^{\varepsilon}_{r,22} & 0 \\ 0 & 0 & G^{\varepsilon}_{r,33} \end{bmatrix},$$

where $G^{\varepsilon}_{r,ij} = (\overline{\mathbf{g}^{\varepsilon,i}_{r,\infty}})^T \mathbf{g}^{\varepsilon,j}_{r,\infty}$, $i,j = 1,2,3$, and the orthogonal vectors $\mathbf{g}^{\varepsilon,j}_{r,\infty} \in \mathbb{C}^{d_r n_r}$ are the column vectors of the matrix $\mathbf{g}^{\varepsilon}_{r,\infty} \in \mathbb{C}^{d_r n_r \times 3}$, which is the leading-order term in the far-field expansion of $\mathbf{g}^{\varepsilon}_{r,z}$.

With the same notation, the same result holds for the matrix $G^{\varepsilon}_{t,z}$: if the array of receivers is symmetric and planar, then we have

$$G^{\varepsilon}_{t,\infty} = \begin{bmatrix} G^{\varepsilon}_{t,11} & 0 & 0 \\ 0 & G^{\varepsilon}_{t,22} & 0 \\ 0 & 0 & G^{\varepsilon}_{t,33} \end{bmatrix},$$

where $G^{\varepsilon}_{t,ij} = (\overline{\mathbf{g}^{\varepsilon,i}_{t,\infty}})^T \mathbf{g}^{\varepsilon,j}_{t,\infty}$, $i,j = 1,2,3$. Here the vectors $\mathbf{g}^{\varepsilon,j}_{t,\infty} \in \mathbb{C}^{d_t n_t}$ are the column vectors of the matrix $\mathbf{g}^{\varepsilon}_{t,\infty} \in \mathbb{C}^{d_t n_t \times 3}$.

We conclude that the matrices $G^{\varepsilon}_{r,\infty}$ and $G^{\varepsilon}_{t,\infty}$ are positive definite and diagonal.

For finite $|z|$, the same results as for $|z| \to +\infty$ hold if the inclusion is centered at the origin. However, if the inclusion is located at a point $z \neq 0$, the matrices $G^{\varepsilon}_{r,z}$ and $G^{\varepsilon}_{t,z}$ are positive definite but not diagonal. By the Taylor expansion of the scalar Green's function $g(\mathbf{r},z) = e^{ik|\mathbf{r}-z|}/(4\pi|\mathbf{r}-z|)$ for $|z|/|\mathbf{r}| < 1$, we can approximate the matrices $G^{\varepsilon}_{p,z}, p = r,t$, to find that they are symmetric with strictly dominant diagonals, which implies that they are invertible.

Now, once the rank of the matrices $G^{\varepsilon}_{r,z}$ and $G^{\varepsilon}_{t,z}$ is known, we can compute the SVD of $\mathbf{A}$ following subsection 4.1. Let $k^{\varepsilon}_r = k^{\varepsilon}_t = 3$. If the inclusion is centered at the origin ($z = 0$) or if $|z|$ is large enough, we have

$$U_r = \mathbf{g}^{\varepsilon}_{r,0}(G^{\varepsilon}_{r,0})^{-\frac{1}{2}}, \quad U_t = \mathbf{g}^{\varepsilon}_{t,0}(G^{\varepsilon}_{t,0})^{-\frac{1}{2}};$$

i.e., the columns of U_r and U_t are normalized column vectors of $\mathbf{g}^{\varepsilon}_{r,0}$ and $\mathbf{g}^{\varepsilon}_{t,0}$, respectively. The nonzero singular values of $\mathbf{A}$ are the singular values of

$$\Xi = (G^{\varepsilon}_{r,0})^{\frac{1}{2}}\mathbf{M}^{\varepsilon}(G^{\varepsilon}_{t,0})^{\frac{1}{2}} = U_{\Xi}\,\Sigma_{\Xi}\,W^{*}_{\Xi}$$

with corresponding singular vectors $U_r U_{\Xi} = \mathbf{g}^{\varepsilon}_{r,0}(G^{\varepsilon}_{r,0})^{-\frac{1}{2}}U_{\Xi}$ and provide an orthonormal left basis of $\mathbf{A}$. The singular vectors $\overline{U_t}W_{\Xi} = \overline{\mathbf{g}^{\varepsilon}_{t,0}}(G^{\varepsilon}_{t,0})^{-\frac{1}{2}}W_{\Xi}$ provide an orthonormal right basis of $\mathbf{A}$.

If our inclusion is an ellipsoid located at the origin with $R = \mathbf{I}_3$, writing $m^{\varepsilon}_j = |m^{\varepsilon}_j|e^{i\theta_j}$ we obtain that

$$\sigma_{\Xi,j} = |m^{\varepsilon}_j|\,(G^{\varepsilon}_{r,0,jj})^{\frac{1}{2}}(G^{\varepsilon}_{t,0,jj})^{\frac{1}{2}},\ j = 1,2,3, \quad U_{\Xi} = \mathrm{diag}(e^{i\theta_1}, e^{i\theta_2}, e^{i\theta_3}), \quad W_{\Xi} = \mathbf{I}_3.$$

It is worth noticing that for the imaging procedure, which can be developed independently of the nature (and geometry) of the inclusions and does rely on the MUSIC-type approach, we do not need to compute the rank of the matrices $G^{\varepsilon}_{r,z}$ and $G^{\varepsilon}_{t,z}$. We shall distinguish between the analysis of the main ingredients of the MSR matrix, which enables us to better understand what to expect from the imaging procedure, and the imaging procedure itself.

5.2. Case of a single permeable inclusion. If we consider the case of a single magnetic inclusion ($m = 1$ and $\varepsilon_1 = \varepsilon_0$), then

$$\mathbf{A} = \mathbf{g}^{\mu}_{r,z}\mathbf{M}^{\mu}(\mathbf{g}^{\mu}_{t,z})^{T},$$

and the matrix $G^{\mu}_{r,z}$ is given by

$$G^{\mu}_{r,z} = (\overline{\mathbf{g}^{\mu}_{r,z}})^{T}\mathbf{g}^{\mu}_{r,z} = \sum_{l=1}^{d_r}\sum_{j=1}^{n_r} -\overline{\nabla{\times}\mathbf{G}(z,\mathbf{R}_j)}\,(\overrightarrow{\gamma}_l\,\overrightarrow{\gamma}_l^{T})\,\nabla'{\times}\mathbf{G}(\mathbf{R}_j,z),$$

where

$$\nabla{\times}\mathbf{G}(\mathbf{r},\mathbf{r}')\cdot\mathbf{a} = \left(ik - \frac{1}{|s|}\right)\frac{e^{ik|s|}}{4\pi|s|}\,\Lambda(s)\cdot\mathbf{a}, \quad s = \mathbf{r}-\mathbf{r}'.$$

If $|z| \to +\infty$, then $G^{\mu}_{r,z}$ can be approximated by

$$G^{\mu}_{r,\infty} = -\sum_{l=1}^{d_r}\sum_{j=1}^{n_r}\Lambda(\mathbf{R}_j)\,(\overrightarrow{\gamma}_l\,\overrightarrow{\gamma}_l^{T})\,\Lambda(\mathbf{R}_j).$$

Again, in the limit that the number of array elements becomes large and the spacing between elements small, the inner products can be replaced with integrals over the aperture of the array. Similarly to the previous subsection, if we consider the far-field expansion of $G^{\mu}_{r,z}$ and write

$$G^{\mu}_{r,\infty} = \sum_{l=1}^{d_r} G^{\mu,l}_{r,\infty},$$

then

$$\overrightarrow{\gamma}_1 = \mathbf{e}_z: \quad G^{\mu,1}_{r,\infty} = \int_{S_r} \frac{1}{r^2} \begin{bmatrix} \eta^2 & -\xi\eta & 0 \\ -\xi\eta & \xi^2 & 0 \\ 0 & 0 & 0 \end{bmatrix} d\xi d\eta,$$

$$\overrightarrow{\gamma}_2 = \mathbf{e}_y: \quad G^{\mu,2}_{r,\infty} = \int_{S_r} \frac{1}{r^2} \begin{bmatrix} z_a^2 & 0 & -z_a\xi \\ 0 & 0 & 0 \\ -z_a\xi & 0 & \xi^2 \end{bmatrix} d\xi d\eta,$$

$$\overrightarrow{\gamma}_3 = \mathbf{e}_x: \quad G^{\mu,3}_{r,\infty} = \int_{S_r} \frac{1}{r^2} \begin{bmatrix} 0 & 0 & 0 \\ 0 & z_a^2 & -z_a\eta \\ 0 & -z_a\eta & \eta^2 \end{bmatrix} d\xi d\eta,$$

where $r = \sqrt{\xi^2 + \eta^2 + z_a^2}$. We can see that for a symmetric array and for any two (nonparallel) directions $\overrightarrow{\gamma}$ the matrix $G^{\mu}_{r,\infty}$ is diagonal with $\operatorname{rank}(G^{\mu}_{r,\infty}) = 3$. The diagonal elements are the squares of the norms of the orthogonal column vectors of the matrix $\mathbf{g}^{\mu}_{r,\infty}$, which is the leading-order term in the far-field expansion of $\mathbf{g}^{\mu}_{r,z}$. The same results hold for the matrix $G^{\mu}_{t,\infty}$.

Now, analogously to the previous subsection, we can estimate the rank of the matrices $G^{\mu}_{r,\infty}$. Thus, for a permeable inclusion located far enough from the set of the points $\{\mathbf{R}_1, \ldots, \mathbf{R}_{n_r}\}$, the matrix $G^{\mu}_{r,\infty}$ has rank 3 for any $d_r \geq 2$ directions $\overrightarrow{\gamma}$ and has rank 2 for any single direction $\overrightarrow{\gamma}$, $\overrightarrow{\gamma} \in \{\mathbf{e}_z, \mathbf{e}_y, \mathbf{e}_x\}$. Moreover, if $z = 0$, then the vector columns of $\mathbf{g}^{\mu}_{r,0}$ are orthogonal. The same conclusion holds for a transmitter array.

Hence, the singular values and the singular vectors of the matrix $\mathbf{A}$ can be found analogously to the case of a single dielectric inclusion.

5.3. Case of a single dielectric and permeable inclusion or perfectly conducting inclusion. In this case,

$$\mathbf{A} = [\mathbf{g}^{\mu}_{r,z}, \mathbf{g}^{\varepsilon}_{r,z}] \mathbf{M}^{\mu,\varepsilon} [\mathbf{g}^{\mu}_{r,z}, \mathbf{g}^{\varepsilon}_{r,z}]^T,$$

and the matrix $G^{\mu,\varepsilon}_{r,z}$ is given by

$$G^{\mu,\varepsilon}_{r,z} = [\overline{\mathbf{g}^{\mu}_{r,z}, \mathbf{g}^{\varepsilon}_{r,z}}]^T [\mathbf{g}^{\mu}_{r,z}, \mathbf{g}^{\varepsilon}_{r,z}] = \begin{bmatrix} G^{\mu}_{r,z} & H_{r,z} \\ H^*_{r,z} & G^{\varepsilon}_{r,z} \end{bmatrix}, \tag{35}$$

where

$$H_{r,z} = (\overline{\mathbf{g}^{\mu}_{r,z}})^T \mathbf{g}^{\varepsilon}_{r,z} = \sum_{l=1}^{d_r} \underbrace{\sum_{j=1}^{n_r} \overline{\nabla \times \mathbf{G}(z, \mathbf{R}_j)} \, (\overrightarrow{\gamma}_l \, \overrightarrow{\gamma}_l^T) \, \mathbf{G}(\mathbf{R}_j, z)}_{:= H^l_{r,z}} \sum_{l=1}^{d_r} H^l_{r,z}.$$

In the limit $|z| \to +\infty$,

$$H^l_{r,\infty} = \sum_{j=1}^{n_r} \Lambda(\mathbf{R}_j) \, (\overrightarrow{\gamma}_l \, \overrightarrow{\gamma}_l^T) \, \Delta(\mathbf{R}_j).$$

Computing $H^l_{r,\infty}$ for each direction $\overrightarrow{\gamma}_l$, $l = 1, \ldots, 3$, for the receiver array $\{\xi, \eta, z_a > 0\}$ which is symmetric about ξ and η axes, we obtain

$$H^1_{r,\infty} = \begin{bmatrix} 0 & -H^1_{r,12} & 0 \\ H^1_{r,21} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad H^2_{r,\infty} = \begin{bmatrix} 0 & -H^2_{r,12} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad H^3_{r,\infty} = \begin{bmatrix} 0 & 0 & 0 \\ H^3_{r,21} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

where $H^l_{r,jk} > 0$, $l = 1,2,3$, $j,k = 1,2$. Here the elements of $H_{r,\infty}$ are the inner products of the column vectors of the matrices $\mathbf{g}^{\mu}_{r,\infty}$ and $\mathbf{g}^{\varepsilon}_{p,\infty}$. On the other hand, for any $d_r \geq 2$ directions $\overrightarrow{\gamma} \in \{\mathbf{e}_z, \mathbf{e}_y, \mathbf{e}_x\}$ the matrix $H_{r,0}$ has the following form:

$$H_{r,0} = \begin{bmatrix} 0 & -H_{r,12} & 0 \\ H_{r,21} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

By computing the nonzero elements of the matrix $G^{\mu,\varepsilon}_{r,\infty}$, for the receiver array $\{\xi, \eta, z_a\}$, which is symmetric about the ξ and η axes, we conclude that for any $d_r \geq 1$ direction $\overrightarrow{\gamma} \in \{\mathbf{e}_z, \mathbf{e}_y, \mathbf{e}_x\}$, the rank of the matrix $G^{\mu,\varepsilon}_{r,\infty}$ is the sum of the ranks of $G^{\mu}_{r,\infty}$ and $G^{\varepsilon}_{r,\infty}$.

For finite $|z|$ and $z \neq 0$, the analysis is more complicated. Fortunately, the elements of the matrix $H_{r,z}$ can be approximated analogously to the previous subsections. Therefore, for a dielectric and permeable or perfectly conducting inclusion located at $z \in \mathbb{R}^3$ far enough from the set of the points $\{\mathbf{R}_1, \ldots, \mathbf{R}_{n_r}\}$ the rank of the matrix $G^{\mu,\varepsilon}_{r,z}$ is the sum of the rank of $G^{\mu}_{r,z}$ and $G^{\varepsilon}_{r,z}$ for any $d_r \geq 1$ direction $\overrightarrow{\gamma}$, $\overrightarrow{\gamma} \in \{\mathbf{e}_z, \mathbf{e}_y, \mathbf{e}_x\}$. The same conclusion holds for a planar transmitter array.

To estimate the singular values and singular vectors of $\mathbf{A}$ we refer to subsection 4.1.

If $k^{\varepsilon}_r = k^{\mu}_t = 3$ and $z = 0$, then the matrix $G^{\mu,\varepsilon}_{r,0}$ of the form of the matrix given by (C.6) has rank 6. Thus, the matrices $G^{\mu,\varepsilon}_{r,0}$ and $G^{\mu,\varepsilon}_{t,0}$ are nonsingular, which implies that

$$U_r = [\mathbf{g}^{\mu}_{r,0}, \mathbf{g}^{\varepsilon}_{r,0}](G^{\mu,\varepsilon}_{r,0})^{-\frac{1}{2}}, \quad U_t = [\mathbf{g}^{\mu}_{t,0}, \mathbf{g}^{\varepsilon}_{t,0}](G^{\mu,\varepsilon}_{t,0})^{-\frac{1}{2}},$$

and

$$\Xi = (G^{\mu,\varepsilon}_{r,0})^{\frac{1}{2}}\, \mathbf{M}^{\mu,\varepsilon}\, (\overline{G^{\mu,\varepsilon}_{t,0}})^{\frac{1}{2}}.$$

Suppose that the inclusion is an ellipsoid centered at the origin and is aligned with the coordinate axes ($R = I$); then $\mathbf{M}^{\mu,\varepsilon}$ is diagonal and $\Xi\Xi^*$ is of the form of (C.6). The eigenvalues and eigenvectors of $\Xi\Xi^*$ are given by (C.7) and (C.8), respectively.

5.4. Case of multiple inclusions. Recall that the matrix $\mathbf{A}$ in the case of multiple inclusions is given by (22) and

$$G_{p,z_1,\ldots,z_m} = (\mathbf{g}_{p,z_1,\ldots,z_m})^* \mathbf{g}_{p,z_1,\ldots,z_m}, \quad p = r, t,$$

where $\mathbf{g}_{p,z_1,\ldots,z_m}$ is defined by (20) with $\mathbf{g}^{\varepsilon}_{p,z}, \mathbf{g}^{\mu}_{p,z}$ given by (19). Set

$$G^{\varepsilon}_{r,z_j,z_l} = (\overline{\mathbf{g}^{\varepsilon}_{r,z_j}})^T\, \mathbf{g}^{\varepsilon}_{r,z_l} = \sum_{q'=1}^{d_r} \sum_{q=1}^{n_r} \overline{\mathbf{G}(z_j, \mathbf{R}_q)}\, (\overrightarrow{\gamma}_{q'}\, \overrightarrow{\gamma}^T_{q'})\, \mathbf{G}(\mathbf{R}_q, z_l),$$

$$G^{\mu}_{r,z_j,z_l} = (\overline{\mathbf{g}^{\mu}_{r,z_j}})^T\, \mathbf{g}^{\mu}_{r,z_l} = -\sum_{q'=1}^{d_r} \sum_{q=1}^{n_r} \overline{\nabla\times\mathbf{G}(z_j, \mathbf{R}_q)}\, (\overrightarrow{\gamma}_{q'}\, \overrightarrow{\gamma}^T_{q'})\, \nabla'\times\mathbf{G}(\mathbf{R}_q, z_l),$$

$$H_{r,z_j,z_l} = (\overline{\mathbf{g}^{\mu}_{r,z_j}})^T\, \mathbf{g}^{\varepsilon}_{r,z_l} = \sum_{q'=1}^{d_r} \sum_{q=1}^{n_r} \overline{\nabla\times\mathbf{G}(z_j, \mathbf{R}_q)}\, (\overrightarrow{\gamma}_{q'}\, \overrightarrow{\gamma}^T_{q'})\, \mathbf{G}(\mathbf{R}_q, r_l),$$

$$G^{\mu,\varepsilon}_{r,z_j,z_l} = [\overline{\mathbf{g}^{\mu}_{r,z_j}, \mathbf{g}^{\varepsilon}_{r,z_j}}]^T [\mathbf{g}^{\mu}_{r,z_l}, \mathbf{g}^{\varepsilon}_{r,z_l}] = \begin{bmatrix} G^{\mu}_{r,z_j,z_l} & H_{r,z_j,z_l} \\ H^*_{r,z_j,z_l} & G^{\varepsilon}_{r,z_j,z_l} \end{bmatrix}, \quad j,l = 1,\ldots,m. \tag{36}$$

The same definition holds for the emitting array. Then

$$G_{p,z_1,\dots,z_m} = \{G_{p,z_j,z_l}\}_{j,l=1}^{m}, \quad p = r, t,$$

where

$$G_{p,z_j,z_l} = \begin{cases} G^{\varepsilon}_{p,z_j,z_l} \in \mathbb{C}^{3\times 3} & \text{in case 1,} \\ G^{\mu}_{p,z_j,z_l} \in \mathbb{C}^{3\times 3} & \text{in case 2,} \\ G^{\mu,\varepsilon}_{p,z_j,z_l} \in \mathbb{C}^{6\times 6} & \text{in case 3.} \end{cases}$$

For $j = l$, $j, l = 1, \dots, m$, we are in the case of a single inclusion centered at the point z_j, and the matrix $G_{p,z_j,z_j} = G_{p,z_j}$ is studied in the previous subsections.

Let $j \neq l$, $j, l = 1, \dots m$. Define

$$F_p(z_j, z_l) = \begin{cases} \displaystyle\sum_{n=1}^{n_r} \frac{e^{-ik|R_n - z_j|}}{|R_n - z_j|} \frac{e^{ik|R_n - z_l|}}{|R_n - z_l|}, & p = r, \\ \displaystyle\sum_{n=1}^{n_t} \frac{e^{-ik|r_n - z_j|}}{|r_n - z_j|} \frac{e^{ik|r_n - z_l|}}{|r_n - z_l|}, & p = t. \end{cases}$$

Similarly to previous sections, the inner product can be replaced with the integrals over the aperture of the array, and in the limit $|z| \to +\infty$ we can write the following:

$$F_p(z_j - z_l) = \int_{S_p} \frac{\exp\left(ik \frac{\xi z^1_{jl} + \eta z^2_{jl} + L_p z^3_{jl}}{\sqrt{\xi^2+\eta^2+L_p^2}}\right)}{\xi^2 + \eta^2 + L_p^2} d\xi d\eta, \quad z_l - z_j = \begin{bmatrix} z^1_{jl} \\ z^2_{jl} \\ z^3_{jl} \end{bmatrix} \in \mathbb{R}^3. \tag{37}$$

Here $S_p =]-A_p, A_p[\times]-B_p, B_p[$, $p = r, t$, and $z = L_p$ is the plane of the array of transmitters or receivers.

Then from (36) we have

$$G^{\mu}_{p,z_j,z_l} = -\sum_{h=1}^{d_p} \left[\nabla\times\mathbf{I}_3(\overrightarrow{\gamma}_h \overrightarrow{\gamma}_h^T)\nabla\times\mathbf{I}_3\right] F_p(z_j - z_l),$$

$$G^{\varepsilon}_{p,z_j,z_l} = \sum_{h=1}^{d_p} \left[\left(\mathbf{I}_3 + \frac{\nabla\nabla}{k^2}\right)(\overrightarrow{\gamma}_h \overrightarrow{\gamma}_h^T)\left(\mathbf{I}_3 + \frac{\nabla\nabla}{k^2}\right)\right] F_p(z_j - z_l),$$

$$H_{p,z_j,z_l} = \sum_{h=1}^{d_r} \left[\nabla\times\mathbf{I}_3(\overrightarrow{\gamma}_h \overrightarrow{\gamma}_h^T)\left(\mathbf{I}_3 + \frac{\nabla\nabla}{k^2}\right)\right] F_p(z_j - z_l).$$

Set $z^3_{jl} = 0$ in (37). If $|A_p|, |B_p| \gg L_p$ and if the inclusions are well separated, then from (D.10) we can approximate

$$G_{p,z_j,z_l} = \mathbf{0} \quad \forall j \neq l, \;\; j, l = 1, \dots, m.$$

Consequently, the columns of the matrix $\mathbf{g}_{p,z_1,\dots,z_m}$ are orthogonal and our problem is partitioned into m independent problems as follows:

$$\mathbf{A}_j = \mathbf{g}_{r,z_j}\, \mathbf{M}_j\, \mathbf{g}^T_{t,z_j}, \quad j = 1, \dots, m.$$

Thus, each inclusion can be imaged independently from the others.

If the values of the matrices G_{r,z_j,z_l} and G_{t,z_j,z_l}, $j \neq l$, $j,l = 1,\ldots,m$, are not very small, then the problem becomes more complicated (we have superposition of m problems). In this case, a priori we do not know which set of singular vectors corresponds to which inclusion. Thus, we can also plot for each right singular vector $\mathbf{w}_j$ and left singular vector $\mathbf{u}_j$ the quantities ${\mathbf{g}_{t,z,l}}^T\mathbf{w}_j$ for the transmitter array and ${\mathbf{g}_{r,z,l}}^*\mathbf{u}_j$ for the receiver array, where $\mathbf{g}_{t,z,l}$ and $\mathbf{g}_{r,z,l}$ are the basis column vectors of the matrices $\mathbf{g}_{t,z}$ and $\mathbf{g}_{r,z}$, respectively. Here $l = 1,\ldots,k_t$ or k_r. Let $z_l = (z_l^1, z_l^2, z_l^3)$, then under the condition $|A_p|, |B_p| \gg L_p$, the spatial resolution is compared to $\lambda/2$ in the plane $z = z_l^3$.

6. Numerical examples of MUSIC reconstructions. As discussed in the theoretical part of the paper the nature (dielectric and/or magnetic) of the inclusions with respect to the one of the embedding medium is essential to the results expected, as are the particular geometrical configuration and electromagnetic polarization of the transmitter and receiver arrays (or plane waves) which one is considering. Below, we display and briefly analyze a small number of numerical results acquired from synthetically generated data in order to illustrate a few features of the method, no attempt being made to cover the diversity of situations that should be encountered in practice. It is worth mentioning that in all the numerical results presented in this section the data are generated from the derived asymptotic expansions and may be affected by adding a white noise. Moreover, the error in the asymptotic expansion is smaller than the added white noise. The magnitudes of the equivalent point sources used in the asymptotic expansion are of order 10^{-3}.

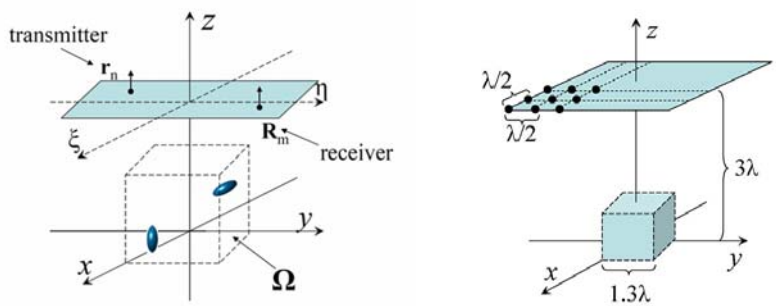


FIG. 6.1. *Sketch of the configuration under numerical study.*

The configuration of interest is schematized in Figure 6.1. We are considering two coincident transmitter and receiver arrays centered within the plane $z = z_a = 5$ and covering a square area of side 10, both consisting of $n_t = n_r = 13^2$ vertical electric dipoles ($\overrightarrow{\beta} = \mathbf{e}_z$) radiating or receiving with unit amplitudes Il and operated at circular frequency $\omega = 4$—the corresponding wavelength in the embedding free space, with as usual unit permittivity and permeability value ($\varepsilon_0 = \mu_0 = 1$), is $\lambda = \pi/2$. The position of the nth transmitter or receiver element is specified by $\mathbf{r}_n = \mathbf{R}_n = (\xi_i, \eta_j, z_a)$, where $n = i + 13(j-1)$ and $\xi_i = -5 + \frac{10}{12}(i-1)$, $\eta_j = -5 + \frac{10}{12}(j-1)$, $i,j = 1,\ldots,13$. Two identical prolate spheroidal inclusions D_1 and D_2 (needle-like, the length of the longest semiaxis being 3 times the ones of the other two semiaxes) with size parameter $\alpha = 0.1$ and orthogonal orientations (the longest semiaxis of D_1 is orientated along ⌣, the longest one of D_2 along ') are to be retrieved within a cubical box $\Omega = [-1,1]^3 \subset \mathbb{R}^3$ assumed situated below the arrays. The said inclusions are,

respectively, characterized by shape equations

$$D_1: \quad (x-0.6)^2+(y+0.6)^2+\frac{(z+0.1)^2}{9}=1,$$
$$D_2: \quad \frac{(x+0.6)^2}{9}+(y-0.6)^2+(z-0.7)^2=1.$$

Their dielectric permittivities are set to $\varepsilon_1=5$, $\varepsilon_2=2$, and their magnetic permeabilities to $\mu_1=5$, $\mu_2=2$; if one is tackling purely dielectric cases, the permeabilities are set to 1 and vice versa for the permittivities whenever tackling purely magnetic cases, the general dielectric magnetic situation involving the full set of nonunit values.

Let us notice that in terms of the probing wavelength, as is also indicated in Figure 6.1, the array step size is about $\lambda/2$, i.e., the traditional half-a-wavelength sampling step. As for the sought inclusions, they are about 1.3λ apart, at about 3.2λ from the center of the array. In short, the inclusions lie in the near field of the array, only strongly aspect-limited data (with view angles of 45 degrees or less) are available, and we are facing the rather severe hypothesis of single polarization electric field data, a good amount of noise being added to them also. However, as detailed below, we appear to be able to number, locate, and further characterize (via singular analysis) inclusions far smaller than the usual half-a-wavelength resolution of imaging procedures.

So, within the above setting, the retrieval involves the calculation of the SVD, $\mathbf{A}=\mathbf{U}\Sigma\mathbf{W}^*$, of the MSR matrix $\mathbf{A}\in\mathbb{C}^{169\times 169}$. Let $\mathbf{a}=(1,1,1)^T\in\mathbb{R}^3$ and $\mathbf{c}=(1,1,1,1,1,1)^T\in\mathbb{R}^6$. Then, for each discrete location $z\in\Omega$ (the sampling step is henceforth chosen as $h=0.06$), the identifiers of interest are

- dielectric inclusions: $W_{r,t}(z)=1/||(\mathbf{P}_r\,\mathbf{g}^{\varepsilon}_{r,z}+\mathbf{P}_t\,\overline{\mathbf{g}^{\varepsilon}_{t,z}})\cdot\mathbf{a}||$,
- permeable inclusions: $W_{r,t}(z)=1/||(\mathbf{P}_r\,\mathbf{g}^{\mu}_{r,z}+\mathbf{P}_t\,\overline{\mathbf{g}^{\mu}_{t,z}})\cdot\mathbf{a}||$,
- dielectric and permeable inclusions:

$$W_{r,t}(z)=1/||(\mathbf{P}_r\,\mathbf{g}^{\mu,\varepsilon}_{r,z}+\mathbf{P}_t\,\overline{\mathbf{g}^{\mu,\varepsilon}_{r,z}})\cdot\mathbf{c}||$$

and

$$W_{r,t}(z)=1/||(\mathbf{P}_r\,\mathbf{g}^{\varepsilon}_{r,z}+\mathbf{P}_t\,\overline{\mathbf{g}^{\varepsilon}_{t,z}})\cdot\mathbf{a}||,$$

which are calculated within Ω. Notice that we are operating herein with coincident transmitter and receiver arrays, which simplifies the formulation of the identifiers. Plots of $z\to W_\cdot(z)$ illustrate the results achieved. Other accompanying results displayed consist of the singular values of the MSR matrix $\mathbf{A}$, shown using a standard log scale.

6.1. Dielectric inclusions. In this purely dielectric case, the singular values are displayed in Figure 6.1.1 (left). In good accord with sections 5.1 and 5.4, six singular values emerge from the 163 others within the noise subspace. Maps of identifier $W_{r,t}(z)=1/||(\mathbf{P}_r\,\mathbf{g}^{\varepsilon}_{r,z}+\mathbf{P}_t\,\overline{\mathbf{g}^{\varepsilon}_{t,z}})\cdot\mathbf{a}||$ in cross-sectional planes $z=\{-0.1,0.7\}$, $x=\{-0.6,0.6\}$, and $y=\{-0.6,0.6\}$ are displayed in Figure 6.1.2, and corresponding three-dimensional (3D) plots are displayed in Figure 6.1.1 (right), the surfaces shown being associated to values of half the peak magnitude.

6.2. Permeable inclusions. In this purely permeable case, the singular values of the matrix $\mathbf{A}\in\mathbb{C}^{169\times 169}$ are displayed in Figure 6.2.1. Again in good accord with previous theoretical material, only four singular values now emerge from the noise,

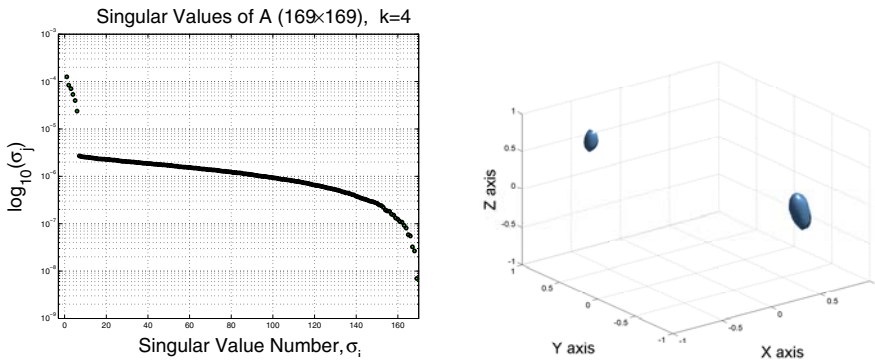


FIG. 6.1.1. *(Dielectric contrasts only): distribution of the singular values of* $\mathbf{A}$ *for* 13^2 *singly polarized dipolar transmitters and receivers in the case of noisy data with* $20dB$ *signal-to-noise ratio (left); corresponding* $3D$ *plots of* $W_{r,t}(z)$ *(right).*

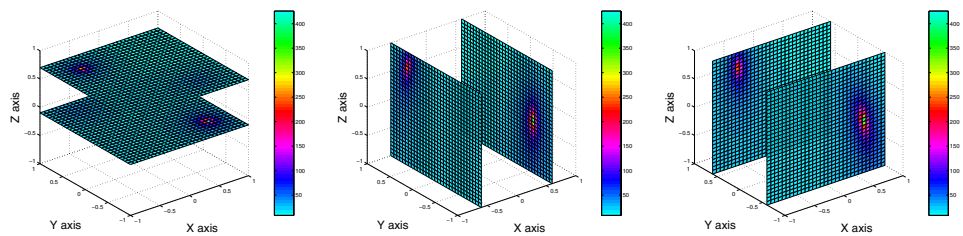


FIG. 6.1.2. *(Dielectric contrasts only): color maps of* $W_{r,t}(z)$ *in cross-sectional planes* $z = \{-0.1, 0.7\}$, $x = \{-0.6, 0.6\}$, *and* $y = \{-0.6, 0.6\}$ *in the case of noisy data with* $20dB$ *signal-to-noise ratio (refer to Figure* 6.1.1*).*

the lack of multiply polarized data implying, for magnetic inclusions and electric fields, the reduction of that number from their maximal number of six. Maps of identifier $W_{r,t}(z) = 1/||(\mathbf{P}_r\, \mathbf{g}^{\mu}_{r,z} + \mathbf{P}_t\, \overline{\mathbf{g}^{\mu}_{t,z}}) \cdot \mathbf{a}||$ in cross-sectional planes $z = \{-0.1, 0.7\}$, $x = \{-0.6, 0.6\}$, and $y = \{-0.6, 0.6\}$ are displayed in Figure 6.2.2, and corresponding 3D plots are displayed in Figure 6.2.1 (right).

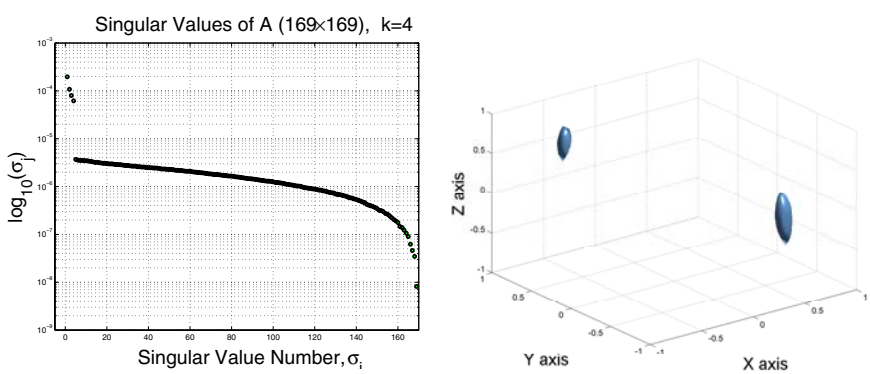


FIG. 6.2.1. *(Permeability contrasts only): distribution of the singular values of* $\mathbf{A}$ *for* 13^2 *singly polarized dipolar transmitters and receivers in the case of noisy data with* $20dB$ *signal-to-noise ratio (left); corresponding* $3D$ *plots of* $W_{r,t}(z)$ *(right).*

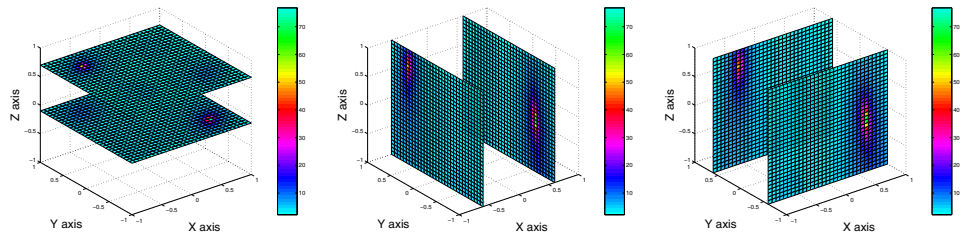


FIG. 6.2.2. *(Permeability contrasts only): color maps of* $W_{r,t}(z)$ *in cross-sectional planes* $z = \{-0.1, 0.7\}$, $x = \{-0.6, 0.6\}$, *and* $y = \{-0.6, 0.6\}$ *in the case of noisy data with* $20dB$ *signal-to-noise ratio (refer to Figure* 6.2.1*).*

6.3. Dielectric and permeable inclusions. In this example, both permittivity and permeability differ from those of the embedding free space. The first 25 singular values of the matrix $\mathbf{A} \in \mathbb{C}^{169\times 169}$ are displayed in Figure 6.3.1 (left); in the case of noisy data the full set is shown in Figure 6.3.1 (right). So, if ten of them (six associated to the dielectric features, four to the magnetic features, as expected from the previous analysis in the case of single polarization) emerge in the absence of noise, this is far less clear cut in the case of a good amount of noise on the data, since the two or three singular values of lesser magnitude are confounded with those outside the signal subspace.

Maps of identifier $W_{r,t}(z) = 1/||(\mathbf{P}_r\, \mathbf{g}^{\mu,\varepsilon}_{r,z} + \mathbf{P}_t\, \overline{\mathbf{g}^{\mu,\varepsilon}_{t,z}}) \cdot \mathbf{c}||$ and $W_{r,t}(z) = 1/||(\mathbf{P}_r\, \mathbf{g}^{\varepsilon}_{r,z} + \mathbf{P}_t\, \overline{\mathbf{g}^{\varepsilon}_{t,z}}) \cdot \mathbf{a}||$ in the cross-sectional planes $z = \{-0.1, 0.7\}$, $x = \{-0.6, 0.6\}$, and $y = \{-0.6, 0.6\}$ are displayed in Figures 6.3.2(a) and (b), respectively, and corresponding 3D plots are displayed in Figure 6.3.3.

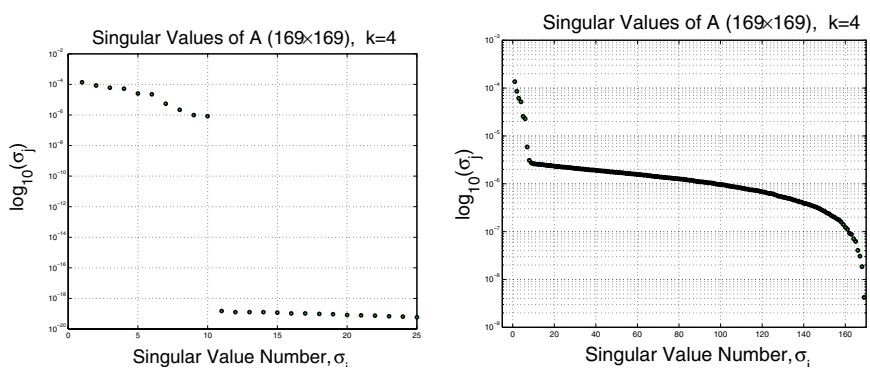


FIG. 6.3.1. *(Dielectric and permeability contrasts): distribution of the first* 25 *singular values of* $\mathbf{A}$ *or* 13^2 *singly polarized dipolar transmitters and receivers in the absence of noise (left) and distribution of the full set in the case of noisy data with* $20dB$ *signal-to-noise ratio (right).*

The singular values of the matrix A and corresponding 3D plots of $W_{r,t}$ are displayed in Figures 6.3.4–6.3.7 in the case of noisy data (white Gaussian noise) with 20dB signal-to-noise ratio.

Finally, we show examples of single inclusions so that the reader can see that, indeed, the number of significant singular values doubles when there are two inclusions.

To conclude this section, we make a remark. If the quantity of data (the dimension of the multistatic response matrix) is not enough to detect multiple inclusions, then,

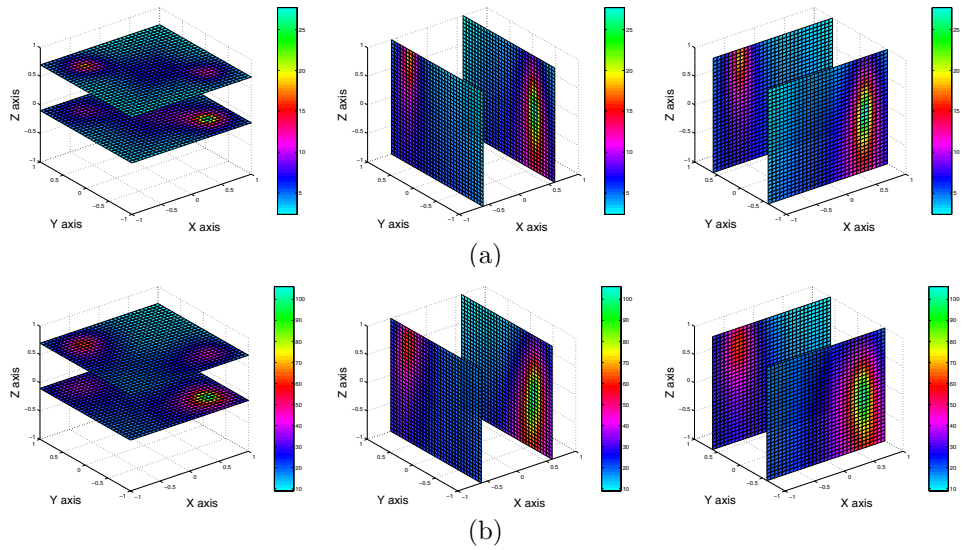


FIG. 6.3.2. *(Dielectric and permeability contrasts): color maps of* $W_{r,t}(z) = 1/||(\mathbf{P}_r\, \mathbf{g}^{\mu,\varepsilon}_{r,z} + \mathbf{P}_t\, \overline{\mathbf{g}^{\mu,\varepsilon}_{t,z}})\cdot\mathbf{c}||$ (a) *and* $W_{r,t}(z) = 1/||(\mathbf{P}_r\, \mathbf{g}^{\varepsilon}_{r,z} + \mathbf{P}_t\, \overline{\mathbf{g}^{\varepsilon}_{t,z}})\cdot\mathbf{a}||$ (b) *in cross-sectional planes* $z = \{-0.1, 0.7\}$, $x = \{-0.6, 0.6\}$, *and* $y = \{-0.6, 0.6\}$ *in the case of noisy data with* $20dB$ *signal-to-noise ratio.*

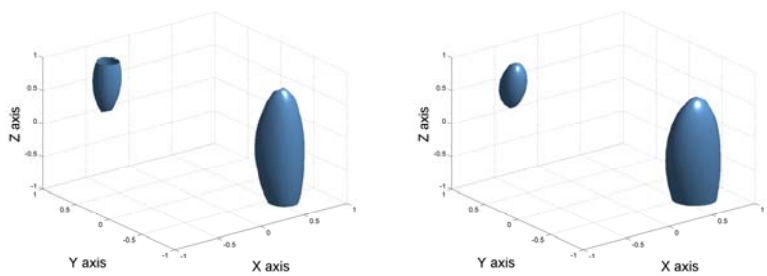


FIG. 6.3.3. *(Dielectric and permeability contrasts):* $3D$ *plots of* $W_{r,t}(z) = 1/||(\mathbf{P}_r\, \mathbf{g}^{\mu,\varepsilon}_{r,z} + \mathbf{P}_t\, \overline{\mathbf{g}^{\mu,\varepsilon}_{t,z}})\cdot\mathbf{c}||$ *(left) and* $W_{r,t}(z) = 1/||(\mathbf{P}_r\, \mathbf{g}^{\varepsilon}_{r,z} + \mathbf{P}_t\, \overline{\mathbf{g}^{\varepsilon}_{t,z}})\cdot\mathbf{a}||$ *(right) in the case of noisy data with* $20dB$ *signal-to-noise ratio (refer to Figure* 6.3.2*).*

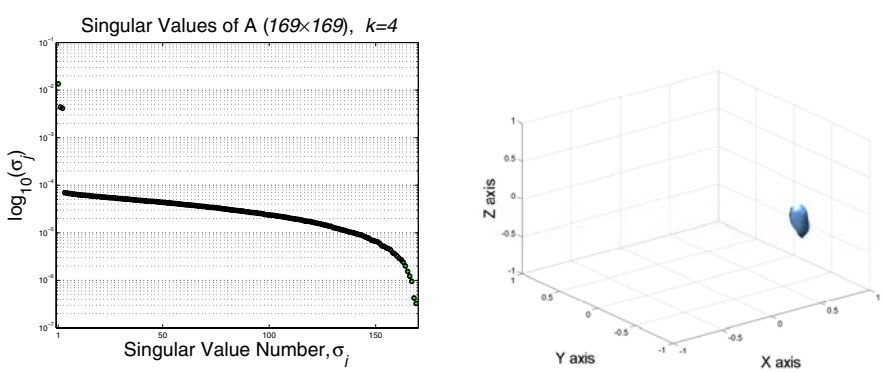


FIG. 6.3.4. *(Dielectric contrasts only): distribution of the singular values of* $\mathbf{A}$ *in the case of a single dielectric inclusion* D_1 *for* 13^2 *singly polarized dipolar transmitters and receivers with* $20dB$ *signal-to-noise ratio (left); corresponding* $3D$ *plots of* $W_{r,t}(z)$ *(right).*

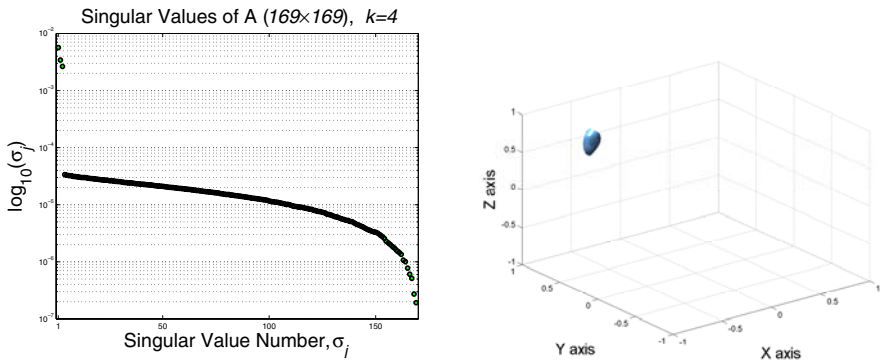


FIG. 6.3.5. *(Dielectric contrasts only): distribution of the singular values of* $\mathbf{A}$ *in the case of a single dielectric inclusion* D_2 *for* 13^2 *singly polarized dipolar transmitters and receivers with* $20dB$ *signal-to-noise ratio (left); corresponding* $3D$ *plots of* $W_{r,t}(z)$ *(right).*

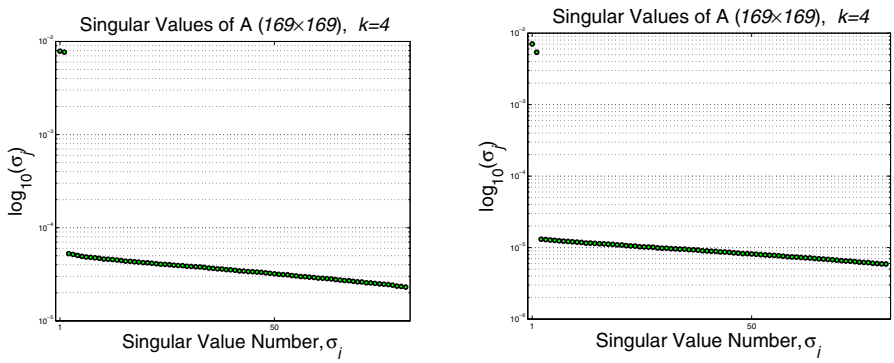


FIG. 6.3.6. *(Permeability contrasts only): distribution of the first* 80 *singular values of* $\mathbf{A}$ *in the case of a single permeable inclusion* D_1 *(left),* D_2 *(right) for* 13^2 *singly polarized dipolar transmitters and receivers in the case of noisy data with* $20dB$ *signal-to-noise ratio.*

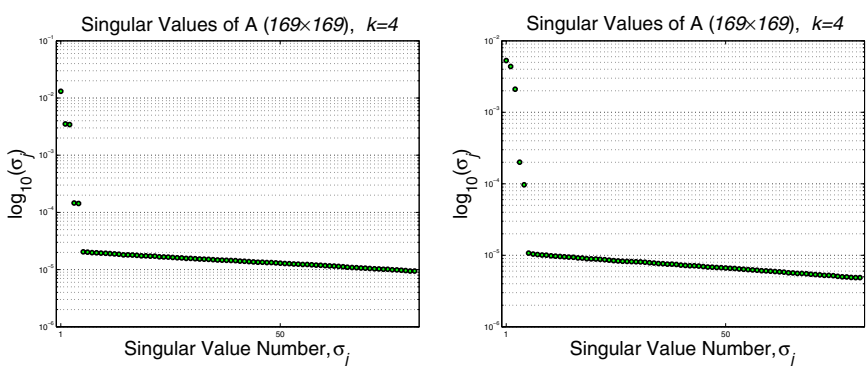


FIG. 6.3.7. *(Dielectric and permeability contrasts): distribution of the first* 80 *singular values of* $\mathbf{A}$ *in the case of a single dielectric and permeable inclusion* D_1 *(left),* D_2 *(right) for* 13^2 *singly polarized dipolar transmitters and receivers in the absence of noise (left) and distribution of the full set in the case of noisy data with* $20dB$ *signal-to-noise ratio (right).*

from the numerical results above, it is clear that our inversion procedure of MUSIC type may miss some of these inclusions or wrongly determine their nature.

7. Conclusion. We have designed a MUSIC-type algorithm for imaging a collection of small 3D inclusions of arbitrary shape and electromagnetic nature. The inclusions are illuminated by an array of electric and/or magnetic dipoles with given polarizations and the resulting electric and/or magnetic field is collected by another array, not necessarily the same as the emitting one. This yields the MSR matrix. By examining the eigenvalue structure of this matrix at a given frequency of operation, we have efficiently located the inclusions and characterized their electromagnetic nature. The polarization tensors, which yield information about the size and orientation (the parameters of equivalent ellipsoids) of the inclusions, can be reconstructed as well. However, the information from these polarization tensors is a mixture of the electromagnetic parameters and the volume. In the case of a single ellipsoidal inclusion with arbitrary electromagnetic parameters and location with respect to the source and receiver arrays, we have provided the single vectors and values in a closed form.

We should emphasize that our approach holds for near-field regimes, as soon as the transmitters and receivers are far enough from the inclusions, and that it easily extends to bona fide far-field data using asymptotic expansions of the Green's functions, as we have done here. Also, the transmitter and receiver arrays can be constituted by electric dipoles (as is the case in the numerical examples) or by magnetic dipoles from application of duality, with one or more of the possible orthogonal polarizations. Conditions for full rank of the multistatic matrix have been spelled out, two orthogonal polarizations being the least number needed for magnetic inclusions while a single polarization suffices in the purely dielectric case.

Furthermore, in the present investigation, we have introduced the configuration when two or more inclusions are close to one another, which are now seen as two equivalent triaxial ellipsoids, one dielectric and one magnetic, through weighted-in electromagnetic and geometric parameters and combined polarization tensors—the case of one inclusion close to the interface could be derived in similar fashion via a multiscale approach.

Finally, the situation of two half-space media, and per extension of any kind of planarly stratified embedding medium, could easily be handled in similar fashion once we have introduced the proper Green's functions of this medium, using known transverse electric/ transverse magnetic spectral representations—this is expected to be true for cylindrical and spherical stratified media as well.

In brief, it appears that the so-called DORT method and 3D time-reversal analysis in electromagnetics are fully encompassed by the present investigation.

We expect that a multifrequency analysis of the MSR matrix will improve the signal-to-noise ratio in cluttered environments and enhance electromagnetic imaging. We also expect that by including higher-order polarization tensors, which in this case might be contributing significantly to scattering, the results given in this paper hold for larger inclusions; see [7]. We intend also to pursue the present investigation in the diffusive regime of electromagnetics (e.g., for eddy-current nondestructive testing and low-frequency subsoil probing), and to extend it to ultrasonic scattering and linear elasticity.

Appendix A. Derivation of the asymptotic formula (10). Since the points $z_j \in \mathbb{R}^3$, $j = 1, \ldots, m$, that determine the location of the inclusions are assumed to satisfy $|z_j - z_l| \geq d_0 > 0$, for all $j \neq l$, and since α, the common order of magnitude of the diameters of the inclusions, is small enough compared to d_0, it suffices to consider

only one inclusion $z + \alpha B$. For simplicity we denote the electric permittivity and magnetic permeability inside $z + \alpha B$ by ε_* and μ_*, respectively.

The asymptotic expansions of $\mathbf{E}_\alpha^{(n)}$ and $\mathbf{H}_\alpha^{(n)}$ are constructed by the method of matched asymptotic expansions for α small, as done in [10, 11]. To reveal the nature of the perturbations in the electric and magnetic fields, we introduce the local variables $\xi = (\mathbf{r} - z)/\alpha$ and set the field $\mathbf{e}_\alpha^{(n)}(\xi) = \mathbf{E}_\alpha^{(n)}(z + \alpha\xi)$ and $\mathbf{h}_\alpha^{(n)}(\xi) = \mathbf{H}_\alpha^{(n)}(z + \alpha\xi)$. We expect that $\mathbf{E}_\alpha^{(n)}(\mathbf{r})$ and $\mathbf{H}_\alpha^{(n)}(\mathbf{r})$ will differ appreciably from $\mathbf{E}_0^{(n)}(\mathbf{r})$ and $\mathbf{H}_0^{(n)}(\mathbf{r})$, respectively, for $\mathbf{r}$ near z, but they will differ little from $\mathbf{E}_0^{(n)}(\mathbf{r})$ and $\mathbf{H}_0^{(n)}(\mathbf{r})$ for $\mathbf{r}$ far from z. Therefore, in the spirit of matched asymptotic expansions, we shall represent each of the fields $\mathbf{E}_\alpha^{(n)}$ and $\mathbf{H}_\alpha^{(n)}$ by two different expansions, an inner expansion for $\mathbf{r}$ near z, and an outer expansion for $\mathbf{r}$ far from z. The outer expansion must begin with $\mathbf{E}_0^{(n)}$ and $\mathbf{H}_0^{(n)}$, respectively, so we write the outer expansions as

$$\begin{aligned} \mathbf{E}_\alpha^{(n)}(\mathbf{r}) &= \mathbf{E}_0^{(n)}(\mathbf{r}) + \alpha^{\tau_1}\mathbf{E}_1^{(n)}(\mathbf{r}) + \alpha^{\tau_2}\mathbf{E}_2^{(n)}(\mathbf{r}) + \cdots, \\ \mathbf{H}_\alpha^{(n)}(\mathbf{r}) &= \mathbf{H}_0^{(n)}(\mathbf{r}) + \alpha^{\tau_1}\mathbf{H}_1^{(n)}(\mathbf{r}) + \alpha^{\tau_2}\mathbf{H}_2^{(n)}(\mathbf{r}) + \cdots \end{aligned} \tag{A.1}$$

for $|\mathbf{r} - z| \gg O(\alpha)$, where $0 < \tau_1 < \tau_2 < \cdots$, and $(\mathbf{E}_j^{(n)}, \mathbf{H}_j^{(n)}), j = 1, 2, \ldots$, are to be found.

We write the inner expansions as

(A.2)

$$\begin{aligned} \mathbf{E}_\alpha^{(n)}(z + \alpha\xi) &= \mathbf{e}_\alpha^{(n)}(\xi)\mathbf{e}_0^{(n)}(\xi) + \alpha\mathbf{e}_1^{(n)}(\xi) + \alpha^2\mathbf{e}_2^{(n)}(\xi) + \cdots, \quad \text{for } |\xi| = O(1), \\ \mathbf{H}_\alpha^{(n)}(z + \alpha\xi) &= \mathbf{h}_\alpha^{(n)}(\xi) = \mathbf{h}_0^{(n)}(\xi) + \alpha\mathbf{h}_1^{(n)}(\xi) + \alpha^2\mathbf{h}_2^{(n)}(\xi) + \cdots, \quad \text{for } |\xi| = O(1), \end{aligned}$$

where $(\mathbf{e}_j^{(n)}, \mathbf{h}_j^{(n)})$, $j = 0, 1, \ldots$, are to be found.

In order to determine the functions $\mathbf{E}_i^{(n)}(\mathbf{r}), \mathbf{H}_i^{(n)}(\mathbf{r}), \mathbf{e}_i^{(n)}(\xi)$, and $\mathbf{h}_i^{(n)}(\xi)$ we have to equate the inner and the outer expansions in some "overlap" domain within which the stretched variable ξ is large and $\mathbf{r} - z$ is small. In this domain the matching conditions are as follows:

(A.3)

$$\begin{aligned} &\mathbf{E}_0^{(n)}(\mathbf{r}) + \alpha^{\tau_1}\mathbf{E}_1^{(n)}(\mathbf{r}) + \alpha^{\tau_2}\mathbf{E}_2^{(n)}(\mathbf{r}) + \cdots \sim \mathbf{e}_0^{(n)}(\xi) + \alpha\mathbf{e}_1^{(n)}(\xi) + \alpha^2\mathbf{e}_2^{(n)}(\xi) + \cdots, \\ &\mathbf{H}_0^{(n)}(\mathbf{r}) + \alpha^{\tau_1}\mathbf{H}_1^{(n)}(\mathbf{r}) + \alpha^{\tau_2}\mathbf{H}_2^{(n)}(\mathbf{r}) + \cdots \sim \mathbf{h}_0^{(n)}(\xi) + \alpha\mathbf{h}_1^{(n)}(\xi) + \alpha^2\mathbf{h}_2^{(n)}(\xi) + \cdots. \end{aligned}$$

Define the stretched coefficients $\varepsilon(\xi)$ and $\mu(\xi)$ by

$$\varepsilon(\xi) = \begin{cases} \varepsilon_0, & \xi \in \mathbb{R}^3 \setminus \overline{B}, \\ \varepsilon_*, & \xi \in B, \end{cases}$$

$$\mu(\xi) = \begin{cases} \mu_0, & \xi \in \mathbb{R}^3 \setminus \overline{B}, \\ \mu_*, & \xi \in B. \end{cases}$$

If we substitute the inner expansions (A.2) into the Maxwell's equations (4) and formally equate coefficients of α^{-1} we get from use of the first matching condition in (A.3) that

$$\begin{cases} \nabla \times \mathbf{e}_0^{(n)}(\xi) = 0 \quad \text{in } \mathbb{R}^3, \\ \nabla \cdot (\varepsilon(\xi)\mathbf{e}_0^{(n)}(\xi)) = 0 \quad \text{in } \mathbb{R}^3, \\ \mathbf{e}_0^{(n)}(\xi) \to \mathbf{E}_0^{(n)}(z) \quad \text{as } |\xi| \to +\infty, \end{cases} \tag{A.4}$$

and

$$\text{(A.5)}\qquad \begin{cases} \nabla \times \mathbf{h}_0^{(n)}(\xi) = 0 \quad \text{in } \mathbb{R}^3, \\ \nabla \cdot (\mu(\xi)\mathbf{h}_0^{(n)}(\xi)) = 0 \quad \text{in } \mathbb{R}^3, \\ \mathbf{h}_0^{(n)}(\xi) \to \mathbf{H}_0^{(l)}(z) \quad \text{as } |\xi| \to +\infty. \end{cases}$$

Therefore,

$$\begin{aligned} \mathbf{e}_0^{(n)}(\xi) &= \nabla v^{\varepsilon}(\xi)(\mathbf{E}_0^{(n)}(z)), \\ \mathbf{h}_0^{(n)}(\xi) &= \nabla v^{\mu}(\xi)(\mathbf{H}_0^{(n)}(z)), \end{aligned}$$

where the function v^q, for $q(\xi) = \varepsilon(\xi)$ or $\mu(\xi)$, is defined by (7).

Define the polarization tensors by (8). Inserting the inner expansions into the Lippman–Schwinger integral representation formula (6) for $\mathbf{E}_\alpha^{(n)}$ we arrive at (10).

Appendix B. Proof of Proposition 4.3. This proof directly follows from [32]. We sketch it for the reader's convenience.

We consider only the first case (permittivity contrast only) and suppose that $k_r = k_t$. The proof in all the other cases is similar. To see the correspondence of $\mathcal{R}(\mathbf{A}\mathbf{A}^*)$ with the different inclusions, we introduce the operator $\Lambda : \mathbb{C}^m \to \mathbb{C}^{d_r}$:

$$(\Lambda\lambda)(\mathbf{r}) := \sum_{j=1}^{m} \lambda_j \left((\mathbf{G}(z_j, \mathbf{r}) \cdot \vec{\gamma}_1) \cdot \mathbf{a}, \dots, (\mathbf{G}(z_j, \mathbf{r}) \cdot \vec{\gamma}_{d_r}) \cdot \mathbf{a} \right)^T$$

for $\mathbf{r} \in \mathbb{R}^3 \setminus \{z_1, z_2, \dots, z_m\}, \lambda \in \mathbb{C}^m, \mathbf{a} \in \mathbb{R}^3 \setminus \{0\}$. We then show that Λ is one-to-one. Let $\lambda \in \mathbb{C}^m$ with $(\Lambda\lambda)(\mathbf{r}) = 0$ for all $\mathbf{r} \in \mathbb{R}^3 \setminus \{z_1, z_2, \dots, z_m\}$. Letting $\mathbf{r}$ tend to one of the points z_j yields that $\lambda_j = 0$ for every $j = 1, \dots, m$.

Assume, without any loss of generality, that the receiver array is in the plane $\{z = 0\}$ and consider $\{\mathbf{R}_p\}_{p\in\mathbb{N}} \subset \{z = 0\}$ to be a countable set of positions with the property that any analytic function which vanishes in $\mathbf{R}_p$ for all $p \in \mathbb{N}$ vanishes identically on $\{z = 0\}$.

Next we show the existence of $n_0 \in \mathbb{N}$ such that the operator

$$\lambda \mapsto ((\Lambda\lambda)(\mathbf{R}_1), \dots, (\Lambda\lambda)(\mathbf{R}_n))^T$$

from $\mathbb{C}^m$ to $\mathbb{C}^{n_r \times d_r}$ is one-to-one for $n_r \geq n_0$. If there existed no such n_0, then there would exist sequences $\{n_l\}_{l\in\mathbb{N}}$ in $\mathbb{N}$ and $\{\lambda^{(l)}\}_{l\in\mathbb{N}}$ in $\mathbb{C}^m$ with $\sum_{j=1}^m |\lambda_j^{(l)}| = 1$ and $(\Lambda\lambda^{(l)})(\mathbf{R}_p) = 0$ for all $p = 1, \dots, n_l$. The sequence $\{\lambda^{(l)}\}_{l\in\mathbb{N}}$ has accumulation points. Without loss of generality we assume that $\lambda^{(l)} \to \lambda$, $l \to \infty$, with $\sum_{j=1}^m |\lambda_j| = 1$. For any $n \in \mathbb{N}$ and l with $n_l \geq n$ we estimate by the triangle inequality

$$\begin{aligned} |(\Lambda\lambda)(\mathbf{R}_p)| &\leq |\Lambda(\lambda - \lambda^{(l)})(\mathbf{R}_p)| + \underbrace{|(\Lambda\lambda^{(l)})(\mathbf{R}_p)|}_{=0} \\ &\leq \sum_{j=1}^{m} |\lambda_j - \lambda_j^{(l)}||\mathbf{G}(z_j, \mathbf{R}_p)| \leq C \sum_{j=1}^{m} |\lambda_j - \lambda_j^{(l)}| \end{aligned}$$

for some constant C independent of l, and this converges to zero as l tends to infinity. Therefore, $(\Lambda\lambda)(\mathbf{R}_p) = 0$ for every $p \in \mathbb{N}$. Since $\Lambda\lambda$ is analytic we conclude that $\Lambda\lambda$

vanishes on all of $\mathbb{R}^3 \setminus \{z_1, z_2, \ldots, z_m\}$, which yields $\lambda = 0$ by the first part of the proof. This contradicts the fact that $\sum_{j=1}^{m} |\lambda_j| = 1$. The proof of the proposition is complete.

Appendix C. Expressions for eigenvalues and eigenvectors. Let $A \in \mathbb{C}^{6\times 6}$ be a matrix of the following form:

$$A = \begin{bmatrix} a_{11} & 0 & 0 & 0 & a_{15} & 0 \\ 0 & a_{22} & 0 & a_{24} & 0 & 0 \\ 0 & 0 & a_{33} & 0 & 0 & 0 \\ 0 & a_{42} & 0 & a_{44} & 0 & 0 \\ a_{51} & 0 & 0 & 0 & a_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & a_{66} \end{bmatrix}. \tag{C.6}$$

Then the eigenvalues are given by

$$\begin{aligned} \lambda_{1,5} &= \frac{1}{2}\left(a_{11} + a_{55} \pm \sqrt{(a_{11} - a_{55})^2 + 4a_{15}a_{51}}\right), \\ \lambda_{2,4} &= \frac{1}{2}\left(a_{22} + a_{44} \pm \sqrt{(a_{22} - a_{44})^2 + 4a_{24}a_{42}}\right), \\ \lambda_3 &= a_{33}, \quad \lambda_6 = a_{66}, \end{aligned} \tag{C.7}$$

and are associated to the eigenvectors

$$\begin{aligned} u_{1,5} &= \frac{1}{\sqrt{a_{15}^2 + (a_{11} - \lambda_{1,5})^2}}\left(a_{15}\mathbf{e}_1 - (a_{11} - \lambda_{1,5})\mathbf{e}_5\right), \\ u_{2,4} &= \frac{1}{\sqrt{a_{24}^2 + (a_{22} - \lambda_{2,4})^2}}\left(a_{24}\mathbf{e}_2 - (a_{22} - \lambda_{2,4})\mathbf{e}_4\right), \\ u_3 &= \mathbf{e}_3, \quad u_6 = \mathbf{e}_6, \end{aligned} \tag{C.8}$$

where $\mathbf{e}_j$, $j = 1, \ldots, 6$, is an orthogonal basis in $\mathbb{R}^6$.

Appendix D. Useful identities to estimate the resolution. Recall the well-known identity

$$\frac{\sin(2\pi a r)}{r} = 2\pi \int_0^\infty J_0(2\pi r x) \frac{\Pi(x/(2a))}{\sqrt{a^2 - x^2}}\, x\, dx, \tag{D.9}$$

where $\Pi(x)$ (rectangular function) is a function that is 0 outside the interval $[-1/2, 1/2]$ and unity inside it.

Let L be fixed. For any point $z = (z_1, z_2, z_3 = 0)^T \in \mathbb{R}^3$, it follows from (D.9) that

(D.10)

$$\begin{aligned} \frac{1}{2\pi ik}\frac{\partial}{\partial z_3}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} dxdy\, \frac{\exp\left(ik\,\frac{xz_1 + yz_2 + Lz_3}{\sqrt{x^2+y^2+L^2}}\right)}{x^2 + y^2 + L^2} &= \frac{L}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} dxdy\, \frac{\exp\left(ik\,\frac{xz_1 + yz_2}{\sqrt{x^2+y^2+L^2}}\right)}{(x^2 + y^2 + L^2)^{3/2}} \\ &= \frac{\sin(k|z|)}{k|z|}. \end{aligned}$$

Appendix E. Some extensions.

E.1. Closely spaced inclusions. Suppose that the set of inclusions $\mathcal{I}_\alpha = \bar{z} + \alpha \cup_{j=1}^m B_j$, where $\overline{B}_j$'s are mutually disjoint, and the electromagnetic parameters of B_j are ε_j and μ_j.

For any $\mathbf{r}$ away from $\mathbf{r}_n$ and $z_j,\ j = 1, \ldots, m$, we can prove that

(E.11)
$$\begin{aligned}\mathbf{E}_\alpha^{(n)}(\mathbf{r}) &= \mathbf{E}_0^{(n)}(\mathbf{r}) \\ &\quad + \alpha^3 \Bigg[- i\omega\mu_0 \, \nabla' \times \mathbf{G}(\mathbf{r}, \bar{z}) \cdot \mathbf{M}\left(\frac{\mu_1}{\mu_0}, \ldots, \frac{\mu_m}{\mu_0}; B_1, \ldots, B_m\right) \mathbf{H}_0^{(n)}(\bar{z}) \\ &\quad + k^2 \, \mathbf{G}(\mathbf{r}, \bar{z}) \cdot \mathbf{M}\left(\frac{\varepsilon_1}{\varepsilon_0}, \ldots, \frac{\varepsilon_m}{\varepsilon_0}; B_1, \ldots, B_m\right) \mathbf{E}_0^{(n)}(\bar{z}) \Bigg] + O(\alpha^4),\end{aligned}$$

where

$$\mathbf{M}\left(\frac{\mu_1}{\mu_0}, \ldots, \frac{\mu_m}{\mu_0}; B_1, \ldots, B_m\right) = \left(\sum_{j=1}^m \left(\frac{\mu_j}{\mu_0} - 1\right) \int_{B_j} \partial_{y_l} v_{l'}^{\mu}(y) \, dy \right)_{ll'},$$

and

$$\mathbf{M}\left(\frac{\varepsilon_1}{\varepsilon_0}, \ldots, \frac{\varepsilon_m}{\varepsilon_0}; B_1, \ldots, B_m\right) = \left(\sum_{j=1}^m \left(\frac{\varepsilon_j}{\varepsilon_0} - 1\right) \int_{B_j} \partial_{y_l} v_{l'}^{\varepsilon}(y) \, dy \right)_{ll'}.$$

Here $\varepsilon(y)$ and $\mu(y)$ are defined by

$$\varepsilon(y) = \begin{cases} \varepsilon_0, & y \in \mathbb{R}^3 \setminus \overline{\cup_{j=1}^m B_j}, \\ \varepsilon_j, & y \in B_j, j = 1, \ldots, m, \end{cases}$$

$$\mu(y) = \begin{cases} \mu_0, & y \in \mathbb{R}^3 \setminus \overline{\cup_{j=1}^m B_j}, \\ \mu_j, & y \in B_j, j = 1, \ldots, m. \end{cases}$$

Next, we define the overall electric permittivity $\bar{\varepsilon}$, the overall magnetic permeability $\bar{\mu}$, and the centers of mass $\bar{z_\varepsilon}$ and $\bar{z_\mu}$ of $B := \cup_{j=1}^m B_j$ as follows:

$$\frac{\bar{\varepsilon} - \varepsilon_0}{\bar{\varepsilon} + \varepsilon_0} \sum_{j=1}^m |B_j| = \sum_{j=1}^m \frac{\varepsilon_j - \varepsilon_0}{\varepsilon_j + \varepsilon_0} |B_j|, \quad \frac{\bar{\mu} - \mu_0}{\bar{\mu} + \mu_0} \sum_{j=1}^m |B_j| = \sum_{j=1}^m \frac{\mu_j - \mu_0}{\mu_j + \mu_0} |B_j|,$$

$$\frac{\bar{\varepsilon} - \varepsilon_0}{\bar{\varepsilon} + \varepsilon_0} \bar{z_\varepsilon} \sum_{j=1}^m |B_j| = \sum_{j=1}^m \frac{\varepsilon_j - \varepsilon_0}{\varepsilon_j + \varepsilon_0} \int_{B_j} x \, dx, \quad \frac{\bar{\mu} - \mu_0}{\bar{\mu} + \mu_0} \bar{z_\mu} \sum_{j=1}^m |B_j| = \sum_{j=1}^m \frac{\mu_j - \mu_0}{\mu_j + \mu_0} \int_{B_j} x \, dx.$$

We can represent and visualize the multiple inclusions $\cup_{j=1}^m B_j$ by means of two ellipsoids, $\mathcal{E}_\varepsilon$ and $\mathcal{E}_\mu$, of centers $\bar{z}_\varepsilon$ and $\bar{z}_\mu$ with the same polarization tensors. We call $\mathcal{E}_\varepsilon$ (resp., $\mathcal{E}_\mu$) the equivalent purely electric (resp., purely magnetic) ellipsoid of $\cup_{j=1}^m B_j$. If we consider the problem of determining the collection of the closely spaced small inclusions $\mathcal{I}_\alpha$, the only information of real interest that could be reconstructed is exactly the sum $\mathcal{E}_\varepsilon + \mathcal{E}_\mu$.

The decomposition of the MSR matrix associated with closely spaced inclusions reduces to the case of a single inclusion, where only the form of the polarization tensor must be changed. And by our MUSIC algorithm we can find the position $\bar{z}$.

E.2. Magnetic field-magnetic dipole. By the duality principle [22],

$$\mathbf{E} \to \mathbf{H}, \quad \mathbf{H} \to \mathbf{E}, \quad \mu \to -\varepsilon, \quad \varepsilon \to -\mu, \quad \mathbf{J} \to -\mathbf{M}, \tag{E.12}$$

where $\mathbf{M}$ is the magnetic current density. The incident magnetic field $\mathbf{H}_0^{(n)}$ in a homogeneous medium that is due to the magnetic dipole $\mathbf{M}_0^{(n)}(\mathbf{r}) = -i\omega\mu_0\,\delta(\mathbf{r} - \mathbf{r}_n)\,\overrightarrow{\beta}\,IA$, at the position $\mathbf{r}_n$ and directed in the $\overrightarrow{\beta}$ direction, is given by

$$\mathbf{H}_0^{(n)}(\mathbf{r}) = \omega^2\varepsilon_0\mu_0\,IA\,\mathbf{G}(\mathbf{r},\mathbf{r}_n)\cdot\overrightarrow{\beta} = k^2\,IA\,\mathbf{G}(\mathbf{r},\mathbf{r}_n)\cdot\overrightarrow{\beta},$$

where IA is the magnetic moment. The electric field is given by

$$\mathbf{E}_0^{(n)}(\mathbf{r}) = i\omega\mu_0\nabla\times IA\,\mathbf{G}(\mathbf{r},\mathbf{r}_n)\cdot\overrightarrow{\beta} = i\omega\mu_0 IA\,\nabla g(\mathbf{r},\mathbf{r}_n)\times\overrightarrow{\beta}.$$

Analogously to (10), for any $\mathbf{r}$ away from $\mathbf{r}_n$ and z_j, $j = 1,\dots,m$, the perturbation of the magnetic field due to the presence of the collection of small electromagnetic inclusions $\mathcal{I}_\alpha$ which is illuminated by the magnetic dipole $\mathbf{M}_0^{(n)}$ is approximated by

$$\begin{aligned}\mathbf{H}_\alpha^{(n)}(\mathbf{r}) - \mathbf{H}_0^{(n)}(\mathbf{r}) = \alpha^3\sum_{j=1}^{m}\Bigg[&i\omega\,(\varepsilon_j-\varepsilon_0)\,\nabla'\times\mathbf{G}(\mathbf{r},z_j)\cdot\mathbf{M}\left(\frac{\varepsilon_j}{\varepsilon_0};B_j\right)\mathbf{E}_0^{(n)}(z_j)\\ &+\omega^2\varepsilon_0\,(\mu_j-\mu_0)\,\mathbf{G}(\mathbf{r},z_j)\cdot\mathbf{M}\left(\frac{\mu_j}{\mu_0};B_j\right)\mathbf{H}_0^{(n)}(z_j)\Bigg]+O(\alpha^4).\end{aligned} \tag{E.13}$$

The duality principle also applies to decompose the MSR matrix $\mathbf{A}$. After multiplying the right-hand side of (18) by $i\omega\mu_0$, the duality principle shows that the decomposition (22) for the MSR matrix $\mathbf{A}$ holds where

$$\begin{gathered}\mathbf{g}_{r,z}^{\varepsilon} \to \mathbf{g}_{r,z}^{\mu}, \quad \mathbf{g}_{e,z}^{\varepsilon} \to \mathbf{g}_{e,z}^{\mu},\\ \mathbf{g}_{r,z}^{\mu} \to \mathbf{g}_{r,z}^{\varepsilon}, \quad \mathbf{g}_{e,z}^{\mu} \to \mathbf{g}_{e,z}^{\varepsilon},\\ \mathbf{M}^{\varepsilon} \to i\omega\mu_0\mathbf{M}^{\mu}, \quad \mathbf{M}^{\mu} \to i\omega\mu_0\mathbf{M}^{\varepsilon},\end{gathered}$$

and the analysis of the MSR matrix and the design of a MUSIC-type reconstruction algorithm are parallel to the ones provided in this paper.

Acknowledgments. Thorough discussions with D. H. Chambers, made possible by the CNRS Math-STIC program, have been very helpful to the authors. Work led by S. Gdoura for her 2005 master's thesis is acknowledged.

REFERENCES

[1] D. L. Alumbaugh and H. F. Morrison, *Electromagnetic conductivity imaging with an iterative Born inversion*, IEEE Trans. Geoscience Remote Sensing, 31 (1993), pp. 758–763.

[2] H. Ammari, R. Griesmaier, and M. Hanke, *Identification of small inhomogeneities: Asymptotic factorization,* Math. Comp., to appear.

[3] H. Ammari, E. Iakovleva, and D. Lesselier, *A MUSIC algorithm for locating small inclusions buried in a half-space from the scattering amplitude at a fixed frequency,* Multiscale Model. Simul., 3 (2005), pp. 597–628.

[4] H. Ammari, E. Iakovleva, and D. Lesselier, *Two numerical methods for recovering small inclusions from the scattering amplitude at a fixed frequency*, SIAM J. Sci. Comput., 27 (2005), pp. 130–158.

[5] H. Ammari, E. Iakovleva, and H. Kang, *Reconstruction of a small inclusion in a two-dimensional open waveguide*, SIAM J. Appl. Math., 65 (2005), pp. 2107–2127.

[6] H. Ammari, E. Iakovleva, and S. Moskow, *Recovery of small inhomogeneities from the scattering amplitude at a fixed frequency*, SIAM J. Math. Anal., 34 (2003), pp. 882–900.

[7] H. Ammari and H. Kang, Reconstruction of Small Inhomogeneities from Boundary Measurements, Lecture Notes in Math. 1846, Springer-Verlag, Berlin, 2004.

[8] H. Ammari and H. Kang, *Boundary layer techniques for solving the Helmholtz equation in the presence of small inhomogeneities,* J. Math. Anal. Appl., 296 (2004), pp. 190–208.

[9] H. Ammari and H. Kang, *Polarization and Moment Tensors with Applications to Inverse Problems and Effective Medium Theory*, Appl. Math. Sci., Springer-Verlag, New York, to appear.

[10] H. Ammari and A. Khelifi, *Electromagnetic scattering by small dielectric inhomogeneities*, J. Math. Pures Appl., 82 (2003), pp. 749–842.

[11] H. Ammari, S. Moskow, and M. S. Vogelius, *Boundary integral formulas for the reconstruction of electromagnetic imperfections of small diameter*, ESAIM Control Optim. Calc. Var., 9 (2003), pp. 49–66.

[12] H. Ammari, M. Vogelius, and D. Volkov, *Asymptotic formulas for perturbations in the electromagnetic fields due to the presence of imperfections of small diameter* II. *The full Maxwell equations*, J. Math. Pures Appl., 80 (2001), pp. 769–814.

[13] H. Ammari and D. Volkov, *The leading order term in the asymptotic expansion of the scattering amplitude of a collection of finite number of dielectric inhomogeneities of small diameter*, Int. J. Multiscale Comput. Engrg., 3 (2005), pp. 285–296.

[14] H. Ammari and D. Volkov, *Correction of order three for the expansion of two-dimensional electromagnetic fields perturbed by the presence of inhomogeneities of small diameter*, J. Comput. Phys., 189 (2003), pp. 371–389.

[15] M. Brühl, M. Hanke, and M. S. Vogelius, *A direct impedance tomography algorithm for locating small inhomogeneities,* Numer. Math., 93 (2003), pp. 635–654.

[16] Y. Capdeboscq and M. Vogelius, *A general representation formula for boundary voltage perturbations caused by internal conductivity inhomogeneities of low volume fraction*, Math. Model. Numer. Anal., 37 (2003), pp. 159–173.

[17] Y. Capdeboscq and M. Vogelius, *Optimal asymptotic estimates for the volume of internal inhomogeneities in terms of multiple boundary measurements*, Math. Model. Numer. Anal., 37 (2003), pp. 227–240.

[18] D. J. Cedio-Fengya, S. Moskow, and M. S. Vogelius, *Identification of conductivity imperfections of small diameter by boundary measurements: Continuous dependence and computational reconstruction*, Inverse Problems, 14 (1998), pp. 553–595.

[19] D. H. Chambers and J. G. Berryman, *Analysis of the time-reversal operator for a small spherical scatterer in an electromagnetic field*, IEEE Trans. Antennas and Propagation, 52 (2004), pp. 1729–1738.

[20] D. H. Chambers and J. G. Berryman, *Time-reversal analysis for scatterer characterization*, Phys. Rev. Lett., 92 (2004), 023902-1.

[21] M. Cheney, *The linear sampling method and the MUSIC algorithm,* Inverse Problems, 17 (2001), pp. 591–595.

[22] W. C. Chew, *Waves and Fields in Inhomogeneous Media*, 2nd ed., IEEE Press, Piscataway, NJ, 1995.

[23] W. C. Chew and Y. M. Wang, *Reconstruction of two-dimensional permittivity distribution using the distorted Born iterative method*, IEEE Trans. Medical Imaging, 9 (1990), pp. 218–225.

[24] D. L. Colton, H. Haddar, and P. Monk, *The linear sampling method for solving the electromagnetic inverse scattering problem,* SIAM J. Sci. Comput., 24 (2002), pp. 719–731.

[25] D. L. Colton and R. Kress, *Inverse Acoustic and Electromagnetic Scattering Theory*, Appl. Math. Sci. 93, Springer-Verlag, New York, 1992.

[26] G. Dassios and R. E. Kleinman, *Low Frequency Scattering*, Oxford University Press, New York, 2000.

[27] A. J. Devaney, *Time reversal imaging of obscured targets from multistatic data*, IEEE Trans. Antennas and Propagation, 523 (2005), pp. 1600–1610.

[28] O. Dorn, H. Bertete-Aguirre, J. G. Berryman, and G. C. Papanicolaou, *A nonlinear inversion method for* 3*D electromagnetic imaging using adjoint fields*, Inverse Problems, 15 (1999), pp. 1523–1558.

[29] M. Fink and C. Prada, *Acoustic time-reversal mirrors,* Inverse Problems, 17 (2001), pp. R1–R38.

[30] B. Gebauer, M. Hanke, A. Kirsch, W. Muniz, and C. Schneider *A sampling method for detecting buried objects using electromagnetic scattering*, Inverse Problems, 21 (2005), pp. 2035–2050.

[31] A. J. Hesford and W. C. Chew, *A comparison of the computational efficiency of three inverse scattering methods*, in Proceedings of the IEEE Antennas and Propagation Society International Symposium, Vol. 2A, 2005, pp. 127–130.

[32] A. Kirsch, *The MUSIC algorithm and the factorization method in inverse scattering theory for inhomogeneous media,* Inverse Problems, 18 (2002), pp. 1025–1040.

[33] A. Kirsch, *The factorization method for Maxwell's equations,* Inverse Problems, 20 (2004), pp. 117–134.

[34] R. E. Kleinman and T. B. A. Senior, *Rayleigh scattering,* in Low and High Frequency Asymptotics, V. K. Varadan and V. V. Varadan, eds., North–Holland, Amsterdam, 1986, pp. 1–70.

[35] T. D. Mast, A. I. Nachman, and R. C. Waag, *Focusing and imaging using eigenfunctions of the scattering operator,* J. Acoust. Soc. Am., 102 (1997), pp. 715–725.

[36] G. Micolau and M. Saillard, *DORT method as applied to electromagnetic subsurface sensing,* Radio Science, 38 (2003), pp. 1038–1049.

[37] J. C. Nédélec, Acoustic and Electromagnetic Equations: Integral Representations for Harmonic Problems, Appl. Math. Sci. 144. Springer-Verlag, New York, 2001.

[38] G. Pólya and G. Szegö, *Isoperimetric Inequalities in Mathematical Physics*, Ann. of Math. Stud. 27, Princeton University Press, Princeton, NJ, 1951.

[39] C. Torres-Verdin and T. M. Habashy, *Rapid* 2.5*-dimensional forward modeling and inversion via a new nonlinear scattering approximation*, Radio Science, 29 (1994), pp. 1051–1079.

[40] M. S. Vogelius and D. Volkov, *Asymptotic formulas for perturbations in the electromagnetic fields due to the presence of inhomogeneities of small diameter*, Math. Model. Numer. Anal., 34 (2000), pp. 723–748.

[41] M. S. Zhdanov and S. Fang, *Three-dimensional quasi-linear electromagnetic inversion*, Radio Science, 31 (1996), pp. 741–754.

[42] M. S. Zhdanov and G. Hursan, 3*D electromagnetic inversion based on quasi-analytical approximation*, Inverse Problems, 16 (2000), pp. 1297–1322.

SIAM J. SCI. COMPUT.
Vol. 29, No. 2, pp. 710–728
© 2007 Society for Industrial and Applied Mathematics

CONVERGENCE IN BACKWARD ERROR OF RELAXED GMRES*

L. GIRAUD†, S. GRATTON‡, AND J. LANGOU§

Abstract. This work is the follow-up of the experimental study presented in [A. Bouras and V. Frayssé, *SIAM J. Matrix Anal. Appl.*, 26 (2005), pp. 660–678]. It is based on and extends some theoretical results in [V. Simoncini and D. B. Szyld, *SIAM J. Sci. Comput.*, 25 (2003), pp. 454–477; J. van den Eshof and G. L. G. Sleijpen, *SIAM J. Matrix Anal. Appl.*, 26 (2004), pp. 125–153]. In a backward error framework we study the convergence of GMRES when the matrix-vector products are performed inaccurately. This inaccuracy is modeled by a perturbation of the original matrix. We prove the convergence of GMRES when the perturbation size is proportional to the inverse of the computed residual norm; this implies that the accuracy can be relaxed as the method proceeds which gives rise to the terminology "relaxed GMRES." As for the exact GMRES we show under proper assumptions that only happy breakdowns can occur. Furthermore, the convergence can be detected using a byproduct of the algorithm. We explore the links between relaxed right-preconditioned GMRES and flexible GMRES (FGMRES). In particular, this enables us to derive a proof of convergence of FGMRES. Finally, we report results of numerical experiments to illustrate the behavior of the relaxed GMRES monitored by the proposed relaxation strategies.

Key words. GMRES, backward error analysis, relaxed GMRES

AMS subject classifications. 65F10, 65G99

DOI. 10.1137/040608416

1. Introduction.

We consider the solution of a linear system of equations $Ax = b$ using the GMRES [12] iterative method, where A is a nonsingular $n \times n$ matrix. In some applications, performing inexact matrix-vector products in this method may be interesting as long as the convergence of GMRES is maintained. Many situations in scientific computing may benefit from such a scheme. For instance, a natural application of this idea occurs in computational electromagnetics, where the fast multipole method provides approximations of the matrix-vector product within a user-defined accuracy; the less accurate the matrix-vector, the faster the computation. The key point is then to design a criterion to control the accuracy of the matrix-vector product so that the iterates achieve a satisfactory convergence level. Another example arises in nonoverlapping domain decomposition where the matrix-vector involving the Schur complement can be approximated.

In [4], a criterion is proposed for general systems and its numerical behavior is illustrated on a large set of numerical experiments. This work is based on some heuristic considerations and the approach is referred to as a relaxation strategy because the perturbation size can grow as the inverse of the residual norm. We denote by "relaxed GMRES" an inexact GMRES that implements a relaxation strategy. The relaxation strategy proposed in [4] attempts to ensure the convergence of the GMRES iterates

*Received by the editors May 15, 2004; accepted for publication (in revised form) August 25, 2006; published electronically April 10, 2007. This work was initiated while the three authors were working at CERFACS.

http://www.siam.org/journals/sisc/29-2/60841.html

†ENSEEIHT, 2, rue Charles Camichel, BP 7122, 31071 Toulouse Cedex 7, France (giraud@enseeiht.fr).

‡CERFACS, 42 av. Gaspard Coriolis, 31057 Toulouse cedex 1, France (gratton@cerfacs.fr).

§Department of Mathematical Sciences, The University of Colorado at Denver and Health Sciences Center, Denver, CO (Julien.Langou@cudenver.edu).

x_k within a relative normwise backward error

$$\begin{aligned}\eta_{A,b}(x_k) &= \min_{\Delta A,\Delta b}\{\tau > 0 : \|\Delta A\| \le \tau\|A\|,\ \|\Delta b\| \le \tau\|b\| \\ &\qquad \text{and } (A+\Delta A)x_k = b + \Delta b\} \\ &= \frac{\|Ax_k - b\|}{\|A\|\|x_k\| + \|b\|}, \end{aligned} \tag{1.1}$$

less than a prescribed quantity $\varepsilon > 0$. A significant number of numerical experiments are presented that illustrate the merits of the strategy but also reveal its lack of robustness; for a few examples the convergence at ε is not obtained. Nevertheless, based on [4], similar strategies have been successfully applied to the solution of heterogeneous diffusion problems using domain decomposition [3], to the preconditioning of a radiation diffusion problem [20], to the solution of electromagnetic problems [11], in lattice quantum chromodynamics [5], and in an ocean circulation model for steady barotropic flow [18]. A significant step toward a theoretical explanation of this observed behavior is proposed in [15, 19]. In these latter works, important justifications are brought to the fact that under some assumptions relaxed GMRES converges in residual norm.

The convergence of iterative solvers is often based on normwise backward error criteria [2, 6, 9]. In this paper we propose criteria to control the accuracy of the matrix-vector products and show that they ensure the convergence of GMRES with respect either to $\eta_{A,b}(x_k)$ defined in (1.1) or to

$$\begin{aligned}\eta_b(x_k) &= \min_{\Delta b}\{\tau > 0 : \|\Delta b\| \le \tau\|b\| \text{ and } Ax_k = b + \Delta b\} \\ &= \frac{\|Ax_k - b\|}{\|b\|}. \end{aligned} \tag{1.2}$$

We mention that $\eta_{A,b}$ and η_b are recommended in [2] when the concern related to the stopping criterion is discussed.

The forthcoming development shows that deriving a sufficient convergence condition of relaxed GMRES for the stopping criterion $\eta_b(x_k) \le \varepsilon$ is far simpler than for $\eta_{A,b}(x_k) \le \varepsilon$. However, we believe that both stopping criteria are of practical interest. Both may be used to account for data uncertainties. It is problem and application dependent to decide whether these uncertainties come chiefly from the right-hand side, when η_b must be used, or from the matrix and the right-hand side, when $\eta_{A,b}$ must be used.

The paper is organized as follows. In section 2 we study the convergence of relaxed GMRES. We first recall and discuss in section 2.1 some theoretical results established in [15, 19]. Section 2.2 is devoted to the convergence proofs. In particular, we show that under suitable assumptions only happy breakdowns can occur and we explain how the perturbation size can be monitored to ensure the convergence of relaxed GMRES with respect to either $\eta_{A,b}$ or η_b. We provide a stopping criterion that uses only byproducts of the algorithm and does not require an exact product by A to compute the backward errors. Numerical experiments that illustrate these theoretical results are given in section 2.3. In section 3, we study the situation when the preconditioner is perturbed. This might happen, for instance, when the application of the preconditioner is obtained by solving iteratively an auxiliary linear system. We also exploit the links between relaxed preconditioned GMRES and flexible GMRES (FGMRES) [13], and we derive a straightforward proof of convergence of FGMRES. In section 4

we consider the use of these strategies in a restarted framework. Finally, we conclude with some comments in section 5.

In this paper, the 2-norm of a vector x is denoted by $\|x\|$, and the spectral 2-norm of a matrix A is denoted by $\|A\|$. We use the notation $\sigma_{\max}(A)$ ($\sigma_{\min}(A)$) for the smallest (resp., the largest) singular value of A. The spectral condition number of a matrix is $\kappa(A) = \|A\|\|A^{-1}\| = \sigma_{\max}(A)/\sigma_{\min}(A)$.

2. GMRES with inexact matrix-vector.

2.1. Background and existing results. The GMRES method is based on the Arnoldi recursion $AV_k = V_{k+1}\bar{H}_k$, where $V_k = [v_1, \ldots, v_k]$ is an orthogonal matrix, and $\bar{H}_k$ is a $(k+1) \times k$ upper-Hessenberg matrix. We assume that we are given an initial guess x_0 for the solution $x^\star$ of the system. In this paper, we suppose $x_0 \neq x^\star$. The GMRES algorithm generates a sequence of iterates $\{x_k\}_{k=1,2,\ldots}$ such that x_k realizes the minimum of the 2-norm of the residual $r_k = b - Ax_k$ over the space $x_0 + \mathcal{K}_k(A, v_1)$. The Krylov subspace $\mathcal{K}_k(A, v_1)$ is defined by $\mathcal{K}_k(A, v_1) = \mathrm{span}(v_1, Av_1, \ldots, A^{k-1}v_1)$, where $\beta v_1 = r_0$ and $\beta = \|r_0\| \neq 0$. From the unitary invariance of the 2-norm, we obtain $x_k = x_0 + \delta x_k$, where $\delta x_k = V_k y_k$ and y_k is the solution of the linear least-squares problem $\min_y \|\bar{H}_k y - \beta e_1\|$, e_1 being the first vector of the canonical basis.

We assume that it is possible to monitor the accuracy of the matrix-vector product Av of the Arnoldi procedure. From a mathematical point of view, the inaccuracy can be modeled by introducing a perturbation matrix E, depending possibly on v, such that $(A + E)v$ is the quantity actually computed. At step k of this perturbed Arnoldi algorithm, the vector $w = (A + E_k)v_k$ is orthogonalized against the vectors v_j, $j = 1, \ldots, k$, so that the following relation holds:

$$[(A + E_1)v_1, \ldots, (A + E_k)v_k] = [v_1, \ldots, v_k, v_{k+1}]\bar{H}_k. \tag{2.1}$$

Note that this matrix perturbation approach was taken in [4, 15]. In [19] the inaccuracies are modeled by introducing the vector f_k such that $w = Av_k + f_k$. We took the former approach because it generalizes to the latter ones by setting $E_k = f_k v_k^T$ since $v_k^T v_k = 1$. In the above equality (2.1), referred to as the [illegible] in [15], $V_k = [v_1, \ldots, v_k]$ is an orthogonal matrix and $\bar{H}_k$ is a $(k+1) \times k$ upper-Hessenberg matrix. We first assume (as in [15, 19]) that the matrix-vector product occurring in the initial residual computation is exact, so that $r_0 = b - Ax_0$ and $\beta = \|b - Ax_0\|$. We define the kth iterate of the inexact method by $x_k = x_0 + \delta x_k$, where $\delta x_k = V_k y_k$ and y_k is the solution of the linear least-squares problem $\min_y \|\bar{H}_k y - \beta e_1\|$. Introducing the perturbation matrix $G_k = [E_1 v_1, \ldots, E_k v_k]$ the inexact Arnoldi relation can also be written [19] as an exact Arnoldi relation $\widetilde{A}_k V_k = (A + G_k V_k^T)V_k = V_{k+1}\bar{H}_k$, with $\widetilde{A}_k = A + G_k V_k^T$. This last equality shows that the quantities $\delta x_i, \bar{H}_i$, and v_i for $i \leq k$ generated by the inexact GMRES on $A\,\delta x = r_0$ until step k are the same as those generated by the first k steps of exact GMRES applied to the linear system $\widetilde{A}_k\,\delta x = r_0$. Using classical results on GMRES [14] this observation implies by induction that the norm of the residual $r_0 - \widetilde{A}_k\,\delta x_i$ is monotonically decreasing as i grows, where $i \leq k$. Furthermore, $\widetilde{A}_k\,\delta x_i = \widetilde{A}_i\,\delta x_i$, because $\widetilde{A}_k\,\delta x_i = (A + G_k V_k^T)V_i y_i = (A + G_i V_i^T)V_i y_i = \widetilde{A}_i\,\delta x_i$. Therefore, the norm of the [illegible] $\tilde{r}_k = r_0 - \widetilde{A}_k\,\delta x_k$ decreases with k. Let us denote by r_k the residual $r_0 - A\,\delta x_k$ and let us define $\tilde{r}_0 = r_0$. The inexact Arnoldi recursion can also be written

$$\widetilde{A}_k V_k = V_k H_k + h_{k+1,k} v_{k+1} e_k^T.$$

Using an analogy with the terminology used for the exact GMRES algorithm, we say that a [illegible] occurs at step m if $h_{m+1,m} = 0$. It is important to notice that, because of the orthogonality of V_k, similarly to the exact GMRES framework, such a breakdown [illegible] for $m \leq n$. In what follows, we denote by m the step where the breakdown occurs. At each step the residual gap is defined by $r_k - \tilde{r}_k = (A - \widetilde{A}_k)\,\delta x_k = G_k y_k = \sum_{i=1}^{k} y_{k,i} E_i v_i$, where $y_k = (y_{k,1}, \ldots, y_{k,k})^T \in \mathbb{R}^k$. Let ϵ be a positive real number. Assuming that the inexact method can be run until step ℓ without breakdown, it is shown in [15, Theorem 5.3] that if

$$\|E_k\| \leq \frac{\sigma_{\min}(\bar{H}_\ell)}{\ell} \frac{\epsilon}{\|\tilde{r}_{k-1}\|} \tag{2.2}$$

for $k \leq \ell$ then $\|r_\ell - \tilde{r}_\ell\| \leq \epsilon$.

This important result is easily derived from $r_\ell - \tilde{r}_\ell = \sum_{k=1}^{\ell} y_{\ell,k} E_k v_k$, $|y_{\ell,1}| \leq \|y_\ell\| = \|\bar{H}_\ell^\dagger \beta e_1\| \leq \|\bar{H}_\ell^\dagger\| \|r_0\| = \|\bar{H}_\ell^\dagger\| \|\tilde{r}_0\|$ and

$$|y_{\ell,k}| = |e_k^T \bar{H}_\ell^\dagger \beta e_1| = \left| e_k^T \bar{H}_\ell^\dagger \left(\beta e_1 - \bar{H}_\ell \begin{pmatrix} y_{k-1} \\ 0 \end{pmatrix} \right) \right| \leq \|\bar{H}_\ell^\dagger\| \|\tilde{r}_{k-1}\|,$$

where $1 < k \leq \ell$. Because the norm of the computed residual, $\|\tilde{r}_k\|$, is monotonically decreasing as k grows, $k \leq \ell$, the inequalities (2.2) and

$$\|r_k\| \leq \|r_k - \tilde{r}_k\| + \|\tilde{r}_k\| \tag{2.3}$$

ensure that the residual norm satisfies at step ℓ

$$\|r_\ell\| \leq \epsilon + \|\tilde{r}_\ell\|. \tag{2.4}$$

However, it can be seen that there is an implicit relation in (2.2) linking all the E_k's, $k \leq \ell$, together. The E_k's depend on $\bar{H}_\ell$ which itself depends on the E_k's. This means that it is not possible to implement (2.2) in an algorithm.

Another interesting feature would also be to monitor $\|\tilde{r}_\ell\|$ in (2.4) so that it can be less than any prescribed value for large enough ℓ. In this way, the residual norm $\|r_\ell\|$, which is a key ingredient of the backward errors $\eta_{A,b}(x_\ell)$ and $\eta_b(x_\ell)$, would also be controlled through (2.4). It turns out that contrary to the exact GMRES, $\|\tilde{r}_\ell\|$ is not necessarily zero at the breakdown of the inexact algorithm. For example, for $A = \mathrm{diag}(1, 2)$, $b = (1/\sqrt{2}, 1/\sqrt{2})^T$, and $x_0 = (0, 0)^T$, if we take $E_1 = 0$ and $E_2 = \mathrm{diag}(-2, 0)$, the inexact Arnoldi relation reads

$$[(A + E_1)v_1, (A + E_2)v_2] = [v_1, v_2] \begin{pmatrix} 3/2 & 3/2 \\ 1/2 & 1/2 \end{pmatrix}.$$

We therefore have a breakdown of the inexact GMRES method, but $\|\tilde{r}_1\| = \|\tilde{r}_2\| = 1/\sqrt{10}$. Note that H_2 is singular, and consequently, the linear least-squares problem $\min_y \|\bar{H}_2 y - \beta e_1\|$ is rank deficient. This means that there is an infinite number of solutions to the linear least-squares problem and neither y_2 nor x_2 is well defined.

The purpose of the next section is to fix the problems related to the control of $\|\tilde{r}_\ell\|$ and to the possible singularity of H_m as well as to remove the implicit relation linking all the E_k's, $k \leq \ell$, together. We shall see that this will enable us to design a relaxed GMRES algorithm that is guaranteed to converge with the backward error stopping criteria $\eta_b \leq \varepsilon$ or $\eta_{A,b} \leq \varepsilon$ for any prescribed tolerance ε.

2.2. Theoretical results. We first design a strategy that ensures the convergence of the inexact GMRES iterates so that η_b can be made smaller than any prescribed tolerance; this theoretical result is closely related to the work of [15, 19]. We consider the splitting

$$\eta_b(x_k) \leq \frac{\|r_k - \tilde{r}_k\|}{\|b\|} + \frac{\|\tilde{r}_k\|}{\|b\|}$$

and set accordingly

$$\varepsilon = \varepsilon_g + \varepsilon_c, \tag{2.5}$$

where ε_g (resp., ε_c) is the targeted tolerance for the scaled residual $\frac{\|r_k - \tilde{r}_k\|}{\|b\|}$ (resp., the relative residual norm $\frac{\|\tilde{r}_k\|}{\|b\|}$).

In the framework of the inexact GMRES the situation where the computed residual $\tilde{r}_m$ is zero at the breakdown is referred to as a . Such a situation is of interest because it ensures that $\frac{\|\tilde{r}_k\|}{\|b\|}$ can be made smaller than any prescribed ε_c provided that k is large enough. The following theorem shows a possible way to control the E_k's such that only a happy breakdown eventually occurs.

THEOREM 1 (breakdown in inexact GMRES). m

1. $r_0 \neq 0$ $k \leq m$ H_k $\tilde{r}_k$ $h_{k+1,k}$ $k = m$
2. $0 < c < 1$ $k \leq m$ $\|E_k\| \leq c\frac{\sigma_{\min}(A)}{n}$ $\bar{H}_k$ $(1-c)\sigma_{\min}(A) \leq \sigma_{\min}(\bar{H}_k)$ H_m m happy $\tilde{r}_m = 0$

1. Owing to the inexact Arnoldi relation (2.1), if $h_{k+1,k} = 0$, then we have $\tilde{r}_k = r_0 - \widetilde{A}_k V_k y_k = V_k(\beta e_1 - H_k y_k)$, where y_k minimizes $\|\beta e_1 - H_k y_k\|$. Because H_k is nonsingular, the minimum value is zero, and consequently $\tilde{r}_k = 0$.

On the other hand, suppose, for the purpose of establishing a contradiction that $\tilde{r}_k = 0$, that $h_{k+1,k} \neq 0$. The inexact Arnoldi relation (2.1) implies that $\tilde{r}_k = r_0 - \widetilde{A}_k V_k y_k = V_{k+1}(\beta e_1 - \bar{H}_k y_k)$. Then $\bar{H}_k(2:k+1, 1:k)$ is a square nonsingular upper-triangular matrix. From $\tilde{r}_k = 0$ it follows that $\|\beta e_1 - \bar{H}_k y_k\| = 0$, which in turn implies that $H_k(2:k+1, 1:k)y_k = 0$. Therefore, $y_k = 0$, since $\bar{H}_k(2:k+1, 1:k)$ is nonsingular. Now $\|\tilde{r}_k\| = \|\beta e_1 - \bar{H}_k y_k\| = \beta$, which contradicts $\tilde{r}_k = 0$.

2. The orthogonality of the V_k's and the relaxed Arnoldi relation read $V_{k+1}\bar{H}_k = (A + G_k V_k^T)V_k$, from which it follows, using the variational characterization of singular values, that

$$\sigma_{\min}(A + G_k V_k^T) \leq \sigma_{\min}(\bar{H}_k).$$

Assuming that there exists a constant c, $0 < c < 1$, such that

$$\|E_k\| \leq c\frac{\sigma_{\min}(A)}{n}, \tag{2.6}$$

we get

$$\|G_k V_k^T\| = \left\| \sum_{i=1}^{k} E_i v_i v_i^T \right\| \leq \sum_{i=1}^{k} \|E_i v_i v_i^T\| \leq c\sigma_{\min}(A), \tag{2.7}$$

and inequality $\sigma_{\min}(A) - \|G_k V_k^T\| \leq \sigma_{\min}(A + G_k V_k^T)$ (see [1, Theorem 3.3.16]) implies that $0 < (1-c)\sigma_{\min}(A) \leq \sigma_{\min}(\bar{H}_k)$. This inequality indicates that $\bar{H}_k$ has full rank. In particular, this is true at breakdown, which shows that H_m is nonsingular. The conclusion follows from 1. ∎

The next theorem gives a sufficient condition for the convergence of relaxed GMRES for η_b.

THEOREM 2 (convergence of relaxed GMRES for η_b). [illegible] m [illegible] c [illegible] $0 < c < 1$ [illegible] ε_c [illegible] ε_g [illegible] $k \leq m$ [illegible]

$$\|E_k\| \leq \frac{1}{n}\sigma_{\min}(A) \min\left(c, \frac{(1-c)\|b\|}{\|\tilde{r}_{k-1}\|}\varepsilon_g\right). \tag{2.8}$$

[illegible] ℓ $0 < \ell \leq m$ [illegible]

$$\|\tilde{r}_\ell\| \leq \varepsilon_c \|b\| \tag{2.9}$$

[illegible]

$$\eta_b(x_\ell) \leq \varepsilon_c + \varepsilon_g.$$

[illegible]. Inequality (2.2) holds with $\epsilon = \varepsilon_g \|b\|$, and it follows that, for all $k \leq m$, $\frac{\|r_k - \tilde{r}_k\|}{\|b\|} \leq \varepsilon_g$. It follows from the second statement of Theorem 1 that, for some $\ell \leq m$, $\|\tilde{r}_\ell\| \leq \varepsilon_c \|b\|$. The bound on $\eta_b(x_\ell)$ then follows from (2.3). The step ℓ can be, for instance, the step where the (happy) breakdown occurs. ∎

[illegible] 1. The stopping criterion (2.9) is based on $\|\tilde{r}_k\| = \|\bar{H}_k y_k - \beta e_1\|$ which is a byproduct of the algorithm and does not require any additional matrix-vector product [7]. Similarly the control on $\|E_k\|$ involves only some constants of the problem (n, $\sigma_{\min}(A)$, $\|b\|$) and the byproduct $\|\tilde{r}_k\|$. Finally, the constant c can be any value between zero and one.

We now show a similar result for a prescribed accuracy on the backward error $\eta_{A,b}$. In a first step, using the same technique as for Theorem 2, we obtain in the following lemma a control that ensures the convergence with respect to $\eta_{A,b}$.

LEMMA 1. [illegible] m [illegible] ε_c [illegible] ε_g [illegible] ℓ $0 < \ell \leq m$ [illegible] $k \leq \ell$ $\tilde{r}_{k-1} \neq 0$ [illegible]

$$\|E_k\| \leq \frac{\sigma_{\min}(\bar{H}_\ell)}{n} \frac{\|A\|\|x_\ell\| + \|b\|}{\|\tilde{r}_{k-1}\|}\varepsilon_g. \tag{2.10}$$

[illegible] $\|\tilde{r}_\ell\| \leq \varepsilon_c \|A\|\|x_\ell\|$ [illegible]

$$\eta_{A,b}(x_\ell) \leq \varepsilon_c + \varepsilon_g.$$

[illegible]. Substituting $(\|A\|\|x_\ell\| + \|b\|)\,\varepsilon_g$ for ϵ in (2.2) yields $\frac{\|r_\ell - \tilde{r}_\ell\|}{\|A\|\|x_\ell\| + \|b\|} \leq \varepsilon_g$. The conclusion follows from (2.3) and $\|\tilde{r}_\ell\| \leq \varepsilon_c \|A\|\|x_\ell\| \leq \varepsilon_c\,(\|A\|\|x_\ell\| + \|b\|)$. ∎

The above result is not implementable because of the forward reference to the quantities $\|x_\ell\|$ and $\sigma_{\min}(\bar{H}_\ell)$. The next lemma shows that the control ensuring a happy breakdown $\|\tilde{r}_m\| = 0$, i.e., $\|E_k\| \leq c\frac{\sigma_{\min}(A)}{n}$, also enables us to derive a lower-bound on $\|x_\ell\|$ when $\|\tilde{r}_\ell\| \leq \varepsilon_c \|A\|\|x_\ell\|$ is satisfied. This lower bound is expressed in terms of c, ε_c, and some constants of the problem.

LEMMA 2. [illegible] m [illegible] c x_0 $x^\star$ [illegible] $2c\|x_0\| \leq \|x^\star\|$ $0 < c < 1$ [illegible] $\ell > 0$ [illegible] $k \leq \ell$ $\|E_k\| \leq c\frac{\sigma_{\min}(A)}{n}$ $\|\tilde{r}_\ell\| \leq \varepsilon_c\|A\|\|x_\ell\|$. [illegible]

$$\frac{\|x^\star\|}{4 + 2\varepsilon_c\kappa(A)} \leq \|x_\ell\|. \tag{2.11}$$

[illegible]. Starting from the definition of $\tilde{r}_\ell$ and from $x_\ell = x_0 + \delta x_\ell$, we get

$$\begin{aligned}\tilde{r}_\ell &= b - Ax_0 - (A + G_\ell V_\ell^T)\,\delta x_\ell \\ &= A(x^\star - x_\ell) - G_\ell V_\ell^T x_\ell + G_\ell V_\ell^T x_0,\end{aligned}$$

which shows that

$$x^\star - x_\ell = A^{-1}\tilde{r}_\ell + A^{-1}G_\ell V_\ell^T x_\ell - A^{-1}G_\ell V_\ell^T x_0. \tag{2.12}$$

From the assumptions $\|E_\ell\| \leq c\frac{\sigma_{\min}(A)}{n}$ and $0 < c < 1$ it follows as in (2.7) that $\|G_\ell V_\ell^T\| \leq c\sigma_{\min}(A)$. Using $\|\tilde{r}_\ell\| \leq \varepsilon_c\|A\|\|x_\ell\|$ and $\|A^{-1}\|\sigma_{\min}(A) = 1$ and taking norms in (2.12) yield

$$\|x^\star - x_\ell\| \leq (c + \varepsilon_c\kappa(A))\|x_\ell\| + c\|x_0\|.$$

Since $\|x^\star\| - \|x_\ell\| \leq \|x^\star - x_\ell\|$, we get

$$\|x^\star\| - c\|x_0\| \leq (1 + c + \varepsilon_c\kappa(A))\|x_\ell\| \leq (2 + \varepsilon_c\kappa(A))\|x_\ell\|,$$

where we have used that $c < 1$. The conclusion follows from the assumption $2c\|x_0\| \leq \|x^\star\|$ that implies $\|x^\star\| - c\|x_0\| \geq \|x^\star\|/2$. ∎

We note that the assumption $2c\|x_0\| \leq \|x^\star\|$ is not very stringent as it is satisfied for $x_0 = 0$. The next theorem gives a sufficient convergence condition for the stopping criterion $\eta_{A,b}$. Similarly as for Theorem 2 (see Remark 1) the control on $\|E_k\|$ and the stopping criterion involved in relaxed GMRES are based on byproducts of the algorithm and some constants of the problem. Provided that the above constants are available or easily estimated, this result can be used in a practical implementation.

THEOREM 3 (convergence of relaxed GMRES for $\eta_{A,b}$). [illegible] m [illegible] c x_0 $x^\star$ [illegible] $2c\|x_0\| \leq \|x^\star\|$ $0 < c < 1$ [illegible] ε_c [illegible] ε_g [illegible] $k \leq m$

$$\|E_k\| \leq \frac{1}{n}\sigma_{\min}(A)\min\left(c, (1-c)\frac{\gamma^\star}{\|\tilde{r}_{k-1}\|}\varepsilon_g\right), \tag{2.13}$$

[illegible] $\gamma^\star = \frac{1}{4+2\varepsilon_c\kappa(A)}\|A\|\|x^\star\| + \|b\|$. [illegible] ℓ $\ell \leq m$ [illegible]

$$\|\tilde{r}_\ell\| \leq \varepsilon_c\|A\|\|x_\ell\| \tag{2.14}$$

[illegible]

$$\eta_{A,b}(x_\ell) \leq \varepsilon_c + \varepsilon_g.$$

. We first observe that $\|E_k\| \leq \frac{c}{n}\sigma_{\min}(A)$ implies (see Theorem 1) that a breakdown will necessarily occur; therefore, for some ℓ, $\ell \leq m$, $\frac{\|\tilde{r}_\ell\|}{\|A\|\|x_\ell\|} \leq \varepsilon_c$. Furthermore, from (2.13) it follows that

$$\|E_k\| \leq \left(\frac{(1-c)\sigma_{\min}(A)}{n}\right)\left(\frac{1}{4+2\varepsilon_c\kappa(A)}\|A\|\|x^\star\|+\|b\|\right)\frac{\varepsilon_g}{\|\tilde{r}_{k-1}\|}. \tag{2.15}$$

The first factor in the right-hand side of (2.15) can be bounded above by $\frac{\sigma_{\min}(\bar{H}_\ell)}{n}$, because $\|E_k\| \leq \frac{c}{n}\sigma_{\min}(A)$ and the second statement of Theorem 1 applies. Using the conclusions of Lemma 2, the second factor is bounded above by $\|A\|\|x_\ell\| + \|b\|$ because of (2.11). Therefore, we have $\|E_k\| \leq \frac{\sigma_{\min}(\bar{H}_\ell)}{n}\frac{\|A\|\|x_\ell\|+\|b\|}{\|\tilde{r}_{k-1}\|}\varepsilon_g$. Consequently the conditions of Lemma 1 hold, which concludes the proof. ∎

2. As illustrated in the forthcoming numerical experiments, it might be noticed that at the drawback of smaller perturbations one can derive a control that does not require the knowledge of $\|x^\star\|$ (or of a lower bound). If we replace $\gamma^\star$ by $\gamma^b = \|b\| < \gamma^\star$ in Theorem 3, the theorem is still valid.

So far, similarly to [15, 19], we have considered that the initial residual involved in the inexact algorithm is computed exactly: $r_0 = b - Ax_0$. We now provide an extension to these results which additionally accounts for inaccuracies in the very first matrix-vector product Ax_0. Such a result is crucial when only approximations of the matrix-vector product are available. This is the case, for instance, when A is a Schur complement matrix in domain decomposition where the local systems are solved iteratively [3].

THEOREM 4 (convergence of relaxed GMRES with inexact initial residual). m ε_A ε_b r_0 $\tilde{r}_0 = b+\Delta b-(A+\Delta A)x_0$ $\|\Delta A\| \leq \varepsilon_A\|A\|$ $\|\Delta b\| \leq \varepsilon_b\|b\|$ 3 ℓ $\ell \leq m$ (2.14)

$$\eta_{A,b}(x_\ell) \leq \varepsilon_c + \varepsilon_g + (1+\varepsilon_c+\varepsilon_g)\left(\varepsilon_A\frac{\|x_0\|}{\|x_\ell\|}+\varepsilon_b\right).$$

. We consider the following two algorithms:

- (A_1) is inexact GMRES with the approximated residual $\tilde{r}_0$ as in the theorem's assumption and with initial guess x_0.
- (A_2) is inexact GMRES without any initial perturbations on A or b for the solution of $Ax = b + \Delta b - \Delta A x_0 = \tilde{b}$ with same initial guess x_0.

If the same perturbations E_k are used for both algorithms (A_1) and (A_2), these two algorithms generate the same matrices V_k and H_k. This follows from writing $\tilde{r}_0 = (b+\Delta b)-(A+\Delta A)x_0 = (b+\Delta b-\Delta A x_0)- Ax_0$ (which shows that both algorithms share the same initial residual norm β) and from the fact that the perturbations E_k are the same.

Using Theorem 3 on (A_2) leads to $\frac{\|Ax_\ell-\tilde{b}\|}{\|A\|\|x_\ell\|+\|\tilde{b}\|} \leq \varepsilon_c+\varepsilon_g$. Furthermore, we have

$$\begin{aligned}
\frac{\|Ax_\ell-b\|}{\|A\|\|x_\ell\|+\|b\|} &\leq \frac{\|Ax_\ell-\tilde{b}\|}{\|A\|\|x_\ell\|+\|b\|} + \frac{\|\Delta b-\Delta A x_0\|}{\|A\|\|x_\ell\|+\|b\|} \\
&\leq \frac{\|Ax_\ell-\tilde{b}\|}{\|A\|\|x_\ell\|+\|\tilde{b}\|}\frac{\|A\|\|x_\ell\|+\|\tilde{b}\|}{\|A\|\|x_\ell\|+\|b\|} + \frac{\|\Delta b-\Delta A x_0\|}{\|A\|\|x_\ell\|+\|b\|} \\
&\leq (\varepsilon_c+\varepsilon_g)\left(1+\frac{\|\Delta b-\Delta A x_0\|}{\|A\|\|x_\ell\|+\|b\|}\right) + \frac{\|\Delta b-\Delta A x_0\|}{\|A\|\|x_\ell\|+\|b\|} \\
&\leq (\varepsilon_c+\varepsilon_g) + (1+\varepsilon_c+\varepsilon_g)\frac{\|\Delta b-\Delta A x_0\|}{\|A\|\|x_\ell\|+\|b\|}.
\end{aligned}$$

The fact that

$$\frac{\|\Delta b-\Delta A x_0\|}{\|A\|\|x_\ell\|+\|b\|} \leq \frac{\|\Delta b\|+\|\Delta A\|\|x_0\|}{\|A\|\|x_\ell\|+\|b\|} \leq \frac{\|\Delta b\|}{\|b\|} + \frac{\|\Delta A\|}{\|A\|}\frac{\|x_0\|}{\|x_\ell\|}$$

concludes the proof. ∎

2.3. Numerical experiments. In this section we report on numerical experiments performed with MATLAB and machine precision $\psi \sim 2.10^{-16}$. They illustrate that the theoretical results established in exact arithmetic in the previous section do ensure the convergence of the relaxed GMRES even in the presence of round-off errors. For those experiments we consider only the strategies related to the convergence with respect to $\eta_{A,b}(x_k)$. Consequently the stopping criterion implemented by the algorithm is defined by (2.14). For those experiments we set x_0, ε_A, and ε_b to zero. The matrices E_k are obtained using random matrices (MATLAB function `rand`) multiplied by a relevant scalar to match the upper bound of the corresponding strategies. We set $c = 1/4$, $\varepsilon_c = \varepsilon_g = \varepsilon/2$. We therefore consider the two strategies defined as follows.

1. Strategy $S^\star$:

$$\|E_k\| = \frac{\sigma_{\min}(A)}{4n}\min\left(1, \frac{3\gamma^\star}{2\|\tilde{r}_{k-1}\|}\varepsilon\right).$$

2. Strategy S^b:

$$\|E_k\| = \frac{\sigma_{\min}(A)}{4n}\min\left(1, \frac{3\gamma^b}{2\|\tilde{r}_{k-1}\|}\varepsilon\right).$$

Note that S^b is derived from Remark 2. The strategy S^b is interesting, because it does not require the knowledge of $\|x^\star\|$, but it enables only perturbations smaller than $S^\star$. The two strategies rely on the knowledge of $\sigma_{\min}(A)$, and $S^\star$ relies on the knowledge of the solution norm $x^\star$. In our experiments, we set $b = Ax^\star$, so that $x^\star$ is known, and $\sigma_{\min}(A)$ is computed using the `svd` MATLAB command. Further research is needed to establish similar convergence results while replacing these quantities by others that are simpler to estimate. Finally, for the sake of completeness we depict in Algorithm 1 the details of the implementation of the relaxed GMRES algorithm.

Typical behaviors are presented for the linear system $Ax = b$ in Figure 1, where A is the matrix `PDE225` preconditioned using the incomplete LU factorization (i.e., ILU(t) [14]) with threshold $t = 10^{-1}$ (i.e., we run the algorithms on

Algorithm 1. Relaxed GMRES with strategy S.

Choose a convergence threshold $\varepsilon = \varepsilon_c + \varepsilon_g$
Choose an initial guess x_0
$r_0 = b - Ax_0$; $\beta = \|r_0\|$
$v_1 = r_0/\|r_0\|$;
for $k = 1, 2, \ldots$ **do**
  $z = (A + E_k)v_k$, E_k being such that strategy S holds
  for $i = 1$ `to` k **do**
    $h_{i,k} = v_i^T z$
    $z = z - h_{i,k}v_i$
  end for
  $h_{k+1,k} = \|z\|$
  $v_{k+1} = z/h_{k+1,k}$
  Solve the least-squares problem $\min \|\beta e_1 - \bar{H}_k y\|$ for y_k
  if $\|\tilde{r}_k\| = \|\beta e_1 - \bar{H}_k y_k\| \leq \varepsilon_c \|A\| \|x_k\|$ **then**
    Set $x_k = x_0 + V_k y_k$
    Exit
  end if
end for

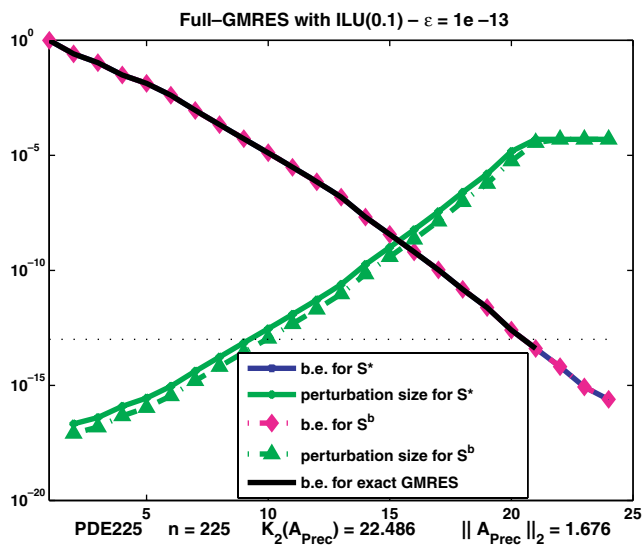


FIG. 1. *Relaxed GMRES with strategy $S^\star$ and S^b,* `PDE225`*, $\varepsilon = 10^{-13}$.*

$A = U^{-1}L^{-1}A_{PDE225}$). The right-hand side b is such that $x^\star = (1, \ldots, 1)^T$ is the solution of $Ax = b$. In this figure, we plot the convergence history, that is, the backward errors $\eta_{A,b}(x_k)$ along the iterations. The line without any tip is the convergence curve with the exact full GMRES; the line with $\diamond$ (resp., $+$) is the convergence of relaxed GMRES with strategy $S^\star$ (resp., S^b). Notice that the three curves perfectly overlap. The dashed horizontal line corresponds to the targeted backward error $\varepsilon = 10^{-13}$, and the line with $\triangle$ (resp., *) is the relative norm of the perturbation $\frac{\|E_k\|}{\|A\|}$ associated

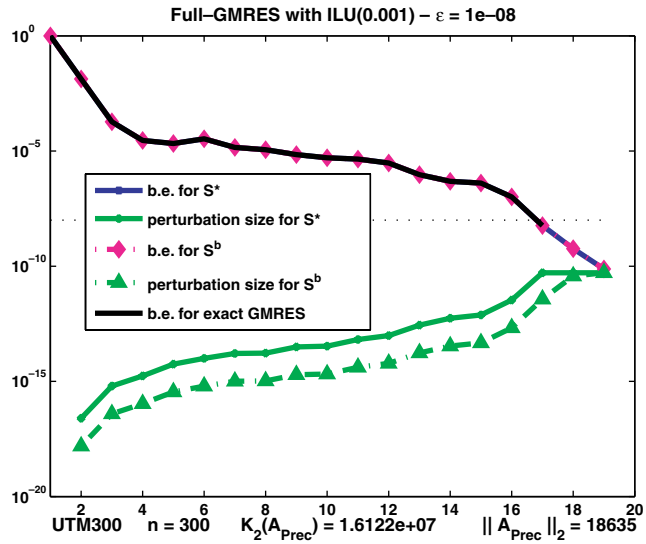


FIG. 2. *Relaxed GMRES with strategy $S^\star$ and S^b,* `UTM300`*, $\varepsilon = 10^{-8}$.*

with strategy $S^\star$ (resp., S^b). It can be seen that as relaxed GMRES converges the accuracy of the matrix-vector product is significantly relaxed without preventing relaxed GMRES from converging to the targeted backward error. We also see that one extra relaxed GMRES iteration is performed before the convergence is detected while $\eta_{A,b}(x_k)$ is already below ε. This delay is due to the stopping criterion implemented in the inexact algorithm that relies on an upper bound for $\eta_{A,b}$ based on (2.3).

In Figure 2, we display the same results for the matrix `UTM 300` preconditioned by ILU(10^{-3}). Again it can be seen that the accuracy of the matrix-vector product can be relaxed without preventing the convergence to occur. However, in that sort of extreme case, the perturbation size is smaller than ψ in the first iterations. This does not make much sense in finite precision calculation because the perturbation below ψ cannot be represented in finite precision. Finally, the example with matrix `UTM 300` better illustrates that S^b is more conservative than $S^\star$.

2.4. Design and behavior of some relaxation heuristics. It is established in [6] that the classical (unperturbed) GMRES algorithm implemented using reliable orthogonal transformation is a backward stable method in finite precision. This means that the quantity $\eta_{A,b}(x_k)$ is of the order of machine precision for some step ℓ, $\ell \leq n$. Let us use this result for a machine precision $\psi = \varepsilon$. Provided that all the operations occurring in the GMRES algorithm are performed with machine precision ε, a backward error $\eta_{A,b}(x_k) \sim C\,\varepsilon$ can be reached, where C depends on the problem size and on the details of the arithmetic. In that context we can define three heuristics that are closely related to the strategies studied in the previous section. They are simply derived by thresholding the norm of the perturbations and preventing them from becoming smaller than ε.

This leads to the definition of the following three heuristics.

1. Heuristic $S^\star(\varepsilon)$:

$$\|E_k\| = \max\left(\varepsilon\|A\|, \frac{\sigma_{\min}(A)}{4n}\min\left(1, \frac{3\gamma^\star}{\|\tilde{r}_{k-1}\|}\varepsilon_g\right)\right).$$

TABLE 2.1
Number of iterations of GMRES with various strategies.

					Heuristics		
Matrix	n	t	ε	N_{ex}	N_ε	$N_\varepsilon^\star$	N_ε^b
e05r0400	236	10^{-3}	10^{-14}	21	21	21	21
e05r0000	236	10^{-2}	10^{-06}	25	25	26	26
GRE115	115	10^{-1}	10^{-10}	15	15	15	15
GRE185	185	10^{-2}	10^{-14}	21	21	21	21
GRE343	343	10^{-1}	10^{-10}	29	29	30	30
CAVITY03	317	10^{-3}	10^{-10}	18	18	18	18
PDE225	225	10^{-1}	10^{-13}	21	21	22	22
SAYLR1	238	10^{-1}	10^{-11}	29	29	30	30
UTM300	300	10^{-3}	10^{-08}	17	17	18	18
WEST0381	381	10^{-2}	10^{-06}	12	12	13	13
BFW398A	398	10^{-1}	10^{-08}	40	40	41	41

2. Heuristic $S^b(\varepsilon)$:

$$\|E_k\| = \max\left(\varepsilon\|A\|, \frac{\sigma_{\min}(A)}{4n}\min\left(1, \frac{3\gamma^b}{\|\tilde{r}_{k-1}\|}\varepsilon_g\right)\right).$$

3. Heuristic $S(\varepsilon)$:

$$\|E_k\| = \varepsilon\|A\|.$$

The latter heuristic is related to exact GMRES run in a floating point arithmetic with machine precision ε.

In Table 2.1 we report on the number of iterations of exact GMRES and relaxed GMRES with the three heuristics on a set of matrices from Matrix Market. In that table we display the size of the matrix n, the threshold used for the $ILU(t)$ preconditioner, and the targeted accuracy ε for $\eta_{A,b}$; the number of exact GMRES iterations is N_{ex}, and the number of relaxed GMRES iterations with heuristic $S(\varepsilon)$ ($S^\star(\varepsilon)$ and $S^b(\varepsilon)$) is N_ε (resp., $N_\varepsilon^\star$ and N_ε^b). For the inexact GMRES the stopping criterion is defined by (2.14). The right-hand side b is such that $x^\star = (1, \ldots, 1)^T$ is the solution of $Ax = b$, where A stands again for the left-preconditioned matrix. We first see that the three variants of inexact GMRES always converge to the targeted accuracy. Even though not displayed for those examples the convergence histories with the three heuristics perfectly overlap the one of exact GMRES along the iterations for all the matrices. We observed this behavior for all the right-hand sides we have considered—not only for $b = A(1, \ldots, 1)^T$. Thus the reason the number of iterations for $S(\varepsilon)$ is sometimes smaller than for $S^\star(\varepsilon)$ and $S^b(\varepsilon)$ solely resides in the stopping criterion which induces a short delay in the convergence detection.

The only example we have encountered that behaves differently is the GRCAR matrix with $b = e_1$ and $\varepsilon \geq 10^{-8}$ (also considered in [15]); in Figure 3 we display for that example the convergence history of $\eta_{A,b}(x_k)$. We see in this peculiar case that exact GMRES converges quickly; that is, the solution lies in a low dimensional Krylov subspace. If inexact matrix-vector products are used, this low dimensional invariant space is not captured as quickly and the convergence is significantly delayed. We mention that this behavior disappears if a smaller value of ε (a larger Krylov space is required) is selected but still exists for larger ε (even smaller "invariant" space). It also disappears for other right-hand sides such as $b = A(1, \ldots, 1)^T$.

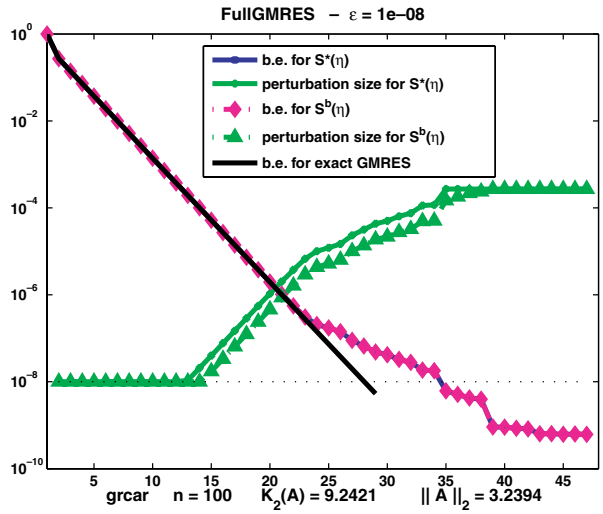


FIG. 3. *Convergence history of the inexact GMRES on* GRCAR *with* $b = e_1$.

3. GMRES with relaxed right-preconditioning. In practice GMRES is very often used with a preconditioner. When the preconditioner is used on the right, we solve the linear system $AM^{-1}u = b$ and we recover the solution by $x^\star = M^{-1}u^\star$. An important class of preconditioners, often referred to as implicit preconditioners, tries to approximate the matrix A and then solves the linear system associated with M (M is chosen such that this operation is much easier with M than with A). We are interested in the case where the application of the preconditioner is inexact: the system $Mz = v$ involved in the preconditioned GMRES is not solved exactly. This happens, for instance, when block preconditioners are used. Examples are Schwarz preconditioners in domain decomposition [17] or, more simply, block Jacobi preconditioners where each block is solved approximately using an iterative scheme. In this case, we assume that the matrix-vector product by A is performed exactly, but we have $Mz = v + p$, where p is the residual associated with the preconditioning operation. Our aim is to find a strategy to monitor $\|p\|$ in such a way that the convergence of GMRES reaches a backward error ε. A related study is developed in [15], where the focus is on the residual norm and the issue related to the recovery of x_ℓ from u_ℓ by the preconditioning $Mx_\ell = u_\ell + p_\ell$ is not addressed.

The inexact Arnoldi relation now reads

$$AM^{-1}V_k + AM^{-1}P_k = (AM^{-1} + AM^{-1}P_kV_k^T)V_k = V_{k+1}\bar{H}_k, \tag{3.1}$$

where $P_k = [p_1, \ldots, p_k]$, p_k being the residual associated with the kth preconditioning operation $Mz_k = v_k + p_k$. We present here results obtained for an unperturbed initial residual r_0. These results extend easily to cover the case where r_0 is perturbed, at the cost of much heavier notation as in Theorem 4. The following theorem, which is a direct application of Theorem 3 on the linear system $AM^{-1}u = b$, shows a possible control of $\|p_k\|$ so that the convergence of the relaxed right-preconditioned GMRES occurs.

THEOREM 5 (convergence of right-preconditioned GMRES for $\eta_{AM^{-1},b}$).

m c u_0 $u^\star$ $2c\|u_0\| \le \|u^\star\|$ $0 < c < 1$

ε_c , ε_g , , ,

TABLE 3.1
GMRES with relaxed right preconditioner.

ρ	$\eta_{A,b}(x_\ell)$	Upper bound
10^{-14}	$4 \cdot 10^{-14}$	$6 \cdot 10^{-14}$
10^{-11}	$3 \cdot 10^{-12}$	$1 \cdot 10^{-11}$
10^{-08}	$3 \cdot 10^{-09}$	$1 \cdot 10^{-08}$
10^{-05}	$3 \cdot 10^{-06}$	$1 \cdot 10^{-05}$
10^{-01}	$7 \cdot 10^{-04}$	$3 \cdot 10^{-03}$

We present a numerical illustration for Theorem 6 in order to demonstrate the effect of the last preconditioning operation on the backward error of the computed solution x_ℓ. We consider the matrix E05R0000 with an $ILU(10^{-2})$ as right-preconditioner and a targeted backward error on the preconditioned system $\eta_{AM^{-1},b}(u_\ell) \leq 10^{-10}$. To recover the unpreconditioned solution we perform a final preconditioning step $Mx_\ell = u_\ell + p$, where a random vector p is chosen such that $\|p\|$ is equal to a prescribed quantity ρ. We report in Table 3.1 the backward errors $\eta_{A,b}(x_\ell)$ obtained at the end of the process (inexact right-preconditioned GMRES and the final preconditioned step $Mx_\ell = u_\ell + p$) and the associated upper bound presented in Theorem 6. We see that the upper bound is tight on this example and that the backward error on x_ℓ strongly depends on the accuracy of the final $Mx_\ell = u_\ell + p$ as monitored by the quantity $\rho = \|p\|$.

REMARK 3. We mention that similar results can be derived for inexact left-preconditioning. In that context, the quantity ε_b from Theorem 4 enables us to account for the inaccuracy in the first preconditioning step to build the right-hand side of the preconditioned system.

REMARK 4. We consider FGMRES with a varying preconditioner satisfying $Mz_k = v + p_k$, where p_k is the residual vector whose norm is used to control the convergence of the algorithm. FGMRES is very close to the inexact right-preconditioned GMRES in the sense that the same Arnoldi relation (3.1) holds, which can be rewritten $AZ_k = V_{k+1}\bar{H}_k$. Contrary to the inexact GMRES framework, [14, Proposition 9.3] shows that when a breakdown occurs at step m in FGMRES and if the associated H_m is nonsingular, not only is $\tilde{r}_m$ zero but also $x_m = Z_\ell y_m = A^{-1}b$. In other words, the nonsingularity of H_m yields that, at the breakdown, x_m is the solution to $Ax = b$. Theorem 1 and (3.2) show that the nonsingularity of H_m is guaranteed if $\|p_k\| \leq \frac{c}{n\kappa(AM^{-1})}$.

Theorem 6 mainly emphasizes the role played by the last preconditioning step. Its use in real applications would deserve additional efforts to find reasonable estimates of the quantities involved in the control such that $\kappa(AM^{-1})$ or $\|AM^{-1}\|$.

4. Some remarks on inexact restarted GMRES. In this paper we are interested in relaxation techniques for which theoretical convergence of GMRES can be established. This is the main reason why we have considered only strategies in the context of full GMRES. There is a clear interest in practice to use the restarted GMRES(m) algorithm [12]. Unfortunately, no general theoretical result exists on the convergence of restarted GMRES. Establishing such a result is probably the first step before studying an inexact restarted algorithm. Nevertheless, we mention that some heuristic strategies [4, 8, 16] as well as those described in this paper enable the convergence of the inexact restarted algorithm in practice.

As previously in this paper we focus on provable convergence theory. In that respect, we consider a variant of restarted GMRES where the restart is not governed by the dimension of the Krylov space affordable in term of memory but rather by a targeted backward error η. We study this variant of restarted GMRES as a fixed point iteration scheme, where at each restart (i.e., each fixed point iteration) the residual can be inaccurately computed. This fixed point iteration is summarized in Algorithm 2, where the function *solve* involved in step 5 is any function ensuring that $\eta_{A,r_k}(s_k) \leq \eta$. If *solve* is exact full GMRES we end up with what we denote by GMRES(η). Notice, that Algorithm 2 differs from classical GMRES(m) in that the calculation of s_k is stopped when the associated backward error is less than η rather than on a memory limit. We define the normwise condition number of the linear system [10, p. 121] by $\kappa(A,b) = \frac{\|A^{-1}\|}{\|x^\star\|}(\|A\|\|x^\star\| + \|b\|)$.

Algorithm 2. Fixed point scheme.

1: Let ε_A, ε_b, and η be positive.
2: Choose an initial guess x_0
3: **for** $k = 1, 2, \ldots$ **do**
4: Compute $r_k = (b + \Delta b_k^{res}) - (A + \Delta A_k^{res})x_k$
5: $s_k = solve(A, r_k, \eta)$
6: $x_{k+1} = x_k + s_k$
7: **end for**

where s_k is such that $\eta_{A,r_k} \leq \eta$ with $\|\Delta A_k^{res}\| \leq \varepsilon_A\|A\|$ and $\|\Delta b_k^{res}\| \leq \varepsilon_b\|b\|$.

THEOREM 7. [illegible] $\{x_k\}$ [illegible] 2 [illegible] $\varepsilon_A\kappa(A) < 1/7$ [illegible] $\eta\kappa(A) < 1/7$ [illegible] $\varepsilon_A = \varepsilon_b = 0$ $\lim_{k\to+\infty} x_k = x^\star$ [illegible] $\max(\varepsilon_A, \varepsilon_b) > 0$ [illegible]

$$\frac{\|x_k - x^\star\|}{\|x^\star\|} \leq 5\kappa(A,b)\max(\varepsilon_A, \varepsilon_b)$$

[illegible] k [illegible] s_k [illegible] 2.2 [illegible]. Let us consider the fixed point iteration

$$x_{k+1} = x_k + s_k.$$

Since $s_k = solve(A, r_k, \eta)$, there exists q_k such that

$$As_k = r_k + q_k, \text{ where } \|q_k\| \leq \eta\left(\|A\|\|s_k\| + \|r_k\|\right). \tag{4.1}$$

We set $e_k = x_k - x^\star$, which yields $e_{k+1} = e_k + s_k$. Taking norms in $r_k = b + \Delta b_k^{res} - (A + \Delta A_k^{res})x_k$ leads to

$$\begin{aligned}\|r_k\| &= \|b + \Delta b_k^{res} - (A + \Delta A_k^{res})(x^\star + e_k)\| \\ &= \|\Delta b_k^{res} - \Delta A_k^{res}x^\star - (A + \Delta A_k^{res})e_k\| \\ &\leq \max(\varepsilon_b, \varepsilon_A)(\|b\| + \|A\|\|x^\star\|) + (1 + \varepsilon_A)\|A\|\|e_k\|\end{aligned}$$

and to

$$\|q_k\| \leq \eta \max(\varepsilon_b, \varepsilon_A)(\|b\| + \|A\|\|x^\star\|) + \eta(2+\varepsilon_A)\|A\|\|e_k\| + \eta\|A\|\|e_{k+1}\|. \tag{4.2}$$

Using $e_{k+1} = e_k + s_k$, multiplying (4.1) by A^{-1}, and using the definition of r_k yield

$$\begin{aligned}\|e_{k+1}\| &\leq \|e_k + A^{-1}b + A^{-1}\Delta b_k^{res} - A^{-1}(A + \Delta A_k^{res})(x^\star + e_k)\| + \|A^{-1}\|\|q_k\| \\ &= \|A^{-1}\Delta b_k^{res} - A^{-1}\Delta A_k^{res}(x^\star + e_k)\| + \|A^{-1}\|\|q_k\| \\ &\leq \max(\varepsilon_b, \varepsilon_A)\|A^{-1}\|(\|b\| + \|A\|\|x^\star\|) + \varepsilon_A\kappa(A)\|e_k\| + \|A^{-1}\|\|q_k\|.\end{aligned}$$

Under the assumption $\eta\kappa(A) < 1/7$, we have $(1 - \eta\kappa(A)) > 0$ and inequality (4.2) yields

$$\|e_{k+1}\| \leq \frac{(1+\eta)\max(\varepsilon_b, \varepsilon_A)}{1 - \eta\kappa(A)}\|A^{-1}\|(\|b\| + \|A\|\|x^\star\|) + \frac{2\eta + \varepsilon_A + \eta\varepsilon_A}{1 - \eta\kappa(A)}\kappa(A)\|e_k\|.$$

Suppose that $\eta\kappa(A) < 1/7$ and $\varepsilon_A\kappa(A) < 1/7$; then

$$\|e_{k+1}\| \leq 4/3 \max(\varepsilon_b, \varepsilon_A)\kappa(A, b)\|x^\star\| + 2/3\|e_k\|.$$

If $\varepsilon_A = \varepsilon_b = 0$, $\lim_{k\to+\infty} x_k = x^\star$; else we have

$$\limsup_{k\to\infty} \|e_k\| \leq 4 \max(\varepsilon_b, \varepsilon_A)\kappa(A, b)\|x^\star\|,$$

which concludes the proof. ∎

The above theorem shows that for a large enough k the forward error is bounded by five times the product of a backward error by the normwise condition number $\kappa(A, b)$. This is reasonable from a perturbation theory point of view.

The calculation of s_k by the function *solve* in Algorithm 2 can be implemented by relaxed GMRES. This latter variant is referred to as relaxed GMRES(η). In Figure 4 we display the convergence history of relaxed GMRES(η) for $\eta = 10^{-4}$ and for two different values of ε_A and ε_b. The curves with $\times$ represent the norm of the perturbations involved in the relaxed GMRES. The curves with + represent the backward error $\eta_{A,b}(x_k)$ at each iteration of relaxed GMRES that implements *solve*. In the experiments we did not implement any stopping criterion; we display the first seven outer iterations that result in 45 iterations of relaxed GMRES. It can be observed that the final backward error reached by the algorithm is related to the perturbation size $\max\{\varepsilon_A, \varepsilon_b\}$ imposed on the residual r_k of step 4 of Algorithm 2. Remarkably the perturbations involved in relaxed GMRES of step 5 are significantly larger than $\max\{\varepsilon_A, \varepsilon_b\}$. The spikes observed in the perturbation size curves correspond to the restarts in relaxed GMRES(η).

5. Conclusion. In this paper we are interested either in relaxation techniques for which theoretical convergence of relaxed GMRES can be established or in heuristics that are closely related to the backward stability of the GMRES algorithm with reliable orthogonalization schemes. The proposed strategies ensure the convergence in backward error $\eta_{A,b}$ or η_b down to a prescribed accuracy ε. Finally, our strategies rely on the knowledge of $\sigma_{\min}(A)$, the solution norm that might be difficult to estimate. Further research is needed to establish similar convergence results while replacing these quantities by others that are simpler to estimate.

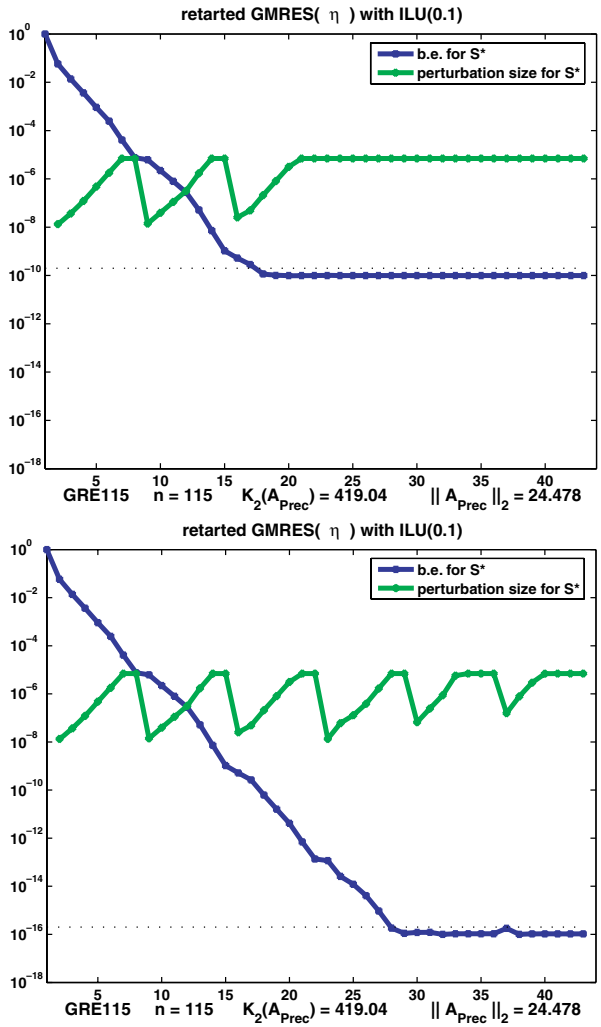


FIG. 4. *Restarted GMRES(ε) with inexact matrix-vector products,* `GRE115`*, with* $\varepsilon_A = \varepsilon_b = 10^{-10}$ *(top), and* $\varepsilon_A = \varepsilon_b = 10^{-16}$ *(bottom).*

Acknowledgments. We would like to thanks the anonymous referees for their valuable remarks, questions, and comments that enabled us to substantially improve this paper. We also express our gratitude to Anne Greenbaum for her continuous encouragement.

REFERENCES

[1] R. A. HORN AND C. R. JOHNSON, *Topics in Matrix Analysis*, Cambridge University Press, Cambridge, UK, 1995.

[2] R. BARRETT, M. BERRY, T. F. CHAN, J. DEMMEL, J. DONATO, J. DONGARRA, V. EIJKHOUT, R. POZO, C. ROMINE, AND H. VAN DER VORST, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, 1993.

[3] A. BOURAS, V. FRAYSSÉ, AND L. GIRAUD, *A Relaxation Strategy for Inner-Outer Linear Solvers in Domain Decomposition Methods*, Technical report TR/PA/00/17, CERFACS, Toulouse, France, 2000.

[4] A. Bouras and V. Frayssé, *Inexact matrix-vector products in Krylov methods for solving linear systems: A relaxation strategy*, SIAM J. Matrix Anal. Appl., 26 (2005), pp. 660–678.

[5] N. Cundy, J. van den Eshof, A. Frommer, S. Krieg, Th. Lippert, and K. Schafer, *Numerical methods for the QCD overlap operator* III: *Nested iterations*, Comput. Phys. Comm., 165 (2005), pp. 221–242.

[6] J. Drkošová, M. Rozložník, Z. Strakoš, and A. Greenbaum, *Numerical stability of the GMRES method*, BIT, 35 (1995), pp. 309–330.

[7] V. Frayssé, L. Giraud, S. Gratton, and J. Langou, *A set of GMRES routines for real and complex arithmetics on high performance computers*, ACM Trans. Math. Software, 31 (2005), pp. 228–238.

[8] L. Giraud, S. Gratton, and J. Langou, *A Note on Relaxed and Flexible GMRES*, Technical report TR/PA/04/41, CERFACS, Toulouse, France, 2004. Available online from http://www.cerfacs.fr/algor/reports.

[9] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, Frontiers Appl. Math. 17, SIAM, Philadelphia, 1997.

[10] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM, Philadelphia, 2002.

[11] K. Mer-Nkonga and F. Collino, *The fast multipole method applied to a mixed integral system for time-harmonic Maxwell's equations*, in Proceedings of the European Symposium on Numerical Methods in Electromagnetics, B. Michielsen and F. Decavèle, eds., ONERA, Toulouse, France, 2002, pp. 121–126.

[12] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems.*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.

[13] Y. Saad, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM J. Sci. Comput., 14 (1993), pp. 461–469.

[14] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.

[15] V. Simoncini and D. B. Szyld, *Theory of inexact Krylov subspace methods and applications to scientific computing*, SIAM J. Sci. Comput., 25 (2003), pp. 454–477.

[16] G. L. G. Sleijpen, J. van den Eshof, and M. B. van Gijzen, *Restarted GMRES with Inexact Matrix–Vector Products*, Technical report TR/PA/04/75, CERFACS, Toulouse, France, 2004.

[17] B. F. Smith, P. E. Bjørstad, and W. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, Cambridge, UK, 1996.

[18] J. van den Eshof, G. L. G. Sleijpen, and M. B. van Gijzen, *Relaxation strategies for nested Krylov methods*, J. Comput. Appl. Math., 177 (2005), pp. 347–365.

[19] J. van den Eshof and G. L. G. Sleijpen, *Inexact Krylov subspace methods for linear systems*, SIAM J. Matrix Anal. Appl., 26 (2004), pp. 125–153.

[20] J. S. Warsa, M. Benzi, T. A. Warein, and J. E. Morel, *Preconditioning a mixed discontinuous finite element method for radiation diffusion*, Numer. Linear Algebra Appl., 11 (2004), pp. 795–811.

SIAM J. SCI. COMPUT.
Vol. 29, No. 2, pp. 729–755
© 2007 Society for Industrial and Applied Mathematics

LIAPUNOV–SCHMIDT REDUCTION AND CONTINUATION FOR NONLINEAR SCHRÖDINGER EQUATIONS*

S.-L. CHANG†, C.-S. CHIEN‡§, AND B.-W. JENG‡¶

Abstract. We study the bifurcation scenario of nonlinear Schrödinger equations (NLS). The Liapunov–Schmidt reduction is applied to show that the simple bifurcations of a single NLS are pitchfork. The pitchfork bifurcation can be subcritical or supercritical, depending on the coefficient of the cubic term we choose. We also describe numerical methods so that the Liapunov–Schmidt reduction can effectively handle a corank-2 bifurcation point. Next, we apply numerical continuation methods to trace solution curves and surfaces of the NLS, where the system is discretized by the centered difference approximations. Numerical results on two- and three-dimensional M-coupled NLS are reported, where the physical properties such as the effect of trapping potentials, isotropic and nonisotropic trapping potentials, mass conservation constraints, and strong and weak repulsive interactions are considered in our numerical experiments.

Key words. Bose–Einstein condensates, time-independent system, bifurcation, mode interactions, mass conservation

AMS subject classifications. 35Q55, 35B32, 65N06

DOI. 10.1137/050642861

1. Introduction.

The M-coupled nonlinear Schrödinger equations (MCNLS) are governed by

$$\begin{aligned} \mathbf{i}\frac{\partial}{\partial t}\Phi_j &= -\Delta\Phi_j + V_j(x)\Phi_j + \mu_j|\Phi_j|^2\Phi_j + \sum_{i\neq j}\beta_{ij}|\Phi_i|^2\Phi_j \\ &\qquad \text{for } x\in\mathbf{R}^n,\ n\le 3,\ t>0, \\ \Phi_j &= \Phi_j(x,t)\in\mathbf{C}, \quad j=1,\dots,M, \\ &\qquad \Phi_j(x,t)\to 0 \text{ as } |x|\to+\infty, \quad t>0. \end{aligned} \tag{1.1}$$

Here $V_j(x)$ is the trapping potential, which usually is harmonic and can be expressed as $V_j(x)=\frac{1}{2}(\alpha_{j,1}x_1^2+\cdots+\alpha_{j,n}x_n^2)$ with $\alpha_{j,1},\dots,\alpha_{j,n}>0$, $j=1,\dots,M$; $\mu_j>0$ being positive constants; and β_{ij} being coupling constants. The trapping potentials are said to be isotropic if $\alpha_{j,1}=\cdots=\alpha_{j,n}$ for $j=1,\dots,M$. Otherwise they are nonisotropic. Equation (1.1) describes a physical model in which M-species Bose–Einstein condensates (BEC) [1, 2, 3] come from ultracold dilute bosonic atoms in a magnetically trapped gas. The solution Φ_j in (1.1) also represents the jth component of the beam in Kerr-like photorefractive media with self-focusing parameter μ_j in nonlinear optics

*Received by the editors October 17, 2005; accepted for publication (in revised form) September 12, 2006; published electronically April 10, 2007.
http://www.siam.org/journals/sisc/29-2/64286.html

†Center for General Education, Southern Taiwan University of Technology, Tainan 710, Taiwan (slchang@mail.stut.edu.tw). This author's research was supported by the National Science Council of R.O.C. (TAIWAN) through project NSC 94-2115-M-218-001.

‡Department of Applied Mathematics, National Chung-Hsing University, Taichung 402, Taiwan (cschien@amath.nchu.edu.tw, d9053401@mail.nchu.edu.tw). The research of the second author was supported by the National Science Council of R.O.C. (TAIWAN) through project NSC 95-2115-M-005-001.

§Corresponding author.

¶Current address: Department of the Application Mathematics, National Chiao Tung University, Hsinchu. 300, Taiwan.

[4]. Equation (1.1) is also known in BEC as the Gross–Pitaevskii equations [5, 6]. Let $\Phi_j(x,t) = e^{-\mathbf{i}\lambda_j t}u_j(x)$ in (1.1); then the time-independent MCNLS can be expressed as

$$
\begin{aligned}
&-\Delta u_j - \lambda_j u_j + V_j(x)u_j + \mu_j u_j^3 + \sum_{i\neq j}\beta_{ij}u_i^2 u_j = 0 \quad \text{in } \mathbf{R}^n,\ n \le 3,\\
&u_j > 0 \quad \text{in } \mathbf{R}^n,\ j = 1,\dots,M,\\
&u_j(x) \to 0 \quad \text{as } |x| \to +\infty.
\end{aligned}
\tag{1.2}
$$

In recent years the BEC has become one of the most popular research areas in experimental and theoretical physics [7, 8, 9, 10], as well as in applied and computational mathematics [11, 12, 13, 14, 15]. In particular, Bao and Tang [11] and Bao, Jaksch, and Markowich [13] studied the numerical solutions of (1.1) for $M = 1$. Hall et al. [16] reported the first experimental results concerning the dynamics of a two-component system of BEC in the different spin states of ^{87}Rb. Then Pu and Bigelow [17] studied the numerical solutions of the one-dimensional two-component BEC using centered difference approximations. Recent research articles concerning numerical solutions of (1.1) and (1.2) for $M > 1$ can be found in [18, 19].

Recently, the authors [20] studied numerical solutions of the steady state MCNLS, $M = 2, 3$, including ground and bound state solutions using continuation methods [21, 22], where the chemical potentials λ_j were treated as the continuation parameter. For $M = 2$ it was shown numerically that the first bifurcation point is pitchfork, which is supercritical if $\beta > 0$ and subcritical if $\beta < 0$. On the other hand, Kuo, Lin, and Shieh [14] studied the local bifurcation behavior of two-coupled nonlinear Schrödinger equations (NLS) by minimizing the potential energy of the system, where the coupling parameter β was used as the continuation parameter. From the viewpoint of numerical continuation, it is more natural to treat the eigenvalues or chemical potentials λ_j as the continuation parameters, since the bifurcation points of the system can easily be determined if $V_j(x) = 0$ for all $j = 1 : M$.

The purpose of this paper is twofold. First, we wish to indicate that the numerical continuation method described in [20] can be used to solve (1.2) with $M = 1$ under the normalization [12],

$$
\int_\Omega |u(x)|^2 dx = 1, \tag{1.3}
$$

where Ω is a subset of $\mathbf{R}^n$. Equation (1.3) means that we are concerned only with mass conservation of the physical system. Starting from the trivial solution curve of the discrete problem, i.e., with $\|u\|_2 = 0$, we numerically trace the nontrivial solution curve, say, branching from the first bifurcation point of (1.2). We stop if $\|u^{(k)}\|_2 = 1$ is reached, where k denotes the kth continuation step and $(u^{(k)}, \lambda^{(k)}, \mu)$ is the desired solution. Note that we may vary the value of μ as we wish. The numerical continuation method can also be used to solve (1.2) for $M > 1$ with the constraints

$$
\int_\Omega |u_j(x)|^2 dx = 1, \quad j = 1,\dots,M. \tag{1.4}
$$

However, as we may see from our numerical result for $M = 2$, we can obtain only

$$
\|u_2(x)\|_2 < \|u_1(x)\|_2 = 1
$$

if the coupling coefficient $\beta < 0$. On the other hand, if $\beta > 0$, then both $u_1(x)$ and $u_2(x)$ satisfy (1.4) but for different values of λ. We may explain this phenomenon as "phase separation."

Next, we will apply the Liapunov–Schmidt reduction [23, Chapters 1 & 7] to study the bifurcation scenario of a single NLS at simple eigenvalues. The Liapunov–Schmidt reduction has been widely used to analyze local bifurcation behavior of bifurcation problems. In [24] Böhmer proposed a numerical Liapunov–Schmidt method for studying the local bifurcation scenario of operator equations. Then Böhmer and Mei [25] used the numerical Liapunov–Schmidt method to compute the solution manifolds bifurcating at a singular zero point of an operator equation. The method was then applied to calculate reduced equations at Hopf bifurcations of the two-dimensional Brusselator equations on a square with Neumann and Dirichlet boundary conditions [26]. It is interesting to see that the Liapunov–Schmidt reduction process is also used in [15] to prove the existence of a bound state solution of (1.2).

In this paper, we consider (1.2) defined in $(0,1)^n,\ n \leq 3$. We show that the first bifurcation of a single NLS is pitchfork. The pitchfork bifurcation can be supercritical or subcritical, depending on the coefficient of the cubic term we choose. For two- and three-dimensional cases the second bifurcation points are of corank-2 and corank-3, respectively. We show that a double bifurcation of a single NLS can be split into two simple ones if the trapping potential is nonisotropic. To the best of our knowledge, the Liapunov–Schmidt reduction cannot be used to treat multiple eigenvalues; see, e.g., [23, Chapter 7, p. 326]. To overcome this difficulty, we describe some numerical methods so that the Liapunov–Schmidt reduction can be effectively applied. Nevertheless, the Liapunov–Schmidt reduction can give only qualitative properties of the system. In order to investigate the quantitative behavior, we need to apply the numerical continuation methods for curve-tracking. We will see how the solutions affect each other using numerical methods. Theoretical study of three-coupled NLS can be found, e.g., in [15].

The bifurcation scenario of the coupled NLS is getting more complicated. We show that mode interactions of the two-coupled NLS cannot occur. It is possible to use the modified Liapunov–Schmidt method described in [27] to investigate the local bifurcation behavior of the two-coupled NLS. But we will not go into detail.

This paper is organized as follows. In section 2 we briefly review the Liapunov–Schmidt reduction for bifurcation problems. Since a single NLS is odd with respect to the state variables, the computations in the Liapunov–Schmidt reduction can be effectively simplified. Moreover, it becomes clear that the bifurcations of a single NLS in one dimension are pitchfork. We give a detailed analysis for the bifurcation scenario of a single NLS in section 3. We also show that the bifurcation scenario is invariant under the effect of linear potentials. In particular, we discuss mode interactions for the coupled one-dimensional NLS. Additionally, we show how the Liapunov–Schmidt reduction can be used to treat a corank-2 bifurcation of the two-dimensional problem. The same techniques also can be used in the three-dimensional case. In section 4 we study the centered difference approximations for MCNLS. We describe a two-grid centered difference discretization algorithm which can compute the first few eigenpairs of the coupled nonlinear Schrödinger equations to the desired accuracy. The normalized eigenvector can be used as the tangent vector at the bifurcation for branch-switching. Furthermore, we indicate that the numerical continuation methods can be used to solve the MCNLS with the normalization constraints (1.4). Our numerical results are reported in section 5. The test problems

include two- and three-dimensional cases, where solution curves branching from the first simple bifurcation point and a corank-2 bifurcation point are numerically traced. Additionally, the physical properties of the NLS are considered in our numerical experiments, including the effect of trapping potentials, strong and weak repulsive interactions, and mass conservation constraints. Finally, some conclusions are given in section 6.

2. A brief review of the Liapunov–Schmidt reduction. To study the local bifurcation scenario of the MCNLS, we need some basic concepts from Golubitsky and Schaeffer [23]. Let $\mathcal{E}_{x,\lambda}$ be the space of all functions $g : \mathbf{R}^2 \to \mathbf{R}$ that are defined and C_∞ on some neighborhood of the origin. We identify any two functions in $\mathcal{E}_{x,\lambda}$ which are equal as germs. Let $g, h : \mathbf{R}^2 \to \mathbf{R}$ be any two smooth mappings that are defined near the origin. We say that g and h are strongly equivalent if there exist functions $X(x,\lambda)$ and $S(x,\lambda)$ such that the relation $g(x,\lambda) = S(x,\lambda)h(X(x,\lambda),\lambda)$ holds near the origin. In this definition we require that $X(0,0) = 0$, $X_x(x,\lambda) > 0$, $S(x,\lambda) > 0$. We refer to [23, p. 95] for the following proposition.

PROPOSITION 2.1. [illegible] $g \in \mathcal{E}_{x,\lambda}$ [illegible] $\alpha x^k + \beta\lambda x$ [illegible] $(x,\lambda) = (0,0)$

(2.1)
$$g = \frac{\partial}{\partial x}g = \cdots = \left(\frac{\partial}{\partial x}\right)^{k-1} g = \frac{\partial}{\partial \lambda}g = 0 \quad [\text{illegible}] \quad \alpha = \operatorname{sign}\left(\frac{\partial}{\partial x}\right)^k g, \quad \beta = \operatorname{sign}\frac{\partial}{\partial \lambda}\frac{\partial}{\partial x}g.$$

Thus if $k = 3$ in Proposition 2.1, then $g(x,\lambda)$ is strongly equivalent to the normal form $\pm x^3 \pm \lambda x = 0$. We say that (2.1) solves the recognition problem for this normal form. Moreover, the bifurcation problem $g(x,\lambda) = x^3 - \lambda x = 0$ is supercritical since the nontrivial solution lies entirely to the right of the bifurcation point $(0,0)$. On the other hand, $x^3 + \lambda x = 0$ is subcritical because the nontrivial solutions lies entirely to the left of the bifurcation point.

Let $\Phi : X \times \mathbf{R}^{k+1} \to Y$ be a smooth mapping between Banach spaces X and Y. We will solve

$$\Phi(u,\alpha) = 0, \tag{2.2}$$

where $u \in X$ and $\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_k) \in \mathbf{R}^{k+1}$. We use α_0 as the continuation parameter and fix $\alpha_1, \ldots, \alpha_k$, say $\alpha_0 = \lambda$. Let L be the differential of Φ with respect to $(u,\lambda) = (0,0)$, i.e., $L = (d\Phi)_{0,0} = \Phi_u(0,0)$. Assume that the differential operator $L : X \to Y$ is Fredholm operator of index 0. The Liapunov–Schmidt reduction is briefly described as follows [23, Chapter 7].

[illegible] 1. Decompose

$$\text{(a) } X = \ker L \oplus M \quad \text{and} \quad \text{(b) } Y = N \oplus R(L), \tag{2.3}$$

where $\ker L$ and $R(L)$ denote the kernel and the range, respectively, of L. For simplicity we assume that $\ker L = \operatorname{span}\{v_0\}$. The discussion given below can be generalized to the case $\dim \ker L \geq 2$.

[illegible] 2. Split (2.2) into an equivalent pair of equations:

$$\text{(a) } E\Phi(u,\lambda) = 0, \quad \text{(b) } (I - E)\Phi(u,\lambda) = 0, \tag{2.4}$$

where $E : Y \to R(L)$ is the projection associated with the splitting.

3. Use (2.3a) to write $u = v + w$, where $v \in \ker L$ and $w \in M$. Define a map $G : \ker L \times M \times \mathbf{R} \to R(L)$ from (2.4a) by

$$G(v, w, \lambda) = E\Phi(v + w, \lambda). \tag{2.5}$$

The differential of G with respect to w at the origin is

$$EL = L.$$

Since the operator L is Fredholm, $R(L)$ is closed. Therefore

$$L : M \to R(L)$$

is invertible. Apply the implicit function theorem to solve (2.4a) for w as a function of v and λ. This leads to a function W, $W : \ker L \times \mathbf{R} \to M$ such that

$$E\Phi(v_0 + W(v_0, \lambda), \lambda) = 0. \tag{2.6}$$

4. Define $\phi : \ker L \times \mathbf{R} \to N$ by

$$\phi(v_0, \lambda) = (I - E)\Phi(xv_0 + W(xv_0, \lambda), \lambda). \tag{2.7}$$

Note that the vector in (2.6) is replaced by xv_0, $x \in \mathbf{R}$, in (2.7).

5. Choose a basis v_0^* for $(R(L))^\perp$. Define $g : \mathbf{R} \times \mathbf{R} \to \mathbf{R}$ by

$$g(x, \lambda) = \langle v_0^*, \Phi(xv_0 + W(xv_0, \lambda), \lambda) \rangle, \tag{2.8}$$

where $\langle r, s \rangle = \int_\Omega r(\xi) s(\xi)\, d\xi$ with Ω the domain of the smooth mapping Φ. The outcome of the Liapunov–Schmidt reduction can be summarized as follows. (See [23, p. 294].)

PROPOSITION 2.2. [illegible] (2.2) [illegible] (2.2) [illegible] $g(x, \lambda) = 0$ [illegible] g [illegible] (2.8)

If the mapping Φ is odd with respect to u, then we have $g_x = g_{xx} = g_\lambda = 0$. Thus if the bifurcation of (2.2) is pitchfork, then we need to compute only

$$\begin{aligned} &\text{(a) } g_{xxx} = \left\langle v_0^*, d^3\Phi(v_0, v_0, v_0) \right\rangle, \\ &\text{(b) } g_{\lambda x} = \left\langle v_0^*, d\Phi_\lambda \cdot v_0 \right\rangle. \end{aligned} \tag{2.9}$$

We refer to [23, Chapter 1] for details.

3. Local bifurcation analysis. We will give a detailed analysis for the local bifurcation behavior of a single NLS. The bifurcation scenario of the MCNLS will be briefly discussed.

3.1. Simple bifurcations of a single NLS. We consider the following equation:

$$\begin{aligned} -\Delta u - \lambda u + \mu u^3 &= 0 \qquad \text{in } \Omega = (0,1)^N,\ N = 1, 2, 3, \\ u &= 0 \qquad \text{on } \partial\Omega. \end{aligned} \tag{3.1}$$

It is well known that the eigenpairs of the linear eigenvalue problem

$$\begin{aligned} -\Delta u + \lambda u &= 0 \quad \text{in } \Omega = (0,1)^N, \\ u &= 0 \quad \text{on } \partial\Omega, \end{aligned} \tag{3.2}$$

for $N = 1$ are

$$\begin{aligned} u_m(x) &= \sin m\pi x, \\ \lambda_m &= m^2\pi^2, \end{aligned} \qquad m = 1, 2, \dots, \tag{3.3}$$

for $N = 2$ are

$$\begin{aligned} u_{m,n}(x, y) &= \sin m\pi x \cdot \sin n\pi y, \\ \lambda_{m,n} &= (m^2 + n^2)\pi^2, \end{aligned} \qquad m, n = 1, 2, \dots, \tag{3.4}$$

and for $N = 3$ are

$$\begin{aligned} u_{m,n,p}(x, y, z) &= \sin m\pi x \cdot \sin n\pi y \cdot \sin p\pi z, \\ \lambda_{m,n,p} &= (m^2 + n^2 + p^2)\pi^2, \end{aligned} \qquad m, n, p = 1, 2, 3, \dots. \tag{3.5}$$

THEOREM 3.1. [illegible] (3.1) [illegible] $\mu > 0$ [illegible] $\mu < 0$ [illegible]. Let $\Phi : X \times \mathbf{R} \to C^0(\bar{\Omega})$ be the mapping defined by

$$\Phi(u, \lambda) = -\Delta u - \lambda u + \mu u^3 = 0,$$

where $X = \{u \in C^2(\Omega) : u = 0 \text{ on } \partial\Omega\}$. First we consider the one-dimensional case. The linear operator $L = (d\Phi)_{0,\lambda} = -\Delta - \lambda$ is singular at $\lambda = \lambda_1$, and $\ker L = \operatorname{span}\{u_1 = \sin \pi x\}$ with $\dim(\ker L) = 1$. Since the linear operator L is self-adjoint, $\ker L^* = \ker L = \operatorname{span}\{u_1\}$. It is obvious that the mapping Φ is odd with respect to u, i.e., $\Phi(-u, \lambda) = -\Phi(u, \lambda)$. Therefore, when $u = 0$ we have $(d^2\Phi)_{0,\lambda} = 0$. We need to compute

$$g_{xxx} = \left\langle u_1, (d^3\Phi)_{0,\lambda_1}(u_1, u_1, u_1) \right\rangle \quad \text{and} \quad g_{\lambda x} = \left\langle u_1, (d\Phi_\lambda)_{0,\lambda_1} \cdot u_1 \right\rangle.$$

Since $\Phi_\lambda(u, \lambda) = -u$, we have $\Phi_\lambda(0, \lambda) = 0$. By definition,

$$(d\Phi_\lambda)_{0,\lambda} \cdot u_1 = \lim_{h \to 0} \frac{\Phi_\lambda(0 + hu_1, \lambda) - \Phi_\lambda(0, \lambda)}{h} = -u_1 = -\sin \pi\xi.$$

Thus, $g_{\lambda x} = \langle u_1, -u_1 \rangle = -\int_0^1 \sin^2 \pi\xi \, d\xi = -\frac{1}{2} < 0$. Next, we compute $(d^3\Phi)_{0,\lambda}(u_1, u_1, u_1)$. By definition,

$$\begin{aligned} d^3\Phi(0, \lambda)(u_1, u_1, u_1) &= \frac{\partial}{\partial t_1}\frac{\partial}{\partial t_2}\frac{\partial}{\partial t_3}\Phi(0 + t_1u_1 + t_2u_1 + t_3u_1, \lambda)\Big|_{(t_1,t_2,t_3)=0} \\ &= 6\mu \cdot \sin^3 \pi\xi. \end{aligned}$$

Since $\int_0^1 \sin^4 \pi\xi \, d\xi = \frac{3}{8}$, we have $(\frac{\partial}{\partial x})^3 g(0, \lambda_1) = \frac{9}{4}\mu$. It follows that the normal form is $\operatorname{sign}(\mu)x^3 - \lambda x = 0$. Thus the pitchfork bifurcation is supercritical if $\mu > 0$, and subcritical if $\mu < 0$. For $N = 2$ we replace u_1 by $u_{1,1} = \sin \pi x \cdot \sin \pi y$ in the proof given above. Note that $\int_0^1 \int_0^1 (\sin \pi\xi \cdot \sin \pi\eta) \, d\xi d\eta = \frac{9}{64}$. The result follows immediately. For $N = 3$, the proof is straightforward. □

COROLLARY 3.2. [illegible] $N = 1$ [illegible] (3.1) [illegible] $\mu > 0$ [illegible] $\mu < 0$ [illegible]. We replace $\sin \pi\xi$ by $\sin m\pi\xi$ in the proof of Theorem 3.1. The result follows immediately. □

Now we impose the term $V(x)u(x)$ on the left-hand side of (3.1) and obtain a single NLS,

$$\begin{aligned} -\Delta u(x) - \lambda u(x) + V(x)u(x) + \mu u^3(x) &= 0 \qquad \text{in } \Omega = (0,1)^N, \\ u &= 0 \qquad \text{on } \partial\Omega. \end{aligned} \tag{3.6}$$

It is clear that the bifurcations of (3.6) are located at the eigenvalues of the Schrödinger eigenvalue problem

$$\begin{aligned} -\Delta u + V(x)u(x) &= \lambda u(x) \qquad \text{in } \Omega = (0,1)^N,\ N = 1,2,3, \\ u &= 0 \qquad \text{on } \partial\Omega. \end{aligned} \tag{3.7}$$

We will show that the bifurcation scenario of (3.6) is the same as that of (3.1).

THEOREM 3.3. [illegible] (3.6) [illegible] $\mu > 0$ [illegible] $\mu < 0$ [illegible]. Let $F : X \times \mathbf{R} \to C^0(\Omega)$ be the mapping defined by

$$F(u,\lambda) = -\Delta u + Vu - \lambda u + \mu u^3,$$

where $X = \{u \in C^2(\Omega) : u = 0 \text{ on } \partial\Omega\}$. Note that the linear operator $L = (dF)_{0,\lambda} = -\Delta + V - \lambda$ is singular at $\lambda = \lambda_1$, and it has a one-dimensional kernel spanned by u_1, where u_1 is the eigenvector corresponding to the first minimum eigenvalue of (3.7). Since $F(-u,\lambda) = -F(u,\lambda)$ for all $\lambda \in \mathbf{R}$, the reduced function g at the bifurcation $(0,\lambda_1)$ satisfies $g = g_x = g_{xx} = g_\lambda = 0$. To prove that the first bifurcation of (3.6) is pitchfork, we only need to compute

$$g_{xxx} = \left\langle u_1,\ (d^3F)_{0,\lambda_1}(u_1,u_1,u_1) \right\rangle \quad \text{and} \quad g_{\lambda x} = \left\langle u_1,\ (dF_\lambda)_{0,\lambda_1} \cdot u_1 \right\rangle.$$

By definition,

$$\begin{aligned} &(d^3F)_{0,\lambda_1}(u_1,u_1,u_1) \\ &= \frac{\partial}{\partial t_1}\frac{\partial}{\partial t_2}\frac{\partial}{\partial t_3} F(0 + t_1u_1 + t_2u_1 + t_3u_1, \lambda_1)\Big|_{t_1=t_2=t_3=0} \\ &= 6\mu u_1^3. \end{aligned}$$

Thus,

$$g_{xxx} = \left\langle u_1, 6\mu u_1^3 \right\rangle = 6\mu \int_\Omega u_1^4 \neq 0 \quad \text{when } \mu \neq 0.$$

Similarly, $F_\lambda(u,\lambda) = -u$, and we have

$$(dF_\lambda)_{0,\lambda_1} \cdot u_1 = \frac{d}{dt} F_\lambda(0 + tu_1, \lambda_1)\Big|_{t=0} = -u_1.$$

Thus,

$$g_{\lambda x} = \langle u_1, -u_1 \rangle = -\int_\Omega u_1^2 < 0.$$

From this it follows that the reduced function g is equivalent to the normal form $\alpha x^3 - \lambda x = 0$, where $\alpha = \operatorname{sign}(\mu)$. Thus the first bifurcation of (3.6) is pitchfork. Moreover, the pitchfork bifurcation is supercritical if $\mu > 0$, and subcritical if $\mu < 0$. □

3.2. Multiple bifurcations of a single NLS in two dimensions. Consider the following semilinear elliptic eigenvalue problem:

$$\begin{aligned} G(u,\lambda) = \Delta u + \lambda f(u) = 0 \quad & \text{in } \Omega = (0,1)^2, \\ u = 0 \quad & \text{on } \partial\Omega, \end{aligned} \tag{3.8}$$

where $f : \mathbf{R} \to \mathbf{R}$ is a smooth odd function which satisfies $f'(0) \neq 0$ and $f'''(0) \neq 0$. A corank-2 bifurcation point of

$$\begin{aligned} -\Delta u - \lambda u + \mu u^3 = 0 \quad & \text{in } \Omega = (0,1)^2, \\ u = 0 \quad & \text{on } \partial\Omega \end{aligned} \tag{3.9}$$

has the form $\{(0, \lambda_{m,n}) \mid \lambda_{m,n} = (m^2+n^2)\pi^2, m \neq n\}$. It is clear that (3.9) satisfies the assumptions of (3.8). For convenience we assume $m = 1$, $n = 2$, and let $u_1 = \sin \pi x \sin 2\pi y$, $u_2 = \sin 2\pi x \sin \pi y$. The kernel of the linear differential operator L in (3.9) is $\ker L = \operatorname{span}\{u_1, u_2\}$. Another choice is $\ker L = \operatorname{span}\{u_1 + u_2, u_1 - u_2\}$. There are four nontrivial solution curves of (3.9) branching from $(0, \lambda_{1,2})$. Solution curves corresponding to $\operatorname{span}\{u_1, u_2\}$ are called rectangular solutions, and those corresponding to $\operatorname{span}\{u_1 + u_2, u_1 - u_2\}$ are called triangular solutions. The multiplicity of solution curves of (3.9) depends on the symmetry of the domain Ω, which we briefly discuss as follows.

Let $C_0^2(\Omega) := \{u \in C^2(\Omega) \mid u|_{\partial\Omega} = 0\}$ and $X = \left(C_0^2(\Omega)\right)^2$, $Y = (C(\Omega))^2$. If we also consider λ as the auxiliary continuation parameter, then $\Phi : X \times \mathbf{R} \to Y$ defined by $\Phi(u, \lambda) = -\Delta u - \lambda u + \mu u^3$ is a smooth mapping. Let $Z_2 := \{1, -1\}$, and let $D_4 := \{S_1, S_2; S_1', S_2', 1, R, R^2, R^3\}$ be the dihedral group of the unit square Ω. Here R denotes a counterclockwise rotation by $\frac{\pi}{2}$, and S_1 and S_2 are reflections about $x = \frac{1}{2}$ and $y = x$, respectively, which are defined by

$$S_1 \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1-x \\ y \end{bmatrix}, \qquad S_2 \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} y \\ x \end{bmatrix} \quad \forall\ (x,y) \in \Omega.$$

Moreover, $S_1' = R^2 S_1$, $S_2' = R^2 S_2$, and we separate the generators of D_4 by " ; ". Note that Φ is an odd function with respect to the state variables u. Since (3.2) is autonomous, we define $\Gamma := Z_2 \times D_4 = \{\pm S_1, \pm S_2; \pm S_1', \pm S_2', \pm 1, \pm R, \pm R^2, \pm R^3\}$ and consider the group action of Γ on the space Y by $\gamma u = \pm u(\delta^{-1}(x,y))$ for all $\gamma = \pm\delta \in \Gamma$, $\delta \in D_4$, and $u \in Y$. It is clear that the spaces X and Y are Γ-invariant. That is, $u \in X$ (respectively, Y) implies $\gamma u \in X$ (respectively, Y) for all $\gamma \in \Gamma$, $u \in X$ (respectively, Y). Note that the mapping Φ is Γ-equivariant; i.e., $\Phi(\gamma u, \lambda) = \gamma \Phi(u, \lambda)$ for all $\gamma \in \Gamma$; $u \in X$, $\lambda \in \mathbf{R}$.

Mei [28] used the group concept to classify the solution curves of (3.8) with their symmetries. In particular, solution curves branching from $(0, \lambda_{m,n})$ with $(m, n) =$ (odd,odd), (odd,even), and (even,even) can be numerically traced on a subdomain of Ω. However, to the best of our knowledge, the Liapunov–Schmidt reduction cannot be used to derive normal forms at a corank-2 bifurcation; see, e.g., [23, pp. 326–328].

To overcome the difficulty, we replace the unit square in Ω by a rectangular domain, say $\Omega_1 = (0, \ell) \times (0, 1)$, $\ell \neq 1$. Then the eigenpairs are

$$\begin{aligned} u_{m,n}(x,y) &= \sin\left(m\pi \frac{x}{\ell}\right) \sin(n\pi y), \\ \lambda_{m,n} &= \left(\frac{m^2}{\ell^2} + n^2\right)\pi^2, \end{aligned} \qquad m, n = 1, 2, 3, \ldots . \tag{3.10}$$

TABLE 1
The first six eigenvalues and residuals of the corresponding eigenvectors of (3.7) *in two dimensions with various choices of* $V(x)$.

	$V(x_1,x_2)=0$		$V(x_1,x_2)=\frac{x_1^2+x_2^2}{2}$		$V(x_1,x_2)=\frac{2x_1^2+x_2^2}{2}$	
	Eigenvalue	Residual	Eigenvalue	Residual	Eigenvalue	Residual
1	19.7382179255601	0.76[-9]	20.0203333007527	0.10[-9]	20.1608349571284	0.12[-8]
2	49.3396000316915	0.30[-9]	49.6410688722418	0.48[-9]	49.7815705286179	0.50[-9]
3	49.3396000316912	0.30[-9]	49.6410688722418	0.50[-9]	49.8016328664392	0.48[-9]
4	78.9409821378226	0.27[-9]	79.2618044437315	0.46[-9]	79.4223684379288	0.50[-9]
5	98.6554245154593	0.42[-9]	98.9603839634467	0.49[-9]	99.1008856198220	0.50[-9]
6	98.6554245154581	0.42[-9]	98.9603839634474	0.48[-9]	99.1243848025148	0.50[-9]

From (3.10) we have

$$\lambda_{1,2}=\left(\frac{1}{\ell^2}+4\right)\pi^2\neq\lambda_{2,1}=\left(\frac{4}{\ell^2}+1\right)\pi^2.$$

That is, the double eigenvalue $\lambda_{1,2}=\lambda_{2,1}$ in the domain Ω is split into two simple ones, $\lambda_{1,2}\neq\lambda_{2,1}$. Accordingly, the corank-2 bifurcation $(0,\lambda_{1,2})$ of (3.9) will be split into two simple bifurcations $(0,\lambda_{1,2})$ and $(0,\lambda_{2,1})$, which are called perturbed bifurcations. In this case the two triangular solution curves of (3.9) branching from the corank-2 bifurcation point $(0,\lambda_{1,2})$ in the domain $\Omega=(0,1)^2$ disappear automatically because of the domain perturbation. Actually, $\operatorname{span}\{u_{1,2}\pm u_{2,1}\}$ belongs to a two-dimensional subspace in the perturbed domain. The above-mentioned technique was discussed in [29] and was used, e.g., in [30]. In this case the Liapunov–Schmidt reduction can be used to treat the perturbed bifurcation again.

REMARK 3.4. It is obvious from [31, p. 459] that if we choose $\Omega=(0,1)^2$ in our numerical computations but with nonsquare grid points on Ω, then the discrete eigenvalues $\lambda_{1,2}$ and $\lambda_{2,1}$ are distinct. Thus the Liapunov–Schmidt reduction can be applied to treating solution curves bifurcating at $\lambda_{1,2}$ and $\lambda_{2,1}$ of the discrete problem as well.

We are ready to study the bifurcation scenario of (3.6) with $N=2$ again. In general, the eigenvalues of (3.7) are obtained by numerical methods. Our numerical results show that if the trapping potential is isotropic, then the second minimum eigenvalue of (3.7) corresponding to the counterpart of (3.2), namely $\lambda_{1,2}$, is still double. However, if a nonisotropic trapping potential is imposed on (3.7), then the double eigenvalues will be separated into two clustered ones. The distance of the clustered eigenvalues is proportional to that of the two coefficients α_1 and α_2 of $V(x)$. Table 1 lists the first six eigenvalues of (3.7) with various choices of $V(x)$, where the eigenvalues were computed using the two-grid centered difference discretization scheme described in [32] with coarse grid size $\widetilde{h}=\frac{1}{8}$ and fine grid size $h=\frac{1}{128}$. We conclude that the Liapunov–Schmidt reduction can be used to analyze the bifurcation scenario of (3.6) if the trapping potential is nonisotropic. For $N=3$ the bifurcations of (3.8) are of corank-1, 3, or 6. There are $13=\frac{(3^3-1)}{2}$ and $264=\frac{(3^6-1)}{2}$ nontrivial solution curves branching from a corank-3 and a corank-6 bifurcation point, respectively. Tracing solution curves bifurcating at a high corank bifurcation point becomes complicated; we refer to [33] for a numerical study of this subject. However, we still can apply the Liapunov–Schmidt reduction to investigate the local bifurcation behavior of the three-dimensional MCNLS. The technique is similar to the discussion described above and is omitted here.

3.3. Mode interactions of the two-coupled NLS. As we pointed out in section 1, the MCNLS is a nonlinear system involving multiple parameters. The parameters of cubic terms disappear as we evaluate the differential of Φ at $w = (u, v) = (0, 0)$. We will show that mode interactions cannot occur in the two-coupled NLS. We consider

$$\begin{aligned} -\Delta u - \lambda_1 u + \mu_1 u^3 + \beta u v^2 &= 0 \\ -\Delta v - \lambda_2 v + \mu_2 v^3 + \beta u^2 v &= 0 \qquad \text{in } \Omega_1 = (0,\ell)^N,\ N = 1, 2, 3, \\ u &= 0 \qquad \text{on } \partial\Omega_1. \end{aligned} \tag{3.11}$$

For convenience we consider $N = 1$ in (3.11). A change of variable $x = \ell\widetilde{x}$ transforms the domain Ω_1 in (3.11) back to $\Omega = (0, 1)$ again. By using the same notation as before, we can express (3.11) as

$$\begin{aligned} -\frac{1}{\ell^2}\frac{d^2}{dx^2}u - \lambda_1 u + \mu_1 u^3 + \beta u v^2 &= 0 \\ -\frac{1}{\ell^2}\frac{d^2}{dx^2}v - \lambda_2 v + \mu_2 v^3 + \beta u^2 v &= 0 \qquad \text{in } \Omega = (0, 1), \\ u(0) = u(1) = v(0) = v(1) &= 0. \end{aligned} \tag{3.12}$$

Now we consider the case $\lambda_1 = \lambda_2 = \lambda$. Let $X = (C_0^2(\Omega))^2$, $Y = (C(\Omega))^2$, and $w = (u, v)$. The linearization of Φ evaluated at $w = (0, 0)$ is

$$L = (d_w\Phi)_{(0,0,\lambda)} = \begin{pmatrix} -\dfrac{1}{\ell^2}\dfrac{d^2}{dx^2} - \lambda & 0 \\ 0 & -\dfrac{1}{\ell^2}\dfrac{d^2}{dx^2} - \lambda \end{pmatrix}, \tag{3.13}$$

where $L : X \times \mathbf{R} \to Y$.

To examine the spectrum of L, we observe that the direct sum

$$X = \sum_{k=1}^{\infty} \oplus X_k, \quad X_k := \left\{ \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \sin k\pi x \mid c_1, c_2 \in \mathbf{R} \right\}, \quad k \in \mathbf{N}, \tag{3.14}$$

holds under the $L^2(\Omega)$-product. It is clear from (3.13) that the linear operator L maps X_k into itself. The restriction of L in the subspace X_k is a 2×2 matrix

$$M_k(\lambda, \ell) := L\Big|_{X_k} = \begin{pmatrix} \dfrac{k^2\pi^2}{\ell^2} - \lambda & 0 \\ 0 & \dfrac{k^2\pi^2}{\ell^2} - \lambda \end{pmatrix}, \quad k = 1, 2, \ldots . \tag{3.15}$$

The eigenvalues of L consist of those of $M_k \in \mathbf{R}^{2\times 2}$, $k = 1, 2, \ldots$. A stationary bifurcation occurs on the solution curve $w = 0$ at $\lambda = \lambda_0$ if and only if there exists at least one integer $k \in \mathbf{N}$ such that $\det M_k(\lambda_0, \ell) = 0$.

The eigenvectors associated with simple eigenvalues of the operator L are called modes. It is well known that a nonlinear system with multiple bifurcation parameters will have multiple modes. Therefore, secondary bifurcations may arise in the system. This process is called mode interaction [34, pp. 412–414]. Now the operator involves two parameters, namely ℓ and λ. We will show that steady/steady mode interactions cannot occur in (3.12).

THEOREM 3.5.

. We choose λ^* and ℓ_0 such that

$$\det M_k(\lambda^*, \ell_0) = 0 \quad \text{and} \quad \det M_m(\lambda^*, \ell_0) = 0, \quad k \neq m. \tag{3.16}$$

For mode interactions to occur we must have $\lambda^* = \frac{(k^2+m^2)\pi^2}{2\ell_0^2}$. However, λ^* cannot satisfy $\det M_k = 0$ and $\det M_m = 0$ simultaneously unless $k = m$. □

Böhmer and Mei [27] studied mode interactions of second order semilinear elliptic systems using a modified Liapunov–Schmidt reduction and symmetries. The technique described in [27] was applied to studying the local bifurcation behavior of the Brusselators in [35]. It is possible to apply the same technique to the MCNLS. However, we will not go into detail. Detailed discussions concerning mode interactions can be found in [34, Chapters 19 and 20].

4. Centered difference approximations and numerical continuation.

4.1. Centered difference approximations. Without loss of generality, we consider the three-coupled nonlinear Schrödinger equations (TCNLS)

$$\begin{aligned} -\Delta u_1 - \lambda_1 u_1 + \mu_1 u_1^3 + \beta_{12} u_1 u_2^2 + \beta_{13} u_1 u_3^2 &= 0 \\ -\Delta u_2 - \lambda_2 u_2 + \mu_2 u_2^3 + \beta_{21} u_2 u_1^2 + \beta_{23} u_2 u_3^2 &= 0 \qquad \text{in } \Omega = (0,1)^2, \\ -\Delta u_3 - \lambda_3 u_3 + \mu_3 u_3^3 + \beta_{31} u_3 u_1^2 + \beta_{32} u_3 u_2^2 &= 0 \\ u_1 = u_2 = u_3 &= 0 \qquad \text{on } \partial\Omega. \end{aligned} \tag{4.1}$$

Let $B = (\beta_{ij})$ with $\beta_{ii} = \mu_i$ be the coefficient matrix associated with the cubic terms in (4.1). We assume that the matrix B is symmetric. For simplicity we also assume that $\lambda_1 = \lambda_2 = \lambda_3 = \lambda$ is the continuation parameter and that the other parameters are fixed. Let $\mathbf{u} = (u_1, u_2, u_3)$, $X = (C_0^2(\Omega))^3$, and $Y = (C^0(\Omega))^3$. Then the system (4.1) can be expressed as

$$F(\mathbf{u}, \lambda) = \begin{bmatrix} -\Delta u_1 + f_1(u, \lambda) \\ -\Delta u_2 + f_2(u, \lambda) \\ -\Delta u_3 + f_3(u, \lambda) \end{bmatrix} = 0, \tag{4.2}$$

where $F : X \times \mathbf{R} \to Y$ is a smooth mapping. In our numerical experiments we will also consider the cases $\lambda_1 \neq \lambda_2 \neq \lambda_3$ and $\lambda_1 = \lambda_2 \neq \lambda_3$.

Differentiating F with respect to $\mathbf{u}$ at the homogeneous equilibrium $\mathbf{u}_0 = 0$, we obtain the linearization L of F, where

$$\begin{aligned} L := D_{\mathbf{u}} F(\mathbf{u}_0, \lambda) &= \begin{bmatrix} -\Delta + \frac{\partial f_1}{\partial u_1}(0,\lambda) & \frac{\partial f_1}{\partial u_2}(0,\lambda) & \frac{\partial f_1}{\partial u_3}(0,\lambda) \\ \frac{\partial f_2}{\partial u_1}(0,\lambda) & -\Delta + \frac{\partial f_2}{\partial u_2}(0,\lambda) & \frac{\partial f_2}{\partial u_3}(0,\lambda) \\ \frac{\partial f_3}{\partial u_1}(0,\lambda) & \frac{\partial f_3}{\partial u_2}(0,\lambda) & -\Delta + \frac{\partial f_3}{\partial u_3}(0,\lambda) \end{bmatrix} \\ &= \begin{bmatrix} -\Delta - \lambda I & 0 & 0 \\ 0 & -\Delta - \lambda I & 0 \\ 0 & 0 & -\Delta - \lambda I \end{bmatrix}. \end{aligned} \tag{4.3}$$

We discretize (4.2) by the centered difference approximations with uniform meshsize $h = \frac{1}{N+1}$ on the x- and y-axes. The centered difference analogue of (4.2) can be

expressed as

$$F(U,\lambda)=\begin{cases} A_1U_1-\lambda U_1+\mu_1U_1^3+\beta_{12}U_1\circ U_2^2+\beta_{13}U_1\circ U_3^2=0,\\ A_1U_2-\lambda U_2+\mu_2U_2^3+\beta_{21}U_2\circ U_1^2+\beta_{23}U_2\circ U_3^2=0,\\ A_1U_3-\lambda U_3+\mu_3U_3^3+\beta_{31}U_3\circ U_1^2+\beta_{32}U_3\circ U_2^2=0. \end{cases} \tag{4.4}$$

Here $A_1\in\mathbf{R}^{N^2\times N^2}$ is the coefficient matrix associated with the discretization of the Laplacian $-\Delta$, $U_i\in\mathbf{R}^{N^2}$, $i=1,2,3$; $U_i\circ U_j$ denotes the Hadamard product of U_i and U_j; and $U_i^r=U_i\circ\cdots\circ U_i$ stand for the r-times Hadamard products of U_i. It is obvious that $F:\mathbf{R}^K\times\mathbf{R}\to\mathbf{R}^K$ is a smooth mapping with $K=3N^2$. We denote the Jacobian matrix of the mapping F in (4.4) by

$$DF=[D_UF,D_\lambda F]\in\mathbf{R}^{(K+1)\times K}, \tag{4.5}$$

where

$$D_UF=\begin{bmatrix} A_1+\operatorname{diag}(S_1) & \operatorname{diag}(2\beta_{12}U_1\circ U_2) & \operatorname{diag}(2\beta_{13}U_1\circ U_3)\\ \operatorname{diag}(2\beta_{21}U_2\circ U_1) & A_1+\operatorname{diag}(S_2) & \operatorname{diag}(2\beta_{23}U_2\circ U_3)\\ \operatorname{diag}(2\beta_{31}U_3\circ U_1) & \operatorname{diag}(2\beta_{32}U_2\circ U_3) & A_1+\operatorname{diag}(S_3) \end{bmatrix},$$

with $S_1=-\lambda+3\mu_1U_1^2+\beta_{12}U_2^2+\beta_{13}U_3^2$, $S_2=-\lambda+3\mu_2U_2^2+\beta_{21}U_1^2+\beta_{23}U_3^2$, and $S_3=-\lambda+3\mu_3U_3^2+\beta_{31}U_1^2+\beta_{32}U_2^2$, is symmetric and

$$D_\lambda F=\begin{bmatrix} -U_1\\ -U_2\\ -U_3 \end{bmatrix}.$$

The discrete operator corresponding to the linear operator L in (4.3) is denoted by

$$A=\begin{bmatrix} A_1-\lambda I & 0 & 0\\ 0 & A_1-\lambda I & 0\\ 0 & 0 & A_1-\lambda I \end{bmatrix}\in\mathbf{R}^{K\times K}. \tag{4.6}$$

The eigenvalues and corresponding eigenvectors of A_1 are (see [31])

$$\begin{aligned} \lambda_{p,q}&=4(N+1)^2\left(\sin^2\frac{p\pi}{2(N+1)}+\sin^2\frac{q\pi}{2(N+1)}\right),\quad 1\le p,q\le N,\\ U_{p,q}(x_j,y_k)&=\sin\frac{jp\pi}{N+1}\sin\frac{kq\pi}{N+1},\quad 1\le j,k,p,q\le N, \end{aligned} \tag{4.7}$$

for $(x_j,y_k)=(\frac{j}{N+1},\frac{k}{N+1})$. Equations (4.6) and (4.7) show that the bifurcation points of the discrete TCNLS are located at $(U,\lambda)=(0,\lambda_{p,q})$, $1\le p,q\le N$.

4.2. Numerical continuation. We denote a solution curve c of (4.4) by

$$c=\{y(s)=(U(s),\lambda(s))\mid F(y(s))=0,s\in I\}.$$

Here I is any interval in $\mathbf{R}$. Assume that a parameterization via arc length is available on c. Let $y_i=(U^{(i)},\lambda^{(i)})\in\mathbf{R}^{K+1}$ be a point which has been accepted as an approximating point for the solution curve c. In the predictor-corrector continuation method, the predicted point is obtained either by the Euler predictor or the higher order Taylor predictor. Perhaps the simplest one is the Euler predictor

$$z_{i+1,1}=y_i+\delta_it_i.$$

Here $\delta_i > 0$ is the step length, and t_i is the unit tangent vector at y_i, which is obtained by solving

$$\begin{bmatrix} D_U F(y_i) & D_\lambda F(y_i) \\ & r_i^T \end{bmatrix} \cdot t_i = \begin{bmatrix} \overline{0} \\ 1 \end{bmatrix} \tag{4.8}$$

for some constraint vector $r_i \in \mathbf{R}^{K+1}$. The accuracy of the predicted point $z_{i+1,1}$ to the solution curve c must be improved via a corrector process. In general one uses Newton's method with constraints, solving

$$\begin{bmatrix} D_U F(z_{i+1,j}) & D_\lambda F(z_{i+1,j}) \\ & t_i^T \end{bmatrix} \cdot w_j = \begin{bmatrix} -F(z_{i+1,j}) \\ 0 \end{bmatrix}, \quad j = 1, 2, \dots , \tag{4.9}$$

and then setting $z_{i+1,j+1} = z_{i+1,j} + w_j$, $j = 1, 2, \dots$. The constraint vector t_i in (4.9) is the solution to the system (4.8). If y_i lies sufficiently near c, then Newton's method will converge if the step size δ_i is small enough. Equations (4.8) and (4.9) can be expressed as

$$\begin{bmatrix} B & p \\ q^T & r \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}, \tag{4.10}$$

where $p, q, f \in \mathbf{R}^K$ and $r, g \in \mathbf{R}$. A well-known numerical method for solving (4.10) is the block elimination algorithm in [22, Chapter 4], where we have to solve linear systems of the form $D_U F \cdot y = b$. Since $D_U F$ is symmetric, the preconditioned Lanczos method with iterative refinement described in [36] can be used as a linear solver for the TCNLS. A more stable numerical method for solving the bordered linear system (4.10) is the mixed block elimination [37, Chapter 3].

4.3. Testing for the bifurcation points. In numerical continuation methods we have to detect bifurcation points along the solution curve of a nonlinear system of equations. For the coupled NLS (4.1), the primary bifurcation points on the trivial solution curve $\{(0, \lambda) \mid \lambda \geq 0\}$ are located at $(0, \lambda_{p,q})$, where $\lambda_{p,q}$ are defined in (4.7). The discrete bifurcation points of the system (4.4) can be obtained by solving the coupled Schrödinger eigenvalue problem. It is evident that the two-grid discretization scheme described in [32] can be used to compute the first few eigenpairs of the coupled Schrödinger eigenvalue problems. We briefly describe the process as follows.

We express the coupled Schrödinger eigenvalue problem as an operator equation of the following form:

$$F(w, \lambda) = (f_1(u_1, \lambda), f_2(u_2, \lambda), f_3(u_3, \lambda)) = 0, \tag{4.11}$$

where $f_i(u_i, \lambda) = -\Delta u_i + V_i(x)u_i - \lambda u_i = 0$, $i = 1 : 3$, and $w = (u_1, u_2, u_3)$. Let $\widetilde{h}, h \in (0, 1)$ be chosen so that $h = O(\widetilde{h}^2)$. Suppose that (4.11) is discretized, say by the centered difference approximations with uniform meshsizes $\widetilde{h} = \frac{1}{\widetilde{N}+1}$ and $h = \frac{1}{N+1}$ on the coarse and the fine grids $\Omega_{\tilde{h}}$ and Ω_h, respectively, where $\widetilde{N} < N$ are two positive integers. Let $F_{\tilde{h}}$ and F_h be the discrete operators corresponding to (4.11), $A_{\tilde{h}} \in \mathbf{R}^{\widetilde{N}^2 \times \widetilde{N}^2}$ and $A_h \in \mathbf{R}^{N^2 \times N^2}$ the coefficient matrices corresponding to the Laplacian $-\Delta$, and $B_{\tilde{h}}$ and B_h the coefficient matrices corresponding to the linear operator L in (4.3) on the coarse and the fine grids, respectively. The eigenvalues of $B_{\tilde{h}}$ and B_h are obtained by solving

$$(A_{\tilde{h}} + (D_i)_{\tilde{h}} - \lambda I)u = 0, \quad i = 1 : 3, \tag{4.12}$$

and

$$(A_h + (D_i)_h - \lambda I)u = 0, \quad i = 1:3, \tag{4.13}$$

respectively. Here $(D_i)_{\tilde{h}} \in \mathbf{R}^{\widetilde{N}^2 \times \widetilde{N}^2}$ and $(D_i)_h \in \mathbf{R}^{N^2 \times N^2}$ are diagonal matrices whose diagonal entries are the values of V_i at each coarse and fine grid point, respectively. The first bifurcation point of (4.1) is determined by computing the minimal eigenvalues of the two diagonal matrices of $B_{\tilde{h}}$ and B_h. Now the two-grid centered difference discretization algorithm in [32] can be generalized to compute the first few eigenpairs of the matrices $B_{\tilde{h}}$ and B_h. Let $I_{\tilde{h}}^h : \Omega_{\tilde{h}} \to \Omega_h$ be the linear interpolation from the coarse grid $\Omega_{\tilde{h}}$ to the fine grid Ω_h.

ALGORITHM 4.1.

1.

2.

$$(C_i)_h u_{i,h}^{(j)} = \lambda_{i,\tilde{h}}^{(j)} (I_{\tilde{h}}^h u_{i,\tilde{h}}^{(j)}), \quad i = 1:3, \ j = 1, \ldots, k,$$

3.

$$\lambda_{i,h}^{(j)} = \frac{(u_{i,h}^{(j)})^T (C_i)_h (u_{i,h}^{(j)})}{(u_{i,h}^{(j)})^T (u_{i,h}^{(j)})}, \quad i = 1:3, \ j = 1, \ldots, k.$$

4.

(i)

$$x_{i,0}^{(j)} = u_{i,h}^{(j)} / \|u_{i,h}^{(j)}\|_2, \qquad \mu_{i,0}^{(j)} = \lambda_{i,h}^{(j)}, \quad i = 1:3, \ j = 1, \ldots, k.$$

(ii) **For** $i = 1:3$
For $j = 1, \ldots, k$
For $l = 0, 1, 2, \ldots$

$$\left((C_i)_h - \mu_{i,l}^{(j)} I + x_{i,l}^{(j)} (x_{i,l}^{(j)})^T\right) y_{i,l+1}^{(j)} = x_{i,l}^{(j)}$$

$$z_{i,l+1}^{(j)} = y_{i,l+1}^{(j)} / (1 - (x_{i,l}^{(j)})^T y_{i,l+1}^{(j)})$$

$$x_{i,l+1}^{(j)} = z_{i,l+1}^{(j)} / \|z_{i,l+1}^{(j)}\|_2$$

$$\mu_{i,l+1}^{(j)} = (x_{i,l+1}^{(j)})^T (C_i)_h (x_{i,l+1}^{(j)})$$

$\|(C_i)_h x_{i,l+1}^{(j)} - \mu_{i,l+1}^{(j)} x_{i,l+1}^{(j)}\|_2 < \tau$

End
End
End

4.2. The coefficient matrix in (ii) of step 4 is a rank-1 update of the matrix $(C_i)_h$. The solution is obtained by using the Sherman–Morrison formula. We solve the perturbed linear system because the condition number of the coefficient matrix increases as the approximate eigenvalue approaches the exact one in the RQI; see [38] for a detailed discussion. In case the eigenvalue we wish to compute is multiple, we can apply Rayleigh–Ritz subspace iteration [39] to solve the linear systems in the RQI.

4.4. Branch-switching techniques. Numerical methods for switching branches at bifurcation points are well known; see, e.g., [22, Chapter 5]. Perhaps the simplest one is the perturbed bifurcation technique. That is, instead of solving (4.4), we solve the following perturbed system:

$$F(U,\lambda)+d=0, \tag{4.14}$$

where $d\in\mathbf{R}^{3N^2}$ is a perturbation vector yet to be determined. In general, one can choose d so that it has the same mode as the eigenvector in (4.7). This technique can be used to handle multiple bifurcations as well. The discussion in subsection 4.3 shows that the first few eigenpairs of the coupled Schrödinger eigenvalue problem can be accurately computed using Algorithm 4.1. It is clear that the normalized eigenvector can be used as tangent vector for branch-switching at the bifurcation point.

Recently, Chien and Jeng [40] developed some two-grid discretization schemes with two-loop continuation algorithms for tracing solution branches of second order semilinear elliptic eigenvalue problems. It is also possible to develop two-grid discretization schemes for reaction-diffusion systems, which also can be used to solve the coupled nonlinear Schrödinger equations. The details will be given elsewhere.

4.5. MCNLS with mass conservation constraints. So far we treat the time-independent or stationary state MCNLS as a parameter-dependent nonlinear system of equations. However, from the physical viewpoint it is more realistic to impose mass conservation constraints on the system. As an example, we consider the single NLS (see [12])

$$\begin{aligned}\mathbf{i}\varepsilon\Phi_t=-\frac{\varepsilon^2}{2}\Delta\Phi+V(x)\Phi+\mu|\Phi|^2\Phi,\quad t>0,\ x\in\Omega=(0,1)^N,\ N=1,2,3,\\ \Phi(x,t)=0,\quad x\in\partial\Omega,\ t\ge 0.\end{aligned} \tag{4.15}$$

We say that a condensate has strong repulsive interactions if $\varepsilon=o(1)$, and weak interactions if $\varepsilon=O(1)$. Two important invariants of (4.15) are the mass conservation constraint of the wave function

$$N(\Phi)=\int_\Omega|\Phi(x,t)|^2\,dx=1,\quad t\ge 0, \tag{4.16}$$

and the energy conservation

$$E_\mu(\Phi)=\int_\Omega\left[\frac{\varepsilon^2}{2}|\nabla\Phi(x,t)|^2+V(x)|\Phi(x,t)|^2+\frac{\mu}{2}|\Phi(x,t)|^4\right]dx,\quad t\ge 0. \tag{4.17}$$

Setting $\Phi(x,t)=e^{-\mathbf{i}\lambda t/\varepsilon}u(x)$ in (4.15), we obtain the following semilinear elliptic eigenvalue problem:

$$\begin{aligned}\lambda u(x)&=-\frac{\varepsilon^2}{2}\Delta u(x)+V(x)u(x)+\mu u(x)^3\quad\text{in }\Omega,\\ u(x)&=0\quad\text{on }\partial\Omega,\end{aligned} \tag{4.18}$$

for $u(x)$ under the normalization condition

$$\int_\Omega |u(x)|^2 \, dx = 1. \tag{4.19}$$

It is obvious that any eigenvalue λ can be computed from its corresponding eigenfunction $u(x)$ by

$$\begin{aligned} \lambda = \lambda_\mu(u) &= \int_\Omega \left[\frac{\varepsilon^2}{2} |\nabla u(x)|^2 + V(x)|u(x)|^2 + \mu |u(x)|^4 \right] dx \\ &= E_\mu(u) + \int_\Omega \frac{\mu}{2} |\phi(x)|^4 \, dx. \end{aligned} \tag{4.20}$$

We wish to indicate that it is a straightforward application of the continuation methods to solve (4.18) with constraint (4.19). As one may see from (4.20), the chemical potential λ is computed by using the eigenfunction $u(x)$ for some fixed parameter μ. Thus, in the context of continuation methods, we treat λ as the continuation parameter and keep the parameter μ fixed. Starting from the trivial solution curve $\{(0, \lambda) \mid \lambda > 0\}$, we will follow, say, the solution curve branching from the first bifurcation point $(0, \lambda_{1,1})$. Since we are interested only in the target point $(\widetilde{u}, \widetilde{\lambda})$ with $\|\widetilde{u}\|_2 = 1$, we may take a relatively large step size to follow the curve until $\|\widetilde{u}\|_2 = 1$ is reached, say at the kth continuation step. In practice, it is quite impossible that we will hit the target point exactly. Therefore we may use the bisection method or linear interpolation to get $(\widetilde{u}, \widetilde{\lambda})$. We can vary the value of μ as we wish. The techniques we describe above can be generalized to treat the MCNLS with mass conservation constraints.

5. Numerical results. The bifurcation scenario of the MCNLS can supply many interesting phenomena, because the system contains $3M$ parameters. We discretized the MCNLS by using centered difference approximations with various uniform meshsizes. In our numerical experiments, we traced solution curves of the MCNLS branching from $(0, \lambda_{1,1})$ and $(0, \lambda_{2,1})$. Our test problems include two- and three- dimensional cases. In particular, the physical properties of the NLS are considered in our numerical experiments, including the effect of linear potentials, strong and weak repulsive interactions, and mass conservation constraints. For simplicity we choose λ_1 as the continuation parameter and keep the other chemical potentials fixed. The accuracy tolerance of the linear solver as well as the Newton corrector is 10^{-11}. Our numerical experiments were executed on a Pentium 4 computer using the MATLAB language.

1. The effect of linear potentials, $M = 3$. We consider (4.1) under the effect of linear potentials, where $V_1(x) = \frac{3x_1^2+x_2^2}{2}$, $V_2(x) = \frac{5x_1^2+3x_2^2}{2}$, $V_3(x) = \frac{7x_1^2+5x_2^2}{2}$, and $\lambda_2 = 40.0$, $\lambda_3 = 30.0$, $\beta_{ij} = \beta = 10.0$, $\mu_1 = 10.0$, $\mu_2 = 15.0$, and $\mu_3 = 30.0$. Algorithm 4.1 was implemented to compute the first eigenpair $(U_1^{(0)}, \lambda_1^{(0)})$ of the first equation, where we chose $\widetilde{h} = \frac{1}{16}$ and $h = \frac{1}{64}$. Note that in implementing Algorithm 4.1 we should set $h = O(\widetilde{h}^2)$. However, because of the storage limitation of our computer, we only have to choose $h = \frac{1}{64}$. The coefficient matrices on the coarse and fine grids are of order 225×225 and 3969×3969, respectively. The normalized eigenvector $(U_1^{(0)}, 0, 0)$ is used as the tangent vector for branch-switching at the bifurcation point $(0, 0, 0, \lambda_1^{(0)})$. The solution curve of u_1 obtained in this way is displayed in Figure 1 with a dashed line, where the bifurcation point is detected

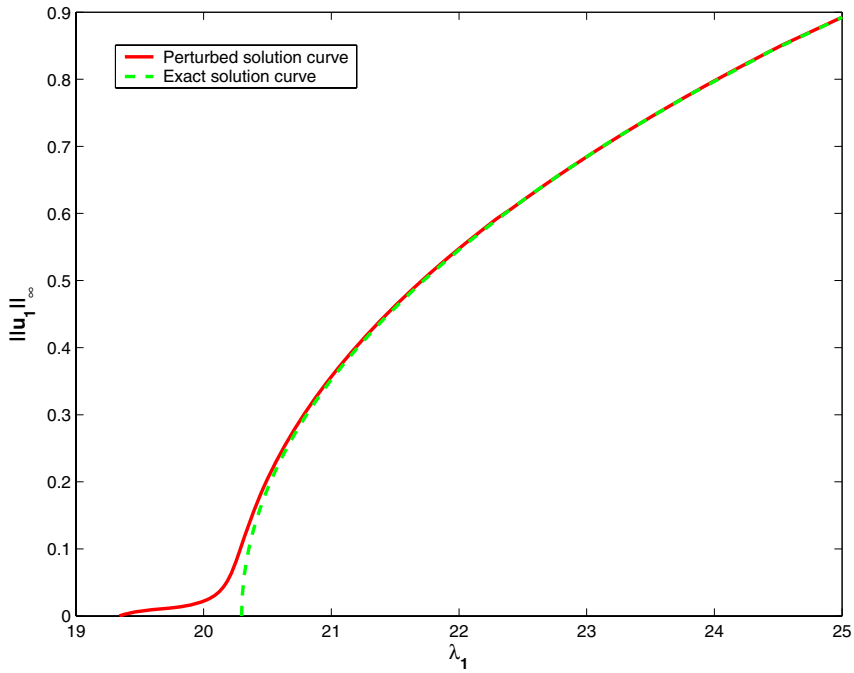


FIG. 1. *The solution curves of* u_1 *at* $\lambda_2 = 40.0$, $\lambda_3 = 30.0$, $\beta = 10.0$, $\mu_1 = 10.0$, $\mu_2 = 15.0$, *and* $\mu_3 = 30.0$.

at $(u_1^*, \lambda_1^*) \approx (0, 20.297808048)$. However, the solid line in Figure 1 is obtained by using the perturbed bifurcation technique, where the bifurcation point was detected at $(u_1^*, \lambda_1^*) \approx (0, 20.2208161)$. Next, we chose $V_1(x) = V_2(x) = V_3(x) = \frac{3x_1^2+x_2^2}{2}$, $\lambda_2 = 49.0$, $\lambda_3 = 30.0$, $\beta = -30.0$, and $\mu_1 = \mu_2 = \mu_3 = 0.1$. The bifurcation was detected at $(u_1^*, \lambda_1^*) \approx (0, 20.2792214)$. Figure 2 shows that a fold is detected at $(\lambda_1, \|u_1\|_\infty) \approx (20.2780507, 0.2655408)$. The contours of the solution curves of u_1 at $\lambda_1 = 10.8328$, -6.7886, -65.2619, and -463.7097 are displayed in Figure 3.

' . ,' 2. Double bifurcation, $M = 2$. We discretized the two-coupled NLS defined in $\Omega = (0, 1)^2$ with uniform meshsize $h = \frac{1}{50}$. The second bifurcation point is double and located at $(0, \lambda_{1,2}) = (0, \lambda_{2,1}) = (0, 5\pi^2)$. Figure 4 shows the two-dimensional contours of the solution curves $U_{1,2}$, $U_{1,2} + U_{2,1}$, $V_{1,2}$, and $V_{1,2} + V_{2,1}$ branching from $(0, \lambda_{1,2})$, where $\lambda_1 = 49.2619$, $\lambda_2 = 60.0$, $\beta = -30.0$, and $\mu_1 = \mu_2 = 0.1$. The contours of $U_{2,1}$, $U_{1,2} - U_{2,1}$, $V_{2,1}$, and $V_{1,2} - V_{2,1}$ can be obtained by rotating their counterparts in Figure 4 clockwise through ninety degrees. The three-dimensional contours of $U_{1,2}$, $V_{1,2}$ and $U_{1,2} + U_{2,1}$, $V_{1,2} + V_{2,1}$ at $\lambda_2 = 60.0$ and $\lambda_1 = 49.2619$, 49.4595, -22.0726, -1018.9563 are displayed in Figures 5 and 6, respectively. Next, we replace the unit square Ω by a rectangular domain $\Omega_1 = (0, \ell) \times (0, 1)$, where we choose $\ell = 0.96$. Thus, the corank-2 bifurcation $(0, \lambda_{1,2})$ of (3.9) is split into two simple bifurcations $(0, \nu_{1,2}) \approx (0, 50.7078)$ and $(0, \nu_{2,1}) \approx (0, 53.1176)$. The solution curves bifurcating at $(0, \nu_{1,2})$ and $(0, \nu_{2,1})$ in the domain Ω_1 and the second bifurcation curve at $(0, \lambda_{1,2})$ in Ω are shown in Figure 7.

' . ,' 3. Strong and weak repulsive interactions. We chose $\varepsilon = 0.1$, $\mu = 1.0$, and $\varepsilon = 1.0$, $\mu = 0.1$ for the former and the later, respectively. Equation (4.18) with $N = 3$ was discretized by the centered difference approximations with uniform

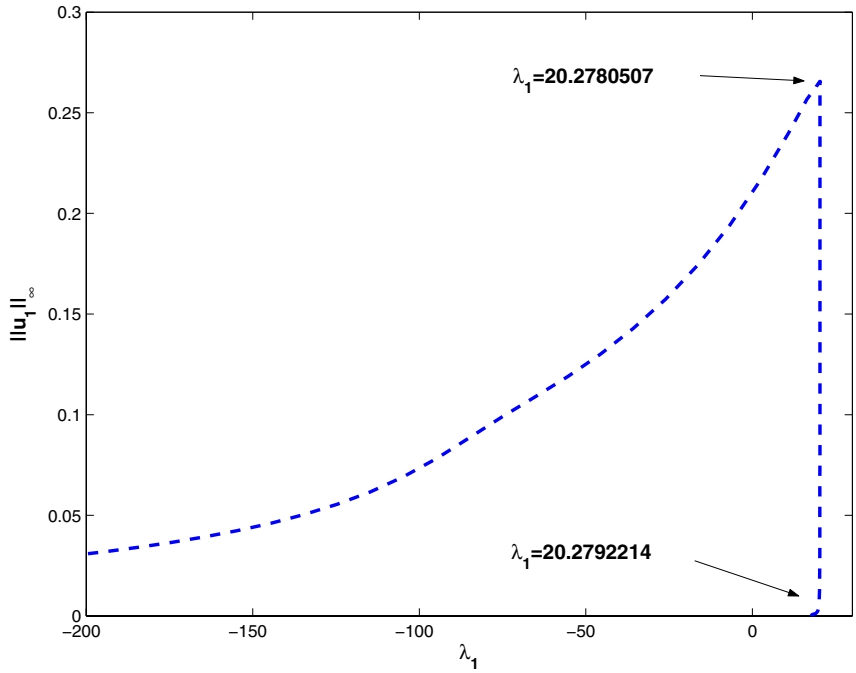


FIG. 2. *The solution curve of* u_1 *at* $\lambda_2 = 49.0$, $\lambda_3 = 30.0$, $\beta = -30.0$, *and* $\mu = 0.1$.

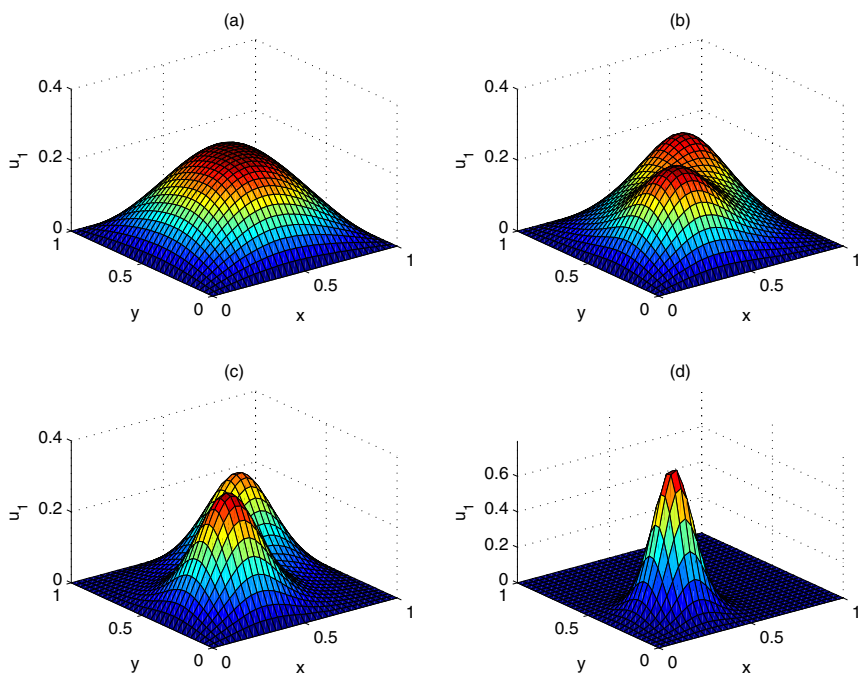


FIG. 3. *The contours of the solutions for* u_1 *at* (a) $\lambda_1 = 10.8328$, (b) $\lambda_1 = -6.7886$, (c) $\lambda_1 = -65.2619$, (d) $\lambda_1 = -463.7097$, *respectively, where* $\lambda_2 = 49.0$, $\lambda_3 = 30.0$, $\beta = -30.0$, *and* $\mu = 0.1$.

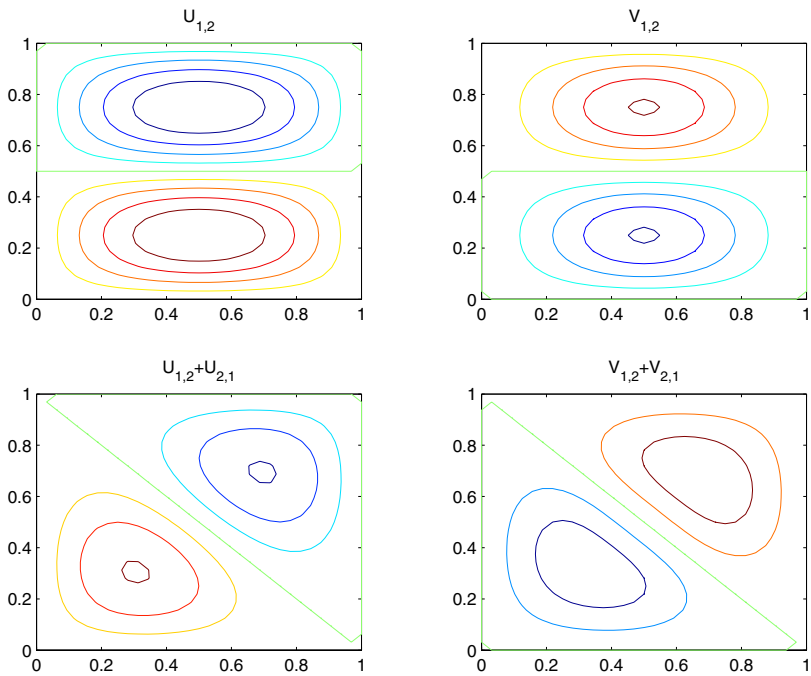


FIG. 4. *The contours of the solution curves bifurcating at* $(0, \lambda_{1,2})$*, where* $\lambda_1 = 49.2619$*,* $\lambda_2 = 60.0$*,* $\beta = -30.0$*, and* $\mu_1 = \mu_2 = 0.1$.

meshsize $h = \frac{1}{32}$, and $V(x) = \frac{1}{2}(x_1^2 + 4x_2^2 + 16x_3^2)$. Figures 8(a) and 8(b) show the solution curves for both cases, where the first bifurcation points were detected at $(0, \lambda^*) \approx (0, 0.977765)$ and $(0, 17.613727)$, respectively. Note that the first eigenvalue is approaching zero if ε is getting smaller.

4. Mass conservation constraints. We discretized (1.2) defined in $(0,1)^3$ with uniform meshsize $h = \frac{1}{20}$ under the normalization (1.3) for $M = 1$ and the constraints (1.4) for $M = 2, 3$. For $M = 1$, we choose $V(x) = \frac{1}{2}(x_1^2 + x_2^2 + x_3^2)$ and $\mu = 0.1$. The solution curve is shown in Figure 9, where the solution $\|u(x)\|_2 = 1$ is reached at the fifteenth continuation step. For $M = 2$, we choose $\lambda_2 = 30.0$, $\mu_1 = \mu_2 = 0.1$. The solution curves with $\beta = -300$ and $\beta = 300$ are displayed in Figures 10 and 11, respectively. In the former, only the component u satisfies (1.4), while in the latter, both u and v satisfy (1.4). For $M = 3$, we choose $\lambda_2 = 29.0$, $\lambda_3 = 50.0$, and $\mu_1 = \mu_2 = \mu_3 = 0.1$. The solution curves of $u_k, k = 1:3$, with $\beta_{ij} = \beta = -300$ and $\beta = 300$ are displayed in Figures 12 and 13, respectively. In the former, both u_1 and u_2 satisfy (1.4), while in the latter, the components u_1 and u_3 satisfy (1.4). For $M = 2$ and $M = 3$ we did not impose linear potentials on the system.

6. Conclusions. We have applied the Liapunov–Schmidt reduction to studying the local bifurcation behavior of a single NLS at simple eigenvalues. We show that the simple bifurcations of the NLS are pitchfork. The pitchfork bifurcation can be supercritical or subcritical, depending on the coefficient of the cubic term we choose. Next, we study how a double bifurcation of a single NLS can be split into two simple ones so that the Liapunov–Schmidt reduction can be applied as well. The methods involved in our study include domain perturbations, choosing nonsquare grid points on the

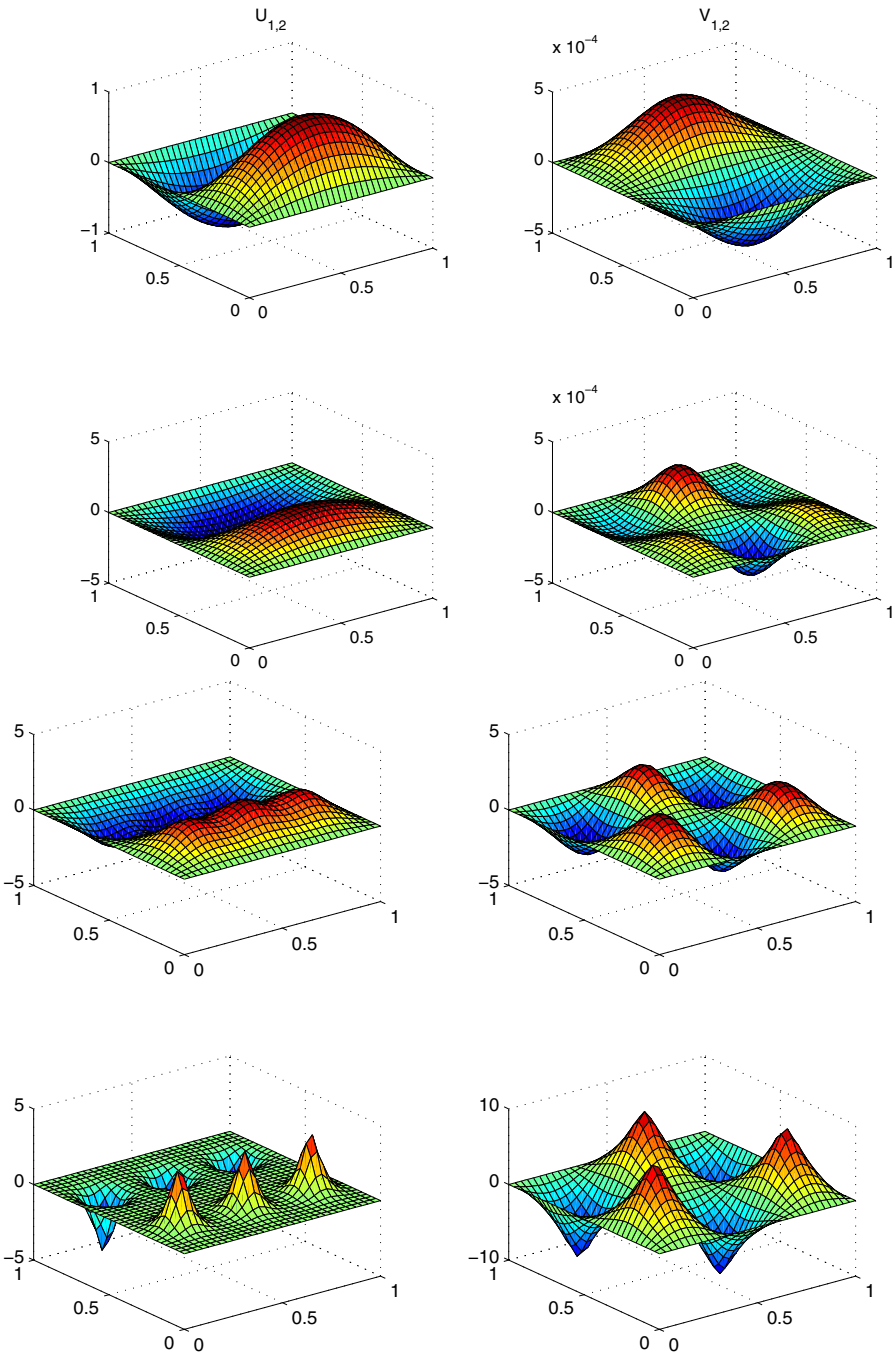


FIG. 5. *The contours of* $U_{1,2}$ *and* $V_{1,2}$ *at* $\lambda_1 = 49.2619,\ 49.4595,\ -22.0726,\ -1018.9563$, *respectively, where* $\lambda_2 = 60.0$, $\beta = -30.0$, *and* $\mu_1 = \mu_2 = 0.1$.

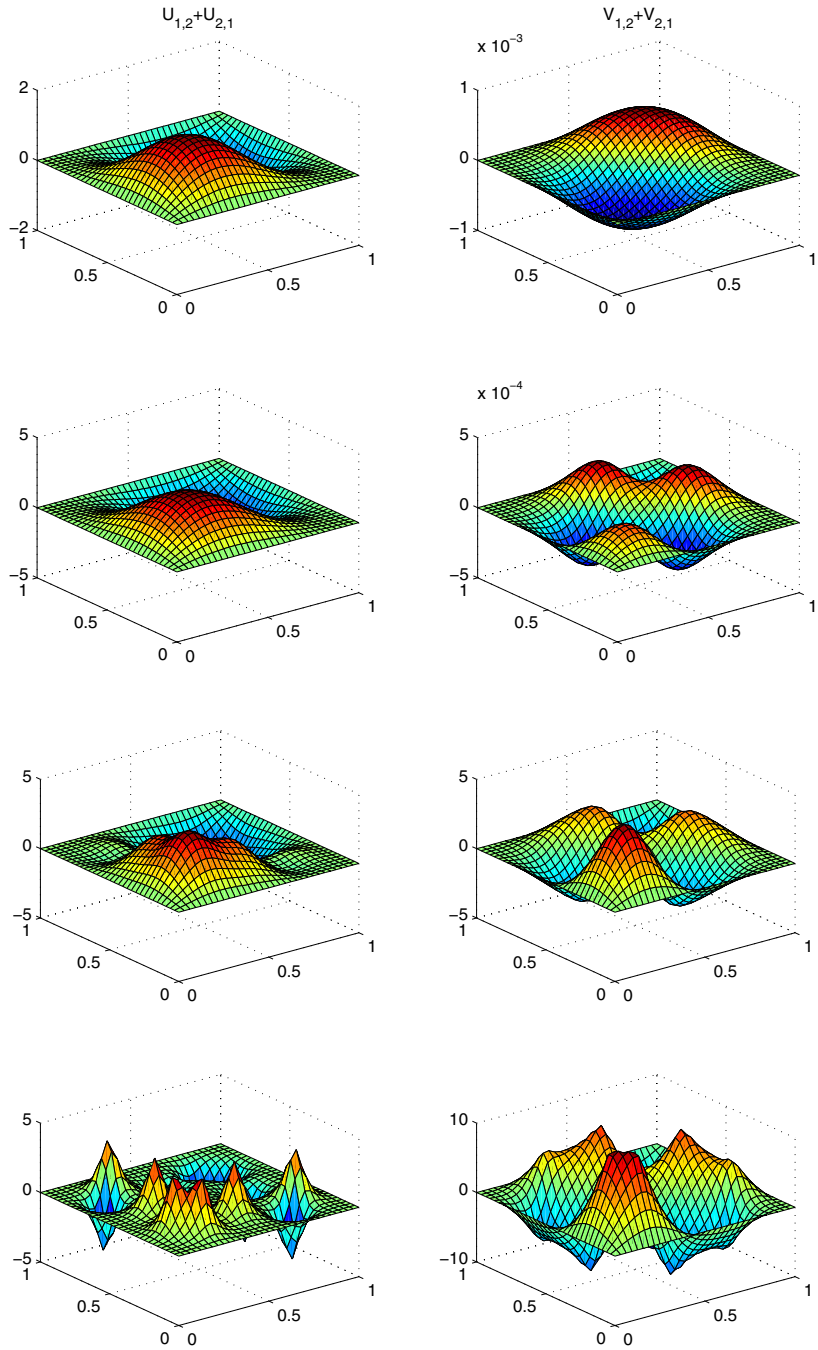


FIG. 6. *The solution surfaces of* $U_{1,2} + U_{2,1}$ *and* $V_{1,2} + V_{2,1}$ *at* $\lambda_1 = 49.2619,\ 49.4595,\ -22.0726,\ -1018.9563$, *respectively, where* $\lambda_2 = 60.0$, $\beta = -30.0$, *and* $\mu_1 = \mu_2 = 0.1$.

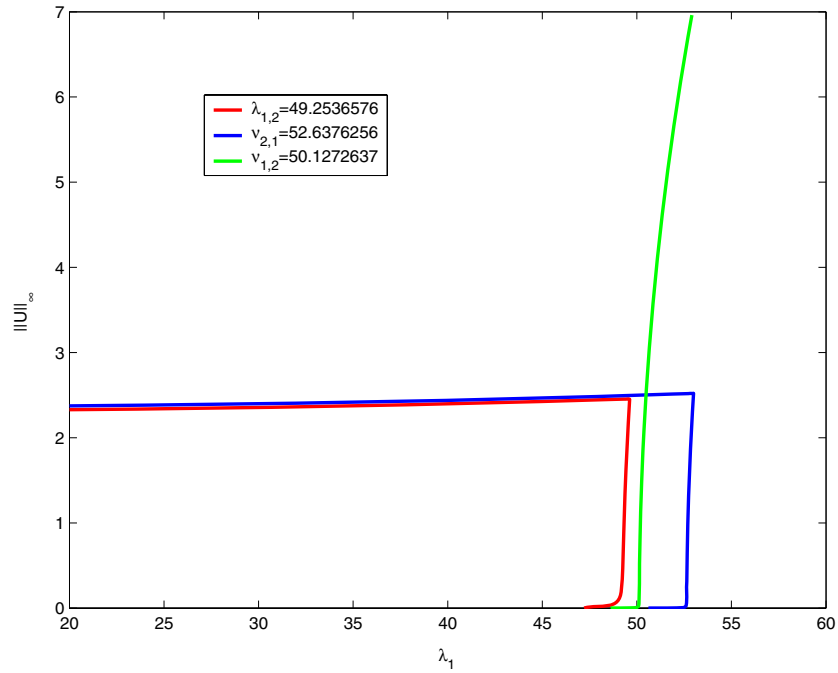


FIG. 7. *The solution curves bifurcating at* $(0, \nu_{1,2})$ *and* $(0, \nu_{2,1})$, $\Omega_1 = (0, 0.96) \times (0, 1)$.

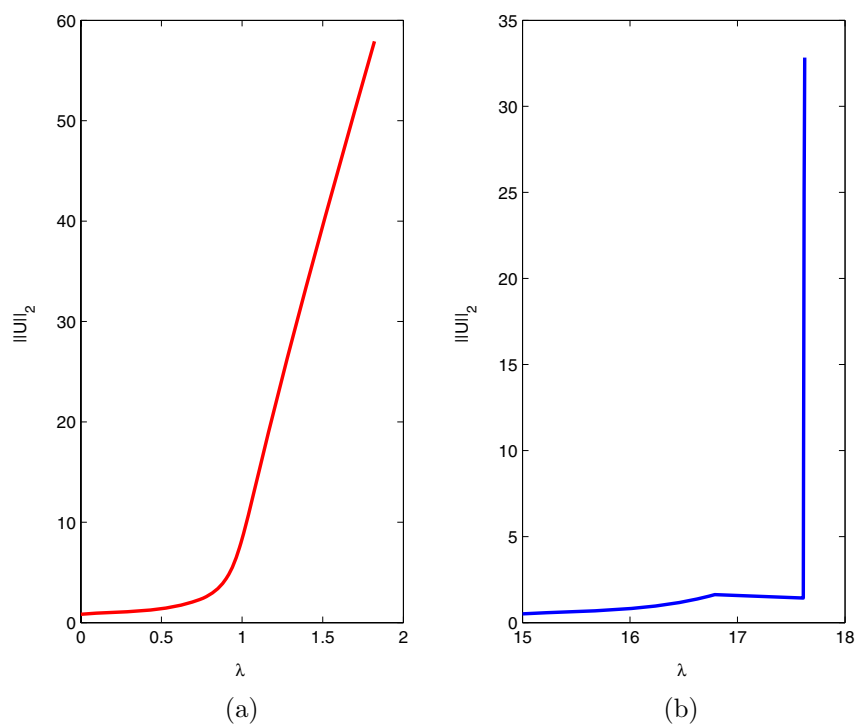


FIG. 8. *The solution curves of* (4.18), $\varepsilon = 0.1$, $\mu = 1.0$ (a), *and* $\varepsilon = 1.0$, $\mu = 0.1$ (b).

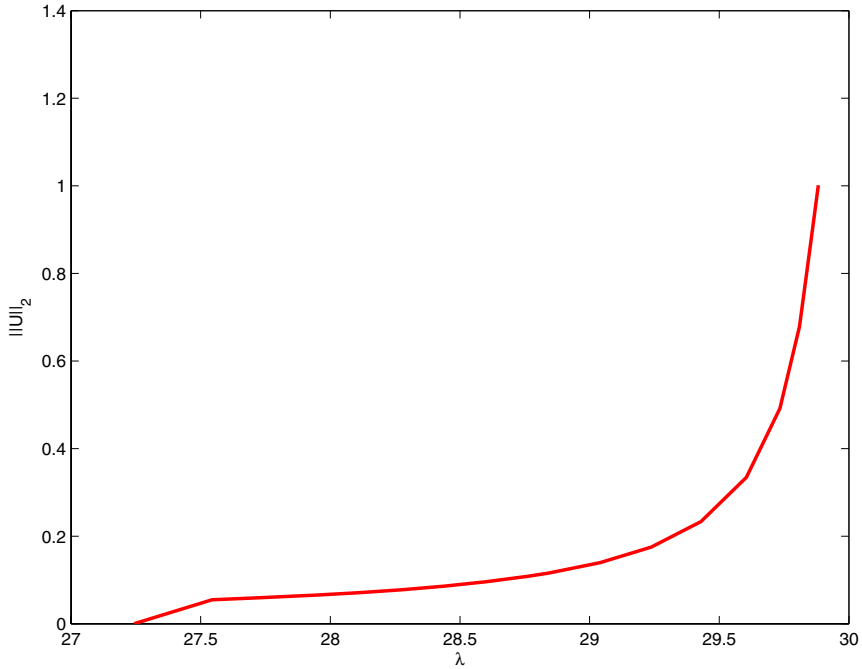


FIG. 9. *The solution curve with* $M = 1$ *and* $\mu = 0.1$.

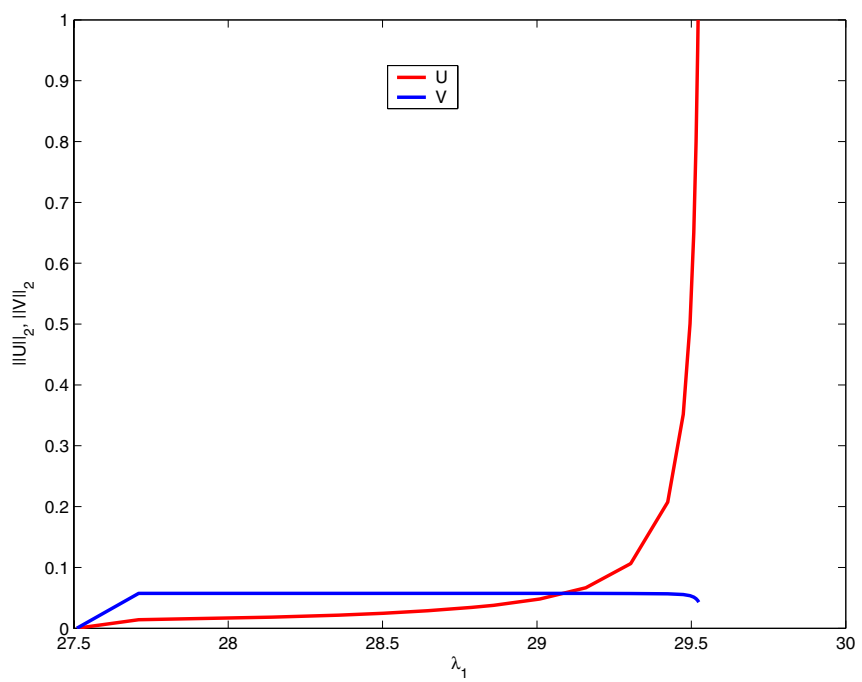


FIG. 10. *The solution curves of* (u, v) *in* (1.2) *with* $M = 2$ *at* $\lambda_2 = 30.0$, $\mu_k = 0.1$, $k = 1, 2$, *and* $\beta = -300.0$.

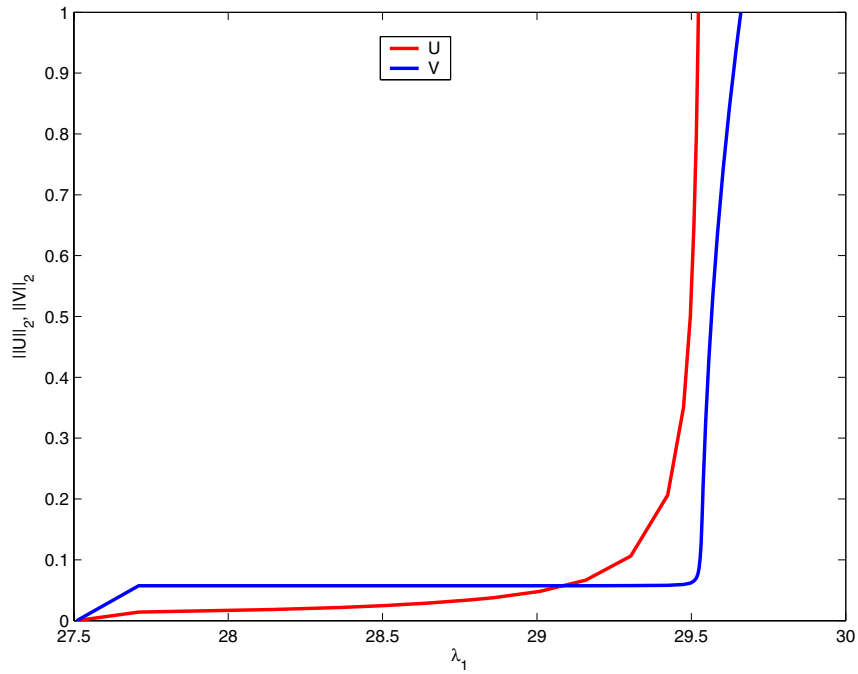


FIG. 11. *The solution curves of* (u, v) *in* (1.2) *for* $M = 2$ *at* $\lambda_2 = 30.0$, $\mu_k = 0.1$, $k = 1, 2$, *and* $\beta = 300.0$.

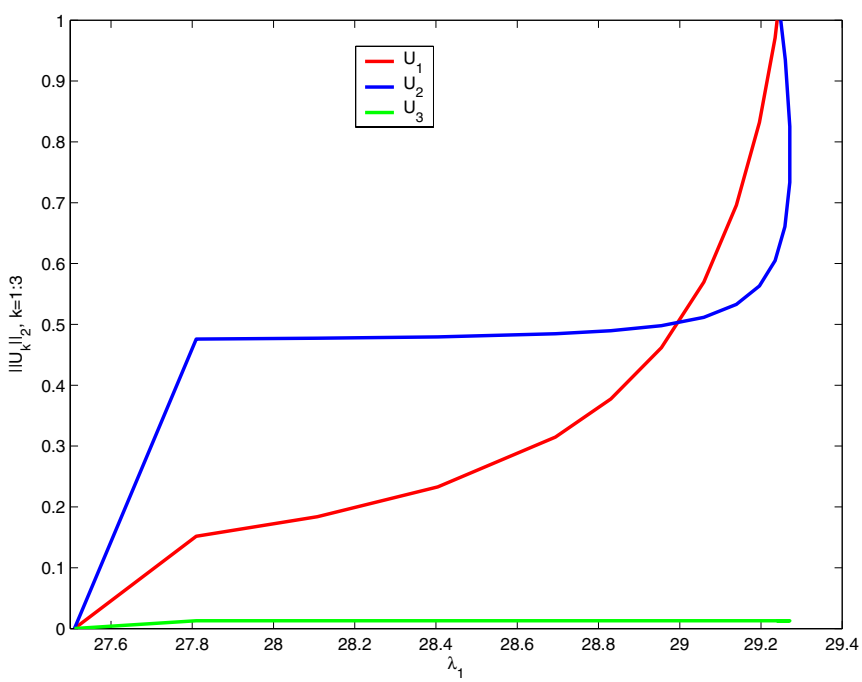


FIG. 12. *The solution curves of* (u_1, u_2, u_3) *in* (1.2) *for* $M = 3$ *at* $\lambda_2 = 29.0$, $\lambda_3 = 50.0$, $\mu_k = 0.1, k = 1 : 3$, *and* $\beta = -300.0$.

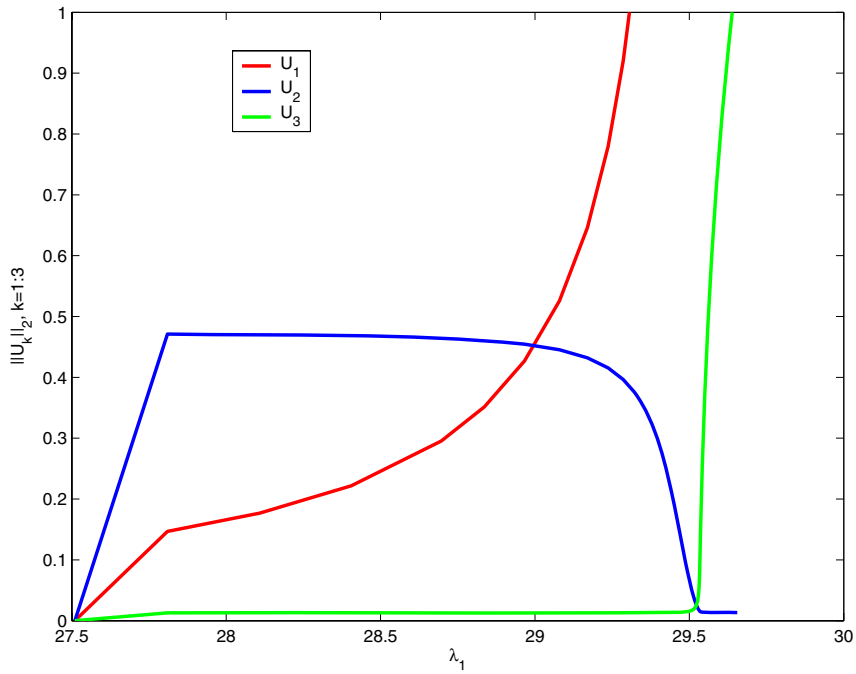


FIG. 13. *The solution curves of* (u_1, u_2, u_3) *in* (1.2) *for* $M = 3$ *at* $\lambda_2 = 29.0$, $\lambda_3 = 50.0$, $\mu_k = 0.1, k = 1:3$, *and* $\beta = 300.0$.

domain, and choosing nonisotropic potentials. A two-grid centered difference method is described for computing the extremum eigenpairs of the coupled linear Schrödinger eigenvalue problems. The normalized eigenvector can be used as the tangent vector at the bifurcation for branch-switching. In our numerical experiments, the effect of linear potentials, double bifurcations, strong and weak repulsive interactions, and the mass conservation constraints were considered. For $M = 2$ the numerical results show that both components of the condensates can reach the mass conservation requirements for different values of chemical potentials if $\beta > 0$, while only one component can reach the requirement if $\beta < 0$.

REFERENCES

[1] M. H. Anderson, J. R. Ensher, M. R. Matthews, C. E. Wieman, and E. A. Cornell, *Observation of Bose-Einstein condensation in a dilute atomic vapor*, Science, 269 (1995), pp. 198–201.

[2] J. R. Anglin and W. Ketterle, *Bose-Einstein condensation of atomic gases*, Nature, 416 (2002), pp. 211–218.

[3] C. C. Bradley, C. A. Sackett, and R. G. Hulet, *Bose-Einstein condensation of Lithium: Observation of limited condensate number*, Phys. Rev. Lett., 78 (1997), pp. 985–989.

[4] G. P. Agrawal, *Nonlinear Fiber Optics*, 2nd ed., Academic Press, New York, 1995.

[5] E. P. Gross, *Structure of a quantized vortex in boson systems*, Nuovo Cimento, 20 (1961), pp. 454–477.

[6] L. P. Pitaevskii, *Vortex lines in an imperfect Bose gas*, Soviet Phys. JETP, 13 (1961), pp. 451–454.

[7] F. T. Hioe, *Solitary waves for N coupled nonlinear Schrödinger equations*, Phys. Rev. Lett., 82 (1999), pp. 1152–1155.

[8] T. Kanna and M. Lakshmanan, *Exact soliton solutions, shape changing collisions, and partially coherent solitons in coupled nonlinear Schrödinger equations*, Phys. Rev. Lett., 86 (2001), pp. 5043–5046.

[9] A. Gammal, T. Frederico, and L. Tomio, *Improved numerical approach for the time-independent Gross-Pitaevskii nonlinear Schrödinger equation*, Phys. Rev. E, 60 (1999), pp. 2421–2424.

[10] M. Mitchell, Z. Chen, M. Shih, and M. Segev, *Self-trapping of partially spatially incoherent light*, Phys. Rev. Lett., 77 (1996), pp. 490–493.

[11] W. Bao and W. Tang, *Ground-state solution of Bose-Einstein condensate by directly minimizing the energy functional*, J. Comput. Phys., 187 (2003), pp. 230–254.

[12] W. Bao and Q. Du, *Computing the ground state solution of Bose–Einstein condensates by a normalized gradient flow*, SIAM J. Sci. Comput., 25 (2004), pp. 1674–1697.

[13] W. Bao, D. Jaksch, and P. A. Markowich, *Numerical solution of the Gross-Pitaevskii equation for Bose-Einstein condensation*, J. Comput. Phys., 187 (2003), pp. 318–342.

[14] Y.-C. Kuo, W.-W. Lin, and S.-F. Shieh, *Bifurcation analysis of a two-component Bose-Einstein condensate*, Phys. D, 21 (2005), pp. 311–346.

[15] T.-C. Lin and J. Wei, *Ground state of N coupled nonlinear Schrödinger equations in $\mathbf{R}^n$, $n \leq 3$*, Comm. Math. Phys., 255 (2005), pp. 629–653.

[16] D. S. Hall, M. R. Matthews, J. R. Ensher, C. E. Wieman, and E. A. Cornell, *Dynamics of component separation in a binary mixture of Bose-Einstein condensates*, Phys. Rev. Lett., 81 (1998), pp. 1539–1542.

[17] H. Pu and N. P. Bigelow, *Properties of two-species Bose condensates*, Phys. Rev. Lett., 80 (1998), pp. 1130–1133.

[18] W. Bao, *Ground states and dynamics of multicomponents Bose-Einstein condensates*, Multiscale Model. Simul., 2 (2004), pp. 210–236.

[19] P. K. Ghosh, *Exact results on the dynamics of a multicomponent Bose-Einstein condensate*, Rhys. Rev. A, 65 (2002), 053601(6).

[20] S.-L. Chang and C.-S. Chien, *Numerical continuation for nonlinear Schrödinger equations*, Inter. J. Bifurcation and Chaos, to appear.

[21] E. L. Allgower and K. Georg, *Introduction to Numerical Continuation Methods*, Classics in Appl. Math. 45, SIAM, Philadelphia, 2003.

[22] H. B. Keller, *Lectures on Numerical Methods in Bifurcation Problems*, Springer-Verlag, Berlin, 1987.

[23] M. Golubitsky and D. G. Schaeffer, *Singularities and Groups in Bifurcation Theory, Vol.* I, Appl. Math. Sci. 51, Springer-Verlag, 1985.

[24] K. Böhmer, *On a numerical Liapunov–Schmidt method for operator equations*, Computing, 51 (1993), pp. 237–269.

[25] K. Böhmer and Z. Mei, *On a numerical Liapunov–Schmidt method*, in Computational Solution of Nonlinear Systems of Equations, E. L. Allgower and K. Georg, eds., Lectures in Appl. Math. 26, AMS, Providence, RI, 1990, pp. 79–98.

[26] P. Ashwin, K. Böhmer, and Z. Mei, *A numerical Liapunov–Schmidt method with applications to Hopf bifurcation on a square*, Math. Comp., 64 (1995), pp. 649–670.

[27] K. Böhmer and Z. Mei, *Mode interactions of an elliptic system on the square*, Int. Ser. Numer. Math. 104, Birkhäuser Verlag, Basel, 1992, pp. 49–58.

[28] Z. Mei, *Path following around corank-2 bifurcation points of a semilinear elliptic problem with symmetry*, Computing, 47 (1991), pp. 69–85.

[29] L. Bauer, H. B. Keller, and E. L. Reiss, *Multiple eigenvalues lead to secondary bifurcation*, SIAM Rev., 17 (1975), pp. 101–122.

[30] C.-S. Chien, S.-Y. Gong, and Z. Mei, *Mode jumping in the von Kármán equations*, SIAM J. Sci. Comput., 22 (2000), pp. 1354–1385.

[31] E. Isaacson and H. B. Keller, *Analysis of Numerical Methods*, John Wiley, New York, 1965.

[32] S.-L. Chang, C.-S. Chien, and B.-W. Jeng, *An efficient algorithm for the Schrödinger-Poisson eigenvalue problem*, J. Comput. Appl. Math., to appear.

[33] E. L. Allgower, C.-S. Chien, and W.-C. Lee, *Bifurcations of corank greater than two*, Comput. Math. Appl., 25 (1993), pp. 51–64.

[34] M. Golubitsky, I. Stewart, and D. G. Schaeffer, *Singularities and Groups in Bifurcation Theory, Vol.* 2, Appl. Math. Sci., 69, Springer-Verlag, 1988.

[35] C.-S. Chien, Z. Mei, and C.-L. Shen, *Numerical continuation at double bifurcation points of a reaction-diffusion problem*, Int. J. Bifurcation and Chaos, 8 (1998), pp. 117–139.

[36] C.-S. Chien and S.-L. Chang, *Application of the Lanczos algorithm for solving the linear systems that occur in continuation problems*, Numer. Linear Algebra Appl., 10 (2003), pp. 335–355.

[37] W. J. F. Govaerts, *Numerical Methods for Bifurcations of Dynamical Equilibria*, SIAM, Philadelphia, 2000.

[38] S. H. Lui and G. H. Golub, *Homotopy method for the numerical solution of the eigenvalue problem of self-adjoint partial differential operators*, Numer. Algorithms, 10 (1995), pp. 363–378.

[39] B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice–Hall, Englewood Cliffs, NJ, 1980.

[40] C.-S. Chien and B.-W. Jeng, *A two-grid discretization scheme for semilinear elliptic eigenvalue problems*, SIAM J. Sci. Comput., 27 (2006), pp. 1287–1304.

SIAM J. SCI. COMPUT.
Vol. 29, No. 2, pp. 756–781
© 2007 Society for Industrial and Applied Mathematics

ANISOTROPIC MEASURES OF THIRD ORDER DERIVATIVES AND THE QUADRATIC INTERPOLATION ERROR ON TRIANGULAR ELEMENTS*

WEIMING CAO†

Abstract. The main purpose of this paper is to present a closer look at how the H^1- and L^2-errors for quadratic interpolation on a triangle are determined by the triangle geometry and the anisotropic behavior of the third order derivatives of interpolated functions. We characterize quantitatively the anisotropic behavior of a third order derivative tensor by its orientation and anisotropic ratio. Both exact error formulas and numerical experiments are presented for model problems of interpolating a cubic function u at the vertices and the midpoints of three sides of a triangle. Based on the study on model problems, we conclude that when an element is aligned with the orientation of $\nabla^3 u$, the aspect ratio leading to nearly the smallest H^1- and L^2-norms of the interpolation error is approximately equal to the anisotropic ratio of $\nabla^3 u$. With this alignment and aspect ratio taken, the H^1-seminorm of the error is proportional to the reciprocal of the anisotropic ratio of $\nabla^3 u$, the L^2-norm of the error is proportional to the $-\frac{3}{2}$th power of the anisotropic ratio, and both of them are insensitive to the internal angles of the element.

Key words. anisotropic mesh, quadratic interpolation, anisotropic ratio, aspect ratio, mesh alignment

AMS subject classifications. 65D05, 65L50, 65N15, 65N50

DOI. 10.1137/050634700

1. Introduction.

In anisotropic mesh generation and refinement, elements or cells can be of different length scales in different directions in order to better fit the behavior of the interpolated functions or the properties of the PDEs to be solved. There are two anisotropic characters of a triangular element. One is the orientation, which is roughly the direction of its longest side. The other is the aspect ratio, which measures how long and thin the triangle is. For linear interpolation over triangular meshes, the error is determined by the Hessian $\nabla^2 u$ of the interpolated function u. There have been extensive studies on the optimal shape and orientation for minimizing the linear interpolation error in various norms. For instance, Nadler [16] studied the model problem of linear interpolation of quadratic functions and concluded that the optimal triangle for minimizing the L^2-error is an equilateral under the Hessian metric. Rippa [17] concluded that to minimize the L^p-error for linear interpolation, the triangular elements should be long and thin in the direction where the second order directional derivative of u is small. D'Azevedo and Simpson [11, 12] made similar conclusions for the maximum error and maximum derivative error in linear interpolation. They also developed several strategies for the anisotropic mesh generation and refinement based on $\nabla^2 u$ [11, 19]. Shewchuk [18] summarized various formulas for the linear interpolation error on a triangle in terms of its area and edge lengths. In [6] the author expresses the H^1- and L^2-errors for the linear interpolation of a quadratic function u on a triangle K in terms of the size, orientation, and aspect ratio of K, and

*Received by the editors June 28, 2005; accepted for publication (in revised form) October 25, 2006; published electronically April 10, 2007. This work was supported in part by the NSF through grant DMS-0209313.

http://www.siam.org/journals/sisc/29-2/63470.html

†Department of Mathematics, University of Texas at San Antonio, San Antonio, TX 78249 (wcao@math.utsa.edu).

the eigenvalues and eigenvectors of $\nabla^2 u$. It is identified that among all possible triangles of a fixed area, the smallest H^1-seminorm of the interpolation error is attained with the acute isosceles triangle that is aligned with the eigenvector (associated with the smaller eigenvalue) of $\nabla^2 u$, and has an aspect ratio $0.8|\lambda_2/\lambda_1|$, where $|\lambda_1| \leq |\lambda_2|$ are the eigenvalues of $\nabla^2 u$. For general triangular elements aligned with $\nabla^2 u$ without the restriction on the maximum internal angles, the best aspect ratio for the H^1-error is about $\sqrt{|\lambda_2/\lambda_1|}$. This is also the best aspect ratio for minimizing the L^2-error.

Meanwhile, there have been quite a few papers on generalizing the classical interpolation error estimates on quasi-uniform meshes to those on anisotropic meshes. For instance, Formaggia and Perotto [13], Huang [14], Huang and Sun [15], Chen, Sun, and Xu [8], and Chen and Xu [9] derived various estimates for the $W^{m,p}$-norm of the error for linear interpolation of general functions. They conclude similarly that the mesh leading to the smallest error bound is the one where all elements are aligned with the eigenvector of $\nabla^2 u$, and their aspect ratios are approximately equal to the square root of the ratio of the eigenvalues of $\nabla^2 u$. Chen, Sun, and Xu [8] further proved that their estimate of the L^p-error is optimal for the class of functions with positive Hessians.

In the case of quadratic interpolation, the understanding of the interplay between the geometric feature of elements and the anisotropic feature of interpolated functions has been mostly qualitative and crude [18]. As pointed by Apel in [1], a more precise understanding of their effects on the interpolation errors relies on "sufficiently fine description of the properties" of the interpolated functions. We introduced in [7] two quantities to measure the anisotropic characters of third order and higher derivative tensors. One is the orientation, along which the absolute value of the higher order directional derivative is about the smallest, while along its perpendicular direction it is about the largest. Another is the anisotropic ratio, which measures the strength of the anisotropic behavior, or how differently the directional derivative behaves in different directions. These measures are invariant under rotation and translation of the coordinates. We also derived an interpolation error estimate in terms of these anisotropic measures and identified the optimal alignment direction and aspect ratio which minimize an upper bounds of the error estimate; see [7] for details.

In this paper, we take a closer look at how the H^1- and L^2-errors for quadratic interpolation on a triangle are determined by the triangle geometry and the anisotropic features of $\nabla^3 u$. Since any smooth function can be approximated locally by a cubic function to a higher order accuracy than by a quadratic function, we simplify our local error analysis over an element by assuming the interpolated functions u are cubics. Unfortunately, unlike the case of linear interpolation studied in [6], a closed formula for the quadratic interpolation error in terms of the magnitude, orientation, and anisotropic ratio of $\nabla^3 u$ turns out to be too complicated. Nevertheless, for isosceles triangles that are aligned with $\nabla^3 u$, we are able to get some handy formulas for the H^1- and L^2-norms of the error. It is found from these formulas that when a triangle is aligned with the orientation of $\nabla^3 u$, the optimal aspect ratio for minimizing both H^1- and L^2-norms of the quadratic interpolation error is approximately equal to the anisotropic ratio of $\nabla^3 u$. This is true regardless of whether the element is an acute or an obtuse isosceles triangle. Therefore, in contrast to the case of linear interpolations, there seems to be no "superaccuracy" in favor of acute isosceles triangles for quadratic interpolations of general cubic functions; see [18]. For general triangle shape and orientations, we study numerically the effects of element alignment, aspect ratio, and internal angles on the H^1- and L^2-norms of the interpolation errors. It is found that

(i) the interpolation error is nearly the smallest when an element is aligned with the orientation of $\nabla^3 u$; (ii) when an element is aligned with $\nabla^3 u$, the optimal aspect ratio is about equal to the anisotropic ratio of $\nabla^3 u$. With this aspect ratio, the H^1-seminorm of the interpolation error is proportional to the magnitude of $\nabla^3 u$ and to the reciprocal of the anisotropic ratio of $\nabla^3 u$. The L^2-norm of the error is proportional to the magnitude of $\nabla^3 u$ and to the $-\frac{3}{2}$th power of the anisotropic ratio of $\nabla^3 u$. (iii) When proper alignment and proper aspect ratio are maintained, the H^1-error is insensitive to the internal angles of the element. Otherwise, it increases as the largest internal angle of the element increases. In this situation the maximum angle condition is essential. The L^2-error is always insensitive to the internal angles.

An outline of this paper is as follows: We first explain in sections 2 and 3 how to measure the orientation and aspect ratio of a triangle, and how to define the orientation and anisotropic ratio of $\nabla^3 u$. Then we derive in section 4 the exact formulas for the H^1- and L^2-norms of the error for quadratic interpolation of cubic functions on a triangle. In section 5 we study the special case with isosceles triangles that are aligned with the orientation of $\nabla^3 u$. In this case the H^1- and L^2-errors can be expressed in simple formulas, and optimal aspect ratios can be identified. For general triangular elements we present a numerical study in section 6 on the effects of the element shape and orientation. We conclude this paper with a few discussions.

2. Anisotropic character of a triangular element. We first recall the definitions of orientation and aspect ratio for a triangular element described in [6]. Let $\hat{K}$ be the standard element with vertices $\boldsymbol{\xi}_1 = (0,1), \boldsymbol{\xi}_2 = (-\frac{\sqrt{3}}{2}, -\frac{1}{2}), \boldsymbol{\xi}_3 = (\frac{\sqrt{3}}{2}, -\frac{1}{2})$. Let K be a general triangular element with vertices $\boldsymbol{x}_1, \boldsymbol{x}_2$, and $\boldsymbol{x}_3$. Let $\boldsymbol{x}_c = \frac{1}{3}(\boldsymbol{x}_1 + \boldsymbol{x}_2 + \boldsymbol{x}_3)$ be the center of K. The affine mapping from $\hat{K}$ to K can be expressed as $\boldsymbol{x} = F\boldsymbol{\xi} + \boldsymbol{x}_c$ with

$$F = \left(\frac{1}{\sqrt{3}}(\boldsymbol{x}_1 - \boldsymbol{x}_3), \boldsymbol{x}_2 - \boldsymbol{x}_c \right).$$

Denote by $F = U\Sigma V^*$ the singular value decomposition (SVD) of matrix F. Without loss of generality, we assume $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2)$ with $\sigma_1 \geq \sigma_2 > 0$, and $U = R_{\phi_u}$ and $V = R_{\phi_v}$ are the matrices for rotation transform by angles ϕ_u and ϕ_v (counterclockwise), respectively.

The mapping $\boldsymbol{x} = F\boldsymbol{\xi}$ transforms $\hat{K}$ into a triangle centered at the origin. It also transforms the unit circle on the $\xi\eta$-plane into an ellipse on the xy-plane, with σ_1 and σ_2 being its semimajor and semiminor axes; see Figure 1. It is easy to see that ϕ_u determines the direction of the semimajor axis of the ellipse. We define this direction as the orientation of triangle K. We define the aspect ratio of K as $r = \frac{\sigma_1}{\sigma_2}$. These definitions are equivalent to the commonly used definition of orientation based on the longest side of K and the definition of aspect ratio based on the diameter of K and the radius of the inscribed circle of K. The angle ϕ_v in the SVD of mapping F determines the internal angles of K. In particular, K is an obtuse isosceles triangle when $\phi_v = 0$ and an acute isosceles triangle when $\phi_v = \pi/2$. If K is an obtuse/acute isosceles triangle, then its orientation is parallel/perpendicular to its base.

3. Anisotropic measures of third order derivatives. By Taylor's expansion, any smooth function u can be approximated locally by lower order polynomials,

$$u(x,y) = \sum_{j=0}^{m} \frac{1}{j!}[(\boldsymbol{x} - \boldsymbol{x}_c) \cdot \nabla]^j u(\boldsymbol{x}_c) + O(\|\boldsymbol{x} - \boldsymbol{x}_c\|^{m+1}).$$

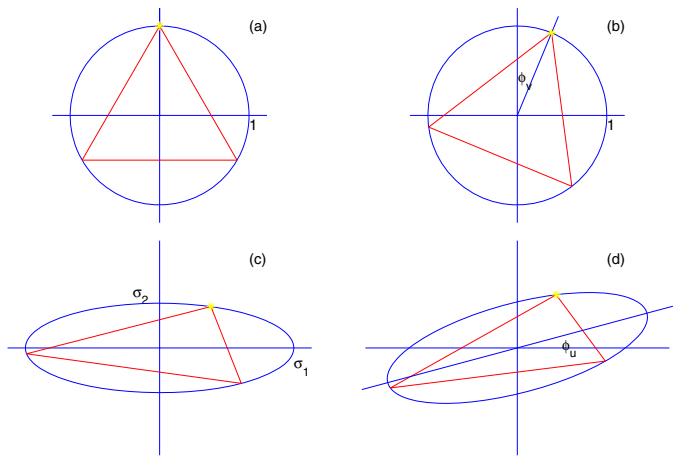


FIG. 1. (a) *standard element* $\hat{K}$; (b) $\hat{K}$ *under* $R^*_{\phi_v}$; (c) $\hat{K}$ *under* $\Sigma R^*_{\phi_v}$; (d) $\hat{K}$ *under* $F = R_{\phi_u}\Sigma R^*_{\phi_v}$.

Let Π_{m-1} be an interpolation operator that preserves the set of polynomials of total degree $\le m-1$. Then the leading term of the interpolation error $u - \Pi_{m-1}u$ is equal to the interpolation error for polynomial $\frac{1}{m!}((\boldsymbol{x}-\boldsymbol{x}_c)\cdot\nabla)^m u(\boldsymbol{x}_c)$, which is controlled by the partial derivative tensor $\nabla^m u(\boldsymbol{x}_c)$. To understand the local behavior of the interpolation error around $\boldsymbol{x}_c$, we need to measure the anisotropic behavior of $\nabla^m u$. Recall that there are three quantities, the area, orientation, and aspect ratio, to characterize the geometric features of a triangular element. We introduced in [7] three quantities to characterize the anisotropic behavior of $\nabla^m u$. The first one is its magnitude, which is equivalent to the Frobenius norm of $\nabla^m u$. The second is the orientation, along which direction the absolute value of the mth order directional derivative at $\boldsymbol{x}_c$ is about the smallest, while along its perpendicular direction it is about the largest. The third quantity is the anisotropic ratio, which measures the strength of the anisotropic behavior of $\nabla^m u$ at $\boldsymbol{x}_c$. These quantities are determined by the distribution of the mth order directional derivatives. They are invariant under the rotation and translation transforms of the independent variables of u. We describe below in detail these definitions in the cases of $m = 2, 3$.

3.1. Anisotropic measure of $\nabla^2 u$. We start from the case of $\nabla^2 u$, which will be needed for introducing the anisotropic measures of $\nabla^3 u$. Note that $\nabla^2 u(\boldsymbol{x}_c)$ is a 2×2 symmetric matrix; it admits the eigendecomposition

$$\nabla^2 u = R_\phi \begin{bmatrix} \lambda_1 & \\ & \lambda_2 \end{bmatrix} R_\phi^T, \tag{3.1}$$

where $|\lambda_1| \le |\lambda_2|$, and R_ϕ is the matrix for rotation with angle $\phi \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ counterclockwise. The two eigenvectors of $\nabla^2 u$ are $\boldsymbol{v}_1 = [\cos\phi, \sin\phi]^T$ and $\boldsymbol{v}_2 = \boldsymbol{v}_1^\perp$, which are in the directions of angle ϕ and $\phi + \frac{\pi}{2}$ from the x-axis. When $|\lambda_1/\lambda_2|$ is small, the absolute value of the second order directional derivative is about the smallest along $\boldsymbol{v}_1$, and it is the largest along its perpendicular direction. We define $\boldsymbol{v}_1$ as the orientation of $\nabla^2 u$ at $\boldsymbol{x}_c$. Furthermore, we define the anisotropic ratio of $\nabla^2 u$ as

$$S(\nabla^2 u) \equiv S_2 \equiv \sqrt{\left|\frac{\lambda_2}{\lambda_1}\right|}$$

and define the magnitude of $\nabla^2 u$ as

$$|\nabla^2 u| \equiv D_2 \equiv \frac{1}{2}(|\lambda_1| + |\lambda_2|).$$

Clearly, $S_2 \geq 1$. The larger S_2 is, the stronger the anisotropic behavior of $\nabla^2 u$. Since $\frac{1}{2}|\lambda_2| \leq D_2 \leq |\lambda_2|$, the magnitude $|\nabla^2 u|$ is equivalent to the 2-norm, and hence the Frobenius norm, of $\nabla^2 u$.

The above definition of the orientation, anisotropic ratio, and magnitude for $\nabla^2 u$ can be interpreted geometrically in terms of the level curves of the following polynomial:

$$p_2(\xi, \eta) = \frac{1}{2}(\boldsymbol{\xi} \cdot \nabla)^2 u(\boldsymbol{x}_c),$$

where $\boldsymbol{\xi} = [\xi, \eta]^T$. For $\|\boldsymbol{\xi}\| = 1$, $p_2(\xi, \eta)$ is one half of the second order directional derivative along $\boldsymbol{\xi}$ for u at $\boldsymbol{x}_c$. We first consider the case $\lambda_1 \lambda_2 < 0$ (assume, more specifically, $\lambda_1 \geq 0$ and $\lambda_2 \leq 0$.) Let (r, t) be the polar coordinates of (ξ, η). Then by the eigendecomposition (3.1) for $\nabla^2 u$,

$$\begin{aligned} p_2(\xi, \eta) &= \tfrac{1}{2}(R_\phi^T \boldsymbol{\xi}) \cdot \begin{bmatrix} \lambda_1 & \\ & \lambda_2 \end{bmatrix} (R_\phi^T \boldsymbol{\xi}) \\ &= \tfrac{1}{2} r^2 [\lambda_1 \cos^2(t - \phi) + \lambda_2 \sin^2(t - \phi)]. \end{aligned} \tag{3.2}$$

Let $\alpha = \tan^{-1}(\sqrt{|\frac{\lambda_1}{\lambda_2}|})$. Then $\alpha \in [0, \frac{\pi}{4}]$ for $|\lambda_1| \leq |\lambda_2|$, and

$$\sin^2 \alpha = \frac{\lambda_1}{|\lambda_1| + |\lambda_2|}, \qquad \cos^2 \alpha = -\frac{\lambda_2}{|\lambda_1| + |\lambda_2|}.$$

Therefore,

$$\begin{aligned} p_2(\xi, \eta) &= \tfrac{1}{2} r^2 (|\lambda_1| + |\lambda_2|)\ [\sin^2 \alpha\ \cos^2(t - \phi) - \cos^2 \alpha\ \sin^2(t - \phi)] \\ &= \tfrac{1}{2} r^2 (|\lambda_1| + |\lambda_2|)\ \sin(\alpha + t - \phi) \cdot \sin(\alpha - t + \phi). \end{aligned}$$

Clearly, $p_2(\xi, \eta) = 0$ along the lines of angles $\phi + \alpha + k\pi$ and $\phi - \alpha + k\pi$ from the ξ-axis. Let $\beta_1 \leq \beta_2$ be the two angles among them that lie in $(-\pi/2, \pi/2]$. Then we can express p_2 as

$$\begin{aligned} p_2(\xi, \eta) &= \pm \tfrac{1}{2} r^2 (|\lambda_1| + |\lambda_2|)\ \sin(t - \beta_1)\ \sin(t - \beta_2) \\ &= \pm \tfrac{1}{2} r^2 (|\lambda_1| + |\lambda_2|)\ (\cos t\ \sin \beta_1 - \sin t\ \cos \beta_1) \cdot (\cos t\ \sin \beta_2 - \sin t\ \cos \beta_2) \\ &= \pm D_2\ \ell_1(\xi, \eta)\ \ell_2(\xi, \eta), \end{aligned} \tag{3.3}$$

where $D_2 = \frac{1}{2}(|\lambda_1| + |\lambda_2|)$, and

$$\ell_i(\xi, \eta) = \xi \sin \beta_i - \eta \cos \beta_i, \qquad i = 1, 2. \tag{3.4}$$

We are ready to see that the orientation of $\nabla^2 u$ is just the bisection direction of the smaller sector formed by the level-0 lines of $p_2(\xi, \eta)$; see Figure 2(a). The angle ϕ in (3.1) for this direction can be expressed as

$$\phi = \frac{1}{2}(\underline{\beta} + \bar{\beta}),$$

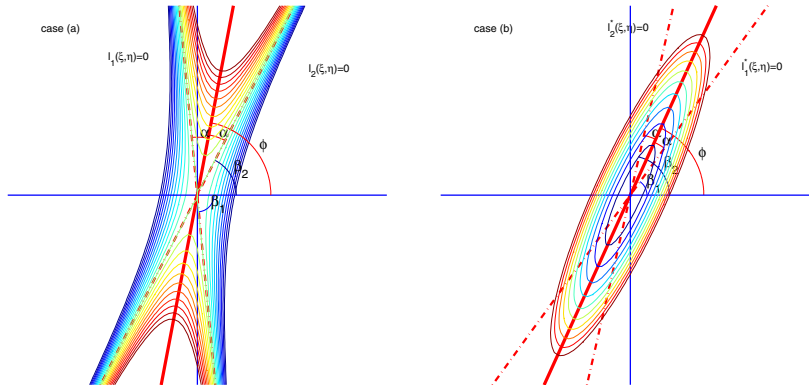


FIG. 2. *Contour lines of $p_2(\xi,\eta)$ and the orientation (bold line) of $\nabla^2 u$.* (a) *$p_2(\xi,\eta)$ has two linear factors. Here $\underline{\beta}=\beta_2$ and $\bar{\beta}=\beta_1+\pi$.* (b) *$p_2(\xi,\eta)$ does not change sign for all (ξ,η). ℓ_1^* and ℓ_2^* are the linear factors of the associated polynomial p_2^*. Here $\underline{\beta}=\beta_1$ and $\bar{\beta}=\beta_2$.*

where $[\underline{\beta},\bar{\beta}]$ is the smaller interval between $[\beta_1,\beta_2]$ and $[\beta_2,\beta_1+\pi]$. Furthermore, the anisotropic ratio $S(\nabla^2 u)=\sqrt{|\frac{\lambda_2}{\lambda_1}|}$ can be expressed in terms of the opening $\bar{\beta}-\underline{\beta}=2\alpha$ of the smaller sector as

$$S(\nabla^2 u)=\cot\alpha=\cot((\bar{\beta}-\underline{\beta})/2).$$

The magnitude $|\nabla^2 u|$ of $\nabla^2 u$ is just the coefficient D_2 in the factoring (3.3) of p_2.

Algebraically, $\lambda_1\lambda_2\le 0$ corresponds to the case where $p_2(1,z)$ has two real roots $z_i=\tan(\beta_i)$. Now we consider the case $\lambda_1\lambda_2>0$. In this case, $p_2(1,z)$ has two conjugate complex roots, and $p_2(\xi,\eta)$ does not change sign for all (ξ,η). In this case, we introduce matrix

$$\hat{H}=R_\phi\begin{bmatrix}\lambda_1 & \\ & -\lambda_2\end{bmatrix}R_\phi^T \tag{3.5}$$

and define an associated polynomial $p_2^*(\xi,\eta)=\frac{1}{2}(\boldsymbol{\xi}\cdot\hat{H}\boldsymbol{\xi})$; see Figure 2(b). Clearly, p_2^* can be factored into the product of two linear functions ℓ_1^* and ℓ_2^* as in (3.4). Since the magnitude, orientation, and anisotropic ratio of $\nabla^2 u$ are identical to those of $\hat{H}$, we can determine them equivalently by the level-0 lines for p_2^* as in the case with $\lambda_1\lambda_2<0$.

In summary, if $p_2(\xi,\eta)$ can be factored into the product of two linear functions, then the orientation and anisotropic ratio of $\nabla^2 u$ can be determined by the level-0 lines of $p_2(\xi,\eta)$. Otherwise, they can be determined equivalently by the level-0 lines of $p_2^*(\xi,\eta)$ defined by $\hat{H}$ in (3.5). In particular, the level-0 lines of p_2 or p_2^* are of angles $\pm\alpha=\pm\tan^{-1}(\sqrt{|\lambda_1/\lambda_2|})$ from the eigenvector $\boldsymbol{v}_1$ of $\nabla^2 u$.

3.2. Anisotropic measure of $\nabla^3 u$. To characterize the anisotropic behavior of $\nabla^3 u$ at a point $\boldsymbol{x}_c$, we study

$$p_3(\boldsymbol{\xi})=\frac{1}{3!}(\boldsymbol{\xi}\cdot\nabla)^3 u(\boldsymbol{x}_c).$$

For $\|\boldsymbol{\xi}\|=1$, $p_3(\boldsymbol{\xi})$ is one sixth of the third order directional derivative along $\boldsymbol{\xi}$ for u at $\boldsymbol{x}_c$. Since $p_3(\boldsymbol{\xi})$ is an homogeneous polynomial, we may write $p_3(\xi,\eta)=\xi^3 p_3(1,\eta/\xi)$,

with $p_3(1,z)$ being a real polynomial of degree ≤ 3. $p_3(1,z)$ can be factored into the product of linear polynomials and/or a positive quadratic polynomial of z. Therefore, we may factor $p_3(\xi,\eta)$ into either the product of three linear functions or the product of a linear function and a positive, homogeneous, quadratic function of ξ,η. We discuss the former case first. Assume

$$p_3(\xi,\eta) = D_3\ \ell_1(\xi,\eta)\ \ell_2(\xi,\eta)\ \ell_3(\xi,\eta), \tag{3.6}$$

where $D_3 \geq 0$, and each ℓ_i is a linear function of the form (3.4) with

$$-\frac{\pi}{2} < \beta_1 \leq \beta_2 \leq \beta_3 \leq \frac{\pi}{2}.$$

These factors can be easily determined by the roots of $p_3(1,z)=0$. Indeed, $\tan\beta_i$'s are just the roots of $p_3(1,z)=0$. Clearly, $\ell_i(\xi,\eta)=0$, $i=1,2,3$, are the level-0 lines of p_3. We define the magnitude of $\nabla^3 u$ as $|\nabla^3 u| = D_3$. This is an upper bound of all the third order directional derivatives at $\boldsymbol{x}_c$. It is not difficult to show that $|\nabla^3 u|$ is equivalent to the Frobenius norm of $\nabla^3 u$ [7]. Moreover, when all three level-0 lines of p_3 are close to a common direction, the absolute value of the third order directional derivative of u is about the smallest along this direction, while it is about the largest along its perpendicular direction. In this case, $\nabla^3 u$ is strongly anisotropic. To determine the common direction to which the level-0 lines are all close, we consider the three sectors formed by every other line of $\ell_i = 0$ for $i=1,2,3$; see Figure 3(a). The ranges of the central angle of these sectors are, respectively,

$$[\beta_1,\beta_3], \quad [\beta_2,\beta_1+\pi], \quad [\beta_3,\beta_2+\pi]. \tag{3.7}$$

Let $[\underline{\beta},\overline{\beta}]$ be the shortest interval among the three of (3.7). Clearly, $\overline{\beta}-\underline{\beta} \leq \frac{2\pi}{3}$. If $\overline{\beta}-\underline{\beta} \ll 1$, all the level-0 lines of $p_3(\xi,\eta)$ are close to the direction of angle $\phi = \frac{1}{2}(\underline{\beta}+\overline{\beta})$ from the ξ-axis. We define this direction as the orientation of $\nabla^3 u$. Furthermore, the smaller $\overline{\beta}-\underline{\beta}$ is, the stronger the anisotropic behavior of $\nabla^3 u$. Hence, we define the anisotropic ratio of $\nabla^3 u$ as

$$S_3 \equiv S(\nabla^3 u) \equiv \max(1, \cot((\bar{\beta}-\underline{\beta})/2)).$$

The artificial cutoff of $S(\nabla^3 u)$ at the value 1 reflects the fact that we consider only stretching an element in two perpendicular directions in anisotropic mesh refinements. When $\overline{\beta}-\underline{\beta} \geq \frac{\pi}{2}$, rescaling an element in two perpendicular directions (while keeping the area fixed) has little advantage in reducing the H^1-norm of the interpolation error; see Remark 2 in section 5 below. Therefore, we simply consider in this case that $\nabla^3 u$ is isotropic and its anisotropic ratio $S(\nabla^3 u)=1$.

Next we study the case where $p_3(\xi,\eta)$ is the product of a linear function and a positive, homogeneous, quadratic function. Assume

$$p_3(\xi,\eta) = \ell_3(\xi,\eta)\ q(\xi,\eta),$$

where $\ell_3(\xi,\eta)$ is in the form of (3.4), and $q(\xi,\eta) > 0$ for all $\|\boldsymbol{\xi}\| \neq \mathbf{0}$. We write $q = \frac{1}{2}\boldsymbol{\xi}^T Q \boldsymbol{\xi}$, where Q is the Hessian of q. Decompose Q into

$$Q = R \begin{bmatrix} \lambda_1 & \\ & \lambda_2 \end{bmatrix} R^T, \tag{3.8}$$

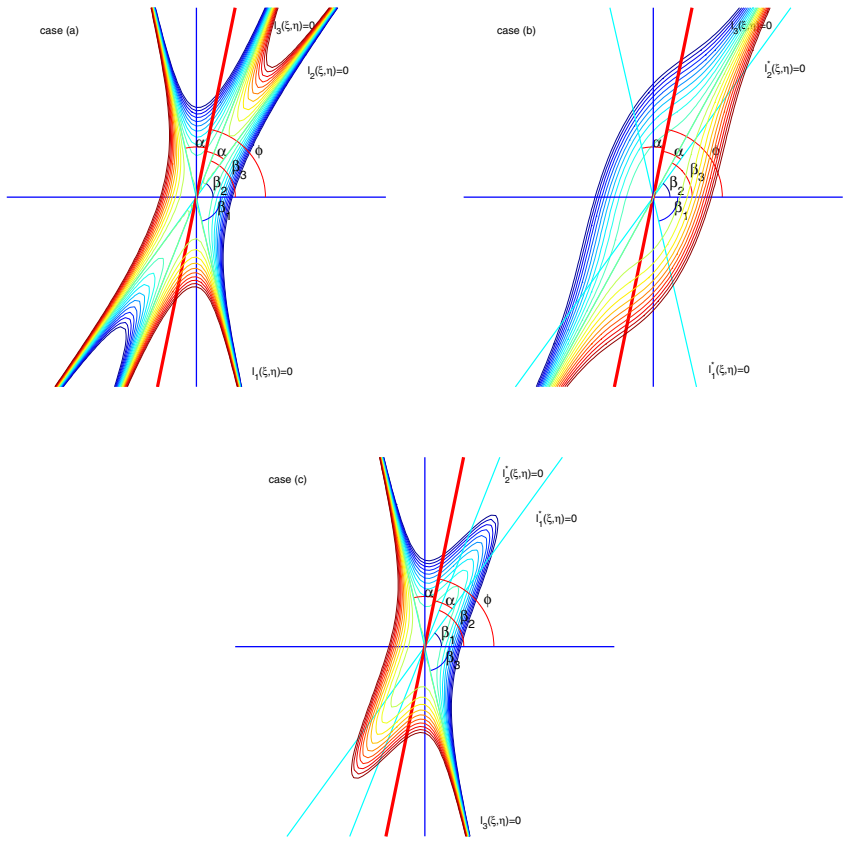


FIG. 3. *Contour lines of* $p_3 = \frac{1}{3!}(\boldsymbol{\xi} \cdot \nabla)^3 u$ *and the orientation (bold line) of* $\nabla^3 u$. (a) p_3 *has three linear (real) factors.* (b) p_3 *has a linear factor* $\ell_3(\xi, \eta)$ *and a positive quadratic factor* $q(\xi, \eta)$. $\ell_3 = 0$ *lies "inside" the level-*0 *lines of* $q^*(\xi, \eta)$. (c) p_3 *has a linear factor* $\ell_3(\xi, \eta)$ *and a positive quadratic factor* $q(\xi, \eta)$. $\ell_3 = 0$ *lies "outside" the level-*0 *lines of* $q^*(\xi, \eta)$.

where R is an orthogonal matrix, and $0 < \lambda_1 \leq \lambda_2$. As explained in the previous subsection, the anisotropic feature of $q(\xi, \eta)$ is identical to that of $q^*(\xi, \eta) = \frac{1}{2}\boldsymbol{\xi} \cdot \hat{Q}\boldsymbol{\xi}$, where

$$\hat{Q} = R \begin{bmatrix} \lambda_1 & \\ & -\lambda_2 \end{bmatrix} R^T. \tag{3.9}$$

Therefore, we define the magnitude, orientation, and anisotropic ratio of $\nabla^3 u$ based on

$$p_3^*(\xi, \eta) = \ell_3(\xi, \eta)\ q^*(\xi, \eta); \tag{3.10}$$

see Figure 3(b) and 3(c). Since $p_3^*(\xi, \eta)$ can be factored into the product of three linear functions ℓ_1^*, ℓ_2^*, and ℓ_3, these anisotropic measures can be determined by the position of the three level-0 lines of p_3^*.

An important feature about the above definitions of magnitude, anisotropic ratio, and orientation for $\nabla^3 u$ is that they are invariant under translation and rotation transforms of the independent variables of u.

1. The above definitions of the anisotropic measures can be generalized to the case of $\nabla^m u$ for $m > 3$; see [7]. Indeed, since $p_m(\xi, \eta) = \frac{1}{m!}(\boldsymbol{\xi} \cdot \nabla)^m u$ is

an homogeneous polynomial, we may write $p_m(\xi,\eta) = \xi^m p_m(1,\eta/\xi)$, with $p_m(1,z)$ being a real polynomial of degree $\le m$. By the fundamental theorem of algebra, $p_m(1,z)$ can be factored into the product of linear functions and/or positive quadratic functions of z. Therefore, $p_m(\xi,\eta)$ can be factored into the product of linear functions and/or positive, homogeneous, quadratic functions of ξ,η. We may convert each positive quadratic function $q(\xi,\eta)$ into $q^*(\xi,\eta)$. Then the magnitude, orientation, and anisotropic ratio of $\nabla^m u$ can be determined by the positions of the level-0 lines of $p_m^*(\xi,\eta)$.

3.3. Examples for anisotropic measures. We give several examples to demonstrate how to calculate the orientation and anisotropic ratio of $\nabla^3 u$. Assume ϵ is a small positive number.

1. $u(x,y) = y^3 + (\epsilon x)^3$. Here

$$p_3(\xi,\eta) = \frac{1}{3!}(\boldsymbol{\xi}\cdot\nabla)^3 u = \eta^3 + (\epsilon\xi)^3 = (\eta+\epsilon\xi)(\eta^2 - \epsilon\xi\eta + \epsilon^2\xi^2),$$

which is the product of a linear function and a positive quadratic function. Let $q(\xi,\eta) = \eta^2 - \epsilon\xi\eta + \epsilon^2\xi^2$. Then

$$\nabla^2 q = \begin{bmatrix} 2\epsilon^2 & -\epsilon \\ -\epsilon & 2 \end{bmatrix}.$$

The eigenvalues of $\nabla^2 q$ are

$$\begin{aligned} \lambda_1 &= (\epsilon^2+1) - \sqrt{\epsilon^4 - \epsilon^2 + 1} \quad \approx (\epsilon^2+1) - (1-\epsilon^2/2) \quad = 3\epsilon^2/2, \\ \lambda_2 &= (\epsilon^2+1) + \sqrt{\epsilon^4 - \epsilon^2 + 1} \quad \approx (\epsilon^2+1) + (1-\epsilon^2/2) \quad = 2+\epsilon^2/2. \end{aligned}$$

The eigenvector of $\nabla^2 q$, associated with the smaller eigenvalue λ_1, is $\boldsymbol{v}_1 = [1, (\epsilon^2 - 1 + \sqrt{\epsilon^4-\epsilon^2+1})/\epsilon]^T \approx [1, \epsilon/2]^T$. Note that the level-0 lines of $q^*(\xi,\eta)$ are along the angles $\phi \pm \alpha$ from the x-axis, where $\phi \approx \tan^{-1}(\frac{\epsilon}{2})$ is the angle of $\boldsymbol{v}_1$ and $\alpha = \tan^{-1}\sqrt{|\frac{\lambda_1}{\lambda_2}|}$. Thus in this case the two level-0 lines of $q^*(\xi,\eta)$ are along the angles $\tan^{-1}(\frac{\epsilon}{2}) \pm \tan^{-1}(\sqrt{\frac{3\epsilon^2}{4+\epsilon^2}}) \approx \frac{\epsilon}{2} \pm \frac{\sqrt{3}}{2}\epsilon$. Combining the direction of linear factor $y + \epsilon x = 0$, we have the three level-0 lines of $p_3^*(\xi,\eta)$ in approximately the directions of angle $-\epsilon$, $\frac{1-\sqrt{3}}{2}\epsilon$ and $\frac{1+\sqrt{3}}{2}\epsilon$, which implies $\underline{\beta} \approx -\epsilon$ and $\bar{\beta} \approx -\frac{1+\sqrt{3}}{2}\epsilon$. Therefore, the orientation of $\nabla^3 u$ is

$$\phi = \frac{1}{2}(\underline{\beta} + \bar{\beta}) \approx \frac{-1+\sqrt{3}}{4}\epsilon,$$

and the anisotropic ratio is

$$S(\nabla^3 u) = \cot(\tfrac{1}{2}(\bar{\beta} - \underline{\beta})) \approx \tfrac{4}{3+\sqrt{3}}\tfrac{1}{\epsilon}.$$

2. $u(x,y) = (x+y)^3 + \epsilon(x^3+y^3)$. Here

$$\begin{aligned} p_3(\xi,\eta) &= (\xi+\eta)^3 + \epsilon(\xi^3+\eta^3) \\ &= (\xi+\eta)\cdot[(1+\epsilon)\xi^2 + (2-\epsilon)\xi\eta + (1+\epsilon)\eta^2], \end{aligned}$$

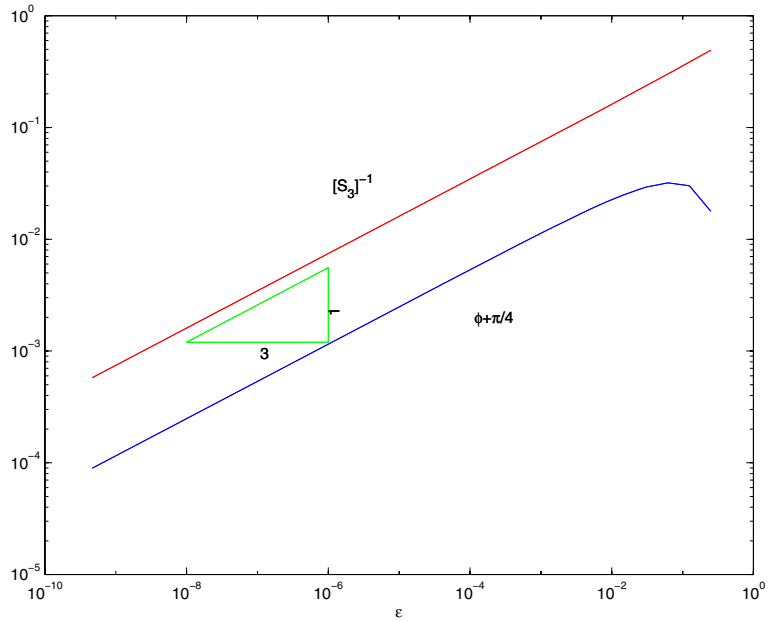


FIG. 4. *Example* 3*: The anisotropic ratio* S_3 *and orientation* $\phi + \frac{\pi}{4}$ *versus parameter* ϵ.

which is the product of a linear function and a positive quadratic function. Let $q(\xi, \eta) = (1+\epsilon)\xi^2 + (2-\epsilon)\xi\eta + (1+\epsilon)\eta^2$. Then

$$\nabla^2 q = \begin{bmatrix} 2+2\epsilon & 2-\epsilon \\ 2-\epsilon & 2+2\epsilon \end{bmatrix}.$$

Its two eigenvalues are 3ϵ and $4+\epsilon$, and the eigenvector associated with the smaller eigenvalue is $[1,-1]$. Therefore, the two level-0 lines of $q^*(\xi,\eta)$ are along angles $\tan^{-1}(-1) \pm \tan^{-1}\sqrt{\frac{3\epsilon}{4+\epsilon}}$. Together with the level-0 line $x+y=0$ due to the linear factor, we are ready to see that the orientation of $\nabla^3 u$ is along angle $\phi = \tan^{-1}(-1) = -\frac{\pi}{4}$, and the anisotropic ratio is $S(\nabla^3 u) = \cot\left[\tan^{-1}(\sqrt{\frac{3\epsilon}{4+\epsilon}})\right] \approx \frac{2}{\sqrt{3\epsilon}}$.

3. $u(x,y) = (x+y)^3 + \epsilon(x^3 - y^3)$. In this case, $p_3(\xi,\eta) = (\xi+\eta)^3 + \epsilon(\xi^3 - \eta^3)$, which is again the product of a linear function and a positive quadratic function. However, the asymptotic analysis for the level-0 lines of $p^*(\xi,\eta)$ is rather complicated. Instead we use a numerical method to determine the angles of the orientation and the anisotropic ratio; see Figure 4. It turns out that the orientation is still close to the line $y+x=0$, but the anisotropic ratio is of the magnitude $O(\sqrt[3]{\frac{1}{\epsilon}})$.

4. Quadratic interpolation error. In this section, we study the model problem of interpolating a cubic function by quadratic polynomials at the three vertices and the midpoints of three sides of a triangular element K. We derive the exact formulas for the H^1-seminorm and L^2-norm of the interpolation error. Let $\boldsymbol{x}_1, \boldsymbol{x}_2$, and $\boldsymbol{x}_3$ be the vertices of K (numbered counterclockwise), and let $\boldsymbol{\ell}_1 = \boldsymbol{x}_3 - \boldsymbol{x}_2, \boldsymbol{\ell}_2 = \boldsymbol{x}_1 - \boldsymbol{x}_3, \boldsymbol{\ell}_3 = \boldsymbol{x}_2 - \boldsymbol{x}_1$. Let $c_i(x,y)$, $i = 1,2,3$, be the barycentric coordinates of a point (x,y) in K. We call a triple index (i,j,k) in cyclic order if it is one of the

following three: $(1,2,3),(2,3,1),(3,1,2)$. For (i,j,k) in cyclic order, define

$$b_0 = c_1c_2c_3 \qquad \text{and} \qquad b_i = c_jc_k(c_j - c_k), \quad i = 1,2,3. \tag{4.1}$$

We are ready to see that b_i's are linearly independent. Let u be a cubic function, and let $\Pi_2 u$ be its quadratic interpolation at the vertices and side midpoints of K. Then the error $e = u - \Pi_2 u$ can be expressed as

$$e = \sum_{i=0}^{3} v_i b_i(x,y). \tag{4.2}$$

To determine the constants v_i, $0 \le i \le 3$, we first list a few basic formulas. Their proofs are elementary.

LEMMA 4.1. [illegible] $\boldsymbol{\tau}_i = \boldsymbol{\ell}_i/|\boldsymbol{\ell}_i|$ [illegible] $\boldsymbol{n}_i$ [illegible] $\boldsymbol{\ell}_i$ [illegible]

$$\nabla c_i = -\frac{|\boldsymbol{\ell}_i|}{2|K|}\boldsymbol{n}_i, \tag{4.3}$$

$$\boldsymbol{\tau}_i \cdot \nabla c_j = \begin{cases} 0 & \text{[illegible] } i = j, \\ \dfrac{-1}{|\ell_i|} & \text{[illegible] } (i,j) = (1,2),(2,3),(3,1), \\ \dfrac{+1}{|\ell_i|} & \text{[illegible] } (i,j) = (2,1),(3,2),(1,3). \end{cases} \tag{4.4}$$

LEMMA 4.2.

$$\begin{cases} \dfrac{\partial^3 b_i}{\partial \boldsymbol{\tau}_i^3} = \dfrac{12}{|\boldsymbol{\ell}_i|^3} & i = 1,2,3, \\ \dfrac{\partial^3 b_i}{\partial \boldsymbol{\tau}_j^3} = 0 & \forall i \ne j. \end{cases} \tag{4.5}$$

Note that $\Pi_2 u$ is of degree 2, and we get from the above two lemmas that

$$v_i = \frac{|\boldsymbol{\ell}_i|^3}{12}\frac{\partial^3 u}{\partial \boldsymbol{\tau}_i^3}, \qquad i = 1,2,3. \tag{4.6}$$

Next we determine the coefficient v_0. It is natural to consider the symmetric differentiation $\frac{\partial^3}{\partial \boldsymbol{\tau}_1 \boldsymbol{\tau}_2 \boldsymbol{\tau}_3}$ on both sides of (4.2). However, since

$$\frac{\partial^3 b_0}{\partial \boldsymbol{\tau}_1\boldsymbol{\tau}_2\boldsymbol{\tau}_3} = 0 \qquad \text{and} \qquad \frac{\partial^3 b_i}{\partial \boldsymbol{\tau}_1\boldsymbol{\tau}_2\boldsymbol{\tau}_3} = \frac{4}{|\boldsymbol{\ell}_1| \cdot |\boldsymbol{\ell}_2| \cdot |\boldsymbol{\ell}_3|}, \qquad i = 1,2,3, \tag{4.7}$$

this operation leads only to an identity

$$\frac{\partial^3 u}{\partial \boldsymbol{\tau}_1\boldsymbol{\tau}_2\boldsymbol{\tau}_3} = \frac{1}{3|\boldsymbol{\ell}_1|\,|\boldsymbol{\ell}_2|\,|\boldsymbol{\ell}_3|}\sum_{i=1}^{3} |\ell_i|^3 \frac{\partial^3 u}{\partial \tau_i^3}. \tag{4.8}$$

Therefore, we consider taking the following operation on both sides of (4.2):

$$\mathcal{L} = \frac{\partial^3}{\partial \boldsymbol{\tau}_1^2 \boldsymbol{\tau}_2} + \frac{\partial^3}{\partial \boldsymbol{\tau}_2^2 \boldsymbol{\tau}_3} + \frac{\partial^3}{\partial \boldsymbol{\tau}_3^2 \boldsymbol{\tau}_1}. \tag{4.9}$$

LEMMA 4.3. [illegible] (i,j,k) [illegible]

$$\begin{cases} \dfrac{\partial^3 b_0}{\partial \boldsymbol{\tau}_i^2 \boldsymbol{\tau}_j} = \dfrac{-2}{|\boldsymbol{\ell}_i|^2 \cdot |\boldsymbol{\ell}_j|}, & \dfrac{\partial^3 b_0}{\partial \boldsymbol{\tau}_i^2 \boldsymbol{\tau}_k} = \dfrac{+2}{|\boldsymbol{\ell}_i|^2 \cdot |\boldsymbol{\ell}_k|}, \\ \dfrac{\partial^3 b_i}{\partial \boldsymbol{\tau}_i^2 \boldsymbol{\tau}_j} = \dfrac{-6}{|\boldsymbol{\ell}_i|^2 \cdot |\boldsymbol{\ell}_j|}, & \dfrac{\partial^3 b_i}{\partial \boldsymbol{\tau}_i \boldsymbol{\tau}_j^2} = \dfrac{+2}{|\boldsymbol{\ell}_i| \cdot |\boldsymbol{\ell}_j|^2}, \\ \dfrac{\partial^3 b_i}{\partial \boldsymbol{\tau}_i^2 \boldsymbol{\tau}_k} = \dfrac{-6}{|\boldsymbol{\ell}_i|^2 \cdot |\boldsymbol{\ell}_k|}, & \dfrac{\partial^3 b_i}{\partial \boldsymbol{\tau}_i \boldsymbol{\tau}_k^2} = \dfrac{+2}{|\boldsymbol{\ell}_i| \cdot |\boldsymbol{\ell}_k|^2}, \\ \dfrac{\partial^3 b_i}{\partial \boldsymbol{\tau}_j^2 \boldsymbol{\tau}_k} = \dfrac{-2}{|\boldsymbol{\ell}_j|^2 \cdot |\boldsymbol{\ell}_k|}, & \dfrac{\partial^3 b_i}{\partial \boldsymbol{\tau}_j \boldsymbol{\tau}_k^2} = \dfrac{-2}{|\boldsymbol{\ell}_j| \cdot |\boldsymbol{\ell}_k|^2}. \end{cases}$$

We have from the above formulas that

$$\mathcal{L} b_0 = \frac{-2}{|\boldsymbol{\ell}_1|^2 |\boldsymbol{\ell}_2|} + \frac{-2}{|\boldsymbol{\ell}_2|^2 |\boldsymbol{\ell}_3|} + \frac{-2}{|\boldsymbol{\ell}_3|^2 |\boldsymbol{\ell}_1|} \tag{4.10}$$

and

$$\mathcal{L} b_i = \frac{-6}{|\boldsymbol{\ell}_i|^2 |\boldsymbol{\ell}_j|} + \frac{-2}{|\boldsymbol{\ell}_j|^2 |\boldsymbol{\ell}_k|} + \frac{+2}{|\boldsymbol{\ell}_k|^2 |\boldsymbol{\ell}_i|}, \qquad i = 1, 2, 3. \tag{4.11}$$

Now the coefficient v_0 can be solved from

$$\mathcal{L} u = v_0 \mathcal{L} b_0 + \sum_{i=1}^{3} v_i \mathcal{L} b_i,$$

which yields

$$v_0 = \left(\mathcal{L} u - \sum_{i=1}^{3} v_i \mathcal{L} b_i \right) / \mathcal{L} b_0. \tag{4.12}$$

Now we derive the formulas for the H^1-seminorm and L^2-norm of the interpolation error. Clearly,

$$\int_K |\nabla e|^2 \, dxdy = \boldsymbol{v} \cdot B_1 \boldsymbol{v}, \tag{4.13}$$

where $\boldsymbol{v} = [v_0, v_1, v_2, v_3]^T$, and $B_1 = \left(\int_K \nabla b_i \cdot \nabla b_j \, dxdy \right)$. By the identity [10]

$$\int_K (c_1)^{\alpha_1} (c_2)^{\alpha_2} (c_3)^{\alpha_3} \, dxdy = 2|K| \frac{\alpha_1! \alpha_2! \alpha_3!}{(\alpha_1 + \alpha_2 + \alpha_3 + 2)!},$$

we have

$$B_1 = \frac{1}{720|K|} \tag{4.14}$$

$$\begin{bmatrix} |\boldsymbol{\ell}_1|^2 + |\boldsymbol{\ell}_2|^2 + |\boldsymbol{\ell}_3|^2 & 2(|\boldsymbol{\ell}_3|^2 - |\boldsymbol{\ell}_2|^2) & 2(|\boldsymbol{\ell}_1|^2 - |\boldsymbol{\ell}_3|^2) & 2(|\boldsymbol{\ell}_2|^2 - |\boldsymbol{\ell}_1|^2) \\ & 4(2|\boldsymbol{\ell}_1|^2 - 3\boldsymbol{\ell}_2 \cdot \boldsymbol{\ell}_3) & -4\boldsymbol{\ell}_1 \cdot \boldsymbol{\ell}_2 & -4\boldsymbol{\ell}_1 \cdot \boldsymbol{\ell}_3 \\ & & 4(2|\boldsymbol{\ell}_2|^2 - 3\boldsymbol{\ell}_1 \cdot \boldsymbol{\ell}_3) & -4\boldsymbol{\ell}_2 \cdot \boldsymbol{\ell}_3 \\ & sym. & & 4(2|\boldsymbol{\ell}_3|^2 - 3\boldsymbol{\ell}_1 \cdot \boldsymbol{\ell}_2) \end{bmatrix}.$$

For the L^2-norm of the interpolation error, we similarly have

$$\int_K |e|^2 \, dxdy = \boldsymbol{v} \cdot B_0 \boldsymbol{v}, \tag{4.15}$$

where

$$B_0 = \left(\int_K b_i b_j \, dxdy \right) = \frac{|K|}{7!} \begin{bmatrix} 2 & 0 & 0 & 0 \\ & 6 & -2 & -2 \\ & & 6 & -2 \\ & sym. & & 6 \end{bmatrix}. \tag{4.16}$$

5. Optimal aspect ratios: A case study. Since error formulas (4.13) and (4.15) involve complicated interaction between the geometric features of K and the orientation and anisotropic ratio of $\nabla^3 u$, it is rather difficult, if not impossible, to decipher from them some quantitative guidelines for anisotropic mesh generation. In particular, among triangles that are of unit area and aligned with the orientation of $\nabla^3 u$, we would like to identify those which produce the smallest quadratic interpolation errors. We study this issue in the case when the triangle is an acute or obtuse isosceles triangle. Because the error is invariant under the rotation transform of the coordinates, we may assume the orientation of $\nabla^3 u$ is simply the x-axis. Furthermore, since we are mainly concerned with the local behavior of the error, we assume u is an homogeneous cubic function. In this case, $p_3(\boldsymbol{\xi}) = u(\boldsymbol{\xi})$ is either the product of three linear functions or the product of a linear function and a positive quadratic function. Suppose the anisotropic ratio of $\nabla^3 u$ is $\frac{1}{\epsilon}$. By the definition of the orientation and anisotropic ratio, u must be a multiple of one of the following three types of functions.

(i). u is the product of three linear functions. In this case u must be a multiple of

$$u_1(x, y) = \frac{1}{3!}(y - \delta x)(y^2 - \epsilon^2 x^2), \tag{5.1}$$

where $|\delta| \leq \epsilon$; see Figure 3(a).

(ii). u is the product of a linear function ℓ_3 and a positive quadratic function q, with the linear factor "lying inside" the quadratic factor (see Figure 3(b)). In this case, the orientation of $\nabla^3 u$ is determined by the directions of ℓ_1^* and ℓ_2^* in q^*. Hence, u is a multiple of

$$u_2(x, y) = \frac{1}{3!}(y - \delta x)(y^2 + \epsilon^2 x^2), \tag{5.2}$$

where $|\delta| \leq \epsilon$.

(iii). u is the product of a linear function ℓ_3 and a positive quadratic function q, with the linear factor "lying outside" the quadratic factor (see Figure 3(c)). In this case, the orientation of $\nabla^3 u$ is determined by the directions of ℓ_3 and one of the factors ℓ_1^* or ℓ_2^* in q^*. Therefore, u must be a multiple of

$$u_3(x, y) = \frac{1}{3!}(y - \epsilon x) q(x, y), \tag{5.3}$$

where

$$q(x, y) = (R_\gamma^T \boldsymbol{x}) \cdot \begin{bmatrix} \sin^2 \gamma & 0 \\ 0 & \cos^2 \gamma \end{bmatrix} (R_\gamma^T \, \boldsymbol{x}),$$

where R_γ is the matrix of rotation by angle γ, and $|\gamma| \le \tan^{-1}(\epsilon)$.

The magnitude of the first two types of functions is $(1+\epsilon^2)\sqrt{1+\delta^2}$, while for the third type it is $\sqrt{1+\epsilon^2}$. Under the assumption $\epsilon \ll 1$, we have for all three cases the magnitude being approximately 1.

We shall simplify the error formulas (4.13) and (4.15) for the first two types of functions when the element is an acute or obtuse isosceles triangle aligned with the $\nabla^3 u$. In this case, optimal aspect ratios leading to the smallest interpolation errors can be identified. The error formulas for the third type of function are still too complicated. For this case we study the optimal aspect ratio numerically in the next section.

5.1. H^1-seminorm of the interpolation error. First, we present the formula for the H^1-seminorm of the interpolation error for function (5.1). We first consider element K being an acute isosceles triangle aligned with the orientation of $\nabla^3 u$, i.e., $\phi_u = 0, \phi_v = \frac{\pi}{2}$. Denote by $|K|$ the area of K, and denote by r its aspect ratio. With the help of Maple, we have in this case

$$\boldsymbol{\ell}_1 = \sqrt{\frac{4|K|}{\sqrt{3}r}}\begin{bmatrix} 0 \\ -1 \end{bmatrix}, \quad \boldsymbol{\ell}_2 = \sqrt{\frac{|K|}{\sqrt{3}r}}\begin{bmatrix} \sqrt{3}r \\ 1 \end{bmatrix}, \quad \boldsymbol{\ell}_3 = \sqrt{\frac{|K|}{\sqrt{3}r}}\begin{bmatrix} -\sqrt{3}r \\ 1 \end{bmatrix},$$

and

$$B_1 = \frac{\sqrt{3}}{540}\begin{bmatrix} \frac{3}{2}(r+\frac{1}{r}) & 0 & \frac{3}{2}(r-\frac{1}{r}) & \frac{3}{2}(r-\frac{1}{r}) \\ & 5r+\frac{9}{r} & \frac{2}{r} & \frac{2}{r} \\ & & 6r+\frac{8}{r} & 3r-\frac{1}{r} \\ & sym. & & 6r+\frac{8}{r} \end{bmatrix}. \tag{5.4}$$

By using formulas (4.6)–(4.12) in section 4, we have the coefficients v_i, $0 \le i \le 3$, as

$$\begin{aligned} v_0 &= 0, \\ v_1 &= -\tfrac{2\sqrt[4]{3}}{9}|K|^{3/2} r^{-3/2}, \\ v_2 &= \tfrac{3^{3/4}}{108}|K|^{3/2}(3r^2\epsilon^2 - 1)(3\delta r - \sqrt{3})\ r^{-3/2}, \\ v_3 &= -\tfrac{3^{3/4}}{108}|K|^{3/2}(3r^2\epsilon^2 - 1)(3\delta r + \sqrt{3})\ r^{-3/2}, \end{aligned}$$

and the H^1-seminorm of the interpolation error $e = u - \Pi_2 u$ is

$$\begin{aligned} \|\nabla e\|^2_{L^2(K)} = \tfrac{|K|^3}{12960}[&r^2\epsilon^4(9+54\delta^2) + (7\epsilon^4 + 12\delta^2\epsilon^2 - 6\epsilon^2 + 16\delta^2) \\ &+ r^{-2}(33 + 6\delta^2 + 6\epsilon^2) + 15r^{-4}]. \end{aligned}$$

Now consider the optimal choice of the aspect ratio r for a given anisotropy ratio ϵ^{-1}. It is easy to verify that the minimum of $\|\nabla e\|_{L^2(K)}$ is attained when r solves the equation

$$(3 + 18\delta^2)r^6\epsilon^4 - (11 + 2\delta^2 + 2\epsilon^2)r^2 - 10 = 0.$$

Use the formula for roots of cubic polynomials, we have the only positive solution of the above equation,

$$r^2 = \frac{2}{3\epsilon^2}\sqrt{\frac{11 + 2\delta^2 + 2\epsilon^2}{1 + 6\delta^2}}\cos\left(\frac{\theta}{3}\right),$$

where

$$\theta = \cos^{-1}\left[45\epsilon^2\sqrt{\frac{1+6\delta^2}{(11+2\delta^2+2\epsilon^2)^3}}\right].$$

When $|\delta| \le \epsilon \ll 1$, we have $\theta \approx \frac{\pi}{2}$ and $r^2 \approx \sqrt{\frac{11}{3}}\epsilon^{-2}$. Therefore, the optimal aspect ratio r_* is approximately

$$r_* \approx \sqrt[4]{\frac{11}{3}}\,\epsilon^{-1} \approx 1.383\,\epsilon^{-1}.$$

With this aspect ratio, the interpolation error is

$$\|\nabla e_*\|^2_{L^2(K)} \approx (0.002196\epsilon^2 + 0.001235\delta^2)|K|^3.$$

Next we consider the case where K is an obtuse isosceles triangle aligned with the orientation of $\nabla^3 u$, i.e., $\phi_u = 0, \phi_v = 0$. Simplifying error formula (4.13), we end up with

$$\|\nabla e\|^2_{L^2(K)} = \tfrac{|K|^3}{12960}[15r^4\delta^2\epsilon^4 + r^2(33\delta^2\epsilon^4 + 6\epsilon^2\delta^2 + 6\epsilon^4) + (-6\epsilon^2\delta^2 + 16\epsilon^4 + 12\epsilon^2 + 7\delta^2) + r^{-2}(54 + 9\delta^2)].$$

The critical points of the right-hand side function are the solutions of

$$10r^6\delta^2\epsilon^4 + r^4(11\delta^2\epsilon^4 + 2\epsilon^2\delta^2 + 2\epsilon^4) - (18 + 3\delta^2) = 0.$$

Let $z = r^{-2}$; then z satisfies the cubic equation

$$z^3 - \frac{11\delta^2\epsilon^4 + 2\epsilon^2\delta^2 + 2\epsilon^4}{18 + 3\delta^2}z = \frac{10\delta^2\epsilon^4}{18 + 3\delta^2}.$$

By studying the positive solution of the above equation for all $|\delta| \le \epsilon \ll 1$, we conclude that the optimal value of the aspect ratio is in the range

$$1.047\epsilon^{-1} \le r_* \le 1.732\epsilon^{-1},$$

and the optimal interpolation error is in the range

$$0.003703|K|^3\epsilon^2 \le \|\nabla e_*\|^2_{L^2(K)} \le 0.007673|K|^3\epsilon^2.$$

We may study the H^1-error for the second type of function (5.2) in a similar way. The error formulas and the optimal aspect ratios are summarized in Table 5.1. A noticeable observation is that the optimal aspect ratio for both types of functions is proportional to the anisotropic ratio ϵ^{-1} of $\nabla^3 u$, and the H^1-seminorm of the error is proportional to ϵ. This is also true for the third type of function (5.3); see the numerical study of this case in the next section. Thus we conclude that when an element is aligned with the orientation of $\nabla^3 u$, the optimal aspect ratio that leads to the smallest H^1-seminorm of the quadratic interpolation errors is about cS_3, with $c \in (1.047, 1.732)$. With this direction and aspect ratio taken, the interpolation error $\|\nabla e\|^2_{L^2(K)}$ is of the magnitude $c(D_3)^2(S_3)^{-2}|K|^3$ for $c \in (0.001852, 0.007673)$. Here D_3 and S_3 are the magnitude and anisotropic ratio of $\nabla^3 u$ defined in section 3.

TABLE 5.1

H^1-seminorm of the quadratic interpolation error for functions (5.1) and (5.2) when element K is an isosceles triangle aligned with $\nabla^3 u$. $|K|$ and r are the area and aspect ratio of the triangle, respectively.

$u = \frac{1}{3!}(y - \delta x)(y^2 - \epsilon^2 x^2)$:

K = acute isosceles

$$\|\nabla e\|^2_{L^2(K)} = \tfrac{|K|^3}{12960}[r^2\epsilon^4(9 + 54\delta^2) + (7\epsilon^4 + 12\delta^2\epsilon^2 - 6\epsilon^2 + 16\delta^2) + r^{-2}(33 + 6\delta^2 + 6\epsilon^2) + 15r^{-4}],$$

$$r_* \approx 1.383\,\epsilon^{-1},$$

$$\|\nabla e_*\|^2_{L^2(K)} \approx (0.002196\epsilon^2 + 0.001235\delta^2)|K|^3.$$

K = obtuse isosceles

$$\|\nabla e\|^2_{L^2(K)} = \tfrac{|K|^3}{12960}[15r^4\delta^2\epsilon^4 + r^2(33\delta^2\epsilon^4 + 6\epsilon^2\delta^2 + 6\epsilon^4) + (-6\epsilon^2\delta^2 + 16\epsilon^4 + 12\epsilon^2 + 7\delta^2) + r^{-2}(54 + 9\delta^2)],$$

$$r_* \approx c\epsilon^{-1}, \text{ with } c \in (1.047, 1.732),$$

$$\|\nabla e_*\|^2_{L^2(K)} \approx c|K|^3\epsilon^2, \text{ with } c \in (0.003703, 0.007673).$$

$u = \frac{1}{3!}(y - \delta x)(y^2 + \epsilon^2 x^2)$:

K = acute isosceles

$$\|\nabla e\|^2_{L^2(K)} = \tfrac{|K|^3}{12960}[r^2\epsilon^4(9 + 54\delta^2) + (7\epsilon^4 - 12\delta^2\epsilon^2 + 6\epsilon^2 + 16\delta^2) + r^{-2}(33 + 6\delta^2 - 6\epsilon^2) + 15r^{-4}],$$

$$r_* \approx 1.383\,\epsilon^{-1},$$

$$\|\nabla e_*\|^2_{L^2(K)} \approx (0.003122\epsilon^2 + 0.001235\delta^2)|K|^3.$$

K = obtuse isosceles

$$\|\nabla e\|^2_{L^2(K)} = \tfrac{|K|^3}{12960}[15r^4\delta^2\epsilon^4 + r^2(33\delta^2\epsilon^4 - 6\epsilon^2\delta^2 + 6\epsilon^4) + (6\epsilon^2\delta^2 + 16\epsilon^4 - 12\epsilon^2 + 7\delta^2) + r^{-2}(54 + 9\delta^2)],$$

$$r_* \approx c\epsilon^{-1}, \text{ with } c \in (1.103, 1.743),$$

$$\|\nabla e_*\|^2_{L^2(K)} \approx c|K|^3\epsilon^2, \text{ with } c \in (0.001852, 0.004752).$$

Another important observation is that the optimal aspect ratio is about the same for both acute and obtuse isosceles triangles. This is in contrast to the case of linear interpolations, where a much higher accuracy can be achieved for acute isosceles triangles with a much larger aspect ratio than that for obtuse isosceles triangles. This phenomenon is termed as "superaccuracy" by Shewchuk in [18]. The study in this subsection indicates that there is no "superaccuracy" in favor of acute isosceles triangles for quadratic interpolation of general cubic functions.

5.2. L^2-norm of the interpolation error. For the L^2-norm of the interpolation error, we may study similarly the optimal choice of the aspect ratio r for the first two types of functions (5.1) and (5.2). The results are summarized in Table 5.2 for

TABLE 5.2

L^2-norm of the quadratic interpolation error for functions (5.1) *and* (5.2) *when element K is an isosceles triangle aligned with $\nabla^3 u$. $|K|$ and r are the area and aspect ratio of the triangle, respectively.*

$u = \frac{1}{3!}(y - \delta x)(y^2 - \epsilon^2 x^2)$:

K = acute isosceles

$$\|e\|^2_{L^2(K)} = \frac{\sqrt{3}|K|^4}{272160}[27\epsilon^4\delta^2 r^3 + 3(\epsilon^2 + 2\delta^2)\epsilon^2 r + (11\delta^2 - 10\epsilon^2)r^{-1} + 19r^{-3}],$$

$$r_* \approx c\epsilon^{-1}, \text{ with } c \in (0.9265, 1.732),$$

$$\|e_*\|^2_{L^2(K)} \approx c|K|^4\epsilon^3, \text{ with } c \in (0.00001959, 0.0003486).$$

K = obtuse isosceles

$$\|e\|^2_{L^2(K)} = \frac{\sqrt{3}|K|^4}{272160}[19\epsilon^4\delta^2 r^3 + (11\epsilon^2 - 10\delta^2)\epsilon^2 r + 3(\delta^2 + 2\epsilon^2)r^{-1} + 27r^{-3}],$$

$$r_* \approx c\epsilon^{-1}, \text{ with } c \in (1.079, 1.732),$$

$$\|e_*\|^2_{L^2(K)} \approx c|K|^4\epsilon^3, \text{ with } c \in (0.0001764, 0.0003486).$$

$u = \frac{1}{3!}(y - \delta x)(y^2 + \epsilon^2 x^2)$:

K = acute isosceles

$$\|e\|^2_{L^2(K)} = \frac{\sqrt{3}|K|^4}{272160}[27\epsilon^4\delta^2 r^3 + 3(\epsilon^2 - 2\delta^2)\epsilon^2 r + (11\delta^2 + 10\epsilon^2)r^{-1} + 19r^{-3}],$$

$$r_* \approx c\epsilon^{-1}, \text{ with } c \in (1.000, 2.517),$$

$$\|e_*\|^2_{L^2(K)} \approx c|K|^4\epsilon^3, \text{ with } c \in (0.00008092, 0.0004073).$$

K = obtuse isosceles

$$\|e\|^2_{L^2(K)} = \frac{\sqrt{3}|K|^4}{272160}[19\epsilon^4\delta^2 r^3 + (11\epsilon^2 + 10\delta^2)\epsilon^2 r + 3(\delta^2 - 2\epsilon^2)r^{-1} + 27r^{-3}],$$

$$r_* \approx c\epsilon^{-1}, \text{ with } c \in (1.000, 1.567),$$

$$\|e_*\|^2_{L^2(K)} \approx c|K|^4\epsilon^3, \text{ with } c \in (0.0001300, 0.0004073).$$

the cases $\phi_u = 0, \phi_v = \frac{\pi}{2}$ (aligned acute isosceles triangle) and $\phi_u = 0, \phi_v = 0$ (aligned obtuse isosceles triangle) separately. When the triangle is aligned with the orientation of $\nabla^3 u$, the aspect ratio leading to the smallest L^2-error is approximately equal to cS_3, with $c \in (0.9264, 2.517)$. With this aspect ratio, $\|e\|^2_{L^2(K)}$ is of the magnitude $c(D_3)^2(S_3)^{-3}|K|^4$ for $c \in (1.959 \times 10^{-5}, 4.072 \times 10^{-4})$.

It is interesting to note that when a triangle is aligned with $\nabla^3 u$, the optimal aspect ratios for both the H^1-seminorm and L^2-norm are close to the anisotropic ratio of $\nabla^3 u$. In practice, an aspect ratio $r \approx S(\nabla^3 u)$ can be used generally.

. . . 2. We emphasize the fact that the optimal aspect ratios listed in Tables 5.1 and 5.2 are derived under the assumption that K is aligned with the orientation of $\nabla^3 u$ defined in section 3. This orientation direction is only an estimate

of the direction along which the absolute value of third order directional derivative is about the smallest, while along its perpendicular direction it is about the largest. It is close to, but may not be exactly, the triangle orientation leading to the minimum interpolation error among all possible choices. Therefore, for elements aligned with the orientation of $\nabla^3 u$ we may expect only a nearly optimal interpolation error. Unlike the case of linear interpolation studied in [6], we do not know precisely in what alignment direction the quadratic interpolation error would be minimum (among all possible triangle shapes and orientations). Furthermore, it seems that the optimal choice of this direction is dependent on the norm used to measure the error. For instance, if we consider the L^2-norm of the interpolation error for function $u(x,y) = \frac{1}{6}xy^2$, then $\|e\|^2_{L^2(K)} = \frac{11\sqrt{3}}{272160}|K|^4 r^{-1}$ and $\frac{3\sqrt{3}}{272160}|K|^4 r^{-1}$ on an acute/obtuse isosceles triangle K aligned with the x-axis, respectively. In this case, the x-axis and a large r are good choices for the alignment and aspect ratio. On the other hand, if we consider the H^1-seminorm of the error for the above function, we have $\|\nabla e\|^2_{L^2(K)} = \frac{1}{12960}|K|^3(16 + 6r^{-2})$ and $\frac{1}{12960}|K|^3(7 + 9r^{-2})$ on an acute/obtuse isosceles triangle aligned with the x-axis, respectively. In this case, using triangles of high aspect ratios helps little to reduce the H^1-error. We may consider in this case $\nabla^3 u$ isotropic with no preferred alignment direction.

. . . 3. In a recent work studying mesh quality measures and variational mesh generation methods [14], Huang suggested use the following matrix to determine the element orientation and aspect ratio in the case of quadratic interpolation:

$$H(\nabla u) = |\nabla^2(\partial_x u)| + |\nabla^2(\partial_y u)|,$$

where $|A| = (A^T A)^{1/2}$ for a matrix A. More precisely, let $(\lambda_1, \boldsymbol{v}_1)$ and $(\lambda_2, \boldsymbol{v}_2)$ be the eigenpairs of $H(\nabla u)$. He suggested choosing the element length scale to be $1/\sqrt{\lambda_i}$ in the $\boldsymbol{v}_i$ direction, $i = 1, 2$, i.e., to use $\boldsymbol{v}_1$ as element orientation and $\sqrt{\lambda_2/\lambda_1}$ as element aspect ratio (assume $\lambda_1 \le \lambda_2$). We believe $H(\nabla u)$ cannot accurately characterize the anisotropic behavior of $\nabla^3 u$, and the guidelines based on $H(\nabla u)$ may not be helpful. While there is no numerical result in [14] for quadratic interpolation, we tested this choice for the function $u = y^3 + (\epsilon x)^3$, i.e., our Example 1 in section 3.3. In this case

$$H(\nabla u) = \begin{bmatrix} 6\epsilon^3 & 0 \\ 0 & 6 \end{bmatrix}.$$

Its eigenvalues are $\lambda_1 = 6\epsilon^3, \lambda_2 = 6$, and the corresponding eigenvectors are along the x-axis and y-axis. Suppose K is an isosceles triangle aligned with the x-axis and of aspect ratio r; then we may derive the exact formula of the interpolation error. Indeed, it is found that

$$\|\nabla e\|^2_{L^2(K)} = \frac{|K|^3}{12960}(33r^{-2} + 15r^{-4} + 54\epsilon^6 r^2),$$
$$\|e\|^2_{L^2(K)} = \frac{\sqrt{3}|K|^4}{272160}(19r^{-3} + 27\epsilon^6 r^3)$$

for acute isosceles triangle K and that

$$\|\nabla e\|^2_{L^2(K)} = \frac{|K|^3}{12960}[54r^{-2} + \epsilon^6(15r^4 + 33r^2)],$$
$$\|e\|^2_{L^2(K)} = \frac{\sqrt{3}|K|^4}{272160}(27r^{-3} + 19\epsilon^6 r^3)$$

for obtuse isosceles triangle K. If r is chosen as $\sqrt{\lambda_2/\lambda_1} = \epsilon^{-3/2}$ as determined in [14], then $\|\nabla e\|^2_{L^2(K)} = O(1) \cdot |K|^3$ for obtuse isosceles triangle K. This H^1-error is in the same magnitude as that over a shape regular triangle of the same area. In other words, using obtuse anisotropic elements with such an aspect ratio helps little to reduce the H^1-error. For the L^2-error, $r = \epsilon^{-3/2}$ leads to $\|e\|^2_{L^2(K)} = O(\epsilon^{3/2}) \cdot |K|^4$ for both acute and obtuse isosceles triangles K. They are a factor of $O(\epsilon^{-3/2})$ larger than with the choice suggested in this paper.

6. Effects of element orientation, aspect ratio, and internal angles: A numerical study. We present in this section a numerical study on the effects of the element alignment, aspect ratio, and internal angles in quadratic interpolations. Let K be a triangle of unit area but in general shape and orientation. Since, under rotation of the coordinates, any cubic function can be written as a multiple of the three types of functions (5.1)–(5.3), we study only the interpolation errors over K for these functions. Note that they are all of anisotropic ratio ϵ^{-1} and approximately of magnitude 1. We test with various values of ϵ in $(0, 1)$. We also experiment with different δ values for functions (5.1) and (5.2) and with various γ for $|\gamma| \le \tan^{-1}(\epsilon)$ for function (5.3).

We calculate exactly the H^1-seminorm and L^2-norm of the error $u - \Pi_2 u$ over K by a numerical quadrature for triangles based on 28 Fekete points described in [20]. This quadrature formula is exact for integrands of total degree up to 6, which is sufficient to evaluate precisely the H^1- and L^2-error for interpolation of cubic functions. We study mainly (1) the effects of the triangle aspect ratios when the element is aligned with $\nabla^3 u$; (2) the effects of the element alignments; and (3) the effects of the element internal angles. Because the numerical results for functions (5.1) and (5.2) are very similar, we report only those for (5.1). In particular, we report the results for (5.1) with $\delta = 0$ and $\delta = \epsilon$. The errors with general $|\delta| \le \epsilon$ are in between these two cases. Our main observations are the following:

(1) , , , r: If the triangle is aligned with the orientation of $\nabla^3 u$, then the optimal aspect ratios to produce the smallest interpolation error in both L^2- and H^2-norms are close to the anisotropic ratio of $\nabla^3 u$. We plot in Figure 5 the H^1-seminorm and L^2-, L^1-, and L^∞-norms of the error versus the aspect ratio of the element for test functions of different anisotropic ratios. The linear dependency of the optimal r on the anisotropic ratio ϵ^{-1} is clearly observed. In addition, with the optimal aspect ratio, the H^1-seminorm of the interpolation error is proportional to ϵ, and the L^2-, L^1-, and L^∞-norms are proportional to $\epsilon^{1.5}$.

(2) , , , ϕ_u: We study the effects of the element alignment by varying the angle ϕ_u in the SVD of the affine mapping for it; see section 2. We plot in Figure 6 the L^2-norm and H^1-seminorm of the error for test functions of various anisotropic ratios. In all the cases, the smallest interpolation error occurs when ϕ_u is close to 0, i.e., when the element is oriented in about the same direction of $\nabla^3 u$. Furthermore, this conclusion is true regardless of whether the element is an acute or obtuse isosceles triangle. It indicates that mesh alignment is crucial for minimizing the interpolation error.

(3) , , , ϕ_v: The internal angles of a triangular element are controlled by ϕ_v in the SVD of its affine mapping. When $\phi_v = m\pi/3$, the triangle is an obtuse isosceles triangle, and when $\phi_v = (2m + 1)\pi/6$, it is an acute isosceles triangle. We plot in Figures 7 and 8 the L^2- and H^1-errors versus ϕ_v for elements with $\phi_u = 0$ and $r = \epsilon^{-1}, 10\epsilon^{-1}$. When the aspect ratio $r = \epsilon^{-1}$, the L^2- and H^1-errors do not depend much on the value of ϕ_v, and thus the element internal angles have little

effect on the interpolation error. When the aspect ratio $r = 10\epsilon^{-1}$, which is far away from the optimal values, the H^1-seminorm of the error is about the smallest when $\phi_v = \pi/6, \pi/2, 5\pi/6$, which correspond to acute isosceles triangles, and the largest error occurs at $\phi_v = 0, \pi/3, 2\pi/3, \pi$, which correspond to obtuse isosceles triangles. The L^2-norm is still insensitive to the internal angles. Hence, we conclude that when proper aspect ratio $r \approx S(\nabla^3 u)$ is chosen and proper alignment is maintained, the H^1-seminorm of the interpolation error is insensitive to the element internal angles. If a much larger aspect ratio is used, then the H^1-error depends on the internal angles. The larger the maximum internal angle is, the larger the H^1-error. In this case, the maximum angle condition is essential [2]. The L^2-norm of the interpolation error is always insensitive to the internal angles.

7. Conclusion and discussions. In the previous sections we described how to measure the anisotropic behavior of the third order derivative $\nabla^3 u$ at a fixed point based on the distribution of the third order directional derivatives at the point. In particular, the orientation of $\nabla^3 u$ is defined as the direction along which the third order directional derivative is about the smallest, while along the perpendicular direction it is about the largest. The anisotropic ratio $S(\nabla^3 u)$ measures the strength of the anisotropic behavior of $\nabla^3 u$. These quantities are invariant under translation and rotation of the independent variables. Then we considered the model problem of quadratic interpolation of cubic functions over a triangular element. In particular, we studied the optimal choice of the aspect ratio for elements aligned with the orientation of $\nabla^3 u$. It turns out that (1) when an element is aligned with the orientation of $\nabla^3 u$, both the H^1-seminorm and L^2-norm of the error are about the smallest if its aspect ratio is approximately equal to $S(\nabla^3 u)$. This is true regardless of whether the element is an acute or an obtuse isosceles triangle. In practice, we may use the orientation of $\nabla^3 u$ for the mesh alignment direction and the anisotropic ratio $S(\nabla^3 u)$ for the element aspect ratio. Our study also indicates that unlike the case of linear interpolation [18, 6], there is no "superaccuracy" in favor of the acute isosceles triangles for quadratic interpolation in general. While there is no rigorous proof of it yet, we believe the reason is because $\nabla^3 u$ has richer geometric features than a triangle can capture. (Indeed, $\nabla^3 u$ has four degree of freedoms, while a triangle has only three: area, aspect ratio, and orientation.) (2) When an element is aligned with $\nabla^3 u$ and is of aspect ratio $S(\nabla^3 u)$, the H^1-seminorm of the error is proportional to $[S(\nabla^3 u)]^{-1}$, and the L^2-norm of the error is proportional to $[S(\nabla^3 u)]^{-3/2}$. Both of them are insensitive to the internal angles of the element. However, if the element aspect ratio is much larger than the anisotropic ratio of $\nabla^3 u$, the H^1-seminorm of the error increases greatly as the largest internal angle increases. The maximum angle condition is essential in this situation. The L^2-error is always insensitive to the internal angles.

The definitions of the orientation and the anisotropic ratio for $\nabla^3 u$ can be generalized to the case of $\nabla^m u$ for $m > 3$. It is also possible to derive more precise and coordinate independent error estimates for higher order interpolation based on these anisotropic measures; see [7]. These results can be used to better understand and improve the anisotropic mesh generation and local refinement. They can also be used for global mesh refinement strategies like the r-refinement or the moving mesh method. In these techniques, one typically needs to define a mesh metric (or monitor function) $M(\boldsymbol{x})$ and looks for a mesh that is quasi-uniform under the metric [3, 4, 5, 8, 9, 15]. For quadratic interpolation, one may choose the metric in the following form based on the anisotropic behavior of the third order derivatives of interpolated function u:

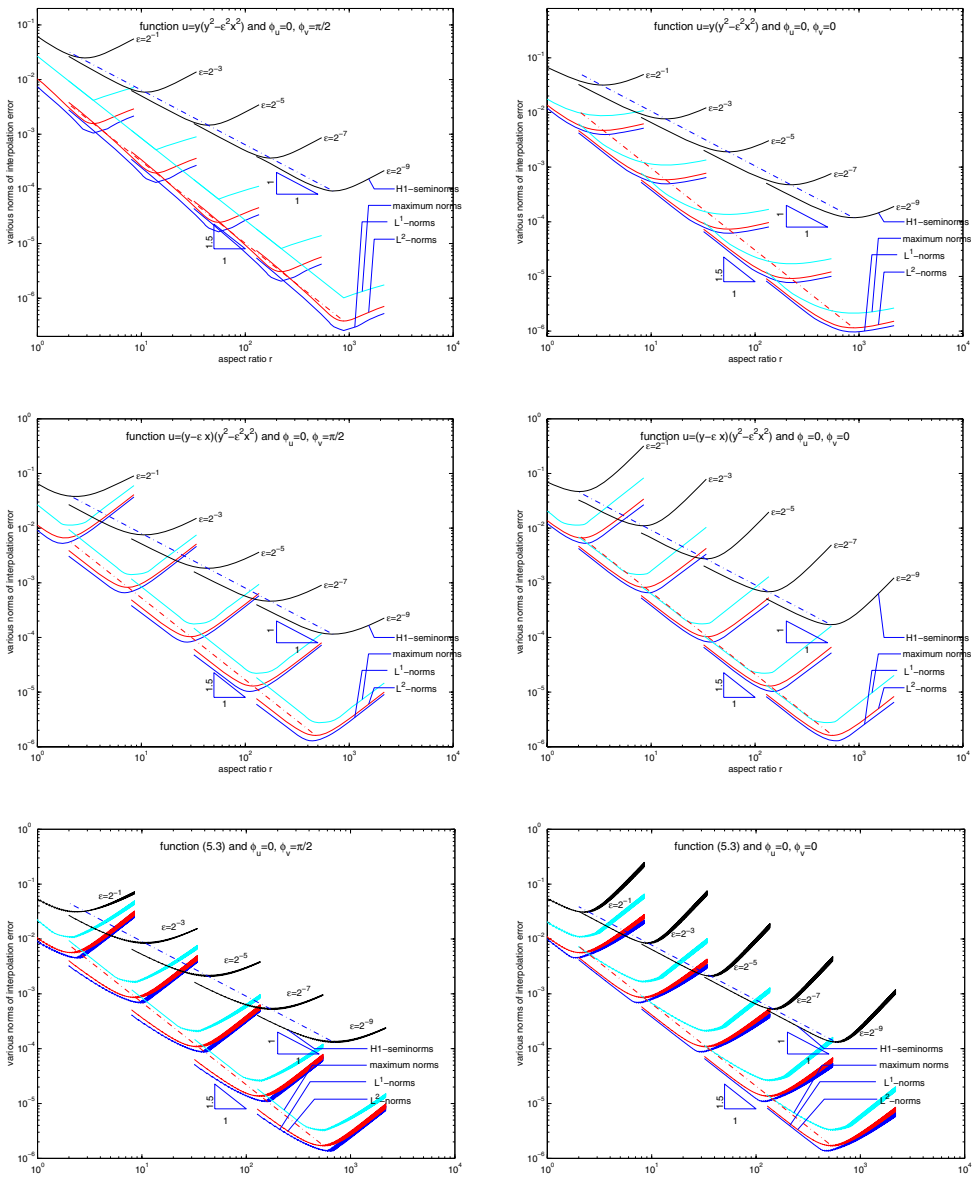


FIG. 5. *The H^1-seminorm and L^1-, L^2-, and maximum norms of the quadratic interpolation errors on an aligned element versus the element aspect ratio r.* (i) *Upper panel: for function $u = y(y^2 - \epsilon x^2)$;* (ii) *middle panel: for function $u = (y - \epsilon x)(y^2 - \epsilon^2 x^2)$;* (iii) *bottom panel: for function* (5.3) *with various $\gamma \in [0, \arctan(\epsilon)]$. The left panel is for the case $\phi_u = 0$ and $\phi_v = \pi/2$ (i.e., aligned acute isosceles triangle), and the right panel is for the case $\phi_u = 0$ and $\phi_v = 0$ (i.e., aligned obtuse isosceles triangle).*

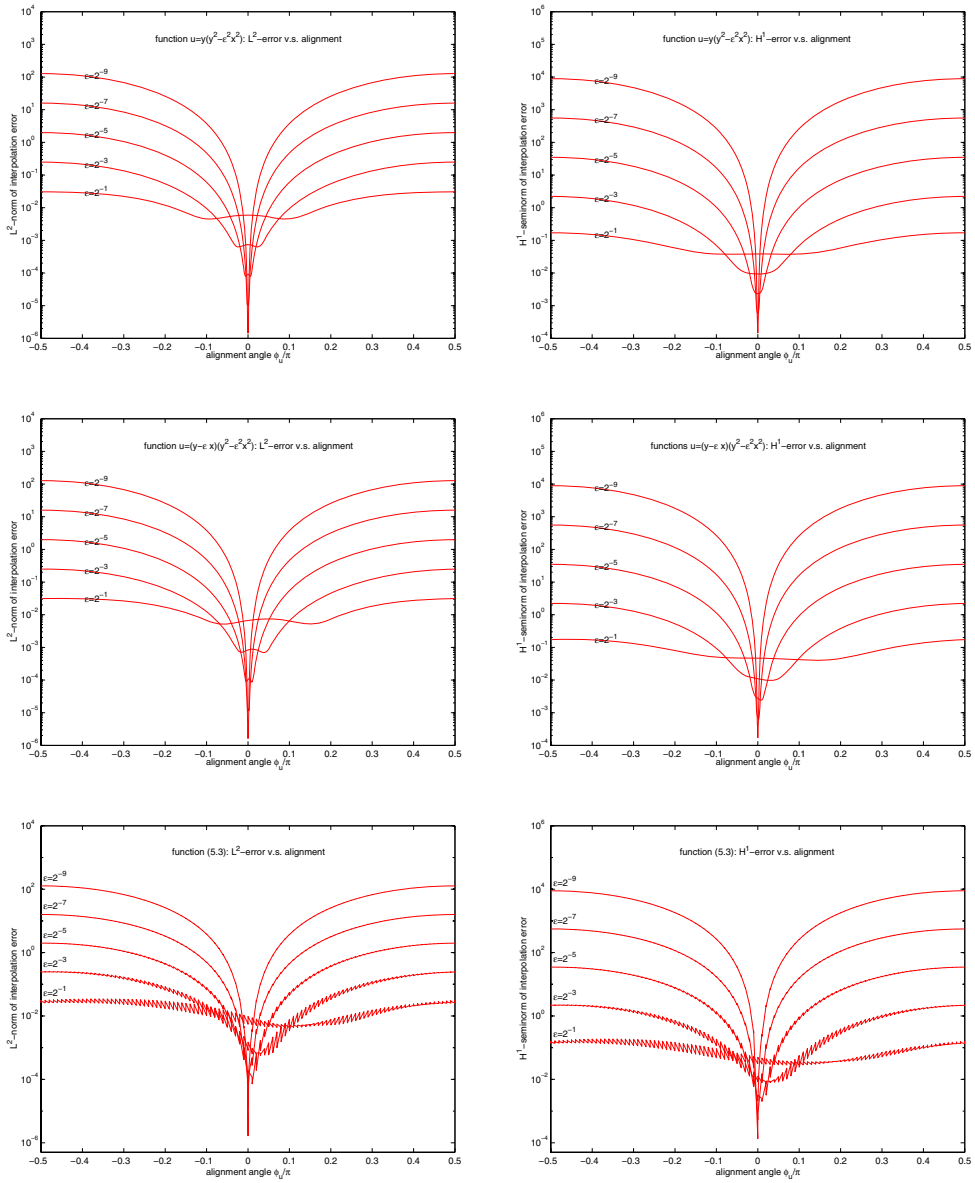


FIG. 6. *The L^2-norm (left) and H^1-seminorm (right) of the quadratic interpolation errors on an obtuse isosceles element (i.e., $\phi_v = 0$) versus its alignment angle ϕ_u/π for the case $r = \epsilon^{-1}$.* (i) *Upper panel: for function $u = y(y^2 - \epsilon x^2)$;* (ii) *middle panel: for function $u = (y - \epsilon x)(y^2 - \epsilon^2 x^2)$;* (iii) *bottom panel: for function* (5.3) *with various $\gamma \in [0, \arctan(\epsilon)]$.*

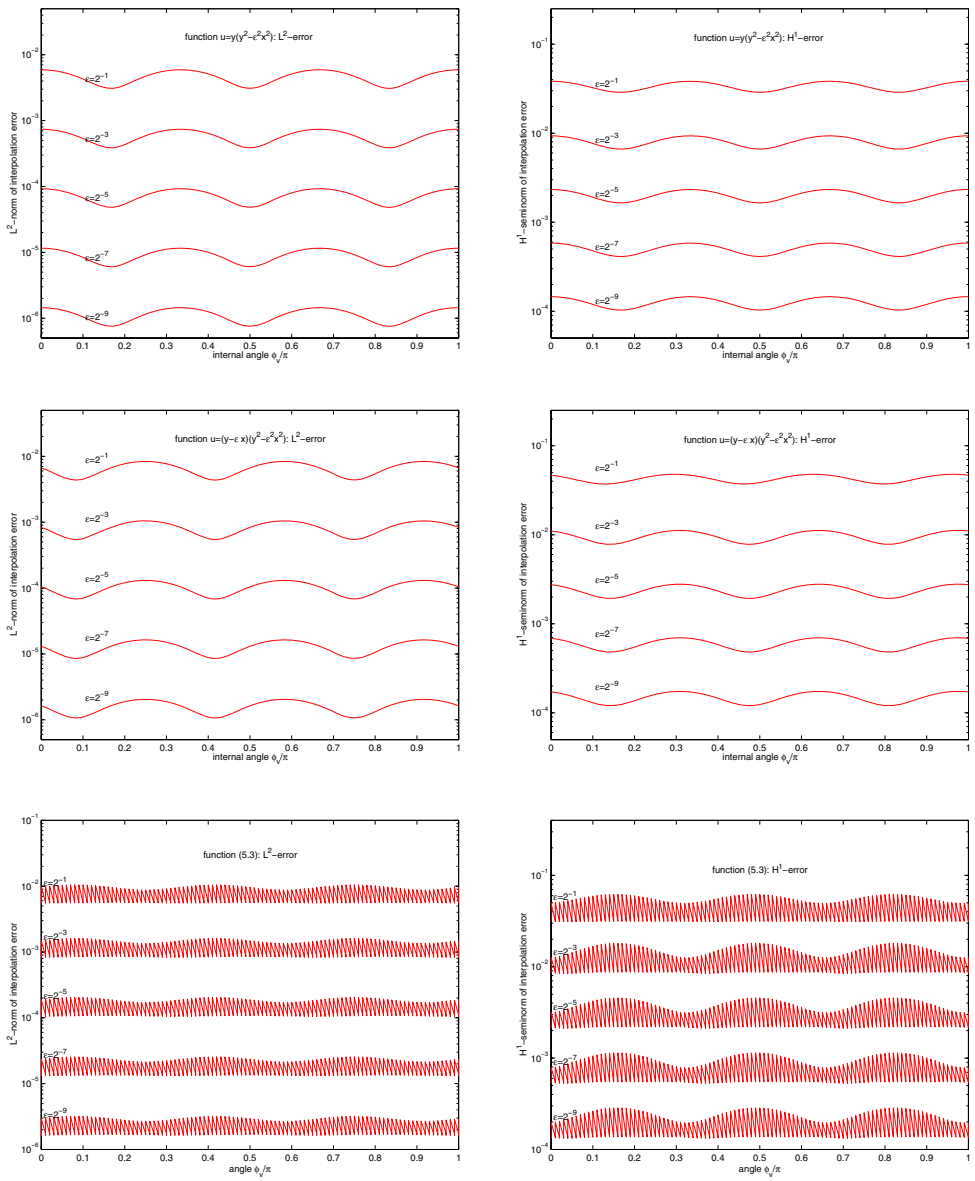


FIG. 7. *The L^2-norm (left) and H^1-seminorm (right) of the quadratic interpolation errors on an element versus the angle ϕ_v that controls the element internal angles. The element is always aligned with $\nabla^3 u$ (i.e., $\phi_u = 0$), and its aspect ratio is fixed as $r = \epsilon^{-1}$.* (i) *Upper panel: for function $u = y(y^2 - \epsilon x^2)$;* (ii) *middle panel: for function $u = (y - \epsilon x)(y^2 - \epsilon^2 x^2)$;* (iii) *bottom panel: for function* (5.3) *with various $\gamma \in [0, \arctan(\epsilon)]$.*

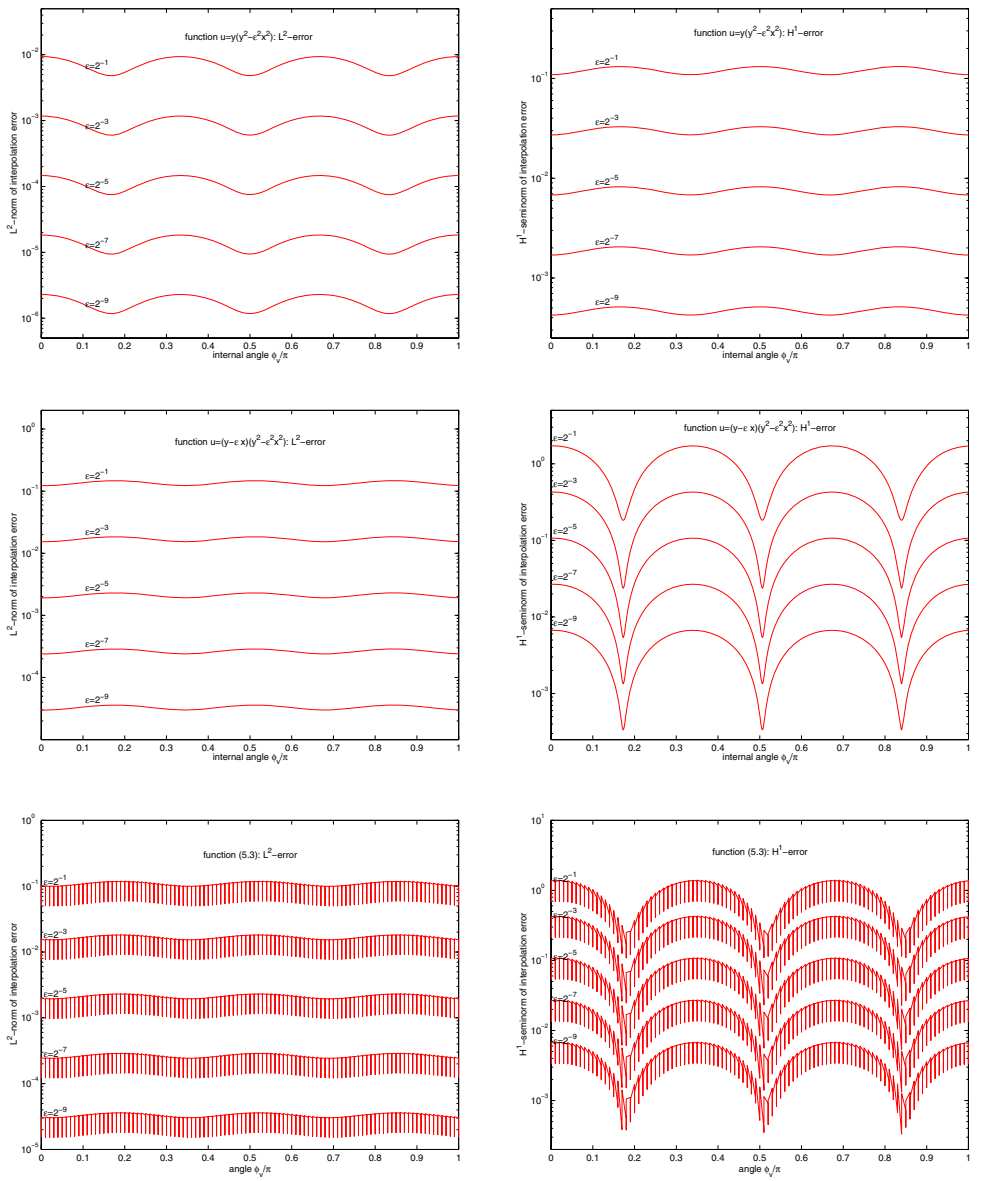


FIG. 8. *The L^2-norm (left) and H^1-seminorm (right) of the quadratic interpolation errors on an element versus the angle ϕ_v that controls the element internal angles. The element is always aligned with $\nabla^3 u$ (i.e., $\phi_u = 0$), and its aspect ratio is fixed as $r = 10\epsilon^{-1}$.* (i) *Upper panel: for function $u = y(y^2 - \epsilon x^2)$;* (ii) *middle panel: for function $u = (y - \epsilon x)(y^2 - \epsilon^2 x^2)$;* (iii) *bottom panel: for function* (5.3) *with various $\gamma \in [0, \arctan(\epsilon)]$.*

$$M(\boldsymbol{x}) = R_\phi \begin{bmatrix} \mu_1 & \\ & \mu_2 \end{bmatrix} R_\phi^T, \tag{7.1}$$

where ϕ is the angle (with the x-axis) of the orientation of $\nabla^3 u$,

$$\mu_1/\mu_2 \approx [S_3]^2, \tag{7.2}$$

and the determinant of $M(\boldsymbol{x})$ is taken as

$$\mu_1\mu_2 = \text{const} \cdot (D_3/S_3)^{4/3} \tag{7.3}$$

for minimizing the H^1-seminorm of the interpolation error and as

$$\mu_1\mu_2 = \text{const} \cdot D_3(S_3)^{-3/2} \tag{7.4}$$

for minimizing the L^2-norm of the interpolation error. Some modification is clearly needed when the function is almost unidirectional (i.e., the anisotropic ratio is ∞).

Finally, we would like to emphasize that the definition of orientation and anisotropic ratio for $\nabla^3 u$ described in [7] and here is only one possible way to characterize the anisotropic behavior of $\nabla^3 u$. Other definitions, possibly dependent on the norms used to measure the errors, can be introduced. In addition, when problems of multicomponents are considered, the anisotropic behavior of each variable should be carefully combined to produce an ideal direction for mesh alignment and an appropriate aspect ratio. These topics need further investigation.

REFERENCES

[1] T. Apel, *Anisotropic Finite Elements: Local Estimates and Applications*, Adv. Numer. Math., Teubner, Stuttgart, 1999.

[2] I. Babuška and A. K. Aziz, *On the angle condition in the finite element method*, SIAM J. Numer. Anal., 13 (1976), pp. 214–226.

[3] H. Borouchaki, P. L. George, F. Hecht, P. Laug, and E. Saltel, *Delaunay mesh generation governed by metric specifications. I. Algorithms.* Finite Elem. Anal. Des., 25 (1997), pp. 61–83.

[4] H. Borouchaki, P. L. George, and B. Mohammadi, *Delaunay mesh generation governed by metric specifications. II. Applications.* Finite Elem. Anal. Des., 25 (1997), pp. 85–109.

[5] W. Cao, W. Huang, and R. D. Russell, *A study of monitor functions for two-dimensional adaptive mesh generation*, SIAM J. Sci. Comput., 20 (1999), pp. 1978–1994.

[6] W. Cao, *On the error of linear interpolation and the orientation, aspect ratio, and internal angles of a triangle*, SIAM J. Numer. Anal., 43 (2005), pp. 19–40.

[7] W. Cao, *An interpolation error estimates in $\mathcal{R}^2$ based on anisotropic measures of higher order derivatives*, Math. Comp., to appear.

[8] L. Chen, P. Sun, and J. Xu, *Optimal anisotropic meshes for minimizing interpolation errors in L^p-norm*, Math. Comp., 76 (2007), pp. 179–204.

[9] L. Chen and J. Xu, *Optimal Delaunay triangulations*, J. Comput. Math., 22 (2004), pp. 299–308.

[10] P. G. Ciarlet, *The Finite Element Methods for Elliptic Problems*, Classics Appl. Math. 40, SIAM, Philadelphia, 2002.

[11] E. F. D'Azevedo, *Optimal triangular mesh generation by coordinate transformation*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 755–786.

[12] E. F. D'Azevedo and R. B. Simpson, *On optimal triangular meshes for minimizing the gradient error*, Numer. Math., 59 (1991), pp. 321–348.

[13] L. Formaggia and S. Perotto, *New anisotropic a priori error estimates*, Numer. Math., 89 (2001), pp. 641–667.

[14] W. Huang, *Measuring mesh qualities and application to variational mesh adaptation*, SIAM J. Sci. Comput., 26 (2005), pp. 1643–1666.

[15] W. Huang and W. Sun, *Variational mesh adaptation* II: *Error estimates and monitor functions*, J. Comput. Phys., 184 (2003), pp. 619–648.

[16] E. J. Nadler, *Piecewise Linear Approximation on Triangulations of a Planar Region*, Ph.D. thesis, Division of Applied Mathematics, Brown University, Providence, RI, 1985.

[17] S. Rippa, *Long and thin triangles can be good for linear interpolation*, SIAM J. Numer. Anal., 29 (1992), pp. 257–270.

[18] J. R. Shewchuk, *What is a good linear finite element? Interpolation, conditioning, anisotropy, and quality measure*, in Proceedings of the 11th International Meshing Roundtable, Sandia National Laboratories, Albuquerque, NM, 2002, pp. 115–126.

[19] R. B. Simpson, *Anisotropic mesh transformations and optimal error control*, Appl. Numer. Math., 14 (1994), pp. 183–198.

[20] M. A. Taylor, B. A. Wingate, and R. E. Vincent, *An algorithm for computing Fekete points in the triangle*, SIAM J. Numer. Anal., 38 (2000), pp. 1707–1720.

SIAM J. SCI. COMPUT.
Vol. 29, No. 2, pp. 782–810
© 2007 Society for Industrial and Applied Mathematics

AVERAGING TECHNIQUES FOR THE A POSTERIORI BEM ERROR CONTROL FOR A HYPERSINGULAR INTEGRAL EQUATION IN TWO DIMENSIONS*

CARSTEN CARSTENSEN† AND DIRK PRAETORIUS‡

Abstract. Averaging techniques or gradient recovery techniques are frequently employed tools for the a posteriori finite element error analysis. Their very recent mathematical justification for partial differential equations allows unstructured meshes and nonsmooth exact solutions. This paper establishes an averaging technique for the hypersingular integral equation on a one-dimensional boundary and presents numerical examples that show averaging techniques can be employed for an effective mesh-refining algorithm. For the discussed test examples, the provided estimator estimates the (in general unknown) error very accurately in the sense that the quotient *error/estimator* stays bounded with a value close to 1.

Key words. averaging techniques, ZZ error estimator, hypersingular integral equation, boundary element method, Galerkin method, a posteriori error estimate, adaptive algorithm

AMS subject classifications. 65N38, 65R20, 65N50

DOI. 10.1137/050623930

1. Introduction.

This paper is devoted to the numerical treatment of the hypersingular integral equation

$$Wu = f \quad \text{on } \Gamma \tag{1.1}$$

on the relatively open boundary piece Γ of $\partial\Omega \subseteq \mathbb{R}^2$ with the hypersingular integral operator

$$(Wu)(x) = \frac{1}{\pi}\frac{\partial}{\partial n_x}\int_\Gamma u(y)\,\frac{\partial}{\partial n_y}\log|x-y|\,ds_y. \tag{1.2}$$

Here, n_x, n_y denote the outer unit normal vectors in x and y, respectively, and ds_y denotes the surface integration on Γ with respect to the variable y. For finite element methods it has recently been shown that any averaging technique [ZZ] yields reliable error estimators [CB, AC, CFuI, CFuII, CFuIII, C2]. This paper establishes analogous results for the Galerkin boundary element method (BEM) for the hypersingular integral equation (1.1). The main idea, also employed in [HSWW], goes back at least to an Oberwolfach conference in the 1980s. The first application to integral equations provides an a posteriori error analysis for Symm's integral equation [CP]. Compared with [CP], the technical details in this paper—such as an approximation result in $H^{1/2}(\Gamma)$—are more involved; cf. Theorem 3.3 below. An overview of other types of a posteriori error estimates for the Galerkin BEM can be found in [CFa].

*Received by the editors February 7, 2005; accepted for publication (in revised form) September 15, 2006; published electronically April 17, 2007. This work was supported by DFG Research Center MATHEON "Mathematics for key technologies" in Berlin and by the Austrian Science Fund FWF under grant P15274.

http://www.siam.org/journals/sisc/29-2/62393.html

†Department of Mathematics, Humboldt-Universität zu Berlin, Unter den Linden 6, D-10099 Berlin, Germany (cc@math.hu-berlin.de).

‡Institute for Analysis and Scientific Computing, Vienna University of Technology, Wiedner Hauptstrasse 8-10, A-1040 Vienna, Austria (Dirk.Praetorius@tuwien.ac.at).

Let $\mathcal{H}$ denote the Hilbert space which allows the solution of (1.1) for a given $f \in \mathcal{H}^*$; see section 2 for the precise setting. Let $\mathcal{T}_H$ be a given triangulation of Γ with mesh-size H, and let $\mathcal{T}_h$ be obtained by uniform refinements of $\mathcal{T}_H$. We consider the discrete spaces $\mathcal{S}_0^1(\mathcal{T}_h)$ and $\mathcal{S}_0^2(\mathcal{T}_H)$ consisting of all globally continuous and $\mathcal{T}_h$-piecewise affine (resp., $\mathcal{T}_H$-piecewise quadratic) splines (with respect to the arclength) which are in $\mathcal{H}$. The corresponding Galerkin projections (with respect to the energy scalar product $\langle\!\langle \cdot\,, \cdot \rangle\!\rangle$; cf. (2.11)) are denoted with $\mathbb{G}_h$ and $\mathbb{G}_H$, respectively.

Let u denote the (in general, [illegible]) exact solution and $u_h = \mathbb{G}_h u \in \mathcal{S}_0^1(\mathcal{T}_h)$ be its Galerkin approximation, and consider the error estimator

$$\eta_M := |\!|\!| u_h - \mathbb{G}_H u_h |\!|\!| = \min_{v_H \in \mathcal{S}_0^2(\mathcal{T}_H)} |\!|\!| u_h - v_H |\!|\!|, \tag{1.3}$$

where $|\!|\!| v |\!|\!| := \langle\!\langle v\,, v \rangle\!\rangle^{1/2}$ denotes the energy norm. Then, Theorem 4.2 implies that

$$C_{\mathrm{eff}}\eta_M - \text{higher-order terms} \le |\!|\!| u - u_h |\!|\!| \le C_{\mathrm{rel}}\eta_M + \text{higher-order terms}, \tag{1.4}$$

where the higher-order terms depend only on the smoothness of the exact solution u. More precisely, (1.4) holds provided that u is $\mathcal{T}_H$-piecewise in $H^{2+\varepsilon}$ for some $\varepsilon > 0$. For the lower estimate in (1.4), we call the error estimator η_M [illegible]. For the upper estimate in (1.4), we say that η_M is [illegible].

[illegible] 1. In order to define spaces $H^m(\mathcal{T}_H)$ of all $\mathcal{T}_H$-piecewise H^m-functions, one has to assume further regularity of the boundary piece Γ provided that $m > 1$. For the analysis below, we restrict our discussion to polygonal boundary pieces.

In contrast to the L^2 norm on Γ, which satisfies $\|\cdot\|^2_{L^2(\Gamma)} = \sum_{\gamma \in \mathcal{T}_H} \|\cdot\|^2_{L^2(\gamma)}$, the energy norm is nonlocal. Thus, the error estimator η_M cannot be used directly for an adaptive mesh-refinement algorithm. Let H indicate the coarse triangulation $\mathcal{T}_H$ and associated quantities v_H, $\mathbb{G}_H$ as well as the (local) mesh-size as a weight in $L^2(\Gamma)$ norms, e.g., in (1.5) below. Replacing the (best approximation) operator $\mathbb{G}_H$ in the definition of η_M by an arbitrary averaging operator $\mathcal{A}_H$ onto $\mathcal{S}_0^2(\mathcal{T}_H)$, we obtain reliable error estimators $\eta_A := |\!|\!| u_h - \mathcal{A}_H u_h |\!|\!|$. By use of an inverse estimate provided in section 3.3, one verifies that

$$\mu_A := \|H^{1/2}(u_h - \mathcal{A}_H u_h)'\|_{L^2(\Gamma)} \le C_{\mathrm{inv}}\eta_A. \tag{1.5}$$

Here and throughout the paper, $(\cdot)' = \partial/\partial s$ denotes differentiation with respect to the arclengths along edges. In particular, this provides efficient error estimators

$$\mu_M := \|H^{1/2}(u_h - \mathbb{G}_H u_h)'\|_{L^2(\Gamma)} \quad \text{and} \quad \mu_\Pi := \|H^{1/2}(u_h' - \Pi_H u_h')\|_{L^2(\Gamma)}, \tag{1.6}$$

where Π_H denotes the L^2 projection onto $\mathcal{P}^1(\mathcal{T}_H)$, i.e., the $\mathcal{T}_H$-piecewise affine functions. Moreover, there also holds $\eta_M \le C\mu_M$ with some constant $C > 0$; i.e., μ_M is also reliable. Since the L^2 norm is local, the μ-estimators can be employed for an adaptive mesh-refinement.

The paper is organized as follows: Section 2 collects the preliminaries on the functional analytic setting of the hypersingular integral equation. Section 3 provides the necessary background on finite element approximation. We prove a local inverse estimate for the $H^{1/2}(\Gamma)$ norm and recall an approximation result from [C1] which is applied to the nodal interpolation operator. Section 4 is the core of this article and proves the reliability and efficiency results for the introduced error estimators. Section 5 provides details on the numerical implementation and the new adaptive mesh-refinement strategy. Four numerical experiments in section 6 conclude the work.

The examples cover a wide range, from an example with smooth solution (covered by the theory in section 4), to two examples with weak corner singularities, up to a final example on a slit which lacks almost any regularity. Even for this case, the proposed adaptive strategy recovers the optimal experimental convergence rate (in terms of number of elements).

2. Preliminaries on the functional analytic setting.

2.1. Fractional-order Sobolev spaces. Let Ω be a bounded domain in $\mathbb{R}^2$ with Lipschitz boundary $\partial\Omega$. Given $0 < \alpha \leq 1$, the Sobolev space $H^\alpha(\partial\Omega)$ is the set of all real-valued functions on $\partial\Omega$ which are the traces of functions in $H^{\alpha+1/2}(\mathbb{R}^2)$ to $\partial\Omega$,

$$H^\alpha(\partial\Omega) := \{u|_{\partial\Omega} \,:\, u \in H^{\alpha+1/2}(\mathbb{R}^2)\}. \tag{2.1}$$

Moreover, it is consistent to define $H^0(\partial\Omega) := L^2(\partial\Omega)$ and to define Sobolev spaces of negative order by duality:

$$H^{-\alpha}(\partial\Omega) := H^\alpha(\partial\Omega)^* := \text{ dual of } H^\alpha(\partial\Omega), \tag{2.2}$$

with corresponding norms and duality brackets $\langle\cdot\,,\cdot\rangle$ which extend the $L^2(\partial\Omega)$ scalar product. For the hypersingular integral equation on $\partial\Omega$, one considers the subspaces $H^\alpha_0(\partial\Omega)$ to factor the constant functions out:

$$H^\alpha_0(\partial\Omega) := H^\alpha(\partial\Omega)/\mathbb{R} = \{u \in H^\alpha(\partial\Omega) \,:\, \langle 1\,, u\rangle = 0\}, \tag{2.3}$$

where 1 denotes the constant function. For a (relatively) open subset $\omega \subseteq \Gamma$ and $\alpha \geq 0$, we define the fractional-order Sobolev space $H^\alpha(\omega)$ by extension:

$$H^\alpha(\omega) := \{u|_\omega \,:\, u \in H^\alpha(\partial\Omega)\}, \tag{2.4}$$

where the norm of $u \in H^\alpha(\omega)$ is defined as the minimal norm of an extension, i.e.,

$$\|u\|_{H^\alpha(\omega)} := \inf\{\|\widehat{u}\|_{H^\alpha(\partial\Omega)} \,:\, \widehat{u} \in H^\alpha(\partial\Omega) \text{ with } \widehat{u}|_\omega = u\}. \tag{2.5}$$

Furthermore, there are Sobolev spaces $\widetilde{H}^\alpha(\omega)$,

$$\widetilde{H}^\alpha(\omega) := \{u \in H^\alpha(\partial\Omega) \,:\, \operatorname{supp}(u) \subseteq \overline{\omega}\}, \tag{2.6}$$

associated with the usual $H^\alpha(\omega)$ norm. The corresponding spaces of negative order are again defined by duality:

$$H^{-\alpha}(\omega) = \widetilde{H}^\alpha(\omega)^* \quad \text{and} \quad \widetilde{H}^{-\alpha}(\omega) = H^\alpha(\omega)^*. \tag{2.7}$$

2. Note that $\widetilde{H}^\alpha(\partial\Omega) = H^\alpha(\partial\Omega)$. For $\omega \subsetneq \partial\Omega$, there holds only $\widetilde{H}^\alpha(\omega) \subseteq H^\alpha(\omega)$ with

$$\|u\|_{H^\alpha(\omega)} \leq \|u\|_{\widetilde{H}^\alpha(\omega)} \quad \text{for } u \in \widetilde{H}^\alpha(\omega). \tag{2.8}$$

3. Note that according to Sobolev's inequality in one dimension each function $u \in H^\alpha(\omega)$ with $\alpha > 1/2$ is continuous. Moreover, each function $u \in H^1(\omega)$ is absolutely continuous; i.e., there holds the fundamental theorem of calculus.

2.2. Hypersingular integral operator and energy norm. Let Γ be either the closed boundary $\Gamma = \partial\Omega$ or a (relatively) open boundary piece $\omega \subseteq \partial\Omega$. We recall the mapping properties of the hypersingular integral operator (1.2).

[illegible] 4. (a) For a closed surface $\Gamma = \partial\Omega$, the hypersingular integral operator is a bijective, linear, and bounded operator

$$W : H_0^\alpha(\partial\Omega) \to H_0^{\alpha-1}(\partial\Omega) \tag{2.9}$$

for all indices $0 \le \alpha \le 1$. It is elliptic for $\alpha = 1/2$.

(b) For an open surface $\Gamma \subsetneqq \partial\Omega$, the hypersingular integral operator is linear and bounded as operator

$$W : \widetilde{H}^\alpha(\Gamma) \to H^{\alpha-1}(\Gamma) \tag{2.10}$$

for all indices $0 \le \alpha \le 1$ and bijective provided that $0 < \alpha < 1$. For $\alpha = 1/2$, W is a bijective and elliptic operator.

To abbreviate the notation for the energy case $\alpha = 1/2$, we define the Sobolev space $\mathcal{H}$ to be either $H_0^{1/2}(\partial\Omega)$ in the case that $\Gamma = \partial\Omega$ is closed or $\widetilde{H}^{1/2}(\Gamma)$ provided that $\Gamma \subsetneqq \partial\Omega$ is an open arc. In both cases, $W : \mathcal{H} \to \mathcal{H}^*$ is bijective and elliptic, whence W induces a scalar product on $\mathcal{H}$,

$$\langle\!\langle u, v \rangle\!\rangle := \langle Wu, v \rangle \quad \text{for } u, v \in \mathcal{H}, \tag{2.11}$$

and the corresponding [illegible]

$$|\!|\!| u |\!|\!| := \langle\!\langle u, u \rangle\!\rangle^{1/2} \tag{2.12}$$

is an equivalent norm on $\mathcal{H} = H_0^{1/2}(\partial\Omega)$ and $\mathcal{H} = \widetilde{H}^{1/2}(\Gamma)$, respectively. According to the Lax–Milgram lemma, given $f \in \mathcal{H}^*$ there is a unique solution $u := W^{-1}f$ in $\mathcal{H}$ of (1.1).

3. Preliminaries on finite element approximation. This section recalls the Galerkin discretization of (1.1) in the Hilbert space $\mathcal{H}$ and provides some estimates for the discrete ansatz functions. Throughout we adopt the notation from the previous section.

3.1. Galerkin discretization. Let $\mathcal{T} = \{\Gamma_1, \dots, \Gamma_n\}$ be a triangulation of Γ. Each element Γ_j of the triangulation $\mathcal{T}$ is supposed to be a connected (affine) boundary piece. The set of all nodes of the triangulation $\mathcal{T}$ is denoted with $\mathcal{N}$. Let $h \in L^\infty(\Gamma)$ denote the local mesh-size $h|_{\Gamma_j} := h_j := \operatorname{diam}(\Gamma_j)$. For an integer $p \ge 0$, $\mathcal{P}_p(\mathcal{T})$ denotes the space of all piecewise polynomials of total degree $\le p$ (defined on a reference element $\Gamma_{\mathrm{ref}} = [0, 1]$). Furthermore, we define globally continuous splines $\mathcal{S}^p(\mathcal{T}) := \mathcal{P}_p(\mathcal{T}) \cap \mathcal{C}(\Gamma) \subseteq H^1(\Gamma) \subseteq H^\alpha(\Gamma)$. Finally, define $\mathcal{S}_0^p(\mathcal{T})$ to be either $\{v_h \in \mathcal{S}^p(\mathcal{T}) : \langle 1, v_h \rangle = 0\}$ provided that $\Gamma = \partial\Omega$ or $\{v_h \in \mathcal{S}^p(\mathcal{T}) : v_h|_{\partial\Gamma} = 0\}$ in case $\Gamma \subsetneqq \partial\Omega$. Here, $\partial\Gamma$ denotes the set of the two endpoints of Γ. The subspace $\mathcal{S}_0^p(\mathcal{T})$ satisfies $\mathcal{S}_0^p(\mathcal{T}) \subseteq \mathcal{H}$ and will be used for a Galerkin discretization.

If $\mathcal{S}$ is a finite dimensional subspace of $\mathcal{H}$, the discrete Galerkin approximation $u_h \in \mathcal{S}$ is uniquely determined by the linear system of equations

$$\langle\!\langle u_h, v_h \rangle\!\rangle = \langle f, v_h \rangle \quad \text{for all } v_h \in \mathcal{S}. \tag{3.1}$$

The Galerkin projection $\mathbb{G} : \mathcal{H} \to \mathcal{S}$, defined by $\mathbb{G}u \in \mathcal{S}$ with

$$\langle\!\langle G_u, v_h \rangle\!\rangle = \langle\!\langle u, v_h \rangle\!\rangle \quad \text{for all } v_h \in \mathcal{S}, \tag{3.2}$$

is the orthogonal projection onto $\mathcal{S}$ with respect to the energy norm.

3.2. Standard approximation estimate. Given a regular triangulation $\mathcal{T}$, real numbers $\alpha, m \in \mathbb{R}$ with $m \geq \alpha$, and an integer $p \geq 1$, define $\widehat{p} := \min\{p+1, m\} - 1/2$. Define the $\mathcal{T}$-piecewise Sobolev space

$$H^m(\mathcal{T}) := \{v \in L^2(\Gamma) : v|_{\Gamma_j} \in H^m(\Gamma_j) \text{ for all } \Gamma_j \in \mathcal{T}\}$$

with norm $\|v\|^2_{H^m(\mathcal{T})} = \sum_{j=1}^n \|v|_{\Gamma_j}\|^2_{H^m(\Gamma_j)}$. Let $\mathbb{G}$ denote the Galerkin projection onto $\mathcal{S}^p_0(\mathcal{T})$. Then, there exists some constant $C_1 > 0$ (depending only on Γ, α, m, and p) such that

$$|||v - \mathbb{G}v||| \leq C_1 h_{\max}^{\widehat{p}} \|v\|_{H^m(\mathcal{T})} \tag{3.3}$$

for all $v \in \mathcal{H} \cap H^m(\mathcal{T})$ and $h_{\max} := \max_{\Gamma_j \in \mathcal{T}} h_j$ [SaS, sections 4.1.9 and 4.1.11].

3.3. Local inverse estimate. This subsection establishes an inverse estimate in the energy norm (see Corollary 3.2), following the arguments from [GHS]. The proof uses the fact that the Sobolev spaces can be obtained from interpolation: $H^1(\Gamma)$ can equivalently be defined as the completion of the Lipschitz continuous functions $\mathrm{Lip}(\Gamma)$ with respect to the norm $\|u\|^2_{H^1(\Gamma)} = \|u\|^2_{L^2(\Gamma)} + \|u'\|^2_{L^2(\Gamma)}$. If one defines $\widetilde{H}^1(\Gamma)$ as the completion of $\{u \in \mathrm{Lip}(\Gamma) : u|_{\partial\Gamma} = 0\}$, the Sobolev spaces $H^\alpha(\Gamma)$ and $\widetilde{H}^\alpha(\Gamma)$ satisfy

$$H^\alpha(\Gamma) = [L^2(\Gamma); H^1(\Gamma)]_\alpha \quad \text{and} \quad \widetilde{H}^\alpha(\Gamma) = [L^2(\Gamma); \widetilde{H}^1(\Gamma)]_\alpha,$$

where the brackets $[\cdot]_\alpha$ denote complex interpolation with exponent $0 \leq \alpha \leq 1$ [BL, McL]. Note that interpolation leads to the same spaces $H^\alpha(\Gamma)$ and $\widetilde{H}^\alpha(\Gamma)$, respectively, but only to equivalent norms. We will use the symbol $\lesssim$ whenever an estimate holds up to a norm equivalence constant.

Remark 5. In the context of Sobolev spaces on domains $\omega \subseteq \mathbb{R}^d$, one writes $H^1_0(\omega)$ instead of $\widetilde{H}^1(\omega)$, but this would conflict with our definition of $H^1_0(\omega)$ in (2.3).

Proposition 3.1. [illegible] $C_2 > 0$ [illegible] α, p, Γ [illegible]

$$\|h^{1-\alpha} v_h'\|_{L^2(\Gamma)} \leq C_2 \|v_h\|_{H^\alpha(\Gamma)} \quad \text{[illegible] } v_h \in \mathcal{S}^p(\mathcal{T}). \tag{3.4}$$

[illegible] We first consider the case $\alpha = 0$. For an element $\Gamma_j \in \mathcal{T}$, there holds $\|v_h'\|_{L^2(\Gamma_j)} \leq C_3\, h_j^{-1} \|v_h\|_{L^2(\Gamma_j)}$ [GHS, Proposition 2.9]. The constant $C_3 > 0$ depends only on the degree p. The summation over all $\Gamma_j \in \mathcal{T}$ leads to

$$\|h v_h'\|_{L^2(\Gamma)} \leq C_3 \|v_h\|_{L^2(\Gamma)}. \tag{3.5}$$

The combination of (3.5) with the trivial estimate $\|v_h'\|_{L^2(\Gamma)} \lesssim \|v_h\|_{H^1(\Gamma)}$ allows an interpolation argument for the operator

$$T_\alpha : \big(\mathcal{S}^p(\mathcal{T}), \|\cdot\|_{H^\alpha(\Gamma)}\big) \to \big(\mathcal{P}_{p-1}(\mathcal{T}), \|h^{1-\alpha}(\cdot)\|_{L^2(\Gamma)}\big), v_h \mapsto v_h'. \tag{3.6}$$

T_0 is continuous with operator norm $\leq C_3$, and T_1 is continuous with operator norm $\lesssim 1$. In particular, T_α is well defined and continuous with operator norm $\lesssim C_3^{1-\alpha}$. Note that the interpolation space of $(L^2, \|h(\cdot)\|_{L^2(\Gamma)})$ and $(L^2, \|\cdot\|_{L^2(\Gamma)})$ for the index $0 \leq \alpha \leq 1$ is given by $(L^2, \|h^{1-\alpha}(\cdot)\|_{L^2(\Gamma)})$ [BL, section 5.4]. □

Remark 6. The given proof of the inverse estimate covers a wider range of problems: The same arguments hold for $d = 3$ (i.e., v_h' is replaced by the surface gradient ∇v_h) and arbitrary $1 \leq q \leq \infty$ instead of the Hilbert case $q = 2$.

For $\alpha = 1/2$, the norm equivalence on $\mathcal{H}$ and (2.8) prove the existence of local inverse estimates in the energy norm.

COROLLARY 3.2. [illegible] $C_{\rm inv} > 0$ [illegible] Γ [illegible] p [illegible]

$$\|h^{1/2} v_h'\|_{L^2(\Gamma)} \le C_{\rm inv} |\!|\!| v_h |\!|\!| \quad \text{[illegible]} \quad v_h \in \mathcal{S}_0^p(\mathcal{T}). \tag{3.7}$$

□

[illegible] 7. Since we will deal later with two discrete spaces, we are going to use superindices to indicate the mesh and the polynomial degree; e.g., $C_{\rm inv}^{h,p}$ denotes the constant in the inverse estimate (3.7) for $\mathcal{S}_0^p(\mathcal{T})$, where $\mathcal{T}$ is a mesh with (local) mesh-size $h \in L^\infty(\Gamma)$.

3.4. Local first-order approximation property. In this section we prove that the (slightly modified) nodal interpolation operator I_h onto $\mathcal{S}^1(\mathcal{T})$, for sufficiently smooth functions $v \in \mathcal{H}$, has a first-order approximation property with respect to the energy norm.

THEOREM 3.3 (see [C1]). [illegible] Ω [illegible] $\mathbb{R}^2$ $\Gamma = \partial\Omega$ [illegible] $\Gamma \subsetneqq \partial\Omega$ [illegible] $C_\alpha > 0$ [illegible] α [illegible] Γ [illegible]

$$C_4 := \begin{cases} C_\alpha & \text{[illegible]} \ \alpha \neq 1/2, \\ C_\alpha \, (\log(1+\kappa))^{1/2} & \text{[illegible]} \ \alpha = 1/2 \end{cases} \tag{3.8}$$

[illegible]

$$\kappa := \max\{h_j/h_k : \Gamma_j \text{ [illegible] } \Gamma_k\}, \tag{3.9}$$

[illegible]

$$\|v\|_{H^\alpha(\Gamma)} \le C_4 \, \|h^{1-\alpha} v'\|_{L^2(\Gamma)} \tag{3.10}$$

[illegible] $v \in H^1(\Gamma)$ [illegible] $\mathcal{T}$ □

For each node $z \in \mathcal{N}$, let $\phi_z \in \mathcal{S}^1(\mathcal{T})$ be the nodal basis function, i.e., $\mathcal{T}$-piecewise affine (with respect to the arclength) and globally continuous with $\phi_z(z) = 1$ and $\phi_z(z') = 0$ for $z' \in \mathcal{N} \backslash \{z\}$. Then, the [illegible] $\widehat{I}_h : \mathcal{C}(\Gamma) \to \mathcal{S}^1(\mathcal{T})$ is defined by

$$\widehat{I}_h v := \sum_{z \in \mathcal{N}} v(z) \phi_z. \tag{3.11}$$

In particular, $\widehat{I}_h v$ is well defined for any $v \in H^\alpha(\Gamma)$ and $\alpha > 1/2$. From this, we may define

$$I_h : \mathcal{C}(\Gamma) \to \mathcal{S}_0^1(\mathcal{T}), \quad I_h v := \begin{cases} \widehat{I}_h v & \text{for } \Gamma \subsetneqq \partial\Omega, \\ \widehat{I}_h v - |\Gamma|^{-1} \int_\Gamma \widehat{I}_h v \, ds & \text{for } \Gamma = \partial\Omega. \end{cases} \tag{3.12}$$

COROLLARY 3.4. [illegible] $v \in H_0^1(\Gamma)$ [illegible] $\Gamma = \partial\Omega$ [illegible] $v \in \widetilde{H}^1(\Gamma)$ [illegible] $\Gamma \subsetneqq \partial\Omega$ [illegible] $C_{\rm apx} \lesssim C_4$ [illegible] C_4 [illegible] (3.8)

$$|\!|\!| v - I_h v |\!|\!| \le C_{\rm apx} \|h^{1/2}(v - I_h v)'\|_{L^2(\Gamma)} \le C_{\rm apx} \|h^{1/2} v'\|_{L^2(\Gamma)}. \tag{3.13}$$

... $\Gamma \subsetneqq \partial\Omega$. Since $w := v - I_h v$ has a zero on each element, namely $w(z) = 0$ for all nodes $z \in \mathcal{N}$, we may apply Theorem 3.3 together with the continuity of $W : H^{1/2}(\Gamma) \to H^{-1/2}(\Gamma)$ to obtain

$$|||w||| \le C\|w\|_{H^{1/2}(\Gamma)} \le CC_4\|h^{1/2}w'\|_{L^2(\Gamma)}.$$

The second inequality is proved elementwise: Note that there holds $\int_{\Gamma_j} w'\,ds = 0$. Therefore, $(I_h v)' \in \mathcal{P}_0(\mathcal{T})$ satisfies $(I_h v)'|_{\Gamma_j} = (1/h_j)\int_{\Gamma_j} v'\,ds$. The elementwise L^2-orthogonality shows that

$$\|w'\|^2_{L^2(\Gamma_j)} = \|v'\|^2_{L^2(\Gamma_j)} - \|(I_h v)'\|^2_{L^2(\Gamma_j)} \le \|v'\|^2_{L^2(\Gamma_j)}.$$

A summation of these estimates over $\mathcal{T}_H$ concludes the proof:

$$\|h^{1/2}w'\|^2_{L^2(\Gamma)} = \sum_{j=1}^n h_j\|w'\|^2_{L^2(\Gamma_j)} \le \sum_{j=1}^n h_j\|v'\|^2_{L^2(\Gamma_j)} = \|h^{1/2}v'\|^2_{L^2(\Gamma)}. \qquad \square$$

... $\Gamma = \partial\Omega$. We define $w := v - I_h v \in H^1_0(\Gamma)$ and $\widehat{w} := w + (1/|\Gamma|)\int_\Gamma w\,ds$. Since $W1 = 0$ for the constant function 1, we have $|||w||| = |||\widehat{w}|||$. Because of $\int_\Gamma v\,ds = 0$, there holds $\widehat{w} = v - \widehat{I}_h v$. With $\widehat{w}' = w'$, we may thus apply the same arguments as in the case $\Gamma \subsetneqq \partial\Omega$ to prove (3.13). $\square$

COROLLARY 3.5. ... $\mathcal{S}$ ... $H^1_0(\Gamma)$ ... $\Gamma = \partial\Omega$ ... $\widetilde{H}^1(\Gamma)$ ... $\Gamma \subsetneqq \partial\Omega$ ... $\mathcal{S}^1_0(\mathcal{T}) \subseteq \mathcal{S}$ ... $\mathbb{G}_h$ ... $\mathcal{S}$ ...

$$|||v - \mathbb{G}_h v||| \le C_{\rm apx} \min\big\{\|h^{1/2}(v - \mathbb{G}_h v)'\|_{L^2(\Gamma)}, \|h^{1/2}v'\|_{L^2(\Gamma)}\big\} \tag{3.14}$$

... $v \in H^1_0(\Gamma)$ ... $v \in \widetilde{H}^1(\Gamma)$ ... $C_{\rm apx} > 0$ ... 3.4 ... According to the best approximation property of $\mathbb{G}_h$ and Corollary 3.4, there holds

$$|||v - \mathbb{G}_h v||| \le |||v - I_h v||| \le C_{\rm apx}\|h^{1/2}v'\|_{L^2(\Gamma)}. \tag{3.15}$$

Then $w := v - \mathbb{G}_h v$ satisfies $v - \mathbb{G}_h v = w - \mathbb{G}_h w$ from the idempotency of the projection $\mathbb{G}_h$. The application of (3.15) with v replaced by w yields $|||v - \mathbb{G}_h v||| \le C_{\rm apx}\|h^{1/2}w'\|_{L^2(\Gamma)}$. $\square$

4. A posteriori error control by averaging techniques. This section aims to provide a new class of error estimators for the hypersingular integral equation and states their reliability and efficiency.

4.1. Assumptions and notation. Let $\mathcal{T}_h = \{\Gamma_1, \dots, \Gamma_n\}$ and $\mathcal{T}_H = \{\gamma_1, \dots, \gamma_N\}$ be triangulations of Γ with (local) mesh-sizes h and H, respectively, and let $\mathcal{S}_h := \mathcal{S}^1_0(\mathcal{T}_h)$ and $\mathcal{S}_H := \mathcal{S}^2_0(\mathcal{T}_H)$ be defined in section 3.1. As in Corollary 3.4, let $I_h : H^1(\Gamma) \to \mathcal{S}_h$ denote the nodal interpolation operator onto $\mathcal{S}_h$ with approximation constant $C_{\rm apx} > 0$, and let $C^{H,2}_{\rm inv} > 0$ denote the constant in the inverse estimate (3.7) for $\mathcal{S}_H$. We assume that the mesh-size h is small enough when compared to H; more precisely,

$$C_{\rm apx}C^{H,2}_{\rm inv} \max_{\gamma_j \in \mathcal{T}_H} (\|h\|_{L^\infty(\gamma_j)}/H_j)^{1/2} =: L < 1. \tag{4.1}$$

Moreover, the analysis in the subsequent sections requires an additional regularity assumption on the exact solution $u \in \mathcal{H}$ of (1.1), namely,

$$u \in H^m(\mathcal{T}_H) \quad \text{for some } m > 2. \tag{4.2}$$

4.2. General results. Let $u \in \mathcal{H}$ denote the unique solution of (1.1), and let $u_h \in \mathcal{S}_h$ be its Galerkin approximation with respect to $\mathcal{S} = \mathcal{S}_h$ in (3.1). The Galerkin projection $\mathbb{G}_H$ onto $\mathcal{S}_H$ (i.e., the orthogonal projection onto $\mathcal{S}_H$ with respect to the energy norm) is compared with an [illegible] (not necessarily linear or continuous) operator

$$\mathcal{A}_H : L^2(\Gamma) \to \mathcal{S}_H. \tag{4.3}$$

The main results of this section are stated in Theorem 4.2 and Corollaries 4.3–4.5.

PROPOSITION 4.1. [illegible] (4.1) [illegible]

$$\|\!|u - u_h|\!\| \le \|\!|(\mathbb{1} - \mathbb{G}_H)(u - u_h)|\!\|/(1 - L). \tag{4.4}$$

[illegible] Define $\widetilde{h} \in \mathcal{P}_0(\mathcal{T}_H)$ by $\widetilde{h}|_{\gamma_j} := \|h\|_{L^\infty(\gamma_j)}$ for each element $\gamma_j \in \mathcal{T}_H$. For the Galerkin error $e := u - u_h$, the approximation property (3.13) of I_h yields

$$\|\!|(\mathbb{1} - I_h)\mathbb{G}_H e|\!\| \le C_{\rm apx}\|h^{1/2}(\mathbb{G}_H e)'\|_{L^2(\Gamma)} \le C_{\rm apx}\|\widetilde{h}^{1/2}(\mathbb{G}_H e)'\|_{L^2(\Gamma)}.$$

Now, we use the inverse estimate (3.7) for $\mathbb{G}_H e \in \mathcal{S}_H$ and infer that

$$\begin{aligned}\|\widetilde{h}^{1/2}(\mathbb{G}_H e)'\|_{L^2(\Gamma)} &\le \|(\widetilde{h}/H)^{1/2}\|_{L^\infty(\Gamma)}\|H^{1/2}(\mathbb{G}_H e)'\|_{L^2(\Gamma)}\\ &\le C_{\rm inv}^{H,2}\|(\widetilde{h}/H)^{1/2}\|_{L^\infty(\Gamma)}\|\!|\mathbb{G}_H e|\!\|.\end{aligned}$$

The combination with the best approximation property $\|\!|\mathbb{G}_H e|\!\| \le \|\!|e|\!\|$ shows that

$$\|\!|(\mathbb{1} - I_h)\mathbb{G}_H e|\!\| \le L\|\!|e|\!\|.$$

Using the Galerkin orthogonality and a Cauchy inequality, we obtain for $I_h\mathbb{G}_H e \in \mathcal{S}_h$

$$\langle\!\langle e, \mathbb{G}_H e\rangle\!\rangle = \langle\!\langle e, \mathbb{G}_H e - I_h\mathbb{G}_H e\rangle\!\rangle \le \|\!|e|\!\|\,\|\!|(\mathbb{1} - I_h)\mathbb{G}_H e|\!\| \le L\|\!|e|\!\|^2.$$

A further application of the Cauchy inequality concludes the proof:

$$\|\!|e|\!\|^2 = \langle\!\langle e, \mathbb{G}_H e\rangle\!\rangle + \langle\!\langle e, e - \mathbb{G}_H e\rangle\!\rangle \le \|\!|e|\!\|\,\big(L\|\!|e|\!\| + \|\!|(\mathbb{1} - \mathbb{G}_H)e|\!\|\big). \qquad \square$$

THEOREM 4.2. [illegible] (4.1) (4.2) [illegible]

$$\eta_M := \|\!|u_h - \mathbb{G}_H u_h|\!\|$$

[illegible]

$$\|\!|u - u_h|\!\| \le \big(\eta_M + \|\!|(\mathbb{1} - \mathbb{G}_H)u|\!\|\big)/\big(1 - L\big) \tag{4.5}$$

[illegible]

$$\eta_M \le \|\!|u - u_h|\!\| + \|\!|(\mathbb{1} - \mathbb{G}_H)u|\!\|. \tag{4.6}$$

[illegible] $\|\!|u - u_h|\!\|$ [illegible] η_M [illegible] $\|\!|(\mathbb{1} - \mathbb{G}_H)u|\!\|$ [illegible]

[illegible]. According to (3.3), the error in the energy norm is of order

$$\|\!|u - u_h|\!\| = O(h_{\rm max}^{\widehat{p}}) \quad \text{with} \quad \widehat{p} := \min\{2, m\} - 1/2 = 3/2.$$

Furthermore, for smooth u we have

$$|||(\mathbb{1}-\mathbb{G}_H)u||| = O(H_{\max}^{\widehat{q}}) \quad \text{with } \widehat{q} := \min\{3,m\} - 1/2 > 3/2.$$

Since $m > 2$, $|||(\mathbb{1}-\mathbb{G}_H)u|||$ is of higher order. The combination of Proposition 4.1 with a triangle inequality shows that

$$|||u-u_h||| \le \big(|||(\mathbb{1}-\mathbb{G}_H)u||| + |||(\mathbb{1}-\mathbb{G}_H)u_h|||\big)/\big(1-L\big).$$

This proves reliability. A simple triangle inequality,

$$\eta_M \le |||(\mathbb{1}-\mathbb{G}_H)(u-u_h)||| + |||(\mathbb{1}-\mathbb{G}_H)u||| \le |||u-u_h||| + |||(\mathbb{1}-\mathbb{G}_H)u|||,$$

shows efficiency since $(\mathbb{1}-\mathbb{G}_H)$ is an orthogonal projection with respect to the energy norm. □

Replacing the best approximation operator $\mathbb{G}_H$ by the operator $\mathcal{A}_H$, we immediately obtain the following corollary.

COROLLARY 4.3. $\eta_M \le \eta_A := |||u_h - \mathcal{A}_H u_h|||$ (4.1) (4.2) η_A □

To develop an adaptive mesh-refining algorithm, we have to provide an error estimator which can be localized. More precisely, we will estimate the nonlocal $\mathcal{H}$ norm by the (weighted) local H^1 seminorm, namely $\mu_M := \|H^{1/2}(u_h - \mathbb{G}_H u_h)'\|_{L^2(\Gamma)}$.

COROLLARY 4.4. $\mathcal{S}_0^1(\mathcal{T}_H) \subseteq \mathcal{S}_H$ $C_{\rm apx} > 0$ 3.4

$$\eta_M \le C_{\rm apx}\mu_M. \tag{4.7}$$

μ_M (4.1) (4.2)

. The proof follows immediately from the approximation property of the Galerkin projection (3.14). □

COROLLARY 4.5. $\mathcal{S}_H \subseteq \mathcal{S}^2(\mathcal{T}_h)$

$$\mu_A := \|H^{1/2}(u_h - \mathcal{A}_H u_h)'\|_{L^2(\Gamma)} \le C_5 C_{\rm inv}^{h,2}\eta_A \tag{4.8}$$

$C_5 := \max\{\|H\|_{L^\infty(\Gamma_j)}/h_j : \Gamma_j \in \mathcal{T}\}$ $C_{\rm inv}^{h,2} > 0$ $\mathcal{S}^2(\mathcal{T}_h)$ Π_H L^2 $\mathcal{P}^1(\mathcal{T}_H)$ μ_M $\mu_\Pi := \|H^{1/2}(u_h' - \Pi_H u_h')\|_{L^2(\Gamma)}$

$$\mu_\Pi \le \mu_M \le C_5 C_{\rm inv}^{h,2}\eta_M \tag{4.9}$$

(4.1) (4.2)

. The estimate (4.8) follows from the inverse estimate (3.7), and so does the second estimate in (4.9). Note that Π_H is the L^2 projection. This proves $\|u_h' - \Pi_H u_h'\|_{L^2(\gamma_j)} \le \|u_h' - (\mathbb{G}_H u_h)'\|_{L^2(\gamma_j)}$ for any $\gamma_j \in \mathcal{T}_H$. Summing the H-weighted L^2 norms over all elements $\gamma_j \in \mathcal{T}_H$, we prove (4.9). □

8. Note that the assumption $\mathcal{S}_H = \mathcal{S}_0^2(\mathcal{T}_H) \subseteq \mathcal{S}^2(\mathcal{T}_h)$ is quite weak and satisfied if $\mathcal{T}_h$ is obtained from $\mathcal{T}_H$ by some (local) refinements.

4.3. Reliability of μ_A for the L^2 projection. As proved in Corollary 4.5, there holds $\mu_A \le C_5 C_{\rm inv}^{h,2}\,\eta_A$ under weak assumptions on $\mathcal{T}_H$. To obtain the converse estimate $\eta_A \le C_6\,\mu_A$ with a constant C_6 which depends on neither the mesh-sizes nor the number of elements, one has to prove

$$|||v_h - \mathcal{A}_H v_h||| \le C_6 \|H^{1/2}(v_h - \mathcal{A}_H v_h)'\|_{L^2(\Gamma)} \quad \text{for all } v_h \in \mathcal{S}_h. \tag{4.10}$$

One then obtains the equivalence of the error estimators μ_A and η_A and, in particular, the reliability of μ_A under the assumptions (4.1)–(4.2). This is done for the L^2 projection $\mathcal{A}_H$ onto $\mathcal{S}_H$ in the following proposition.

PROPOSITION 4.6. [illegible] $\mathcal{A}_H$ [illegible] L^2 [illegible] $\mathcal{S}_H$ [illegible] 4.5, [illegible] μ_A [illegible] (4.8) [illegible]

$$C_6^{-1}\eta_A \le \mu_A \le C_5 C_{\mathrm{inv}}^{h,2}\,\eta_A. \tag{4.11}$$

[illegible] $\Gamma = \partial\Omega$. Since the L^2 projection $\mathbb{P}_H : H_0^1(\Gamma) \to \mathcal{S}^2(\mathcal{T}_H)$ satisfies

$$0 = \langle 1\,, (\mathbb{1} - \mathbb{P}_H)u_h\rangle = \langle 1\,, u_h\rangle - \langle 1\,, \mathbb{P}_H u_h\rangle = -\langle 1\,, \mathbb{P}_H u_h\rangle,$$

there holds $\mathbb{P}_H u_h \in \mathcal{S}_H$. Consequently, $\mathbb{P}_H u_h = \mathcal{A}_H u_h$ and $(\mathbb{1} - \mathcal{A}_H)u_h$ is L^2-orthogonal to $\mathcal{S}^2(\mathcal{T}_H)$. Let $\gamma_j \in \mathcal{T}_H$ be a coarse grid element with nodes $z, z' \in \mathcal{N}_H$. We define the bubble function $\beta_j(x) := \phi_{H,z}(x)\phi_{H,z'}(x)$ for $x \in \Gamma$ as the product of the corresponding hat functions. Then, $\beta_j \in \mathcal{S}^2(\mathcal{T}_H)$ is positive on the interior γ_j. With $v := (\mathbb{1} - \mathcal{A}_H)u_h$ there holds

$$0 = \langle \beta_j\,, v\rangle = \int_{\gamma_j} \beta_j v\, ds. \tag{4.12}$$

Therefore, v has at least one zero in γ_j so that Theorem 3.3 can be applied for $\mathcal{T}_H$. ∎

[illegible] $\Gamma \subsetneqq \partial\Omega$. The bubble functions vanish in all nodes $z \in \mathcal{N}_H$ and hence satisfy $\beta_j \in \mathcal{S}_H$ for all $1 \le j \le n$. Therefore, (4.12) holds and we can apply Theorem 3.3 again. ∎

[illegible] 9. The approximation argument in the proof of Proposition 4.6 works for sufficiently smooth $\widetilde{u} \in \mathcal{H}$ instead of $u_h \in \mathcal{S}_h$; e.g., u_h may be replaced by some $\widetilde{u}$ in $H_0^1(\Gamma)$ (resp., $\widetilde{H}^1(\Gamma)$).

In the case that $\Gamma \subsetneqq \partial\Omega$ is an open boundary piece, it is easy to construct operators which satisfy (4.10) since we only have to satisfy the zero boundary conditions for discrete functions at the endpoints of Γ.

[illegible] 10. Suppose that $\Gamma \subsetneqq \partial\Omega$. Let $\mathcal{A}_H$ be the pointwise interpolation operator satisfying

$$\mathcal{A}_H u_h \in \mathcal{S}^2(\mathcal{T}_H) \quad \text{and} \quad \mathcal{A}_H u_h(z) = u_h(z) \quad \text{for } z \in \{a_j, m_j, b_j\},$$

where a_j and b_j denote the nodes of γ_j and m_j denotes the element's midpoint. Note that $\mathcal{A}_H u_h$ vanishes at the endpoints of Γ. Therefore, $\mathcal{A}_H u_h \in \mathcal{S}_H$ and according to Theorem 3.3 there holds (4.10).

[illegible] 11. Suppose that $\mathcal{A}_H$ is idempotent; i.e., $\mathcal{A}_H^2 = \mathcal{A}_H$. Then, an elementary algebraic manipulation shows that (4.10) follows from the approximation estimate

$$\|\!|v_h - \mathcal{A}_H v_h|\!\| \le C_5 \|H^{1/2} v_h'\|_{L^2(\Gamma)}$$

for all $v_h \in \mathcal{S}_H$.

5. Numerical realization. As in the previous section, we write $\mathcal{S}_h = \mathcal{S}_0^1(\mathcal{T}_h)$ and $\mathcal{S}_H = \mathcal{S}_0^2(\mathcal{T}_H)$. The finer mesh $\mathcal{T}_h$ is obtained from the coarser $\mathcal{T}_H$ by uniform refinements; see section 5.5 for details. This ensures $\mathcal{S}_H = \mathcal{S}_0^2(\mathcal{T}_H) \subseteq \mathcal{S}^2(\mathcal{T}_h)$ as in Corollary 4.5.

5.1. Neumann problem and hypersingular integral equation. In the numerical experiments we consider examples where the right-hand side f in (1.1) comes from a Neumann problem

$$\Delta U = 0 \text{ in } \Omega \quad \text{and} \quad \partial U/\partial\nu = g \text{ on } \Gamma = \partial\Omega \tag{5.1}$$

with given Neumann data g on Γ (plus boundary conditions at infinity if Ω is unbounded) and the normal derivative $\partial U/\partial\nu$, where ν denotes the outer unit vector on $\partial\Omega$. This problem is equivalent to the hypersingular integral equation (1.1), where f takes the form

$$f = (\mathbb{1} - K^*)g \tag{5.2}$$

with the adjoint double-layer potential operator K^*, defined as Cauchy principal value by

$$(K^*g)(x) := -\frac{1}{\pi} \, \text{⨍}\!\!\!\!\!\!\int_\Gamma g(y) \, \frac{(y-x)\cdot\nu(x)}{|x-y|^2} \, ds_y \quad \text{for } x \in \Gamma. \tag{5.3}$$

Up to an additive constant, the exact solution of (1.1) is just the Dirichlet data $U|_\Gamma$ of U on the boundary Γ. More precisely, there holds $u = U|_\Gamma - \langle U|_\Gamma \, , 1\rangle/\langle 1 \, , 1\rangle$.

5.2. Computation of the discrete solution for $\Gamma = \partial\Omega$. The problem to solve (1.1) in the Hilbert space $H^\alpha_0(\Gamma)$ can equivalently be formulated as follows [CoS, CaS]: Given $(f, b) \in H^{\alpha-1}(\Gamma) \times \mathbb{R}$, find $(u, a) \in H^\alpha(\Gamma) \times \mathbb{R}$ such that

$$Wu + a = f \quad \text{and} \quad \langle 1 \, , u\rangle = b. \tag{5.4}$$

The corresponding discrete formulation reads as follows: Find $(u_h, a_h) \in \mathcal{S}^1(\mathcal{T}) \times \mathbb{R}$ such that

$$\langle Wu_h + a_h \, , v_h\rangle = \langle f \, , v_h\rangle \quad \text{for all } v_h \in \mathcal{S}^1(\mathcal{T}_h) \quad \text{and} \quad \langle 1 \, , u_h\rangle = b. \tag{5.5}$$

Notice that $\langle 1 \, , u - u_h\rangle = 0$, whence $u - u_h \in H^\alpha_0(\Gamma)$, and $\langle 1 \, , f - a_h\rangle = \langle 1 \, , Wu_h\rangle = 0$; i.e., $a_h = \langle f \, , 1\rangle/\langle 1 \, , 1\rangle$ and $f - a_h \in H^{\alpha-1}_0(\Gamma)$. In particular, for $b = 0$ and $f \in H^{\alpha-1}_0(\Gamma)$, the solution of (5.4) (resp., (5.5)) is just the solution of (1.1) (resp., (3.1)). The benefit of the generalized formulation is that we are allowed to consider the full discrete space $\mathcal{S}^1(\mathcal{T}_h)$ instead of $\mathcal{S}^1_0(\mathcal{T}_h)$. This simplifies the implementation since one can use the usual nodal hat functions instead of a basis of $\mathcal{S}^1_0(\mathcal{T}_h)$.

Let $\mathcal{N}_h$ denote the set of all nodes of $\mathcal{T}_h$ and, with $n := |\mathcal{N}_h|$, define the (symmetric) stiffness matrix $\mathbf{A}$ by

$$\mathbf{A}_{jk} = \langle\!\langle \phi_j \, , \phi_k \rangle\!\rangle \quad \text{for } 1 \le j, k \le n, \tag{5.6}$$

with $\phi_j, \phi_k \in \mathcal{S}^1(\mathcal{T}_h)$ denoting the hat functions corresponding to the jth and kth node, and

$$\mathbf{A}_{j,n+1} = \mathbf{A}_{n+1,j} = \langle 1 \, , \phi_j\rangle, \quad \mathbf{A}_{n+1,n+1} = 0. \tag{5.7}$$

With the right-hand side

$$\mathbf{b}_j = \langle f \, , \phi_j\rangle \quad \text{for } 1 \le j \le n, \tag{5.8}$$

the discrete solution u_h of (3.1) has the basis representation

$$u_h = \sum_{j=1}^{n} \mathbf{x}_j \phi_j, \tag{5.9}$$

where the coefficient vector $\mathbf{x} \in \mathbb{R}^{n+1}$ solves the linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ with $\mathbf{b}_{n+1} := 0$. Moreover, there holds $x_{n+1} = a_h$ with a_h from (5.5).

[illegible] 12. (a) The hypersingular integral operator (1.2) and the single-layer potential

$$(Vu)(x) := -\frac{1}{\pi} \int_\Gamma u(y) \log|x-y| \, ds_y \tag{5.10}$$

are linked through Nédélec's equation (recall $(\cdot)' = \partial/\partial s$)

$$\langle Wu \, , v \rangle = \langle Vu' \, , v' \rangle \quad \text{for all } u, v \in \mathcal{H}. \tag{5.11}$$

(b) According to (a), the entries of $\mathbf{A}$ in (5.6) can be computed analytically using (5.11) and exact integration for the single-layer potential; see [Ma].

5.3. Computation of the discrete solution for $\Gamma \subsetneqq \partial\Omega$. Let $\mathcal{K}_h := \mathcal{N}_h \backslash \partial\Gamma$ denote the set of all interior nodes of $\mathcal{T}$. Then, the implementation for an open curve is less involved since the hat functions ϕ_j with $j \in \mathcal{K}_h$ form the nodal basis of $\mathcal{S}^1_0(\mathcal{T}_h)$. With $n := |\mathcal{K}_h|$, we compute $\mathbf{A}$ and $\mathbf{b}$ as in (5.6) and (5.8), respectively. Then, the discrete solution u_h is represented as in (5.9) with $\mathbf{x} \in \mathbb{R}^n$ satisfying $\mathbf{A}\mathbf{x} = \mathbf{b}$.

[illegible] 13. For implementational reasons, the matrix $\mathbf{A}$ and the right-hand side $\mathbf{b}$ are computed for all hat functions, i.e., $\mathcal{N}_h$, and the resulting linear system is solved only on a subblock corresponding to $\mathcal{K}_h$.

5.4. Computation of the right-hand side for academic examples. If the exact solution $u \in \mathcal{H}$ is known, the entries for the right-hand side vector (cf. (5.8)), can easily be obtained by use of Remark 12: According to (5.11), there holds

$$\mathbf{b}_j = \langle f \, , \phi_j \rangle = \langle Wu \, , \phi_j \rangle = \langle u' \, , V\phi_j' \rangle.$$

Notice that the integrand $b(x) = u'(x)(V\phi_j')(x)$ may have logarithmic singularities. For the numerical experiments, we used the `quad` function implemented in MATLAB, which is based on an h-adaptive Simpson formula and Romberg extrapolation.

5.5. Adaptive algorithm. All mesh-refinements are performed with the following adaptive algorithm from [CP] based on the refinement indicators $\mu_{M,j}$, $\mu_{A,j}$, and $\mu_{\Pi,j}$ defined as follows: Given the coarse mesh $\mathcal{T}_H = \{\gamma_1, \dots, \gamma_N\}$ and the L^2 projections $\Pi^1_H : \mathcal{S}_h \to \mathcal{S}^1_0(\mathcal{T}_H)$ and $\Pi^2_H : \mathcal{S}_h \to \mathcal{S}^2_0(\mathcal{T}_H)$, we define, for $j = 1, \dots, N$,

$$\mu_{M,j} := H_j^{1/2} \| u_h' - (\mathbb{G}_H u_h)' \|_{L^2(\gamma_j)}, \tag{5.12}$$

$$\mu_{A,j} := H_j^{1/2} \| u_h' - (\Pi^2_H u_h)' \|_{L^2(\gamma_j)}, \tag{5.13}$$

$$\mu_{\Pi,j} := H_j^{1/2} \| u_h' - \Pi^1_H(u_h') \|_{L^2(\gamma_j)}, \tag{5.14}$$

i.e., $\mu_M = \big(\sum_{j=1}^N \mu_{M,j}^2\big)^{1/2}$, $\mu_A = \big(\sum_{j=1}^N \mu_{A,j}^2\big)^{1/2}$, and $\mu_\Pi = \big(\sum_{j=1}^N \mu_{\Pi,j}^2\big)^{1/2}$, respectively.

ALGORITHM 5.1. [illegible] $\mathcal{T}_H^{(0)}$ $k = 0$ $\ell \in \mathbb{N}_{\geq 2}$ $0 \leq \theta \leq 1$

(i) $\mathcal{T}_h^{(k)} = \{\Gamma_1, \dots, \Gamma_n\}$ $\mathcal{T}_H^{(k)} = \{\gamma_1, \dots, \gamma_N\}$
$\gamma_j \in \mathcal{T}_H^{(k)}$ ℓ
(ii) $u_h^{(k)}$ $\mathcal{T}_h^{(k)}$
(iii) η_M η_A $\mu_{M,j}$ $\mu_{A,j}$ $\mu_{\Pi,j}$
(iv) γ_j $\mu_{M,j} \geq \theta \max\{\mu_{M,1}, \dots, \mu_{M,N}\}$ $\mu_{M,j}$ $\mu_{A,j}$ $\mu_{\Pi,j}$
(v) $\gamma_j \in \mathcal{T}_H^{(k)}$ $\mathcal{T}_H^{(k+1)}$ (i)

14. The choice of $\theta = 0$ in Algorithm 5.1 leads to uniform mesh-refinement, whereas we used $\theta = 1/2$ for adaptive mesh-refinement in the subsequent numerical examples. The parameter $\ell \in \mathbb{N}_{\geq 2}$ is chosen between 2 and 4 and is empirically discussed later.

15. Since the local mesh-ratio $\kappa(\mathcal{T}_h^{(h)})$ defined in (3.9) enters the constant $C_{\rm apx}$ and thus affects the reliability of our estimators, we mark further elements for refinement to guarantee $\kappa(\mathcal{T}_H^{(k)}) \leq 2\kappa(\mathcal{T}_H^{(0)})$. Note that there holds $\kappa(\mathcal{T}_h^{(k)}) = \kappa(\mathcal{T}_H^{(k)})$.

5.6. Computation of the error estimators. Let $\phi_1, \dots, \phi_n$ be a basis of $\mathcal{S}_h$ and $\Phi_1, \dots, \Phi_N$ be a basis of $\mathcal{S}_H$ and define the matrices $\mathbf{A} \in \mathbb{R}^{n\times n}_{sym}$, $\mathbf{B} \in \mathbb{R}^{N\times N}_{sym}$, and $\mathbf{C} \in \mathbb{R}^{N\times n}$ by

$$\mathbf{A}_{jk} := \langle\!\langle \phi_j, \phi_k \rangle\!\rangle, \quad \mathbf{B}_{jk} := \langle\!\langle \Phi_j, \Phi_k \rangle\!\rangle, \quad \text{and} \quad \mathbf{C}_{jk} := \langle\!\langle \Phi_j, \phi_k \rangle\!\rangle. \tag{5.15}$$

Let $\mathbf{x} \in \mathbb{R}^n$ be the coefficient vector of $u_h \in \mathcal{S}_h$ and $\mathbf{y} \in \mathbb{R}^N$ the coefficient vector of $\mathbb{G}_H u_h \in \mathcal{S}_H$. Note that the vector $\mathbf{y}$ is the solution of the linear system $\mathbf{By} = \mathbf{Cx}$. Then, Galerkin orthogonality yields

$$\eta_M^2 = |\!|\!| u_h - \mathbb{G}_H u_h |\!|\!|^2 = |\!|\!| u_h |\!|\!|^2 - |\!|\!| \mathbb{G}_H u_h |\!|\!|^2 = \mathbf{x}\cdot\mathbf{Ax} - \mathbf{y}\cdot\mathbf{By} = \mathbf{x}\cdot(\mathbf{Ax} - \mathbf{C}^T\mathbf{y}).$$

The computation of the error estimator η_A involves the L^2 mass matrices $\widehat{\mathbf{B}} \in \mathbb{R}^{N\times N}_{sym}$ and $\widehat{\mathbf{C}} \in \mathbb{R}^{N\times n}$ defined by

$$\widehat{\mathbf{B}}_{jk} := \langle \Phi_j, \Phi_k \rangle \quad \text{and} \quad \widehat{\mathbf{C}}_{jk} := \langle \Phi_j, \phi_k \rangle. \tag{5.16}$$

If $\mathcal{A}_H$ denotes the L^2 projection onto $\mathcal{S}_H$, the coefficient vector $\widehat{\mathbf{y}} \in \mathbb{R}^N$ of $\mathcal{A}_H u_h$ is the solution of the linear system $\widehat{\mathbf{B}}\mathbf{y} = \widehat{\mathbf{C}}\mathbf{x}$. Then, η_A reads as

$$\eta_A^2 = |\!|\!| u_h - \mathcal{A}_H u_h |\!|\!|^2 = \mathbf{x}\cdot\mathbf{Ax} - \widehat{\mathbf{y}}\cdot(2\,\mathbf{Cx} - \mathbf{B}\widehat{\mathbf{y}}).$$

16. (a) In case $\Gamma = \partial\Omega$, it is convenient to extend $\mathbf{A}$, $\mathbf{B}$, and $\widehat{\mathbf{B}}$ by a side-constraint analogously to (5.7) and then to consider a basis of $\mathcal{S}^1(\mathcal{T}_h)$ and $\mathcal{S}^2(\mathcal{T}_h)$ instead of $\mathcal{S}_h$ and $\mathcal{S}_H$, respectively.

(b) The entries of $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are computed analytically by use of Remark 12 for ϕ_j being a hat function in $\mathcal{S}^1(\mathcal{T}_h)$ and Φ_j being either a hat function in $\mathcal{S}^1(\mathcal{T}_H)$ or a bubble function in $\mathcal{S}^2(\mathcal{T}_H)$, i.e., the product of two hat functions with overlapping support.

(c) The entries of $\widehat{\mathbf{B}}$ are computed analytically by transforming the arising integrals to the reference element $[0, 1]$, where the hat functions are given by t and $1 - t$ and the corresponding bubble is $t(1 - t)$.

(d) The entries of $\widehat{\mathbf{C}}$ are easily computed since a fine grid element $\Gamma_j \in \mathcal{T}_h$ lies in precisely one coarse grid element $\gamma_k \in \mathcal{T}_H$. If we use the arclength parametrization of γ_k, Γ_j then corresponds to a subinterval $[a,b] \subseteq [0, H_k]$.

(e) The error estimators μ_M and μ_A are computed $\mathcal{T}_H$-elementwise as indicated in (5.12).

6. Numerical experiments. This section reports on four numerical experiments to study the accuracy of the introduced error estimators and the performance of the proposed adaptive strategy. All experiments have been conducted using MATLAB. All examples are academic in the sense that the exact solution is always known and we can compare the convergence of the error $E_h = |\!|\!| u - u_h |\!|\!|$ with the convergence of the error estimators. All experiments were performed with the parameter $\ell = 2, 3, 4$ in Algorithm 5.1. Since the numerical outcome looks similar, we restrict the presentation to the simplest case $\ell = 2$. The right-hand side for the Galerkin method is always computed as explained in section 5.4.

The first two examples are related to Neumann problems; cf. section 5.1. The solution for the example in section 6.1 is smooth and satisfies the regularity assumptions of section 4 so that the higher-order term in the error estimates satisfies $|\!|\!| u - \mathbb{G}_H u |\!|\!| = \mathcal{O}(h^{5/2})$. The solution for the example in section 6.2 corresponds to a classical finite element test problem on the (rotated) L-shaped domain which lacks H^2-regularity. For the generic corner singularity $U(r,\varphi) = r^\alpha \cos(\alpha\varphi)$ with $\alpha = 2/3$ for the reentrant corner $3\pi/2$, the corresponding Neumann data are (piecewise) smooth. In the example in section 6.3, we therefore consider the same example with $\alpha = 3/7$. Then, the Neumann data are singular at the reentrant corner and uniform mesh-refinement leads to a suboptimal convergence rate for the numerical solution of the hypersingular integral equation. This can be overcome by use of adaptive mesh-refinement. Finally, the example in section 6.4 considers a slit problem on the slit $\Gamma = [-1,1] \times \{0\}$. The right-hand side simplifies since the adjoint double layer potential vanishes; i.e., we do not have to deal with further quadrature errors to compute the right-hand side. The exact solution lies in $\widetilde{H}^{1-\varepsilon}(\Gamma)$ for any $\varepsilon > 0$.

6.1. Neumann problem with smooth solution. We consider the Neumann problem (5.1) on the halved unit square $\Omega = [0, 1/2]^2$ with exact solution

$$U(x,y) = \sinh(2\pi\, x)\, \cos(2\pi\, y) \tag{6.1}$$

and solve the corresponding hypersingular integral equation $Wu = f$ with right-hand side $f = (1 - K^*)g$ and $g = \partial U/\partial\nu$. Note that $\int_\Gamma U(x)\, ds_x = 0$. Therefore, the exact solution u of the hypersingular integral equation is just the trace on the boundary $u = U|_\Gamma$. The computation of the right-hand side for the Galerkin method is performed as shown in section 5.4 with the arclength derivative $u' = \nabla U(x) \cdot \tau(x)$ for the tangential vector $\tau(x)$ on Γ. Figure 1 shows initial mesh and the corresponding discrete solution u_h (for $\ell = 2$ in Algorithm 5.1) as a function of the arclength parameter $0 \le s \le 2$, where $s = 0$ corresponds to the point $(0,0)$ and counter-clockwise parametrization of the boundary $\Gamma = \partial\Omega$. The exact solution u is shown for comparison. Figure 2 shows the error

$$E_h := |\!|\!| u - u_h |\!|\!| = (|\!|\!| u |\!|\!|^2 - |\!|\!| u_h |\!|\!|^2)^{1/2},$$

where $|\!|\!| u |\!|\!| = 258.4278864$ (obtained from Aitkin's Δ^2 extrapolation of the sequence on uniformly refined meshes) as well as the introduced error estimators for uniform mesh-refinement. The values are plotted over the number $n^{(k)}$ of elements in the

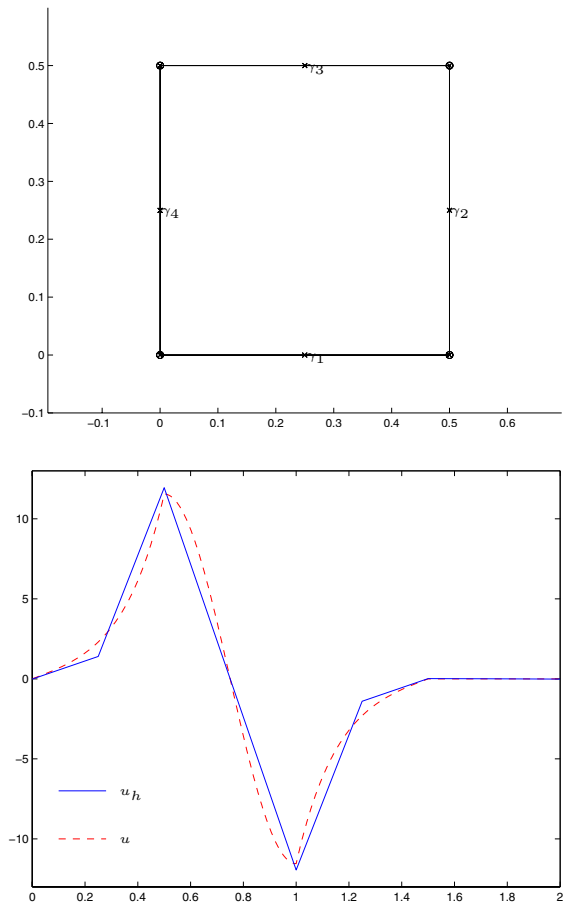


FIG. 1. *Initial coarse mesh* $\mathcal{T}_H^{(0)}$ *in the Neumann problem in section* 6.1 *with* $N = 4$ *elements (nodes indicated by* o*) (top) and the corresponding fine mesh* $\mathcal{T}_h^{(0)}$ *for* $\ell = 2$ *in Algorithm* 5.1*; i.e.,* $n = 8$ *(nodes indicated by* x*). The related discrete solution* u_h *on* $\mathcal{T}_h^{(0)}$ *(bottom) is plotted over the arclength* $s = 0, \dots, 2$ *of* Γ*. The exact solution* u *(i.e., the trace of* U *from* (6.1)*) is shown for comparison (bottom).*

kth fine grid $\mathcal{T}_h^{(k)}$. As is to be expected, uniform mesh-refinement leads to optimal experimental convergence rate

$$\kappa^{(k)}(E_h) := \log(E_h^{(k-1)}/E_h^{(k)}) \,/\, \log(n^{(k)}/n^{(k-1)}). \tag{6.2}$$

Here, $E_h^{(j)}$ denotes the error corresponding to the jth mesh (i.e., on the fine mesh in the jth step of Algorithm 5.1) with $n^{(j)}$ elements. The experimental convergence rate is visualized in Figure 2 through the slope of the interpolated values plotted in double-logarithmic scale. From the regularity of the exact solution $u = U|_\Gamma$ one expects convergence rate 3/2 for the error E_h and 5/2 for the error $E_H := |||u - \mathbb{G}_H u|||$ in a Galerkin scheme with $\mathcal{S}_0^2$ ansatz functions (cf. section 3.2), and this is in fact observed in Figure 2. Note that the assumptions of Theorem 4.2 are satisfied and η_M is an upper bound for the error E_h up to terms of higher order $\mathcal{O}(h^{5/2})$. We

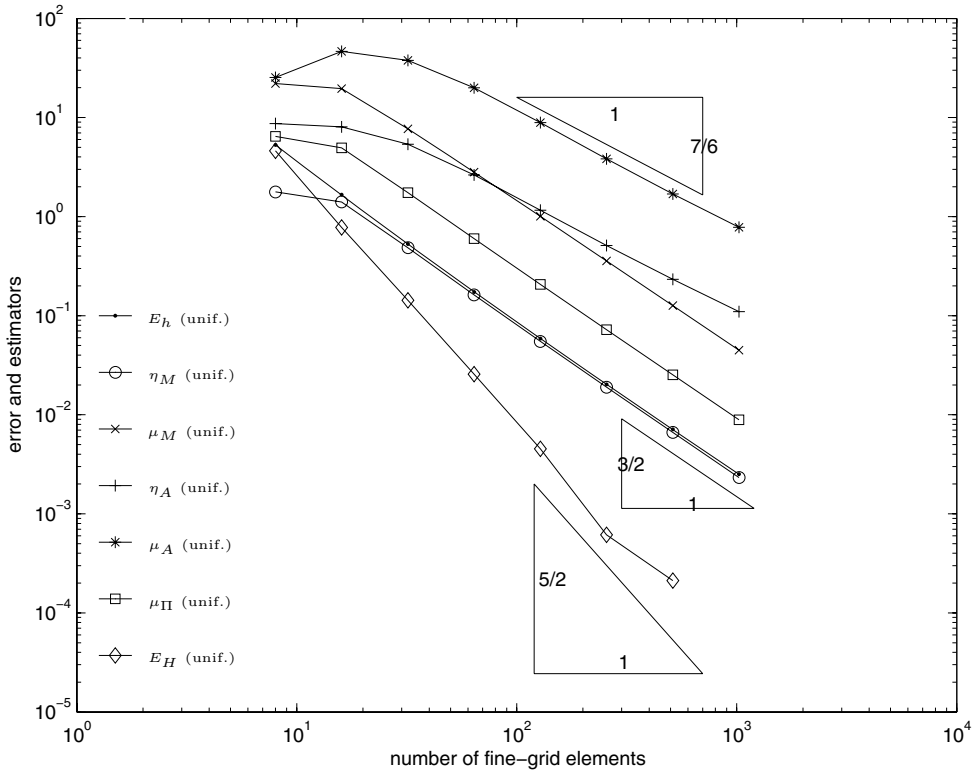


FIG. 2. *Error* $E_h = \|(\mathbb{1} - \mathbb{G}_h)u\|$ *and error estimators* η_M, μ_M, η_A, μ_A, *and* μ_Π *for* $\ell = 2$ *and uniform mesh-refinement in the Neumann problem in section* 6.1. *For comparison, we further show the* S_H*-error* $E_H = \|(\mathbb{1} - \mathbb{G}_H)u\|$ *which is of higher order: As expected from section* 3.2, *we observe* $\kappa(E_h) = 3/2$ *and* $\kappa(E_H) = 5/2$. *The estimators* η_A *and* μ_A *lack the efficiency, showing an experimental convergence rate less than* 3/2. *On the other hand, we observe reliability of* μ_Π. *Note that* η_M *is a very accurate estimator for* E_h *in the sense that both curves almost coincide.*

see that η_M is a very accurate estimate for E_h: The curves for E_h and η_M almost coincide. (See also Table 1, where we provide some numbers for $\ell = 1, \dots, 8$.) The error estimator μ_M is expected and observed to be reliable (for uniformly refined meshes) and efficient: The curve of μ_M is parallel to that of E_h and η_M. The same holds for the error estimator μ_Π which was only proved to be efficient in Corollary 4.5. The error estimators μ_A and η_A show reliability but obviously lack the efficiency: The curves for μ_A and η_A are parallel (as expected from Proposition 4.6) with slope 7/6. The failing efficiency estimate probably results from the lack of H^1 stability of the L^2 projection $\mathcal{A}_H : H^1(\Gamma) \to \mathcal{S}^2_0(\mathcal{T}) \subseteq H^1(\Gamma)$.

Figure 3 visualizes the error E_h and the error estimator μ_Π for uniform and adaptive mesh-refinement. As for uniform mesh-refinement, we obtain the experimental reliability of μ_Π: The curve of E_h and the corresponding curve of μ_Π are always parallel. Even for μ_A-adaptive mesh-refinement, where the error curve shows some zigzag behavior, this is imitated by the curve of μ_Π. All mesh-refining strategies recover the optimal convergence rate $\kappa(E_h) = 3/2$. Although adaptive mesh-refinement is not necessary for this example, the absolute values of E_h recommend the use of μ_M- or μ_Π-adaptive mesh-refinement. This is caused by the vanishing solution $u(x, y)$ for

TABLE 1

Index $\varrho = \|u-u_h\|/\eta_M$ for the Neumann problem in section 6.1 *with respect to the number $|\mathcal{T}_H|$ of coarse grid elements and the refinement parameter ℓ in Algorithm* 5.1. *One expects that ϱ tends to* 1 *for $\#\mathcal{T}_H, \ell \to \infty$ (cf. Theorem* 4.2*): In this case, the higher-order term $\|u - \mathbb{G}_H u\|$ tends to zero, and the constant $1/(1-L)$ tends to* 1. *Note that we used Aitkin's extrapolation rowwise to compute $\|u\|$ since the extrapolated value is affected by the quadrature errors to compute the right-hand side; cf. section* 5.4.

	$\ell = |\mathcal{T}_h|/|\mathcal{T}_H|$						
$|\mathcal{T}_H|$	2	3	4	5	6	7	8
4	2.98e+00	6.54e-01	3.75e-01	2.53e-01	1.85e-01	1.43e-01	1.15e-01
8	1.18e+00	8.49e-01	6.06e-01	4.46e-01	3.41e-01	2.70e-01	2.20e-01
16	1.09e+00	9.50e-01	8.08e-01	6.70e-01	5.52e-01	4.58e-01	3.84e-01
32	1.07e+00	9.96e-01	9.37e-01	8.65e-01	7.84e-01	7.02e-01	6.24e-01
64	1.07e+00	1.01e+00	9.87e-01	9.60e-01	9.28e-01	8.88e-01	8.43e-01
128	1.07e+00	1.02e+00	1.00e+00	9.91e-01	9.91e-01	9.68e-01	9.51e-01
256	1.07e+00	1.02e+00	1.00e+00	9.85e-01	1.04e+00	1.09e+00	9.91e-01
512	1.06e+00	1.03e+00	1.00e+00	9.70e-01	1.09e+00	1.20e+00	1.00e+00

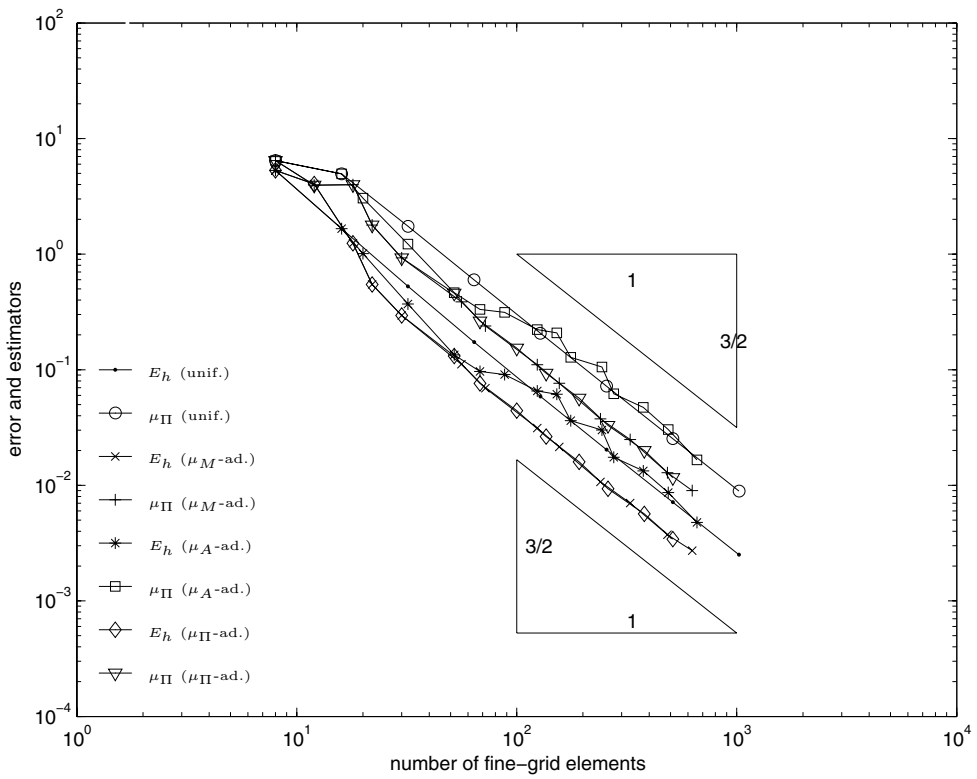


FIG. 3. *Error $E_h = \|(\mathbf{1} - \mathbb{G}_h)u\|$ and error estimator μ_Π in the example in section* 6.1 *for uniform (unif.) and adaptive (ad.) mesh-refinement and $\ell = 2$. Besides the efficiency of μ_Π proven in Corollary* 4.5 *one observes reliability of μ_Π: In all cases, the curves of E_h and μ_Π are parallel—even for μ_A-adaptive mesh-refinement.*

$x = 0$. We do not need any boundary elements there. The μ_A-adaptive mesh-refinement leads to absolute values of E_h which are even worse than the uniform mesh-refinement.

Some adaptively generated meshes are shown in Figure 4. For μ_M- and μ_Π-adaptive mesh-refinement, we see a strong refinement around the arclength parameter $s = 1/2$ and $s = 1$, where the exact solution has strong gradients; cf. Figure 1.

Finally, Figure 5 provides a comparison of the different estimators in dependence on the mesh-refining strategy. To make the subplots comparable we always print the outcome for uniform mesh-refinement. For all mesh-refining strategies, we see that η_A and μ_A are equivalent and reliable but lack the efficiency. The convergence rates for η_A and μ_A for adaptive mesh-refinement become even worse than for uniform mesh-refinement. On the other hand, we prove experimentally the reliability of μ_M (and μ_Π) even for non–quasi-uniform meshes.

6.2. Neumann problem with slightly singular solution. For a fixed parameter $\alpha > 0$, we consider the Neumann problem (5.1) on the L-shaped domain Ω shown in Figure 6 with exact solution

$$U(x) = \mathrm{Re}(x^\alpha) = r^\alpha \cos(\alpha\varphi) \quad \text{in polar coordinates} \quad x = r\,(\cos\varphi, \sin\varphi). \tag{6.3}$$

With the Dirichlet data $U|_\Gamma$, the exact solution u of (1.1) in $H^1_0(\partial\Omega)$ is given by $u = U|_\Gamma - \langle U|_\Gamma\,, 1\rangle/\langle 1\,, 1\rangle$. Numerical quadrature yields $\langle U|_\Gamma\,, 1\rangle/\langle 1\,, 1\rangle = 0.2095275$. The exact solution u is plotted over the arclength in Figure 6 for comparison with the discrete Galerkin solution. As in the previous section, the right-hand side in the Galerkin scheme is computed by use of the exact solution; cf. section 5.4. The gradient of $U(x)$ reads in polar coordinates as

$$\nabla U(x) = \alpha r^{\alpha-1} \begin{pmatrix} \cos(\varphi)\cos(\alpha\varphi) + \sin(\varphi)\sin(\alpha\varphi) \\ \sin(\varphi)\cos(\alpha\varphi) - \cos(\varphi)\sin(\alpha\varphi) \end{pmatrix}. \tag{6.4}$$

For the numerical experiment we choose $\alpha = 2/3$, which corresponds to the typical corner singularity at a reentrant corner with angle $3\pi/2$. The Neumann problem then leads to $U \notin H^2(\Omega)$. Nevertheless, the Dirichlet data are smooth as can be seen in Figure 6. Thus, we observe optimal experimental convergence rate 3/2 [in terms of number of elements] even for uniform mesh-refinement. Figure 7 visualizes the error E_h and the error estimators η_M and μ_Π for uniform and adaptive mesh-refinement. As in the example in section 6.1, we observe that both error estimators are reliable and efficient. μ_M- and μ_Π-adaptive mesh-refinement lead to optimal experimental convergence rate $\kappa(E_h) = 3/2$ and even the absolute values of E_h are superior to the values obtained from uniform mesh-refinement. The μ_A-adaptive mesh-refinement leads to a curve with absolute values E_h even worse than for uniform mesh-refinement.

Some adaptively generated meshes are shown in Figure 8 for comparison. The meshes generated by μ_M-related and μ_Π-related mesh-refinement are very similar. This can also be observed from the plot in Figure 7, where the curves corresponding to μ_M and μ_Π, respectively, almost coincide.

Independent of the mesh-refining strategy, the estimator η_M provides an accurate estimate for the error E_h: The curves for E_h and η_M, which are related to the same mesh-refinement, coincide.

6.3. Problem with singular solution. We consider the same example as in section 6.2 but now with $\alpha = 3/7$, which leads to a singular exact solution u. Figure 9 shows the discrete solution u_h on the initial mesh $\mathcal{T}_h^{(0)}$ and, for comparison, the Dirichlet data $U|_\Gamma$ (up to an additive constant). Note that we subtracted the constant $\langle U|_\Gamma\,, 1\rangle/\langle 1\,, 1\rangle = 0.4205278$ from the Dirichlet data to obtain the unique solution u

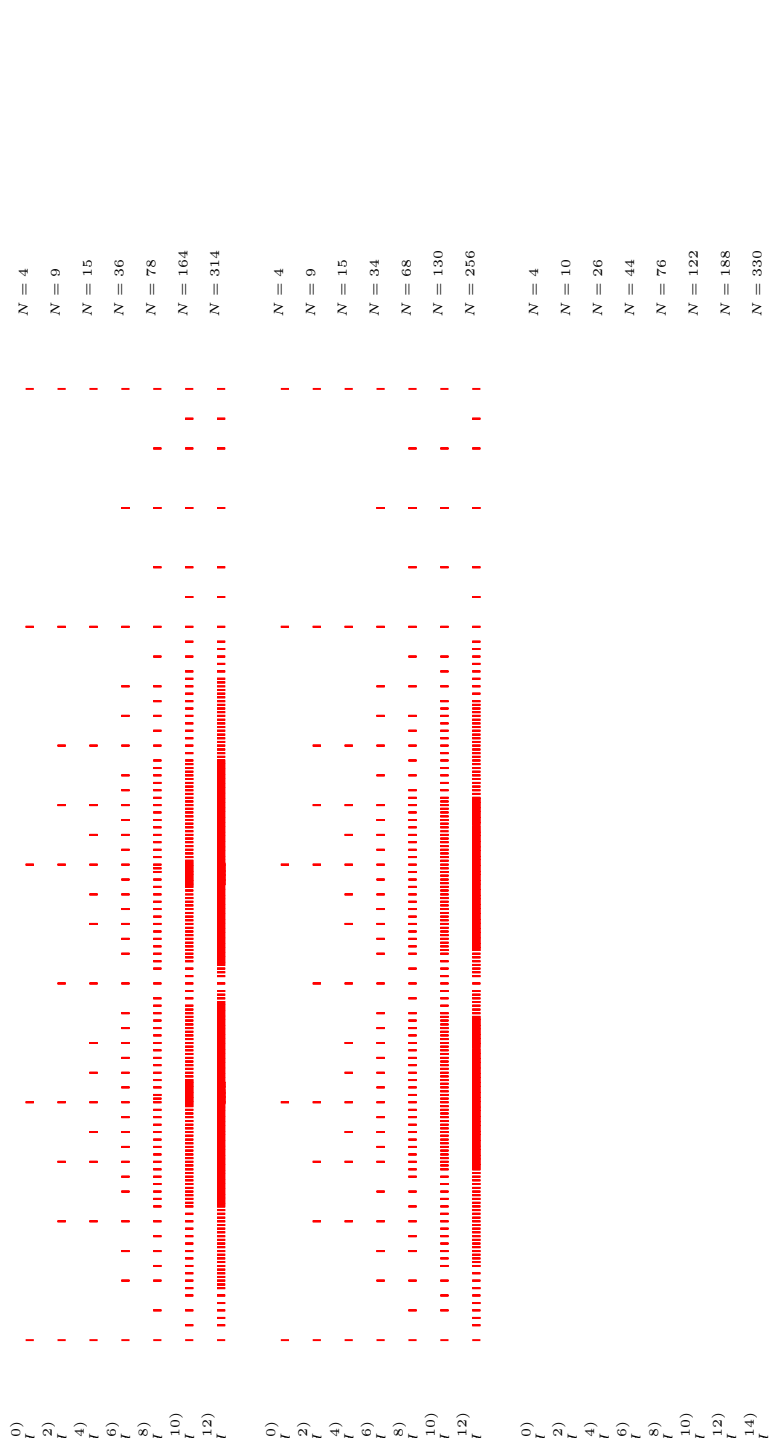


FIG. 4. *Adaptive mesh-refinement in the Poisson problem in section* 6.1 *with* $\ell = 2$ *related to the error estimator* μ_M *(top),* μ_Π *(middle), and* μ_A *(bottom), respectively.*

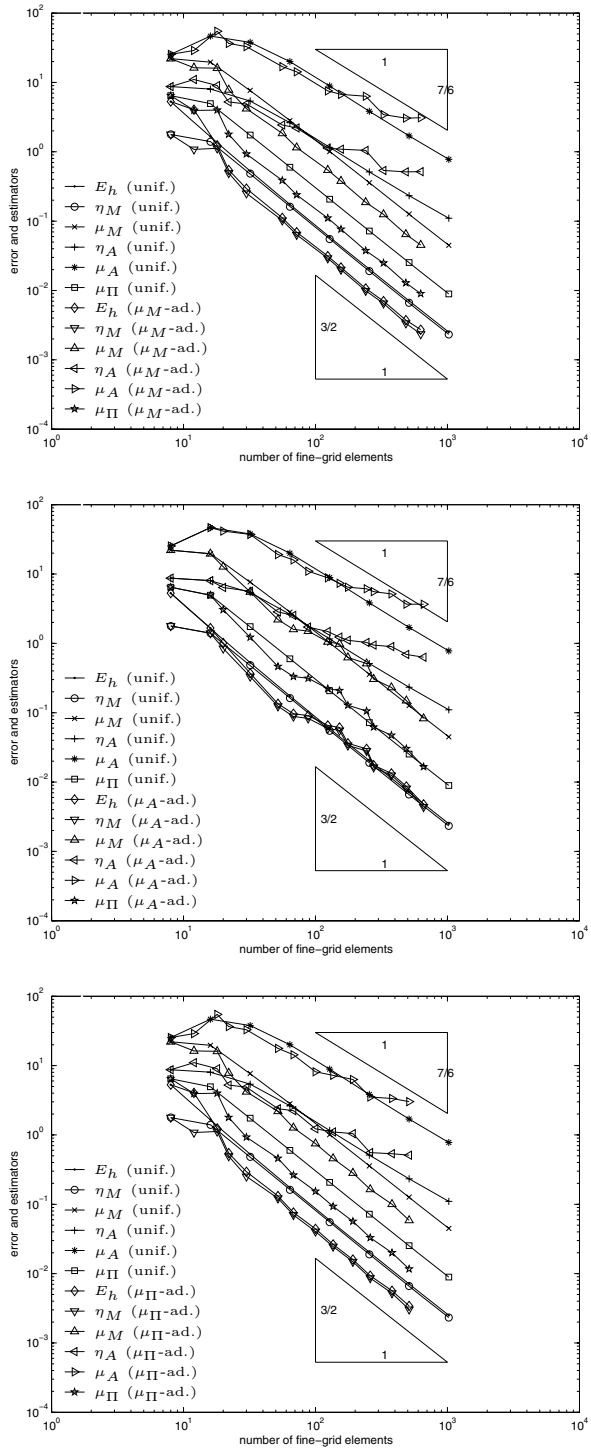


FIG. 5. *Error* $E_h = \|(\mathbf{1} - \mathbb{G}_h)u\|$ *and error estimator* μ_Π *in the example in section* 6.1 *for* $\ell = 2$ *and uniform (unif.) (resp., different adaptive (ad.)) mesh-refining strategies. In any case, we observe that the error estimators* η_M, μ_M, *and* μ_Π *are reliable and efficient. All mesh-refining strategies lead to optimal convergence rate* $\kappa(E_h) = 3/2$, *and* η_M *and* E_h *almost coincide. The error estimators* η_A *and* μ_A *lack the efficiency.*

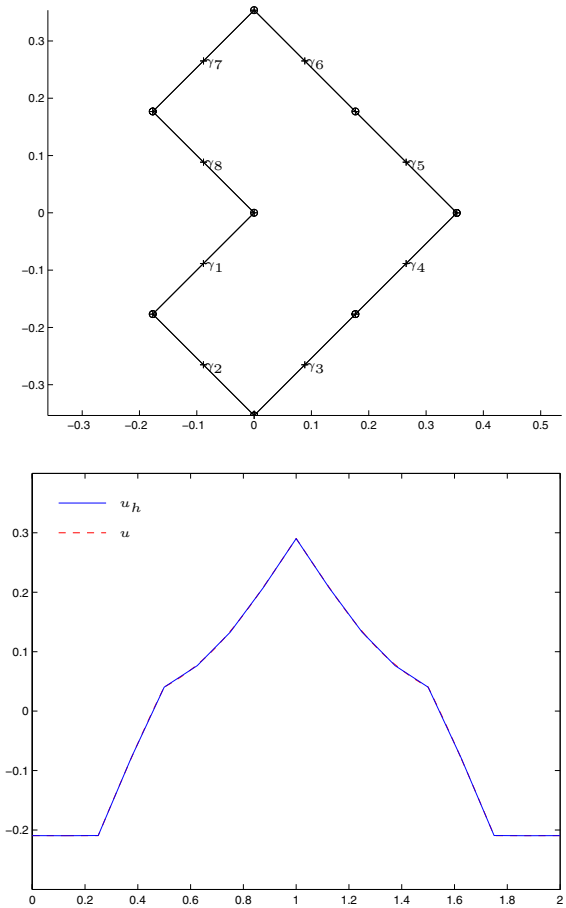


FIG. 6. *Initial coarse mesh* $\mathcal{T}_H^{(0)}$ *in the example in section* 6.2 *with* $N = 8$ *elements (nodes indicated by* ∘*) (top) and the corresponding fine mesh* $\mathcal{T}_h^{(0)}$ *for* $\ell = 2$ *in Algorithm* 5.1*, i.e.,* $n = 16$ *(nodes indicated by* +*). The related discrete solution* u_h *on* $\mathcal{T}_h^{(0)}$ *(bottom) is plotted over the arclength* $s = 0, \dots, 2$ *of* Γ*. The exact solution* u *is shown for comparison (bottom). Note that, due to the resolution of the bottom figure, there is almost no visible difference between* u *and* u_h*.*

of (1.1) in $H_0^1(\partial\Omega)$. As can be seen, the exact solution now yields a singularity at the reentrant corner at arclength $s = 0$ and $s = 2$, respectively.

Uniform mesh-refinement leads to a suboptimal experimental convergence rate of 3/7 for the error E_h. Figure 10 visualizes the error E_h as well as the error estimators η_M and μ_Π for uniform, and μ_M-, μ_A-, and μ_Π-adaptive, mesh-refinement. The μ_M- and μ_A-adaptive strategies recover the optimal convergence rate 3/2, which is experimentally obtained from the very start; i.e., there is no preasymptotic behavior as is normally observed for adaptive methods. The adaptively generated meshes, visualized in Figure 11, look very similar for both strategies, and the related E_h-curves coincide. This is very different for the μ_A-adaptive mesh-refinement. After a small preasymptotic phase (for $n \leq 40$) and compared with the uniform mesh-refinement, we observe an improvement of the experimental convergence rate. However, the μ_A-adaptive strategy does not seem to recover the optimal convergence rate. The μ_A-adaptively

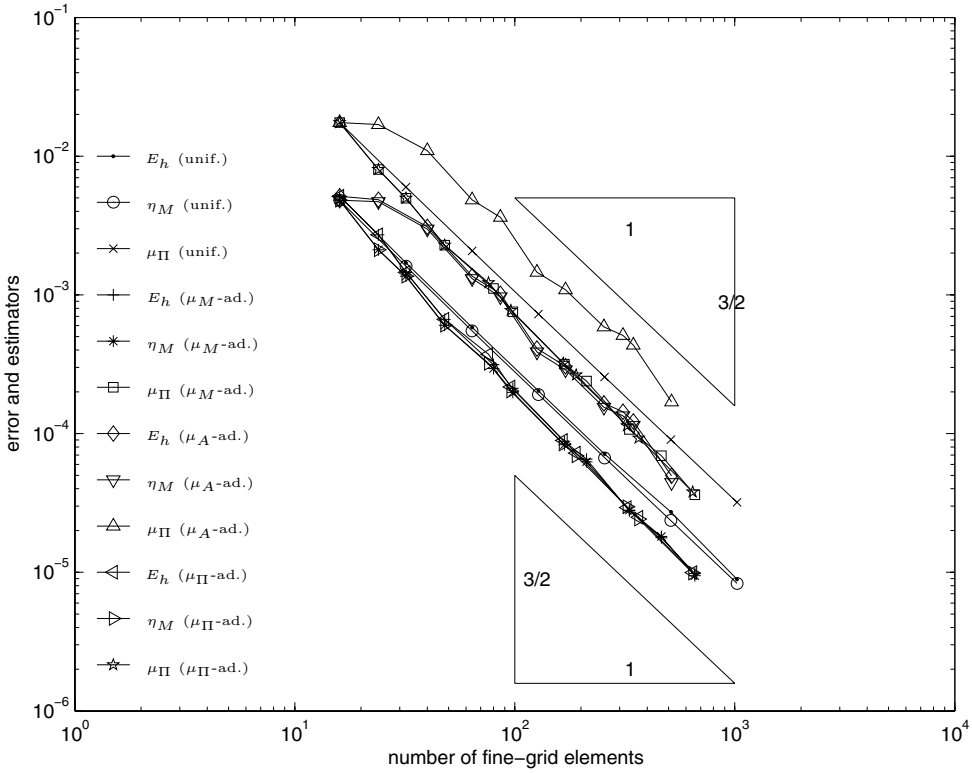


FIG. 7. *Error $E_h = \|(\mathbf{1} - \mathbb{G}_h)u\|$ and error estimator μ_Π in the example in section* 6.2 *for uniform (unif.) and adaptive (ad.) mesh-refinement and $\ell = 2$. Besides the efficiency of μ_Π proven in Corollary* 4.5 *one observes reliability of μ_Π. The error estimator η_M estimates E_h very accurately: The values almost coincide independent of the mesh-refining strategy. Whereas μ_M- and μ_Π-adaptive mesh-refinement lead to improved absolute values for the error E_h, the E_h-curve for μ_A-adaptive mesh-refinement is above the curve corresponding to uniform mesh-refinement; i.e., the absolute value of E_h for a fixed number of unknowns is even worse than for uniform mesh-refinement.*

generated meshes show some further mesh-refinement where the exact solution u has some local maxima (cf. Figures 9 and 11), which is not necessary when compared with the optimal results for μ_M- and μ_Π-adaptive mesh-refinement.

The error estimation by η_M is not as good as that observed in the preceding examples. This is probably due to the fact that the term $\|u - \mathbb{G}_H u\|$ now is a higher-order term.

6.4. Slit problem with exact solution. Our last example is concerned with the hypersingular integral equation (1.1) with constant right-hand side $f \equiv 1$ on the slit $\Gamma = [-1, 1] \times \{0\}$. The exact solution u is given by

$$u(x, 0) = \sqrt{1 - x^2} \quad \text{for } (x, 0) \in \Gamma. \tag{6.5}$$

There holds $u \in \widetilde{H}^{1-\varepsilon}(\Gamma)$ for all $\varepsilon > 0$, but $u \notin \widetilde{H}^1(\Gamma)$. The energy norm of u can be computed analytically, $\|u\| = \sqrt{\pi/2}$. Figure 12 shows the experimental results of our computations. As in the previous examples, the error estimators η_M, μ_M, and μ_Π show

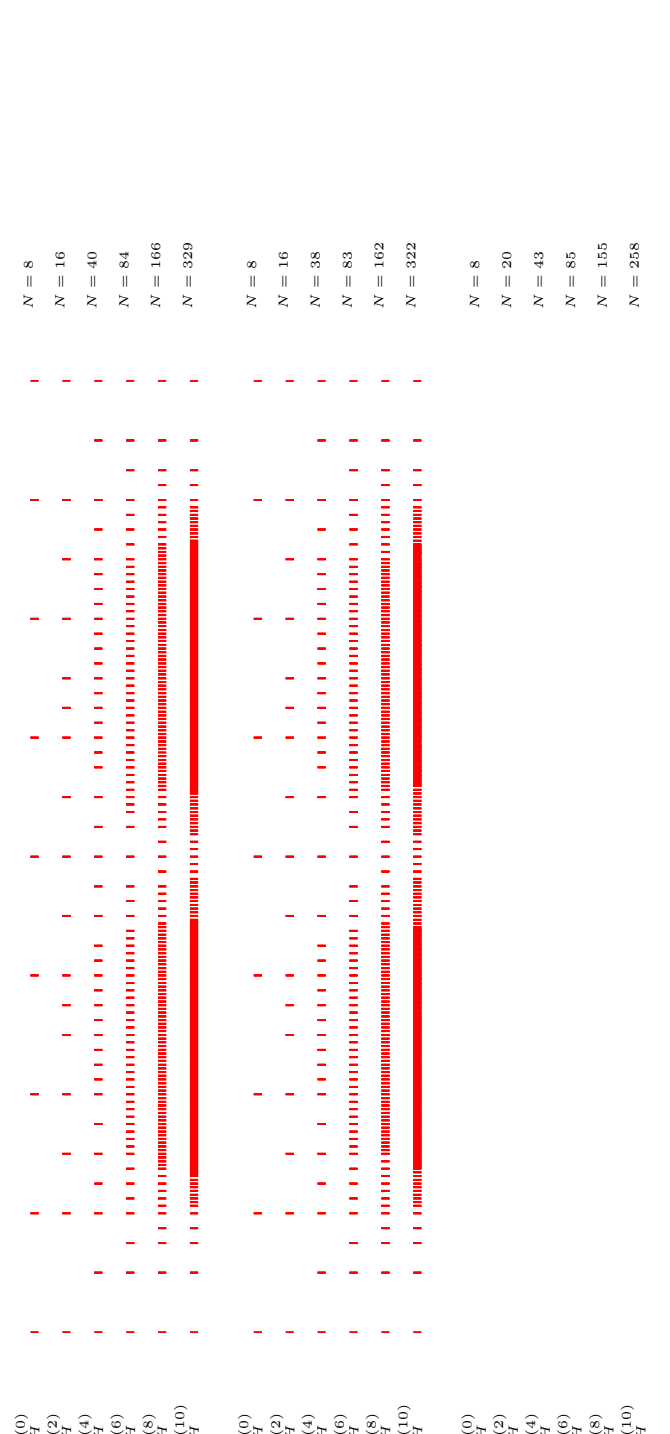


FIG. 8. *Adaptive mesh-refinement in the Poisson problem in section* 6.2 *with* $\ell = 2$ *related to the error estimator* μ_M *(top),* μ_Π *(middle), and* μ_A *(bottom), respectively. For the* μ_M*- and* μ_Π*-adapted meshes, we see a high mesh-refinement towards the reentrant corner, where* u *is slightly singular.*

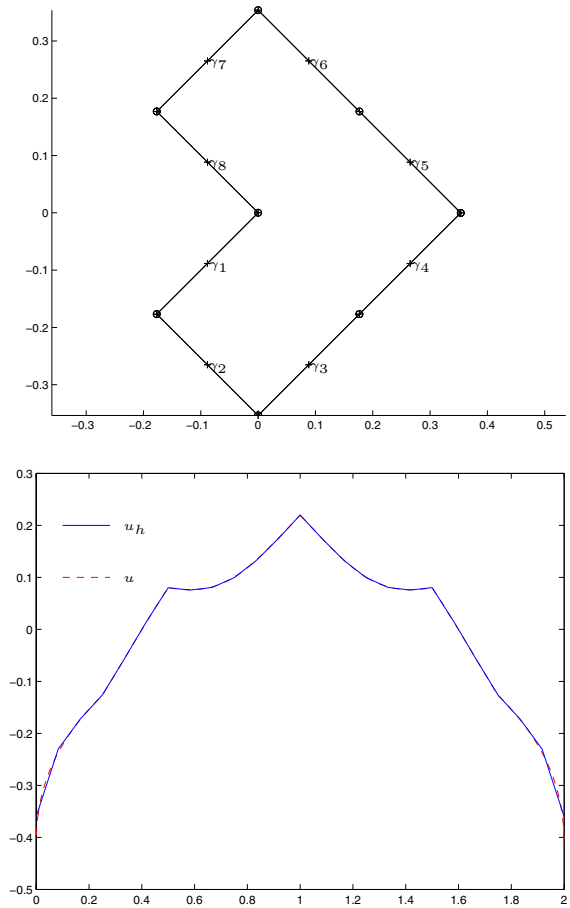


FIG. 9. *Initial coarse mesh* $\mathcal{T}_H^{(0)}$ *in the example in section* 6.3 *with* $N = 8$ *elements (nodes indicated by* ∘*) (top) and the corresponding fine mesh* $\mathcal{T}_h^{(0)}$ *for* $\ell = 2$ *in Algorithm* 5.1. *The related discrete solution* u_h *on* $\mathcal{T}_h^{(0)}$ *as well as the exact solution* u *is plotted over the arclength* $s = 0, \dots, 2$ *of* Γ *(bottom).*

efficiency and reliability independent of the mesh-refining strategy. Uniform mesh-refinement leads to a suboptimal experimental convergence rate $\kappa(E_h) = \mathcal{O}(h^{1/2})$. This can be improved by use of μ_M- and μ_Π-adaptive mesh-refinement. Both adaptive strategies recover the optimal convergence rate. As in the example in section 6.3, μ_A-adaptive mesh-refinement leads to an improved convergence rate of the error, but this is still suboptimal.

7. Conclusions.

7.1. Empirical results. The overall impressions of our numerical experiments, partly repeated from the preceding subsections, give us the subsequent empirical results.

(a) μ_Π is the most simple estimator (from the conceptual and implementational point of view); it is theoretically shown only to be efficient, but performs efficiently and reliably in our numerical examples.

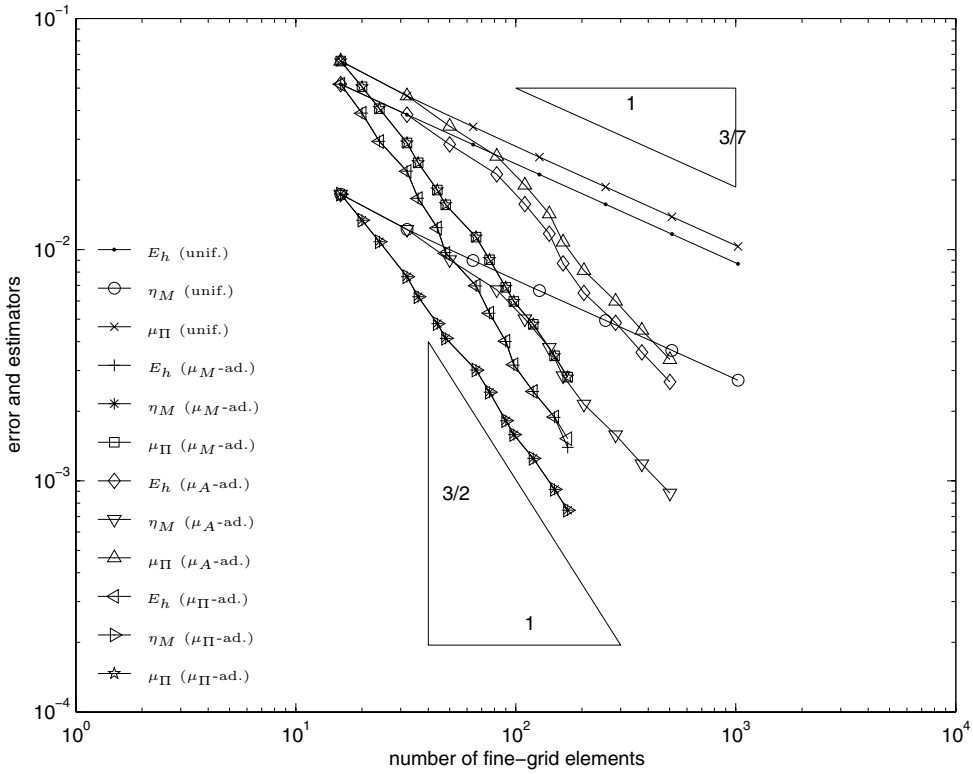


FIG. 10. *Error* $E_h = \|(\mathbf{1} - \mathbb{G}_h)u\|$ *and error estimators* η_M, μ_M, *and* μ_Π *for uniform (unif.),* μ_M*-adaptive (*μ_M*-ad.), and* μ_Π*-adaptive (*μ_Π*-ad.) mesh-refinement in the example in section* 6.3 *with* $\ell = 2$ *in Algorithm* 5.1.

(b) The error control of η_M is extremely accurate at least if the exact solution u has some regularity. The error estimation is far superior in comparison with the other error estimators.

(c) Algorithm 5.1 steered by the refinement indicators μ_M and μ_Π, respectively, leads to the optimal experimental convergence rate of the Galerkin method whenever the regularity of the exact solution leads to a reduced convergence order for uniform mesh-refinement.

(d) For $\mathcal{A}_H$, the L^2 projection onto $\mathcal{S}_H$, the error estimators η_A and μ_A lack the efficiency. Although μ_A-adaptive mesh-refinement leads to an improvement of the experimental convergence rate for the singular examples, this adaptive strategy does not regain the optimal convergence rate.

(e) The numerical outcome appears to be very sensitive with respect to implementational details. The estimators η_M, μ_M, and η_A involve stiffness matrices with respect to $\mathcal{S}_0^2$, while μ_Π relies on the much more robust L^2 projection.

7.2. Theoretical results. The heart of our analysis is Proposition 4.1 and, obtained from this, the efficiency and reliability of the error estimator $\eta_M = \|u_h - \mathbb{G}_H u_h\|$ up to terms of higher order. Those higher-order terms depend only on the regularity of the exact solution u which is in general unknown.

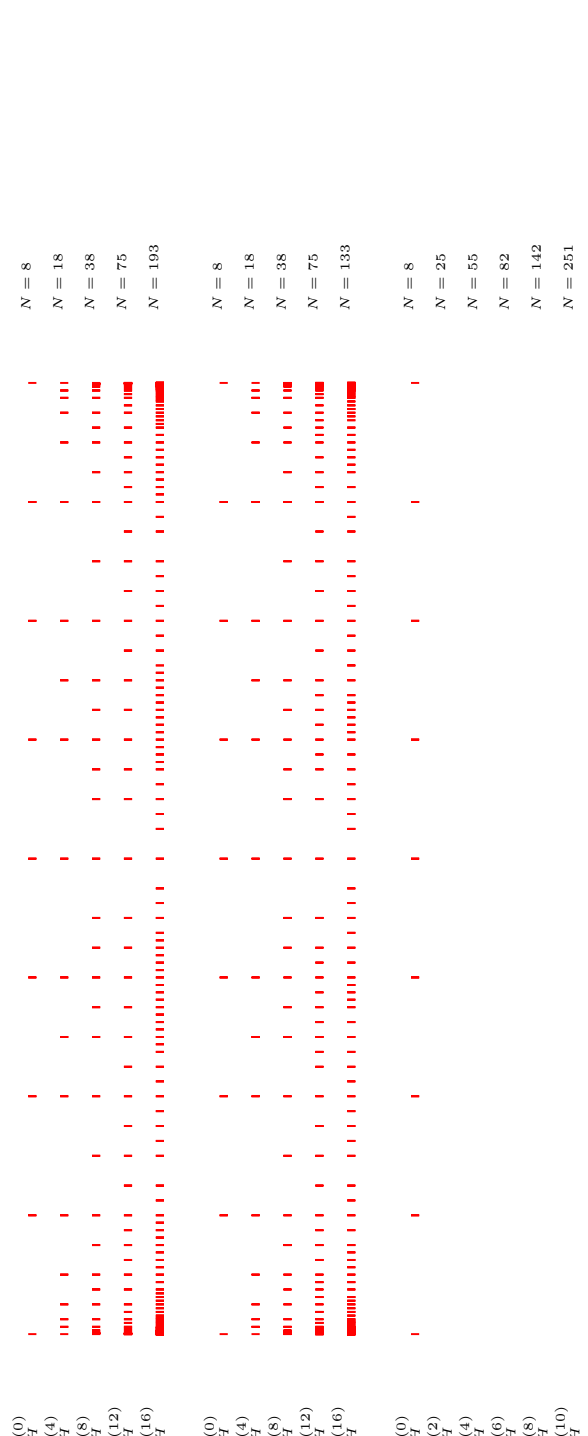


FIG. 11. *Adaptive mesh-refinement in the Poisson problem in section* 6.3 *with* $\ell = 2$ *related to the error estimator* μ_M *(top),* μ_Π *(middle), and* μ_A *(bottom), respectively.*

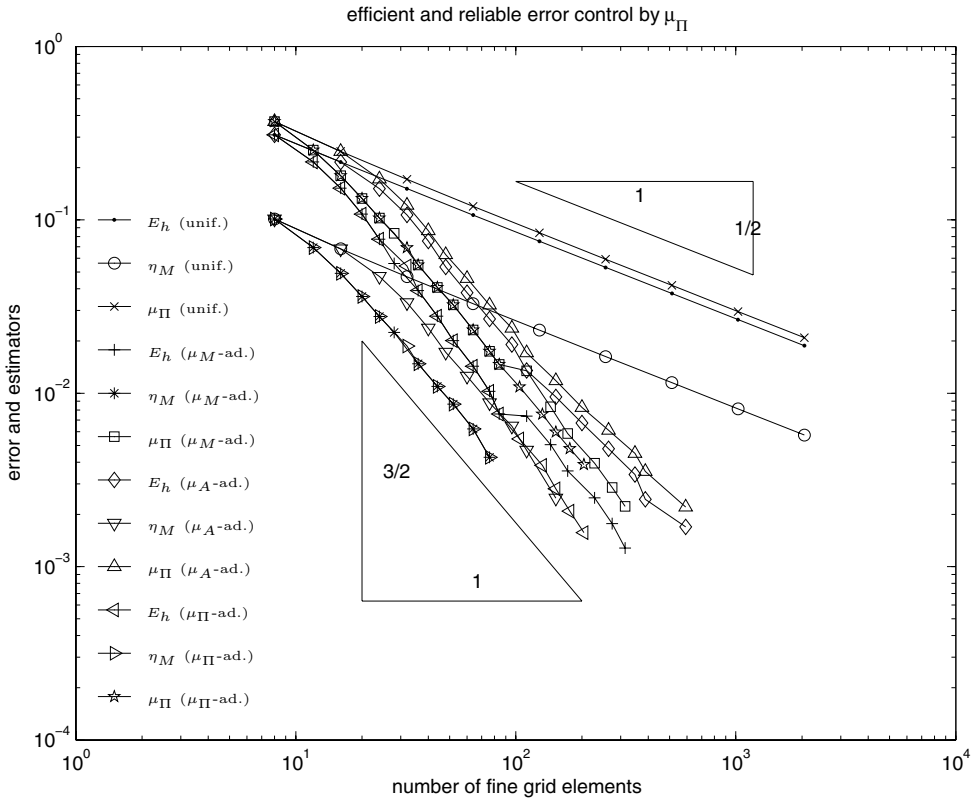


FIG. 12. *Error* $E_h = \|(\mathbf{1} - \mathbb{G}_h)u\|$ *and error estimators* η_M *and* μ_Π *for uniform (unif.) and adaptive (ad.) mesh-refinement in the example in section* 6.4 *with* $\ell = 2$ *in Algorithm* 5.1.

The proof of Proposition 4.1 has essentially two ingredients: first, an approximation operator $I_h : \mathcal{S}_H \to \mathcal{S}_h$ which has a local first-order approximation property, and second, a local inverse estimate in $\mathcal{S}_H$. Whereas we provided the inverse estimate for two- and three-dimensional problems, we did not provide the operator I_h in the latter case. For two dimensions, one can simply choose $I_h v_H$ to be the nodal interpolant. For three dimensions, one choice is (a slightly modified) Clément interpolation operator [FFP].

From the best approximation property of the Galerkin projection, one obtains that each averaging operator $\mathcal{A}_H : \mathcal{S}_h \to \mathcal{S}_H$ yields a reliable error estimator $\eta_A = \|u_h - \mathcal{A}_H u_h\|$.

Since the energy norm $\|\cdot\|$ is nonlocal, the obtained estimators cannot be employed for an adaptive mesh-refinement directly. We use localization in terms of the weighted H^1 seminorm $\mu_A = \|H^{1/2}(u_h - \mathcal{A}_H u_h)'\|_{L^2(\Gamma)}$. An inverse estimate shows the lower bound $C^{-1}\mu_A \leq \eta_A$. The verification of $\eta_A \leq C\,\mu_A$ needs some approximation property of $\mathcal{A}_H$ and holds, e.g., for the Galerkin projection $\mathcal{A}_H = \mathbb{G}_H$.

7.3. Future development. There remain several open questions for future research.

(a) From a comparison with the finite element method, the most natural error estimator is $\mu_\Pi = \|H^{1/2}(u_h' - \Pi_H u_h')\|_{L^2(\Gamma)}$ with Π_H the L^2 projection onto $\mathcal{P}^1(\mathcal{T}_H)$.

The proof of the reliability seems to need a different analytical technique, which is far beyond the scope of this paper.

(b) As for Symm's integral equation [CP], an analytical justification of Algorithm 5.1 would be desirable. The analytical verification of the introduced error estimators needs high regularity assumptions on u which are in general not satisfied and—according to our experiments—not necessary in praxis.

(c) The justification of η_M for the higher-order term $\|(\mathbb{1} - \mathbb{G}_H)u\|$ reads more precisely as follows, where $L < 1$ denotes the constant from (4.1): Provided that

$$\frac{1}{1-L}\,\frac{\|u - \mathbb{G}_H u\|}{\|u - u_h\|} =: \varrho_{hH} < 1, \tag{7.1}$$

Theorem 4.2 states that $\|u - u_h\| \le \frac{1}{1-L}\,\eta_M + \varrho_{hH}\|u - u_h\|$ and thus

$$\|u - u_h\| \le \frac{1}{(1-L)(1-\varrho_{hH})}\,\eta_M. \tag{7.2}$$

Sufficient smoothness of u (cf. (4.2)) justifies (7.1) for sufficiently small mesh-sizes h and H. However, singularities yield much poorer smoothness, e.g., $u \in H^s(\mathcal{T}_H)$ for $s \le \frac{3}{2} - \varepsilon$. Then, uniform meshes result in

$$\|u - u_h\| = C_7 h^\alpha + o(h^\alpha) \quad \text{and} \quad \|u - \mathbb{G}_H u\| = C_8 H^\alpha + o(H^\alpha)$$

for the same order of convergence $\alpha > 0$ as $h \to 0$. Then, there is no guarantee that

$$\varrho_{hH} = \frac{(H/h)^\alpha}{1-L}\,\frac{C_8}{C_7} + o(1)$$

satisfies $\varrho_{hH} < I$. However, because higher-order discrete spaces are better, one may expect that $C_8 < C_7$ so that there is room for (7.1). Since details depend on all the constants, a further analysis lies beyond the scope of this work.

(d) Three-dimensional experiments for Symm's integral equation and the hypersingular integral equation are postponed to a forthcoming paper [FFP].

Acknowledgment. Parts of this work were done when the second author (DP) enjoyed the hospitality of the Humboldt-Universität zu Berlin.

REFERENCES

[AC] J. Alberty and C. Carstensen, *Averaging techniques for reliable a posteriori FE-error control in elastoplasticity with hardening*, Comput. Methods Appl. Mech. Engrg., 192 (2003), pp. 1435–1450.

[BL] J. Bergh and J. Löfström, *Interpolation Spaces*, Grundlehren Math. Wiss. 223, Springer-Verlag, Berlin, 1976.

[C1] C. Carstensen, *An a posteriori error estimate for a first-kind integral equation*, Math. Comp., 66 (1997), pp. 69–84.

[C2] C. Carstensen, *Some remarks on the history and future of averaging techniques in a posteriori finite element analysis*, ZAMM Z. Angew. Math. Mech., 84 (2004), pp. 3–21.

[CB] C. Carstensen and S. Bartels, *Each averaging technique yields reliable a posteriori error control in FEM on unstructured grids.* I. *Low order conforming, nonconforming, and mixed FEM*, Math. Comp., 71 (2002), pp. 945–969.

[CFa] C. Carstensen and B. Faermann, *Mathematical foundation of a posteriori error estimates and adaptive mesh-refining algorithms for boundary integral equations of the first kind*, Eng. Anal. Bound. Elem., 25 (2001), pp. 497–509.

[CFuI] C. Carstensen and S. A. Funken, *Averaging technique for FE—A posteriori error control in elasticity.* I. *Conforming FEM*, Comput. Methods Appl. Mech. Engrg., 190 (2001), pp. 2483–2498.

[CFuII] C. Carstensen and S. A. Funken, *Averaging technique for FE—A posteriori error control in elasticity.* II. *λ-independent estimates*, Comput. Methods Appl. Mech. Engrg., 190 (2001), pp. 4663–4675.

[CFuIII] C. Carstensen and S. A. Funken, *Averaging technique for FE—A posteriori error control in elasticity.* III. *Locking-free nonconforming FEM*, Comput. Methods Appl. Mech. Engrg., 191 (2001), pp. 861–877.

[CP] C. Carstensen and D. Praetorius, *Averaging techniques for the effective numerical solution of Symm's integral equation of the first kind*, SIAM J. Sci. Comput., 27 (2006), pp. 1226–1260.

[CaS] C. Carstensen and E. P. Stephan, *Adaptive boundary element methods for some first kind integral equations*, SIAM J. Numer. Anal., 33 (1996), pp. 2166–2183.

[Co] M. Costabel, *Boundary integral operators on Lipschitz domains: Elementary results*, SIAM J. Math. Anal., 19 (1988), pp. 613–626.

[CoS] M. Costabel and E. P. Stephan, *The normal derivative of the double layer potential on polygons and Galerkin approximation*, Appl. Anal., 16 (1983), pp. 205–228.

[FFP] S. Funken, S. Ferraz-Leite, and D. Praetorius, *Averaging on Large Patches for First Kind Integral Equations in* 3*D*, work in progress, 2007.

[GHS] I. G. Graham, W. Hackbusch, and S. A. Sauter, *Finite elements on degenerate meshes: Inverse-type inequalities and applications*, IMA J. Numer. Anal., 25 (2005), pp. 379–407.

[HSWW] W. Hoffmann, A. H. Schatz, L. B. Wahlbin, and G. Wittum, *Asymptotically exact a posteriori estimators for the pointwise gradient error on each element in irregular meshes, Part* I*: A smooth problem and globally quasi-uniform meshes*, Math. Comp., 70 (2001), pp. 897–909.

[Ma] M. Maischak, *The Analytical Computation of the Galerkin Elements for the Laplace, Lamé and Helmholtz Equation in* 2*D-BEM*, preprint, Institut für Angewandte Mathematik, Universität Hannover, Hannover, Germany, 1999.

[McL] W. McLean, *Strongly elliptic systems and boundary integral equations*, Cambridge University Press, Cambridge, UK, 2000.

[SaS] S. A. Sauter and C. Schwab, *Randelemente, Analyse und Implementierung schneller Algorithmen*, Teubner, Stuttgart, Germany, 2004.

[ZZ] O. C. Zienkiewicz and J. Z. Zhu, *A simple error estimator and adaptive procedure for practical engineering analysis*, Internat. J. Numer. Methods Engrg., 24 (1987), pp. 337–357.

SIAM J. SCI. COMPUT.
Vol. 29, No. 2, pp. 811–826
© 2007 Society for Industrial and Applied Mathematics

ERROR ANALYSIS FOR OPERATIONS IN SOLID MODELING IN THE PRESENCE OF UNCERTAINTY*

LARS-ERIK ANDERSSON†, NEIL F. STEWART‡, AND MALIKA ZIDANI‡

Abstract. The problem of maintaining consistent representations of solids in computer-aided design and of giving rigorous proofs of error bounds for operations such as regularized Boolean intersection has been widely studied for at least two decades. One of the major difficulties is that the representations used in practice not only are in error but are fundamentally inconsistent. Such inconsistency is one of the main bottlenecks in downstream applications. This paper provides a framework for error analysis in the context of solid modeling, in the case where the data is represented using the standard representational method, and where the data may be uncertain. Included are discussions of ill-condition, error measurement, stability of algorithms, inconsistency of defining data, and the question of when we should invoke methods outside the scope of numerical analysis. A solution to the inconsistency problem is proposed and supported by theorems: it is based on the use of Whitney extension to define sets, called Quasi-NURBS sets, which are viewed as realizations of the inconsistent data provided to the numerical method. A detailed example illustrating the problem of regularized Boolean intersection is also given.

Key words. solid modeling, NURBS, regularized Boolean operations, Whitney extension

AMS subject classifications. 65D07, 65D17, 68U07

DOI. 10.1137/040604303

1. Introduction.

The problem of error analysis, in the context of representation and manipulation of solids (certain subsets of R^3), has generated a large literature over the last two decades—overviews can be found in [1] and [2]—but the fundamental problems have not yet been resolved. Indeed, Farouki [3] observes that inconsistent geometry is "... the key bottleneck in the use of CFD [Computational Fluid Dynamics] as a practical design tool" and, further, that the "... lack of a robust solution to the surface intersection problem is the culprit behind most problems in CAD data (e.g., gaps/overlaps between abutting surfaces, topological inconsistencies).... A satisfactory theory of how to approximate [edges and surfaces of represented solids] in a mutually consistent manner has failed to emerge." While certain solutions have been proposed [4, Chap. 20], in the setting considered here they do not provide the satisfactory theory referred to above.

In this paper we consider the case where a certain standard representation of solids is used and develop a framework for error analysis in this context. To establish the framework, the idea of Quasi-NURBS sets is introduced: these are regular-closed sets with boundary faces that are slight perturbations of nonuniform rational B-splines (NURBS) patches. Such sets will be viewed as realizations in R^3 of given representational data.

In the solid-modeling and computational-geometry literature our topic is called

*Received by the editors February 19, 2004; accepted for publication (in revised form) October 9, 2006; published electronically April 17, 2007.
http://www.siam.org/journals/sisc/29-2/60430.html
†Department of Mathematics, University of Linköping, SE-581 83 Linköping, Sweden (leand@mai.liu.se).
‡Département IRO, Université de Montréal, CP6128, Succ. CentreVille, H3C 3J7, QC, Canada (stewart@iro.umontreal.ca, zidanima@iro.umontreal.ca). The research of the second author was supported in part by a grant from the Natural Sciences and Engineering Research Council of Canada and by grants from the National Science Foundation under grants DMS-9985802 and DMS–0138098.

"robustness." It is usually unclear, however, whether lack of robustness is related to [illegible] of the problem to be solved, to [illegible] of the methods used to implement operations on solids, to [illegible] in the underlying representations, or to some combination of these. Indeed, the distinction between ill-condition and instability [5, 6] is seldom made, and the possibility of uncertainty in the input data is often forgotten or eliminated by hypothesis. The fundamental inconsistencies mentioned may be due to the approximation of high-degree intersection curves by lower-degree curves, or to the use of floating-point arithmetic. Thus, standard methods for the representation of solids include multiple and inconsistent descriptions of geometric entities, and these descriptions will also be inconsistent with the specified topological information. This leads to the question of exactly which subset of R^3 is defined by the inconsistent representations of the input operands. We cannot prove rigorous theorems about, say, the computed union of two solids, until this has been specified.

Another serious shortcoming of much of the literature in this area is the lack of a metric permitting us to evaluate the quality of a computed result. In fact, robustness of a numerical method, in solid-modeling operations, is often taken to mean simply that the computer program does not "crash." Such a criterion is, of course, completely inadequate (for example, it can easily be satisfied by an algorithm that always outputs the unit cube, independently of its inputs).

In section 2 we describe the framework for error analysis. Section 3 describes the standard approach to representing solids and our method of specification of a well-defined subset of R^3 given well-formed data. Section 4 gives a specific proposal for the definition of a metric, and section 5 discusses, primarily, Boolean set operations. Section 6 concludes the paper.

2. Error analysis in the presence of uncertainty. A useful approach to error analysis, in the context of uncertain input operands, is to show that the numerical method produces a "slightly wrong solution to a slightly wrong problem" [7, sect. 4]. In particular, this idea was used with great success for methods in numerical linear algebra [5]. The fact that the data is uncertain permits us to associate all or part of the computational error with the problem itself, since we must in any case tolerate the error arising from the difference between the problem to be solved and the problem submitted to the numerical method. Thus, if we have performed an analysis of the sort suggested in the above quote, then the potential size of the error will be determined by the condition of the problem. If the problem is ill-conditioned (small perturbations of the input data may cause large changes in the solution), then the error may be large, even though the method has done as well as possible, and in this case we must be prepared to accept a large error in the solution, due to the combined effect of the uncertainty in the data and the ill-condition of the problem: improving the numerical method cannot guarantee a small error.

We will examine these ideas in the context of computing regularized[1] Boolean operations between [illegible] (i.e., compact, semialgebraic, regular-closed sets [8]).

2.1. Choice of metric. We will postpone the question of the choice of the metric d, to measure the distance between two solids, until section 4. For now we observe only that it may or may not be appropriate to include sameness of topological

[1]Boolean set operations in solid modeling are interpreted as operations in the algebra of regular-closed sets: $A \circ^* B = cl(int(A \circ B))$, $\circ = \cap, \cup$, or $\sim$ [8]. The r-sets, referred to here as *solids* or *objects*, have become a de facto standard for geometric modeling.

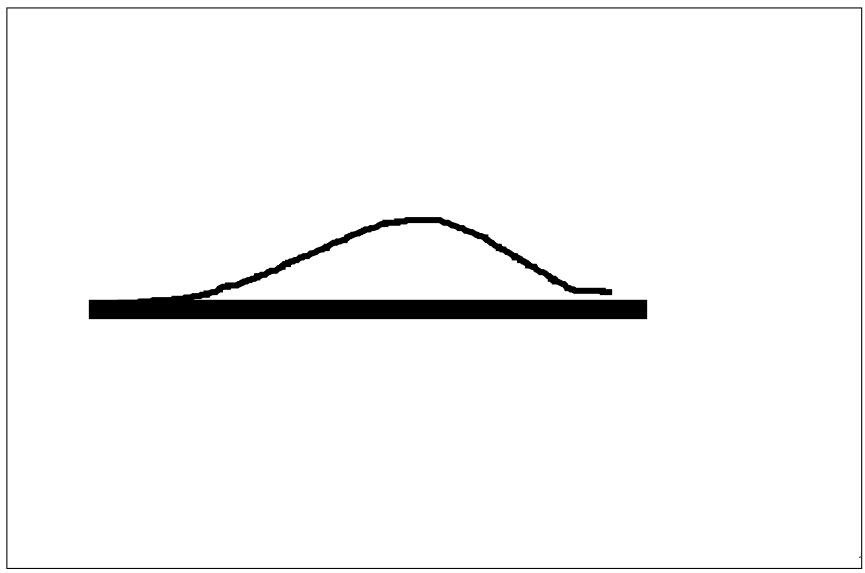


FIG. 1. *A springclip on a block: the set-union problem is ill-conditioned.*

form in the metric, depending upon the application.

The springclip shown in Figure 1 touches the block only at the left. If the block and the springclip are considered as separate objects, then there is no ambiguity about the topological form of the objects, notwithstanding possible inconsistencies in their representations: both are 3-balls, with boundaries that are 2-spheres. If these objects are submitted to a set-union algorithm, the topological form of the output is well defined, and if the algorithm does not produce an object with correct topological form (e.g., if it incorrectly joins the springclip to the block at the right end), then it has produced a wrong result. Furthermore, in a design application, the topological form of the result [illegible] to the user. Thus, it is often appropriate to include sameness of topological form [9, 10] in a metric that determines whether two objects are proximate. In contrast, in applications such as collision detection, use of the Hausdorff metric [11] may be sufficient.

2.2. Condition of the problem. Whether in the example of Figure 1 a computer program should be [illegible] to produce a result with the correct topological form is a completely different question. In fact, depending on the separation between the block and the right-hand end of the springclip, the problem submitted to the method may be ill-conditioned: small perturbations of the input data lead to a completely different answer (a set with the wrong topological form). This is relevant, because it is quite likely that there was uncertainty in the data defining the two original objects. For example, there may have been error in decimal-binary conversion or in the methods used to create or input the two original objects. In such circumstances, if the error metric requires sameness of topological form, then it is futile, for example, to apply exact arithmetic to obtain the exact result for the given input data.

Our second example illustrates the inevitability of uncertainty in the practical context of operations on solids. Let $S_0 = conv(\{(0,0,z),(1,0,z),(0,1,z) : z = 0,1\})$, $S_1 = conv(\{(0,0,z),(\sqrt{2},0,z),(\sqrt{2},\sqrt{2},z),(0,\sqrt{2},z) : z = 0,1\})$ (each of which has

a rectangular face with sides of length 1 and $\sqrt{2}$), and suppose that S_0 and S_1 are to be joined together by rotating and translating S_1 so that its vertices are $\{(1,0,z),(2,1,z),(1,2,z),(0,1,z),\ z=0,1\}$ and computing the set union of S_0 and S_1. In this set union, the rectangular faces mentioned above may or may not coincide: because of the uncertainty in the data defining S_1, due to the original approximation of $\sqrt{2}$ and to the error introduced by rotation and translation, the numerical set-union algorithm cannot tell. As in the case of the springclip, if sameness of topological form is required, then the set-union problem is [illegible]. Small perturbations of the input data could cause the union to be either a smooth connected object or two disjoint objects separated by a void corresponding to, say, a thin membrane or viscous fluid. The essential point is that the numerical method cannot possibly decide, using the information available, which answer is correct. An ideal set-union algorithm should not return either answer but should instead signal that it is unable to solve the problem, due to insufficient information. Possible approaches to resolving the ambiguity are outside the domain of numerical analysis. (They are, rather, in domains like human-computer interaction: for example, we might consider a "snap-to-grid" interface.) It is not the business of the numerical analyst to add clever tricks to the algorithm so that it happens to give a particular valid solid.

The above ideas may be illustrated by discussing what may be the most widely used test of the robustness of set-intersection algorithms in the solid-modeling literature. A well-formed cube and a slightly rotated version of the cube are presented to a set-difference algorithm; the smaller the angle of rotation that can be tolerated without the program "crashing," the more robust the algorithm is considered to be. First, as already suggested, this criterion gives no consideration to the accuracy of the result. Second, even if we introduce a stringent accuracy requirement, in the presence of uncertainty it is not appropriate to judge a method by the smallness of the angle for which it can produce accurate results for the [illegible] input solids. Once the angle and the data uncertainty preclude determination of the topological form of the difference set, an algorithm should not use expensive means to compute this difference; rather, it should signal that it is not possible to compute a reliable answer.

2.3. Requirements for a stability proof. We wish to provide rigorous error bounds for implementations of operations such as rigid motions or regularized Boolean operations. In the latter case, a typical problem is to compute $S_0 \cap^* S_1$, given solids S_0 and S_1. According to the prescription above, if S_c is the computed result, then we should show that

i. there exists S_0' and
ii. there exists S_1'

such that

1. $d(S_0', S_0)$ is small,
2. $d(S_1', S_1)$ is small, and
3. $d(S_0' \cap^* S_1', S_c)$ is small,

where d is the metric mentioned in section 2.1. A satisfactory value for "small" will depend on the uncertainty in the input data, but in the context of solid-modeling applications, relative errors as large as 1% are useful and appropriate.

There is, however, a fundamental difficulty: the representations of solids used in practice do not rigorously define regular-closed sets. The standard approach to representing such sets, widely used in commercial software [12] and in international standards [13], involves a finite collection of compact 2-manifolds-with-boundary in R^3 that are intended to fit together to form a topological 2-cycle and thus define the

boundary of a homogeneously three-dimensional subset of R^3 [8]. Unfortunately, the separate computations performed to obtain the boundary of each 2-manifold virtually ensure that manifolds which are indicated as being adjacent will not meet along the two evaluations of their common boundary. Moreover, there will typically be a third representation of the boundary between two patches, namely, an explicit boundary curve, which will be inconsistent with the other two; further, the endpoints of the boundary curve may be represented explicitly by points in R^3, inconsistently with the actual endpoints of the three curves approximating the patch boundary. The reasons for maintaining such redundant and almost certainly inconsistent information arise from the needs of various algorithms, such as graphical display.

As already mentioned, in order to prove theorems about operations on solids, we must define the actual subset S of R^3 defined by the inconsistent data Δ. This will be done in section 3.1: we will define conditions for [illegible] of Δ, which ensure that a corresponding subset S of R^3 is [illegible], and in this case we will say that S is the [illegible] of Δ, written $S \models \Delta$.

Since we want to consider only well-defined subsets of R^3, and since the distance d between two such subsets S_0 and S_1 will be defined (see section 4) in terms of their representations Δ_0 and Δ_1, we must revise the requirements given above. Suppose (S_0, Δ_0) and (S_1, Δ_1) are given with $S_0 \models \Delta_0$ and $S_1 \models \Delta_1$, and let Δ_c be the data representing the computed result. We wish to show that

i. there exist S_0' and Δ_0' such that $S_0' \models \Delta_0'$,
ii. there exist S_1' and Δ_1' such that $S_1' \models \Delta_1'$, and
iii. there exists S_c such that $S_c \models \Delta_c$

such that

1. $d(S_0', S_0)$ is small,
2. $d(S_1', S_1)$ is small, and
3. $d(S_0' \cap^* S_1', S_c)$ is small.

3. Well-formedness of Δ and well-definition of S.

3.1. The standard approach to representing r-sets. More formally, the representation Δ contains symbolic or [illegible], describing how the faces, edges, and vertices of the cellular decomposition of the boundary ∂S of S fit together. This data defines a topological 2-cycle.[2] Further, the representation contains [illegible]. Corresponding to each 2-cell in the topological data is an oriented (as specified by an outer normal $\boldsymbol{n}$) trimmed surface patch in R^3. Two of these, $\boldsymbol{F}[D] = \{\boldsymbol{F}(u,v) : (u,v) \in D \subseteq D_0\}$ and $\boldsymbol{F}'[D'] = \{\boldsymbol{F}'(u,v) : (u,v) \in D' \subseteq D_0\}$, are shown in Figure 2. Here, the mappings $\boldsymbol{F}$ and $\boldsymbol{F}'$ define NURBS surfaces[3] in R^3, defined over a (triangular [14] or rectangular [15]) domain D_0. The patches are called [illegible] NURBS patches because the spline function is restricted to a subset D of D_0, as delineated by certain curves (so called [illegible] [12]) in the parametric domain, which may arise, for example, from intersecting two surfaces used for the boundary of the solid.

[2] A topological 2-cycle is a collection of 0-cells, 1-cells, and 2-cells forming a closed boundary for which every 1-cell has an even number of incident 2-cells. This includes 2-manifolds (e.g., a torus) for which the even number of incident 2-cells is exactly 2, as well as nonmanifolds (e.g., two tetrahedra sharing a common edge). Surface patches are 2-manifolds-with-boundary that correspond to 2-cells, but the set $S \subseteq R^3$ is not necessarily a 3-manifold.

[3] In section 3, primes have been used to distinguish distinct faces within a single object. Elsewhere, primes denote perturbed versions of complete objects.

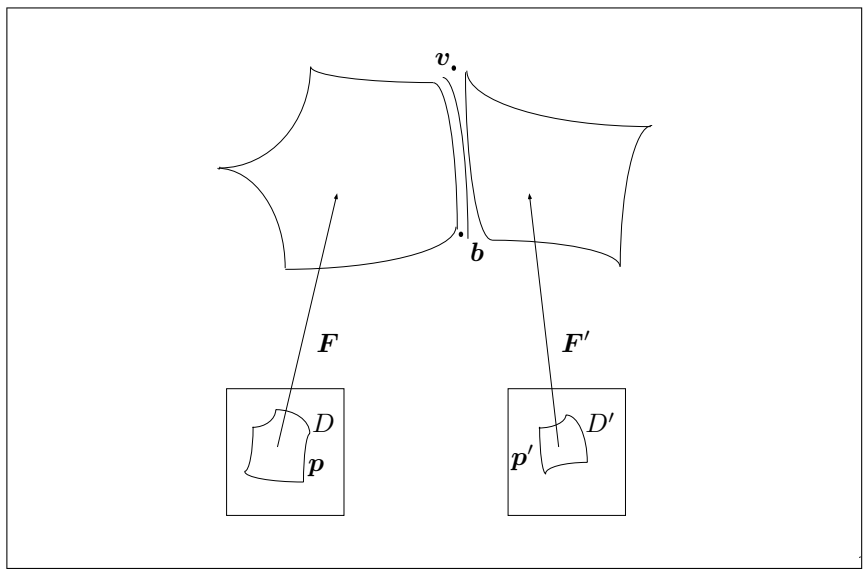


FIG. 2. *The geometric data includes adjoining trimmed NURBS patches.*

The actual boundaries of the trimmed patches are stored implicitly by their appropriate preimages (p-curves such as $\boldsymbol{p}$ and $\boldsymbol{p}'$). (Numerical algorithms [16, 17] compute these preimages as the output for an intersection of surfaces.) The data representation for each preimage is a collection of floating-point numbers, stored exactly, defining the control points (in (u-v)-space) for the p-curves, and it is easy to arrange that the p-curves interpolate their exactly stored endpoints [15]. There is an even number of p-curves corresponding to each 1-cell in the topological data, one associated with each adjacent face. There is also an explicit representation in R^3 of the boundary curve between $\boldsymbol{F}[D]$ and $\boldsymbol{F}'[D']$, denoted $\boldsymbol{b}(t)$, $t \in [0,1]$ (one such curve for each 1-cell). As shown in Figure 2, the curve $\boldsymbol{b}(t)$ does not usually coincide exactly with the corresponding edges of $\boldsymbol{F}[D]$ or $\boldsymbol{F}'[D']$. Finally, corresponding to each 0-cell in the topological data, there may also be stored an explicit corner-vertex $\boldsymbol{v}$, which in practice may not coincide exactly with the ends of the neighboring boundary curves, such as $\boldsymbol{b}(t)$, or with the exact corners of the trimmed patches $\boldsymbol{F}[D]$ and $\boldsymbol{F}'[D']$.

We would like to specify under what conditions such a representation is
and, when it is, exactly which subset of R^3 it represents.[4]

3.2. Specification of a well-defined r-set S. The question of whether to give precedence to the topological information or to the geometric information in Δ has been discussed often in the literature on robustness in solid modeling [18]. We will suppose that the topological information is correct and modify the geometric data slightly in order to bring them into agreement with each other and with the topological information.

In our context, it is easy to justify such slight modifications of the geometric

[4]Note that the purpose of introducing these well-defined sets is that we must agree on exactly which set is defined by the inconsistent data in Δ in order to prove rigorous theorems. There is no need to actually *compute* the well-defined set determined by a well-formed representation.

data.[5] First, the geometric data provided on input [illegible] be considered exact: the fact that, even for solids of very simple form, the edges of $\boldsymbol{F}[D]$ and $\boldsymbol{F}'[D']$ do not coincide shows that the geometric data is, patently, inexact. Second, suppose the user provides, say, a cylindrical surface as input. It may be important in practice that this surface be very nearly cylindrical, but the user does not believe that the eventually manufactured object will be an [illegible] mathematical cylinder. Such a user should be prepared to accept that the inconsistent data, provided in Δ, models an object with surfaces varying slightly from cylindrical. Third, in the case when there is uncertainty in the data provided, this uncertain data may correspond to uncertainty in the actual form of the object surfaces.

We now define the subset S of R^3 specified by the inconsistent data Δ. The boundary of S will be made up of slightly perturbed trimmed NURBS patches mutually consistent with the explicit vertices $\boldsymbol{v}$ and with slightly perturbed versions of the boundary curves $\boldsymbol{b}(t)$, and consistent with the topological information.

The collection of p-curves $\boldsymbol{p}^k(t)$ defining a domain $D \subseteq D_0$ will do so by specifying the boundary ∂D of D. Since D may have holes, ∂D is a collection of simple closed curves, each with segments that are p-curves. Each mapping $\boldsymbol{p}^k$ is supposed to be injective, each segment intersects its neighboring segments only at mutual endpoints, and distinct p-curves do not intersect. (These hypotheses can be easily verified in the case of Bézier curves [9].) We also suppose that if part of the boundary of the trimmed patch is defined by a line segment forming part of the edge of the u-v domain D_0, then there is an explicit p-curve that coincides with this line segment.

It will be desirable below that $\|\boldsymbol{b}^k(t) - \boldsymbol{F}(\boldsymbol{p}^k(t))\|$ be small, $0 \le t \le 1$, and at this point, if desired, a reparametrization [15, p. 241] of $\boldsymbol{b}^k(t)$ may be introduced to reduce the size of the difference just mentioned. This reparametrization need not be the same one that is used when other patches corresponding to $\boldsymbol{b}^k(t)$ are treated. Given the explicit endpoints $\boldsymbol{v}$ and $\boldsymbol{v}'$, corresponding to $t = 0$ and $t = 1$, respectively, we map the curves $\boldsymbol{b}^k(t)$ onto curves $\boldsymbol{\beta}^k(t)$, $t \in [0,1]$, for which $\boldsymbol{\beta}^k(0) = \boldsymbol{v}$ and $\boldsymbol{\beta}^k(1) = \boldsymbol{v}'$, using linear interpolation of the endpoint differences:

$$\boldsymbol{\beta}^k(t) = \boldsymbol{b}^k(t) + (1-t)[\boldsymbol{v} - \boldsymbol{b}^k(0)] + t[\boldsymbol{v}' - \boldsymbol{b}^k(1)], \quad 0 \le t \le 1,$$

so that $\|\boldsymbol{\beta}^k(t) - \boldsymbol{b}^k(t)\| \le \max\{\|\boldsymbol{v} - \boldsymbol{b}^k(0)\|, \|\boldsymbol{v}' - \boldsymbol{b}^k(1)\|\}$, $0 \le t \le 1$. It may be that $\boldsymbol{\beta}^k$ is identical to $\boldsymbol{b}^k$ (e.g., if $\boldsymbol{b}^k$ is a piecewise-linear curve obtained by "curve-tracing," or a NURBS curve interpolating its initial and final control points).

PROPOSITION 3.1. [illegible]

1. [illegible] $\boldsymbol{F}$ [illegible]
2. [illegible] k [illegible] $\boldsymbol{p}^k$ [illegible]
3. [illegible] k $\left\| \frac{\boldsymbol{b}^k(t) - \boldsymbol{b}^k(t')}{t - t'} \right\| > (\|\boldsymbol{v} - \boldsymbol{b}^k(0)\| + \|\boldsymbol{v}' - \boldsymbol{b}^k(1)\|)$, $t < t'$

[illegible] k [illegible] $\boldsymbol{F}(\boldsymbol{p}^k(t))) \to \boldsymbol{\beta}^k(t)$ [illegible] $(\boldsymbol{F}(\boldsymbol{p}^k(t)))$ [illegible] $(\boldsymbol{\beta}^k(t)))$ [illegible]. Since conditions 1 and 2 ensure that $\boldsymbol{F} \circ \boldsymbol{p}^k$ is injective on $[0,1]$ for each k, the mappings $\boldsymbol{F}(\boldsymbol{p}^k(t))) \to \boldsymbol{\beta}^k(t)$ are well defined. Condition 3 guarantees that for each k, $\boldsymbol{\beta}^k(t)$ is injective, and the mappings are therefore bijective. □

In order for condition 3.1 of Proposition 3.1 to be satisfied, $\boldsymbol{b}^k$ must itself be injective for each k. Since the mappings $\boldsymbol{p}^k = \boldsymbol{p}^k(t)$ are injective, and distinct p-

[5] In other applications, such as engineering analysis, deducing topology may be attempted, and in this case it may be appropriate to modify the topology to agree with the geometry.

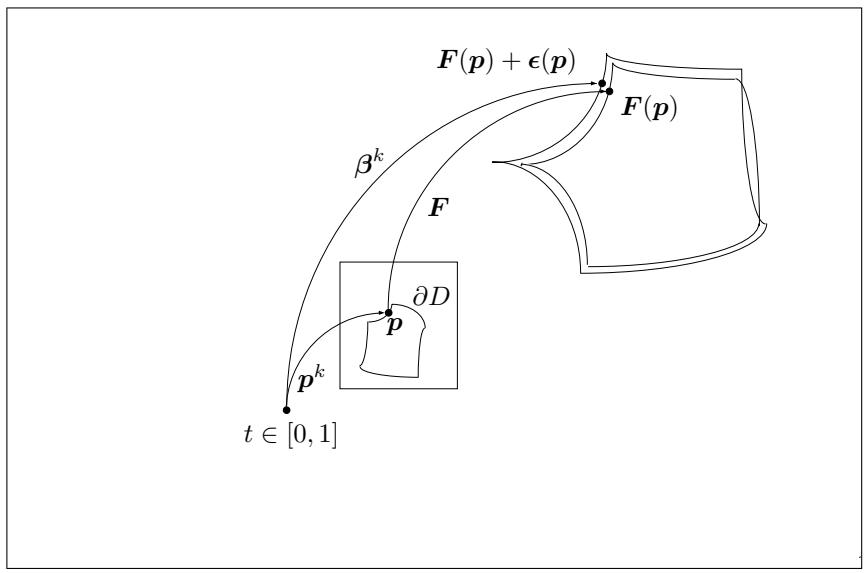


FIG. 3. *Defining the mapping* $\boldsymbol{\epsilon} = \boldsymbol{\epsilon}(\boldsymbol{p})$ *on* ∂D.

curves do not intersect, except at appropriate endpoints, we may define the mapping $\boldsymbol{\epsilon} : \partial D \to R^3$ by

$$\boldsymbol{\epsilon}(\boldsymbol{p}) = \boldsymbol{\beta}^k([\boldsymbol{p}^k]^{-1}(\boldsymbol{p})) - \boldsymbol{F}(\boldsymbol{p}), \quad \boldsymbol{p} \in \partial D, \tag{3.1}$$

where $[\boldsymbol{p}^k]^{-1}$ is the inverse [19] of the p-curve $\boldsymbol{p}^k$ for which $\boldsymbol{p}^k(t) = \boldsymbol{p}$ for some $t \in [0,1]$. Thus, $\boldsymbol{\epsilon}$ is the difference[6] between the edge of the trimmed patch $\boldsymbol{F}[D]$ and the curves $\boldsymbol{\beta}^k$, viewed as a function of $\boldsymbol{p} \in \partial D$ (see Figure 3).

We now extend $\boldsymbol{\epsilon}$ to all of D, in order to define a new trimmed patch

$$\boldsymbol{F}(\boldsymbol{p}) + \boldsymbol{\epsilon}(\boldsymbol{p}), \quad \boldsymbol{p} \in D,$$

that interpolates the modified boundary curves $\boldsymbol{\beta}$ at all points $\boldsymbol{p} \in \partial D$. These new trimmed patches will comprise the boundary of the Quasi-NURBS set S. (This defines "Quasi-NURBS set.")

Let $\epsilon(\boldsymbol{p})$ denote a component of the vector $\boldsymbol{\epsilon}(\boldsymbol{p})$, and define

$$L = \sup_{\boldsymbol{p}_1, \boldsymbol{p}_2 \in \partial D,\ \boldsymbol{p}_1 \neq \boldsymbol{p}_2} \frac{|\epsilon(\boldsymbol{p}_1) - \epsilon(\boldsymbol{p}_2)|}{\|\boldsymbol{p}_1 - \boldsymbol{p}_2\|} \quad . \tag{3.2}$$

We can extend ϵ to a continuous function on all of R^2 that satisfies a Lipschitz condition with the same Lipschitz constant L [20]. Define

$$l(\boldsymbol{p}) = \sup_{\boldsymbol{q} \in \partial D} \{\epsilon(\boldsymbol{q}) - L \cdot \|\boldsymbol{p} - \boldsymbol{q}\|\}, \quad \boldsymbol{p} \in R^2,$$

$$u(\boldsymbol{p}) = \inf_{\boldsymbol{q} \in \partial D} \{\epsilon(\boldsymbol{q}) + L \cdot \|\boldsymbol{p} - \boldsymbol{q}\|\}, \quad \boldsymbol{p} \in R^2,$$

[6]Depending upon how $\boldsymbol{b}(t)$ was obtained, it may be possible to obtain an estimate of $\boldsymbol{\epsilon}$ directly as a function of $\boldsymbol{p}$ (without examining the parametric forms) [15, p. 431], [16, sects. 4.29, 4.73].

and

$$a(\boldsymbol{p}) = \frac{1}{2}[l(\boldsymbol{p}) + u(\boldsymbol{p})], \quad \boldsymbol{p} \in R^2. \tag{3.3}$$

Any continuous function satisfying the Lipschitz condition on R^2 is bracketed by the [illegible], $l(\boldsymbol{p})$, and the [illegible], $u(\boldsymbol{p})$, and $l(\boldsymbol{p})$ and $u(\boldsymbol{p})$ are [21] themselves solutions to the extension problem, as is $a(\boldsymbol{p})$. Furthermore,

$$|a(\boldsymbol{p})| \leq \sup_{\boldsymbol{q} \in \partial D} |\epsilon(\boldsymbol{q})|, \quad \boldsymbol{p} \in R^2.$$

We will extend $\epsilon(\boldsymbol{p})$ to D using the method suggested by (3.3):

$$\epsilon(\boldsymbol{p}) = \frac{1}{2}[l(\boldsymbol{p}) + u(\boldsymbol{p})], \quad \boldsymbol{p} \in D. \tag{3.4}$$

Thus, the patch comprising the boundary of the set S differs (in each component) from the trimmed patch $\boldsymbol{F}[D]$, given as part of Δ, by no more than the error $\sup_{\boldsymbol{q} \in \partial D} |\epsilon(\boldsymbol{q})|$ in the original input data.

PROPOSITION 3.2. [illegible] $\epsilon(\boldsymbol{p})$ [illegible] (3.4) [illegible] D [illegible] (3.1) [illegible] ∂D [illegible] (3.4) [illegible] D [illegible] L [illegible] (3.2) [illegible]

$$|\epsilon(\boldsymbol{p})| \leq \sup_{\boldsymbol{q} \in \partial D} |\epsilon(\boldsymbol{q})|, \quad \boldsymbol{p} \in D.$$

The Lipschitz constants for each of the components of $\boldsymbol{\epsilon}$ can be used to define a Lipschitz constant for the vector-valued function $\boldsymbol{\epsilon}$.

We have now completed the task of defining a new patch $\boldsymbol{F}(\boldsymbol{p}) + \boldsymbol{\epsilon}(\boldsymbol{p})$, defined for all $\boldsymbol{p} \in D$, that interpolates the modified boundary curves $\boldsymbol{\beta}^k$ at all points $\boldsymbol{p} \in \partial D$, and if we apply the same process to each trimmed patch, they will fit together perfectly along the modified boundary curves. Furthermore, we have established bounds on the difference between $\boldsymbol{\beta}^k(t)$ and $\boldsymbol{b}^k(t)$, and bounds on the magnitude and rate of change of $\boldsymbol{\epsilon}(\boldsymbol{p})$ in terms of their values on ∂D.

3.3. Well-formedness of $\boldsymbol{\Delta}$. We summarize now the conditions thus far introduced, in order that Δ be well-formed. First, the topological data must define a topological 2-cycle. Second, the geometric data must include a trimmed patch corresponding to each 2-cell, an explicit boundary curve corresponding to each 1-cell, and an explicit corner point corresponding to each 0-cell. Furthermore, for each trimmed patch, there must be one p-curve corresponding to each 1-cell in the boundary of the 2-cell associated with the patch. Also, the geometric data must satisfy the conditions in the hypotheses of Proposition 3.1.

The remaining conditions, to be added now, ensure that the perturbed surface patches form an embedding of the 2-cycle in R^3: each perturbed patch

$$\boldsymbol{F}[D] + \boldsymbol{\epsilon}[D] = \{\boldsymbol{F}(\boldsymbol{p}) + \boldsymbol{\epsilon}(\boldsymbol{p}) : \boldsymbol{p} \in D\}$$

must not self-intersect, adjacent perturbed patches

$$\boldsymbol{F}[D] + \boldsymbol{\epsilon}[D] \text{ and } \boldsymbol{F}'[D'] + \boldsymbol{\epsilon}'[D']$$

may intersect only at a common corner point or along a common boundary curve, and nonadjacent perturbed patches must be disjoint. If all of the above conditions are satisfied, then S is well defined, and we write $S \models \Delta$.

In simple cases, it is possible to give a priori guarantees of non–self-intersection. If $\boldsymbol{q} = \boldsymbol{F}(\boldsymbol{p})$, the mapping $\boldsymbol{q} \mapsto \boldsymbol{q} + \boldsymbol{\epsilon}(\boldsymbol{p})$ is injective provided that the product of L and a Lipschitz constant for $\boldsymbol{F}^{-1}$ is less than 1, and, consequently, under the latter condition, injectiveness of $\boldsymbol{F}$ implies that $\boldsymbol{F}(\boldsymbol{p}) + \boldsymbol{\epsilon}(\boldsymbol{p})$ is injective in $\boldsymbol{p}$. For example, if $\boldsymbol{F}$ is a trimmed Bézier patch defined on a subset D that is contained in a triangular subset of the triangular domain D_0, then [9, Criteria 3.1 and 3.1*] and [22] can be used to guarantee injectiveness of $\boldsymbol{F}$ on D. On the other hand, adjacent patches sharing a common edge or vertex cannot be dealt with in this way. For this case and for self-intersection of single non-Bézier patches, it will be necessary to rely on a posteriori (computational) verification. Algorithms to verify non–self-intersection, based on properties of the surface normals, are available [23] (but note that care is required in applying such algorithms to composite patches [24]). If $\boldsymbol{F}$ is a rational tensor-product Bézier patch, then the Gauss map [25] of the patch can be computed by the methods of [26]. The problem of finding the Gauss map of $\boldsymbol{F} + \boldsymbol{\epsilon}$, given the Gauss map of $\boldsymbol{F}$, is discussed below.

4. Metrics for the measurement of error. Definition of a metric that adequately reflects our intuition, and practical requirements, in the context of solid modeling is not straightforward. For example, as mentioned in section 2.1, we must first decide whether the metric should require that two objects considered to be close have the same topological form, and if so, how demanding this requirement should be. A stringent requirement would be that the representations have equivalent topological data, and that the geometric objects be linked by an ambient isotopy.[7] The following is a possible definition.

Suppose $S \models \Delta$ and $S' \models \Delta'$ are given, that the 2-cycles associated with S and S' are isomorphic, and that there is an ambient isotopy linking S and S'. In this case, we define the penalty $\pi(S, S')$ to be zero; otherwise, it is infinite. Now, let $\boldsymbol{G}_i[D] = \boldsymbol{F}_i[D] + \boldsymbol{\epsilon}_i[D]$, $i = 1, \ldots, n$, be the embedded patches of S corresponding to the 2-cells in the topological 2-cycle, and let $\boldsymbol{G}'_i[D']$ be the corresponding patches of S'. Then the distance between S and S' may be defined by

$$d(S, S') = \max_{i=1,\ldots,n} \{d(\boldsymbol{G}_i[D], \boldsymbol{G}'_i[D'])\} + \pi(S, S'),$$

where the symbol d has also been used to denote the Hausdorff metric (no confusion should arise). In summary, for the distance between two representations to be small, they must define topologically equivalent embeddings, and the maximum Hausdorff distance between corresponding embedded patches must be small.[8]

We may envisage also the use of less demanding metrics: for example, the Hausdorff metric alone could be used to measure the distance between a true and a computed solution. Which problems are ill-conditioned depends strongly on the choice of metric. Thus, the example set-union problems discussed in section 2 are ill-conditioned

[7] In the case of objects defined by curvilinear complexes, not necessarily Bézier, sufficient conditions that perturbations of the control points should produce an object, linked to the original by an ambient isotopy, were given in [10].

[8] Even such a demanding metric is not always sufficient, however, to capture our intuitive idea of what constitutes closeness between solutions in the case of solid-modeling operations, since the Hausdorff metric is not sensitive to thin protrusions and cracks in an object [11].

if the metric requires equivalence of topological form, but well-conditioned if only the Hausdorff metric is used.

5. The example of set intersection or difference. A complete error analysis of the sort suggested by the requirements of section 2.3 is possible, at present, only for very simple representations and operations. Consider first an operation simpler than set intersection: suppose that an object is represented by a curvilinear Bézier complex (involving untrimmed triangular Bézier patches) [9] and that the operation is rigid motion. In this case, the representation of section 3.1 simplifies greatly. There are no p-curves other than the three boundaries of the triangular domain of each patch, and the images of the boundary segments of the parametric domain are Bézier curves; adjacent patches sharing an edge intersect in a common Bézier curve with common control points; the common curve can also play the role of $\boldsymbol{b}(t)$ and $\boldsymbol{\beta}(t)$; and, $\boldsymbol{\epsilon}(\boldsymbol{p}) \equiv \mathbf{0}$. The error introduced by applying the rigid motion to the control points of the Bézier mapping $\boldsymbol{F}$ defining each face, using finite-precision arithmetic, can be estimated by standard means [5]. Provided that $\boldsymbol{F}$ has no nearly critical points, these estimates can be used in conjunction with [9, 10] to verify that the computed solid (S_c, Δ_c) remains a well-formed Bézier complex, linked to the true transformed object by an ambient isotopy. The Hausdorff distance between the pairs of computed and true patches can also be computed. Thus, we can do a simple forward analysis: there is no need to associate any of the error with the problem data.

Even in the case of Bézier patches, however, the introduction of trimming complicates the problem greatly. The roundoff error incurred in the transformation of the control points of $\boldsymbol{F}$ can be taken into account in the application of the a priori self-intersection test described in section 3.3 (and in a similar way for tests of noninterference of disjoint patches) in order to verify that the computed set remains a well-formed Bézier complex. Verification of noninterference of adjacent patches would, however, depend on the use of computational methods [23, 24].

The [illegible] of two well-formed curvilinear objects, an operation that has occupied a central place in the robustness literature in solid modeling, is a vastly more difficult problem. We illustrate using the operations $\cap^*$ or $\sim^*$ with input solids that are modified versions of those in an example used by Hoffmann [2] to illustrate how Boolean algorithms can fail. The numerical results were obtained using the DT_NURBS software package [16].

The two ideal operands are shown in Figure 4. The first solid, denoted S_0, is a tetrahedron with four planar triangular faces: one horizontal, in the plane $z = 21$; another vertical, in the plane $x = 10/3$; and two others, partially hidden and separated by a heavy solid line partially visible in the figure. The extreme simplicity of this object means that the horizontal and vertical faces are exactly defined, in the representation of section 3.1, by their corner vertices, which respectively have third components equal to 21, or first components equal to (the floating-point representation of) 10/3. This permits use of a simple version of the Grandine–Klein algorithm, the planar-cut algorithm described in [17].

The second ideal mathematical solid shown in Figure 4, denoted S_1, has a curvilinear face defined by a Bézier mapping

$$\boldsymbol{F}(u,v) = \sum_{i=0}^{1}\sum_{j=0}^{3} \binom{1}{i}\binom{3}{j} u^i(1-u)^{1-i}v^j(1-v)^{3-j}\boldsymbol{P}_{ij}, \quad 0 \le u, v \le 1,$$

where $\boldsymbol{P}_{00} = (0,-10,0)$, $\boldsymbol{P}_{01} = (0,-10,40)$, $\boldsymbol{P}_{02} = (0,10,40)$, $\boldsymbol{P}_{03} = (0,10,0)$,

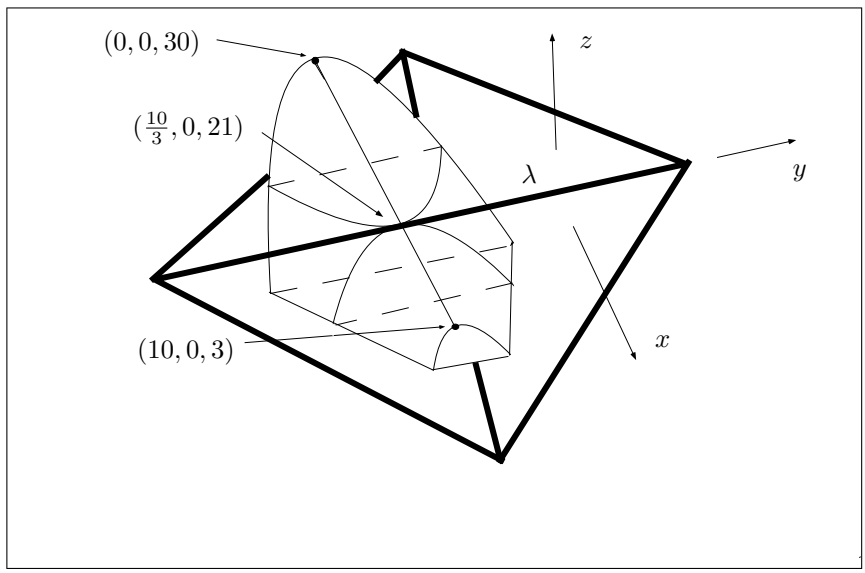


FIG. 4. *Two input objects (operands).*

and $\boldsymbol{P}_{10} = (10, -1, 0)$, $\boldsymbol{P}_{11} = (10, -1, 4)$, $\boldsymbol{P}_{12} = (10, 1, 4)$, $\boldsymbol{P}_{13} = (10, 1, 0)$. This face is symmetric with respect to the plane $y = 0$ and intersects each of the planes $z = 21$ and $x = 10/3$, as shown in Figure 4. There is a single point of intersection, namely $(10/3, 0, 21)$, between the curvilinear face and the line λ defined by $z = 21$ and $x = 10/3$. The other faces of this second object do not intersect the two rear triangular faces of S_0. For example, in the plane $x = 0$ there is a hidden face of S_1 with upper boundary defined by the Bézier curve $\sum_{j=0}^{3} v^j (1-v)^{3-j} \boldsymbol{P}_{1j}$, which has a maximum z value at $(0, 0, 30)$, and in the plane $x = 10$ there is a visible face of S_1 with upper boundary defined by the Bézier curve $\sum_{j=0}^{3} v^j (1-v)^{3-j} \boldsymbol{P}_{0j}$, which has a maximum z value at $(10, 0, 3)$.

The interesting region [2] for this example is the region near $(10/3, 0, 21)$. When the surface $\boldsymbol{F}$ is numerically intersected with the triangular face lying in the plane $z = 21$, the corresponding boundary curve may not intersect the plane $x = 10/3$ exactly once, as it theoretically should: it may intersect zero or two times. Similarly, when $\boldsymbol{F}$ is intersected with the triangular face lying in the plane $x = 10/3$, the corresponding boundary curve may intersect the plane $z = 21$ zero, one, or two times, depending on the surface-surface intersector. We used the routines D2PCUT and D2CLXT [16, sect. 4] with a tolerance of 10^{-7} to find fourth-order B-spline intersection curves $\boldsymbol{b}(t)$ lying in the $z = 21$ plane (see Figure 5) and the corresponding pair of p-curves in $(u$-$v)$-space. The two curves shown in Figure 5 can be viewed as parts of a single curve with a maximum x value approximately $3 \cdot 10^{-5}$ units beyond $(10/3, 0, 21)$ in the positive x direction. The intersections of $\boldsymbol{b}(t)$ with the line $x = 10/3$ are quite far (0.0022 units) from the true intersection $(10/3, 0, 21)$. This illustrates a major open problem in the analysis of regularized Boolean operations, namely, when and how various geometric entities, such as the two vertices indicated in Figure 5, should be merged. Commercial kernel systems [12] depend on the use of complicated (and partially undisclosed) systems of tolerances to decide these questions. The question of "topology resolution" of the boundary [17, 27] is related to this difficulty, and it is

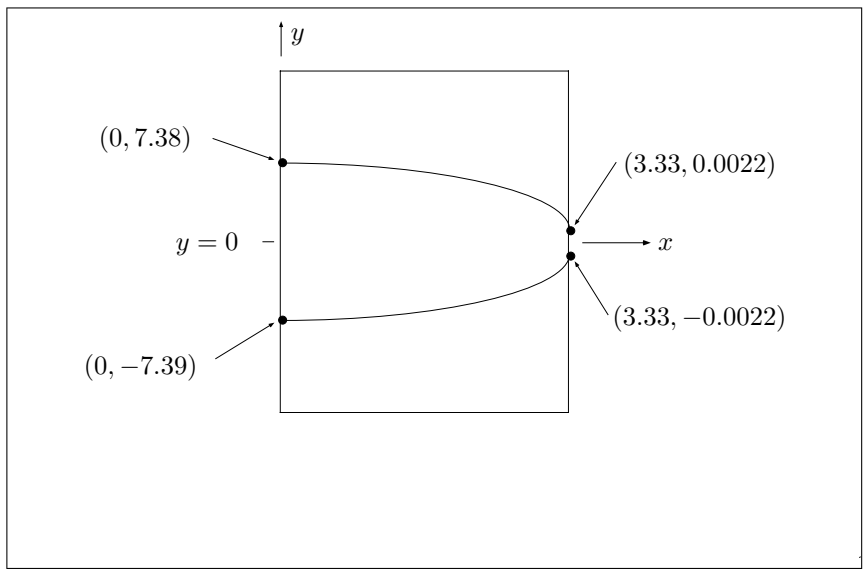


FIG. 5. *Two boundary curves in the plane* $z = 21$ *do not meet.*

aggravated by the use of finite-precision arithmetic.

Since ill-condition is possible in the case of $S_0 \sim^* S_1$, a backward error analysis will be appropriate ($S_0 \neq S_0'$ and $S_1 \neq S_1'$ in the notation of section 2.3). A promising first step in this direction has been made in [27], where it was shown, at least in special cases, how to perturb the control points of two input surfaces so that their exact intersection coincides with a given computed solution. In the context of regularized Boolean operations, however, the surface perturbations must take account of the simultaneous intersection of , , , pairs of surfaces, and the general problem remains open. To illustrate, we began with a pair of p-curves in (u-v)-space (see Figure 6), corresponding to the intersection between $\boldsymbol{F}$ and the plane $z = 21$. These p-curves (obtained by modifying the output of D2PCUT and D2CLXT) are quadratics. It was then possible to make perturbations in the third components of the $\boldsymbol{P}_{ij}$ which maintain the symmetry of $\boldsymbol{F}$ with respect to the plane $y = 0$ and for which the image of the p-curves is the , , intersection of the perturbed $\boldsymbol{F}$ and a perturbed version of the plane $z = 21$:

$$\begin{aligned}
\boldsymbol{P}^3_{0j} &\leftarrow \boldsymbol{P}^3_{0j} + 0.824\ldots, \quad j = 0, 3,\\
\boldsymbol{P}^3_{0j} &\leftarrow \boldsymbol{P}^3_{0j} - 0.747\ldots, \quad j = 1, 2,\\
\boldsymbol{P}^3_{1j} &\leftarrow \boldsymbol{P}^3_{1j} + 0.863\ldots, \quad j = 0, 3,\\
\boldsymbol{P}^3_{1j} &\leftarrow \boldsymbol{P}^3_{1j} + 0.654\ldots, \quad j = 1, 2.
\end{aligned}$$

The perturbed version of the plane $z = 21$ is a ruled surface: the perturbation is defined by a piecewise polynomial, where each polynomial is of degree 6 in a single parameter θ, $0 \leq \theta \leq 1$, and for each u the surface is constant as a function of v. The coefficients of the polynomials are bounded in magnitude by .00057.

The preimage of the intersection between $\boldsymbol{F}$ and the plane $x = 10/3$ is the straight line $u = \frac{1}{3}$ in (u-v)-space, and the first components of the $\boldsymbol{P}_{ij}$ can be perturbed (simultaneously with the perturbations of the third components, described above) in order to ensure that the image of the straight-line p-curve $u = u_o$, $u_o \cong \frac{1}{3} - 1.8 \cdot 10^{-5}$,

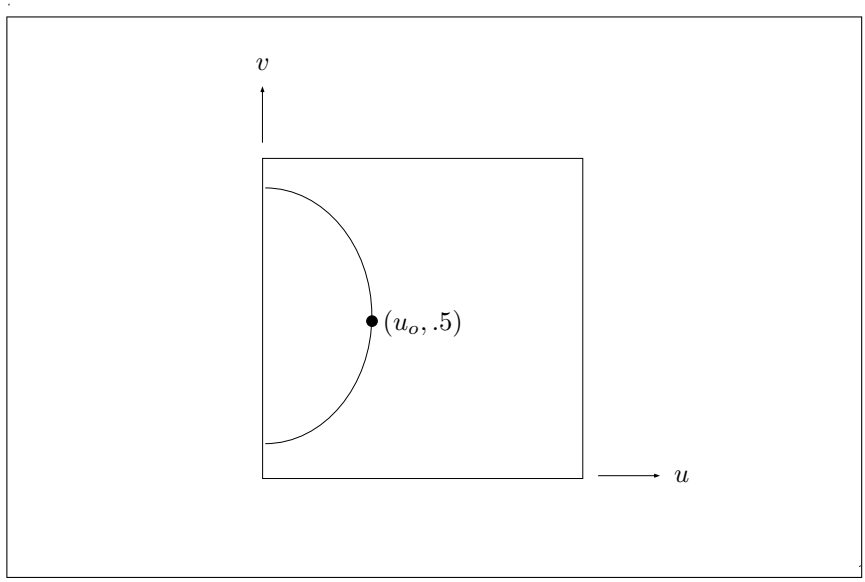


FIG. 6. *Preimage of the intersection with the plane* $z = 21$.

is the intersection of the perturbed $\boldsymbol{F}$ and a perturbed plane. In this case the perturbations are

$$\boldsymbol{P}^1_{0j} \leftarrow \boldsymbol{P}^1_{0j} + \frac{30u_o - 10}{3u_o - 3}, \quad \boldsymbol{P}^1_{1j} \leftarrow \boldsymbol{P}^1_{1j}, \quad j = 0, 1, 2, 3,$$

and the perturbed plane remains a perfect plane.

The simultaneous perturbations just described define sets S_0' and S_1' such that S_0' is close to S_0, S_1' is close to S_1, and the images of the p-curves in the parametric spaces define $S_0' \cap^* S_1'$. These sets S_0' and S_1' are, however, not necessarily sets such that $S_0' \models \Delta_0'$ and $S_1' \models \Delta_1'$, since we have not taken account of the computed intersection curves $\boldsymbol{b}(t)$. Nonetheless, note that the size of the Lipschitz constant obtained in the example ($L = 0.01114$) is promising, in the context of guaranteeing that perturbed versions $\boldsymbol{F} + \boldsymbol{\epsilon}$ and $\boldsymbol{F}' + \boldsymbol{\epsilon}'$ fit together seamlessly, as discussed at the end of section 3. To obtain such guarantees we need information about the Gauss map of $\boldsymbol{F} + \boldsymbol{\epsilon}$, given the Gauss map of $\boldsymbol{F}$.

Consider, for example, the case where $\boldsymbol{F}$ is a planar face and, for simplicity, suppose that the parametric domain is the plane itself:

$$\boldsymbol{F}(u, v) = Id(u, v),$$

and $\boldsymbol{F}(\boldsymbol{p}(t))$ is a simple closed planar curve $t \mapsto \boldsymbol{\alpha}(t)$. Suppose that we also have a similar curve, perturbed by $\boldsymbol{\epsilon}(t)$,

$$t \mapsto \begin{bmatrix} \boldsymbol{\alpha}(t) \\ 0 \end{bmatrix} + \boldsymbol{\epsilon}(t) \in R^3,$$

where it is known that $\|\boldsymbol{\epsilon}(t)\| \leq \epsilon_{max}$, and that for each component ϵ of $\boldsymbol{\epsilon}$,

$$|\epsilon(u_1, v_1) - \epsilon(u_2, v_2)| \leq L \cdot \|(u_1, v_1) - (u_2, v_2)\|$$

for (u_1, v_1), (u_2, v_2) on $t \mapsto \boldsymbol{\alpha}(t)$, as in section 3.2. Extending the mapping $(u, v) \mapsto \boldsymbol{\epsilon}(u, v)$, which is defined on the curve $t \mapsto \boldsymbol{\alpha}(t)$, to the region by $t \mapsto \boldsymbol{\alpha}(t)$, as described in section 3.2, we obtain a map $\boldsymbol{G} = \boldsymbol{F} + \boldsymbol{\epsilon}$:

$$\boldsymbol{G}(u, v) = \begin{bmatrix} u \\ v \\ 0 \end{bmatrix} + \boldsymbol{\epsilon}(u, v).$$

Assume that we have information about the normals of the patches adjacent to $\boldsymbol{F}$. In order to guarantee that $\boldsymbol{G}$ does not have extraneous intersections with any adjacent patch, we must bound the angle between the unit normal $\boldsymbol{n} = \boldsymbol{n}(u, v)$ of $\boldsymbol{G}$ and the unit normal $(0, 0, 1)$ of the planar surface $\boldsymbol{F}$.

We have $\frac{\partial \boldsymbol{G}}{\partial u} = (1, 0, 0) + \frac{\partial \boldsymbol{\epsilon}}{\partial u}$ and $\frac{\partial \boldsymbol{G}}{\partial v} = (0, 1, 0) + \frac{\partial \boldsymbol{\epsilon}}{\partial v}$. Computing $\frac{\partial \boldsymbol{G}}{\partial u} \times \frac{\partial \boldsymbol{G}}{\partial v}$ to obtain the surface normal of $\boldsymbol{G}$, normalizing it, and using $(1 + x)^{-\frac{1}{2}} = 1 - \frac{x}{2} + \frac{3}{8}x^2 + O(x^3)$, we obtain a vector $\boldsymbol{n}$ whose third component is no smaller than $1 - L^2 - O(L^3)$, since $\boldsymbol{\epsilon}$ is piecewise differentiable and each component of $\frac{\partial \boldsymbol{\epsilon}}{\partial u}$ and $\frac{\partial \boldsymbol{\epsilon}}{\partial v}$ is bounded by L. Taking $L = 0.01114$, this provides a lower bound of $1 - L^2 = 0.99987$ on the inner product between $\boldsymbol{n}$ and $(0, 0, 1)$ and an upper bound of $\sqrt{2}L = 0.0158$ radians on the angle between the two vectors.

The example illustrates that in order to obtain useful bounds precluding extraneous intersections between adjacent patches, it is important that the numerical method control not only the size of $\boldsymbol{\epsilon}$ but also the size of its derivative (and L).

6. Conclusion. In this paper a definition for the actual subset of R^3, defined by the solid-model representations that are used in practice, was proposed. Such representations are not only in error, they contain intrinsic inconsistencies. The proposed definition for the subset defined by such data, which was called a Quasi-NURBS set, produces patches that are perturbed by an amount smaller than the error already present along the boundaries of the given trimmed patches and are constrained by the same Lipschitz constants that describe the variation of the given trimmed patches along their boundaries.

The paper gave the basic results necessary to provide a framework for error analysis when using the representations referred to above. The discussion included the question of ill-condition and when it is appropriate to invoke methods outside the scope of numerical analysis. It also illustrated certain of the steps that must be taken to obtain a rigorous proof of stability for regularized Boolean operations on solids. The discussion and example (section 5) showed that much remains to be done to achieve this latter goal. On the other hand, other results may be more tractable. For example, the rigorous formulation of well-definition (section 3.2) may make it possible to prove that well-formedness (verified a posteriori, as discussed in section 3.3) is sufficient to guarantee the correctness of surface meshing algorithms, which are empirically very reliable when the underlying solid model does not contain errors [3]. Such a result would have significant practical importance.

Acknowledgments. The authors are grateful to T. J. Peters, who made many useful comments, and to J. Bisceglio, who provided many of the numerical results mentioned in section 5. Responsibility for any errors or omissions lies solely with the authors.

REFERENCES

[1] N. Patrikalakis and T. Maekawa, *Shape Interrogation for Computer Aided Design and Manufacturing*, Springer, Berlin, 2002.

[2] C. M. Hoffmann, *Robustness in geometric computations*, J. Comput. Inform. Sci. Engrg., 1 (2001), pp. 143–156.

[3] R. Farouki, *Closing the gap between CAD model and downstream application*, SIAM News, 32, No. 5 (1999).

[4] F. Yamaguchi, *Computer-Aided Geometric Design*, Springer, Berlin, 2002.

[5] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, UK, 1965.

[6] P. Deuflhard and A. Hohmann, *Numerical Analysis in Modern Scientific Computating*, 2nd ed., Springer, New York, 2003.

[7] W. M. Kahan, *A survey of error analysis*, in Information Processing '71 (Ljubljana, 1971), North–Holland, Amsterdam, pp. 1214–1239.

[8] A. A. G. Requicha, *Representations for rigid solids: Theory, methods and systems*, ACM Comput. Surveys, 12 (1980), pp. 437–464.

[9] L.-E. Andersson, T. J. Peters, and N. F. Stewart, *Selfintersection of composite curves and surfaces*, Comput. Aided Geom. Design 1, No. 5 (1998), pp. 507–527.

[10] L.-E. Andersson, T. J. Peters, and N. F. Stewart, *Equivalence of topological form for curvilinear geometric objects*, Internat. J. Comput. Geom. Appl., 10 (2000), pp. 609–622.

[11] M. Boyer and N. F. Stewart, *Imperfect-form tolerancing on manifold-objects: A metric approach*, Int. J. Robot. Res., 11 (1992), pp. 482–490.

[12] *ACIS 3D Toolkit*, Spatial Technology, Boulder, CO, 1998.

[13] *STEP International Standard, ISO* 10303-42*, Industrial Automation Systems and Integration—Product Data Representation and Exchange—Part* 42, International Organization for Standardization (ISO), Reference Number ISO 10303-42: 1994(E), First Edition 1994-12-15, 1997.

[14] H. Qin and D. Terzopoulos, *Triangular NURBS and their dynamic generalizations*, Comput. Aided Geom. Design, 14 (1997), pp. 325–347.

[15] L. Piegl and W. Tiller, *The NURBS Book*, Springer, New York, 1997.

[16] *DT_NURBS Spline Geometry Subprogram Library User's Manual, Version* 3.6, Carderock Division Naval Surface Warfare Center, Bethesda, MD, 20084-5000, CARDEROCKIV-94/000, 1998.

[17] T. A. Grandine and F. W. Klein, *A new approach to the surface intersection problem*, Comput. Aided Geom. Design, 14 (1997), pp. 111–114.

[18] G. Shen, T. Sakkalis, and N. M. Patrikalakis, *Boundary representation model rectification*, Graphical Models, 63 (2001), pp. 177–195.

[19] T. W. Sederberg and E.-W. Chionh, *The algebra and geometry of curve and surface intersection*, in Geometric Modeling and Processing—Theory and Applications, IEEE Computer Society, Los Alamitos, CA, 2002, pp. 11–20.

[20] H. Whitney, *Analytic extensions of differentiable functions defined in closed sets*, Trans. Amer. Math. Soc., 36 (1934), pp. 63–89.

[21] G. Aronsson, *Extension of functions satisfying Lipschitz conditions*, Ark. Mat., 6 (1967), pp. 551–561.

[22] W. Böhm, G. Farin, and J. Kahmann, *A survey of curve and surface methods in CAGD*, Computer Aided Geom. Design, 1 (1984) pp. 1–60.

[23] P. Volino and N. M. Thalmann, *Efficient self-collision detection on smoothly discretized surface animations using geometrical shape regularity*, Eurographics, 13 (1994), pp. C155–C164.

[24] L.-E. Andersson, N. F. Stewart, and M. Zidani, *Proof of a non-selfintersection conjecture*, Comput. Aided Geom. Design, 23 (2006), pp. 599–611.

[25] M. P. do Carmo, *Differential Geometry of Curves and Surfaces*, Prentice–Hall, Englewood Cliffs, NJ, 1976.

[26] T. Saito, G.-J. Wang, and T. W. Sederberg, *Hodographs and normals of rational curves and surfaces*, Comput. Aided Geom. Design, 12 (1995), pp. 417–430.

[27] X. Song, T. W. Sederberg, J. Zheng, R. T. Farouki, and J. Hass, *Linear perturbation methods for topologically consistent representations of free-form surface intersections*, Comput. Aided Geom. Design, 21 (2004), pp. 303–319.

SIAM J. SCI. COMPUT.
Vol. 29, No. 2, pp. 827–841
© 2007 Society for Industrial and Applied Mathematics

GEOMETRIC OPTIMIZATION OF THE EVALUATION OF FINITE ELEMENT MATRICES*

ROBERT C. KIRBY† AND L. RIDGWAY SCOTT‡

Abstract. This paper continues earlier work on mathematical techniques for generating optimized algorithms for computing finite element stiffness matrices. These techniques start from representing the stiffness matrix for an affine element as a collection of contractions between *reference tensors* and an element-dependent *geometry tensor*. We go beyond the complexity-reducing binary relations explored in [R. C. Kirby, A. Logg, L. R. Scott, and A. R. Terrel, *SIAM J. Sci. Comput.*, 28 (2006), pp. 224–240] to consider geometric relationships between three or more objects. Algorithms based on these relationships often have even fewer operations than those based on complexity-reducing relations.

Key words. finite element, variational form, optimized algorithm, graph theory

AMS subject classifications. 65N30, 65Y20, 68R05, 68R10

DOI. 10.1137/060660722

1. Introduction.

In our recent work, we have presented mathematical and computational techniques for the automatic generation of optimized code for evaluating local stiffness matrices for finite elements. These techniques are applicable to multilinear variational forms over affine elements using general basis functions. Beginning in [10], we suggested that the local evaluation of the stiffness matrix for multilinear forms for a single affine element could be written as contractions of a set of , with a single tensor encoding the geometry and coefficients.

This formulation allows us to generate code for the innermost part of a finite element computation. The FEniCS form compiler, FFC [11, 12], takes a variational form and approximating space as input. As output, it produces a function that maps the geometry and material coefficients for any element in an unstructured mesh to the element stiffness matrix. It also generates the higher-level code that loops over all elements in a mesh, computing the element stiffness matrices and inserting them into a global sparse matrix. The current implementation of FFC is fully functional for multilinear forms with arbitrary order Lagrange elements, and works for general unstructured meshes in one, two, or three space dimensions. In [12], we show how the technique can be applied to more general variational forms, curved elements, and $H(\mathrm{div})$ or $H(\mathrm{curl})$ elements. Although our techniques can be applied to quadrilateral or hexahedral meshes, spectral element techniques that work dimension by dimension are probably more effective.

The innermost computational kernel of the FFC-generated code is the collection of tensor contractions on each element of a mesh. Since this code is a single function executed over all elements of a mesh, it makes sense to find optimizations that allow us to perform the computations for an arbitrary element as efficiently as possible.

*Received by the editors May 24, 2006; accepted for publication (in revised form) October 11, 2006; published electronically April 24, 2007.

http://www.siam.org/journals/sisc/29-2/66072.html

†Department of Computer Science, University of Chicago, Chicago, IL 60637-1581 (robert.c.kirby@ttu.edu). This author's work was supported by the U.S. Department of Energy Early Career Principal Investigator Program under award DE-FG02-04ER25650.

‡The Computation Institute and Departments of Computer Science and Mathematics, University of Chicago, Chicago, IL 60637-1581 (ridg@cs.uchicago.edu).

In [10], we noticed that relations exist between the reference tensors that allow one to perform all the contractions needed to compute a single element matrix in many fewer arithmetic operations than would otherwise seem necessary. Our rather crude heuristic for exploiting these dependencies allowed us to generate code that was notably faster than the standard method for forming local stiffness matrices on our test problems.

Then, in [13], we formalized part of the optimization process in [10] by introducing the concept of [illegible]. These are derived from metrics that model the cost of computing one of the tensor contractions once another contraction has been computed. This theory effectively translates optimizations based on pairwise dependencies between the reference tensors into a minimum spanning tree. While the strategy is quite effective at reducing the operation count for many variational forms, it does not address how to utilize relations between three or more reference tensors, such as when one reference tensor may be written as a linear combination of two others. In this paper, we study efficient algorithms for detecting such geometric relations and hence finding fast algorithms for building finite element stiffness matrices.

Our attempts to mechanize the optimization of evaluating finite element matrices bear strong resemblance to projects in other areas. In [2], a tool is developed for synthesizing high-performance FORTRAN code for high-dimensional structured tensor contractions arising in quantum chemistry. The authors search for ways of reducing operation counts by heuristics for common subexpression elimination. While our techniques seem to search for a wider class of arithmetic dependencies, their work pushes harder on data locality and other architecture and performance-related issues. Other attempts to mathematize the derivation of efficient numerical code include the FLAME project of van de Geijn and coworkers for dense linear algebra [8, 3] and the SMART project of Egner and Püschel in signal processing [6]. The latter project has developed a code that takes a (symbolic representation of) a matrix such as the discrete Fourier transform as input and factors it via representation theory into a short sequence of sparse structured matrices. Their techniques automatically replicate the Cooley–Tukey algorithm [4], but also find fast algorithms for other, more complicated, transforms. Many other projects, such as ATLAS [20] and SPIRAL [19], attempt to map a given algorithm as closely as possible to a given computer architecture.

We begin the remainder of this paper by reviewing the formulation of the local matrix computation as tensor contractions and stating the abstract computational problem to optimize in section 2. Then, we study our geometric optimization process in section 3. This includes both an expected quadratic-time algorithm for detecting coplanarity among a set of input vectors as well as using such relations to speed up the finite element computation. These ideas generalize readily to higher-dimensional dependencies, which we do at the end of this section. In section 4, we study the actual reduction of operation count generated by our algorithms. In many cases for both the Laplacian and weighted Laplacian, it gives reduction in arithmetic superior to that of complexity-reducing relations. This leads us to consider how the two approaches can be combined into a single optimization process. While we do not have a good answer to this yet, we discuss a simple heuristic and also phrase the issue as a combinatorial optimization problem in section 5. Throughout the paper, readers familiar with the subjects of finite linear spaces and ordered ternary relations may find many of the concepts somewhat familiar. To focus on the exposition of the algorithms and application to finite element computation, we defer these issues and make several remarks on these relationships in section 6.

2. Background. As in previous work, we let $\{V_i\}_{i=1}^r$ be a given set of finite-dimensional function spaces defined on a triangulation $\mathcal{T} = \{e\}$ of a domain $\Omega \subset \mathbb{R}^d$. We consider multilinear forms a defined on the product space $V_1 \times V_2 \times \cdots \times V_r$:

$$a : V_1 \times V_2 \times \cdots \times V_r \to \mathbb{R}. \tag{2.1}$$

In finite element methods, a is an integral of products of (derivatives) of its arguments.

We are interested in forming the element tensor A^K for each $K \in \mathcal{T}_h$. This element tensor depends on the basis chosen for each function space. We let $\{\varphi_i^1\}_{i=1}^{M_1}, \{\varphi_i^2\}_{i=1}^{M_2}, \ldots, \{\varphi_i^r\}_{i=1}^{M_r}$ be bases of $V_1, V_2, \ldots, V_r$ over K, and let $i = (i_1, i_2, \ldots, i_r)$ be a multi-index. The multilinear form a then defines a rank r tensor given by

$$A_i^K = a(\varphi_{i_1}^1, \varphi_{i_2}^2, \ldots, \varphi_{i_r}^r). \tag{2.2}$$

For affine elements such as triangles and tetrahedra, A^K is evaluated by mapping each K to a fixed reference element $\hat{K}$. As we have written extensively about this representation elsewhere, we recall the Laplacian as an example here. For other examples and a more general representation theorem for a class of variational forms, we refer the reader to [11, 12].

We let F_K be the affine map from $\hat{K}$ to K. We denote by x the coordinates on K and by X the coordinates on $\hat{K}$. We identify $\phi_i^j = \Phi_i^j \circ F_K^{-1}$, and by the chain rule we obtain

$$\begin{aligned} A_i^K &= \int_e \nabla\varphi_{i_1}^{e,1}(x) \cdot \nabla\varphi_{i_2}^{e,2}(x)\,\mathrm{d}x \\ &= \det F_e' \frac{\partial X_{\alpha_1}}{\partial x_\beta} \frac{\partial X_{\alpha_2}}{\partial x_\beta} \int_E \frac{\partial \Phi_{i_1}^1(X)}{\partial X_{\alpha_1}} \frac{\partial \Phi_{i_2}^2(X)}{\partial X_{\alpha_2}}\,\mathrm{d}X = A_{i\alpha}^0 G_e^\alpha. \end{aligned} \tag{2.3}$$

In [13], we abstracted the computation of A^K to the following scenario. Let $V \subset \mathbb{R}^d$ with $|V| = n < \infty$. By default, for any $g \in \mathbb{R}^d$, we may compute $\{v^t g : v \in V\}$ in nd multiply-add pairs (hence MAPs). However, the kinds of V that arise in finite element methods often possess special structure, so that the overall process may be performed on a general g in many fewer MAPs. In order to formalize an optimization procedure based on pairs of vectors in V, we introduced the notion of [illegible]. These are distance measures ρ such that if $\rho(u, v) = k$, then $u^t g$ may be computed using $v^t g$ in no more than k MAPs. For example, if u, v are colinear, then we may define $\rho(u, v) = 1$, since if $u = \alpha v$, then $u^t g = \alpha(v^t g)$. However, we left unstudied how to exploit relations between three or more members of V, such as when three vectors lie in a two-dimensional subspace.

The point of this work is to formalize and employ these geometric relations in optimizing the computation of element stiffness matrices. Throughout this work, we shall present illustrations of our geometric notions on a particular set of vectors. In this case, we will work with the vectors from the Laplacian reference tensor for cubic Lagrange elements over triangles. We will use the symmetry transformation described in [13] to map from $\mathbb{R}^4$ to $\mathbb{R}^3$ and work only with the triangular part of the matrix. A table of the vectors can be found in Table 2.1. In particular, we recall that many of the vectors are zero or else equal or colinear to other vectors. Hence, the number of vectors among which we need to consider coplanar relations is actually much smaller. In Table 2.2, we show the entries of the reference tensor after removing the zero entries and all but one of any collection of equal or colinear vectors.

TABLE 2.1
Reference tensor entries for Laplacian with cubic polynomials on triangles.

(0, 0)	0.42500	0.85000	0.42500	(3, 4)	-0.33750	0.67500	-0.33750
(0, 1)	-0.08750	-0.08750	0.00000	(3, 5)	0.33750	0.33750	0.00000
(0, 2)	0.00000	-0.08750	-0.08750	(3, 6)	0.33750	0.33750	0.00000
(0, 3)	-0.03750	-0.07500	-0.03750	(3, 7)	0.00000	0.33750	0.33750
(0, 4)	-0.03750	-0.07500	-0.03750	(3, 8)	0.00000	-1.68750	-1.68750
(0, 5)	0.03750	0.37500	0.33750	(3, 9)	-2.02500	-2.02500	0.00000
(0, 6)	0.03750	-0.63750	-0.67500	(4, 4)	1.68750	1.68750	1.68750
(0, 7)	-0.67500	-0.63750	0.03750	(4, 5)	-1.68750	-1.68750	0.00000
(0, 8)	0.33750	0.37500	0.03750	(4, 6)	0.33750	0.33750	0.00000
(0, 9)	0.00000	0.00000	0.00000	(4, 7)	0.00000	0.33750	0.33750
(1, 1)	0.42500	0.00000	0.00000	(4, 8)	0.00000	0.33750	0.33750
(1, 2)	0.00000	0.08750	0.00000	(4, 9)	0.00000	-2.02500	-2.02500
(1, 3)	0.03750	0.71250	0.00000	(5, 5)	1.68750	1.68750	1.68750
(1, 4)	0.03750	-0.30000	0.00000	(5, 6)	-0.33750	-1.35000	-1.35000
(1, 5)	-0.03750	0.00000	0.00000	(5, 7)	0.00000	-0.33750	0.00000
(1, 6)	-0.03750	0.00000	0.00000	(5, 8)	0.00000	-0.33750	0.00000
(1, 7)	0.33750	0.30000	0.00000	(5, 9)	0.00000	2.02500	0.00000
(1, 8)	-0.67500	-0.71250	0.00000	(6, 6)	1.68750	1.68750	1.68750
(1, 9)	0.00000	0.00000	0.00000	(6, 7)	0.00000	1.68750	0.00000
(2, 2)	0.00000	0.00000	0.42500	(6, 8)	0.00000	-0.33750	0.00000
(2, 3)	0.00000	-0.30000	0.03750	(6, 9)	-2.02500	-2.02500	0.00000
(2, 4)	0.00000	0.71250	0.03750	(7, 7)	1.68750	1.68750	1.68750
(2, 5)	0.00000	-0.71250	-0.67500	(7, 8)	-1.35000	-1.35000	-0.33750
(2, 6)	0.00000	0.30000	0.33750	(7, 9)	0.00000	-2.02500	-2.02500
(2, 7)	0.00000	0.00000	-0.03750	(8, 8)	1.68750	1.68750	1.68750
(2, 8)	0.00000	0.00000	-0.03750	(8, 9)	0.00000	2.02500	0.00000
(2, 9)	0.00000	0.00000	0.00000	(9, 9)	4.05000	4.05000	4.05000
(3, 3)	1.68750	1.68750	1.68750				

TABLE 2.2
Reference tensor entries for Laplacian using cubics on triangles, after zero entries and all but one entry from any set of equal or colinear entries have been removed.

(0, 3)	-0.03750	-0.07500	-0.03750	(2, 4)	0.00000	0.71250	0.03750
(0, 5)	0.03750	0.37500	0.33750	(2, 5)	0.00000	-0.71250	-0.67500
(0, 6)	0.03750	-0.63750	-0.67500	(2, 6)	0.00000	0.30000	0.33750
(0, 7)	-0.67500	-0.63750	0.03750	(2, 8)	0.00000	0.00000	-0.03750
(0, 8)	0.33750	0.37500	0.03750	(3, 3)	1.68750	1.68750	1.68750
(1, 3)	0.03750	0.71250	0.00000	(3, 4)	-0.33750	0.67500	-0.33750
(1, 4)	0.03750	-0.30000	0.00000	(4, 7)	0.00000	0.33750	0.33750
(1, 6)	-0.03750	0.00000	0.00000	(5, 6)	-0.33750	-1.35000	-1.35000
(1, 7)	0.33750	0.30000	0.00000	(5, 9)	0.00000	2.02500	0.00000
(1, 8)	-0.67500	-0.71250	0.00000	(6, 9)	-2.02500	-2.02500	0.00000
(2, 3)	0.00000	-0.30000	0.03750	(7, 8)	-1.35000	-1.35000	-0.33750

3. Geometric relations. As with complexity-reducing relations, we will consider the following abstract problem. Suppose that $\{v_i\}_{i=1}^n \equiv V \subset \mathbb{R}^d$ is a finite collection of vectors. We desire to derive a process by which $\{v_i^t g\}_{i=1}^n$ may be constructed efficiently. This process must take an arbitrary $g \in \mathbb{R}^d$ as an input. Effectively, this corresponds to a matrix-vector multiplication Ag, where the elements of V form the rows of A. Such a process may then be encoded in some low-level C or FORTRAN program and used as the innermost computation for building finite element matrices. As finding such a process is a "compile-time" process that need only be performed once per variational form and set of basis functions, one may be willing to perform a relatively expensive computation to obtain it.

If any $u, v \in V$ are equal or colinear, then $u^t g$ may be computed in zero or one operation using $v^t g$. If such relations occur in V, we need only search for coplanarity among the noncolinear elements of V. Similarly, when we search for higher-dimensional dependencies, we will want to filter out coplanarity.

DEFINITION 3.1. $V \subset \mathbb{R}^d$ [illegible] $k+1 \leq |V| < \infty$ [illegible] k-independent [illegible] $k+1$ [illegible]

DEFINITION 3.2. [illegible] $U \subset V$ [illegible] k [illegible] U [illegible] k-maximal [illegible] $U \subset U'$ [illegible] U [illegible] U' [illegible]

It stands to reason that if there are many linear dependencies among the vectors, this should allow us to construct $\{v_i^t g\}_{i=1}^n$ efficiently, for if $w = \alpha u + \beta v$, then $w^t g$ may be computed in only two MAPs once $u^t g$ and $v^t g$ are known. In the rest of the section, we study a process to make thorough use of these dependencies. We first examine the case of two-dimensional dependencies, and then move on to present the general case.

Now, suppose that V is 1-independent (no two vectors are colinear or equal). Two issues must be resolved to make use of coplanar dependencies in computing the dot products. First, we must enumerate all of the 2-maximal subsets of V. Second, we must find a way to construct $\{v_i^t g\}_{i=1}^n$ in such a way that as many of the dot products are computed by using linear relations as possible. Equivalently, as few dot products as possible should be performed explicitly.

3.1. Search algorithms. The first step one might take in finding the 2-maximal sets is obvious; loop over all triples of vectors. This process runs in $O(n^3)$ time, and the innermost check for coplanarity will typically require $O(d^3)$ operations (for example, Gaussian elimination or singular value decomposition). It then remains to determine the 2-maximal sets from the coplanar triples. To do this, let (G, E) be a graph where $G = \{\{i, j, k\} : \dim \operatorname{span}\{v_i, v_j, v_k\} = 2\}$ and an edge is drawn between $\{i, j, k\}$ and $\{i', j', k'\}$ if $|\{i, j, k\} \cap \{i', j', k'\}| = 2$. Then the connected components of this graph encode the 2-maximal sets. Actually, the vertices of a connected component are a set of sets of vector labels; one must take the union of all sets of labels in each connected component.

The 2-maximal sets may be enumerated much more effectively. By assumption, each pair of elements in V spans a plane. If we consider two pairs of vectors (these pairs may share a common vector), then we should be able to determine whether they correspond to the same plane or not.

Rather than searching for equality among the planes themselves, we will work a [illegible] condition for equality, leading to Algorithm 1 below. Let $\Pi : \mathbb{R}^d \to \mathbb{R}^3$ be some full-rank linear transformation. We compute $\{\Pi v_i\}_{i=1}^n$. Then, if $\{v_i, v_j\}$ and $\{v_{i'}, v_{j'}\}$ encode the same two-dimensional subspace, then of necessity so do $\{\Pi v_i, \Pi v_j\}$ and $\{\Pi v_{i'}, \Pi v_{j'}\}$. Conveniently, each is a pair of vectors in $\mathbb{R}^3$, so its cross product provides a vector normal to the spanned plane. If $\{v_i, v_j\}$ and $\{v_{i'}, v_{j'}\}$ span the same plane, then $n_{i,j} = \Pi v_i \times \Pi v_j$ and $n_{i',j'} = \Pi v_{i'} \times \Pi v_{j'}$ must be colinear. We have already seen in [13] that we may search for colinearity among a set of vectors in expected linear time by using hash tables. We form a table T whose keys n are the unique normal vectors to the pairs of projected vectors and whose values are sets of vector indices. Consequently, this entire process runs in expected (thanks to hashing) $O(n^2)$ time and has a lower dependence on d than the brute force algorithm. Furthermore, this hash table maps the normal vector to the set of all vectors that lie in the projected plane, so we do not need to find connected components as above.

Of course, some information is lost when mapping vectors from $\mathbb{R}^d$ down to $\mathbb{R}^3$.

It is possible that the mappings of vectors are coplanar while the vectors themselves are not. The candidates for 2-maximal sets are table values $T[n]$ containing more than two indices. We must first verify that each set of vectors is indeed 2-dependent. If it is not, we may recur to find its 2-independent subsets by selecting a new Π. In practice, however, the size of these sets is much smaller than $|V|$, so even using brute force followed by connected components is effective.

ALGORITHM 1. One phase of projection-based algorithm for determining 2-maximal sets. This process finds candidates for 2-maximal sets, and we verify and possibly recur on each.

Given $\{v_i\}_{i=1}^n \subset \mathbb{R}^d$, $n < \infty$, 1-independent.
Compute $\Pi \in \mathbb{R}^{3,d}$, a full-rank transformation.
Let T be an (initially empty) hash table mapping vectors in R^3 to sets of integers in $[1, n]$.
for i=1 to n **do**
  Compute Πv_i
end for
for $i = 1$ to n **do**
  for $j = i + 1$ to n **do**
    $n_{i,j} = \Pi v_i \times \Pi v_j$
    Normalize $n_{i,j}$ to have the unit ∞-norm and first nonzero entry positive.
  end for
end for
for $i = 1$ to n **do**
  for $j = i + 1$ to n **do**
    if $n_{i,j}$ is a key of T **then**
      $T[n_{i,j}] \leftarrow T[n_{i,j}] \cup \{i, j\}$
    else
      $T[n_{i,j}] \leftarrow \{i, j\}$
    end if
  end for
end for
Output $T[n]$ for each key n of T such that $|T[n]| > 2$.

It is interesting to know how close to being optimal this algorithm is. To know this requires knowing just how many common planes there can be. Consider a set of vectors in three dimensions, for simplicity, in the positive orthant ($x \geq 0$, $y \geq 0$, $z \geq 0$). Now consider the projection of the vectors on the triangle T defined by

$$x + y + z = M, \quad x \geq 0, y \geq 0, z \geq 0, \tag{3.1}$$

where $M > 0$ could be arbitrary, but we will take it to be sufficiently large to simplify our notation. Three such vectors lie in a plane through the origin if and only if the projections onto T are collinear. We now construct a set of n points with $\mathcal{O}(n^2)$ common planes.

Let k be a positive integer, and consider the points in the rectangular lattice

$$(i, j), \quad i = 1, \ldots, 2k, \quad j = 1, 2, 3. \tag{3.2}$$

We see that for each point with $j = 0$ we can associate k lines going through three points, and thus there are at least $2k^2$ common planes. Figure 3.1 shows an example with $k = 4$ showing only four of the eight sets of four planes for $i = 1, 2, 3, 4$.

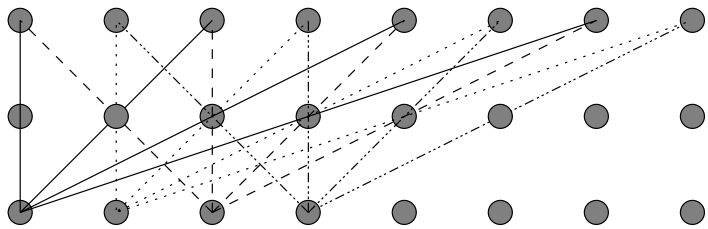


FIG. 3.1. *Example of lattice with $k = 4$. For each point on the lower line, there are exactly four planes. Only the planes for $i = 1, 2, 3, 4$ are shown.*

Since the number of planes to be determined is quadratic in the number of initial vectors, a quadratic algorithm for determining them is the best we would expect as a worst-case bound.

3.2. Generators. Now, let $\mathcal{P}$ be the set of all 2-maximal subsets of V. We would like to use V and $\mathcal{P}$ to infer an efficient process for computing $\{v_i^t g\}_{i=1}^n$ for arbitrary g. In order to do this, we will identify a (hopefully small) subset $S \subset V$ by which we may recursively construct the rest of the members of V via pairwise linear combinations. Once this construction is found, it will tell us how to efficiently compute the dot products. That is, we start with S and will compute $\{v^t g : v \in S\}$ by brute force. Then, we find all $v \in V - S$ with v in the span of two members of S. We then let S_1 be the union of all such v and S. This implies that we can compute each element of $\{v^t g : v \in S_1 - S\}$ in two MAPs. We then identify all $v \in V - S_1$ spanned by two members of S_1, letting S_2 be the union S_1 with all such v. Again, if $v \in S_2 - S_1$, then $v^t g$ may be computed in two MAPs. If S is chosen suitably, then this process will yield $S_i = V$ for some $i < \infty$. We will call such a set S a [illegible]. Then, the total arithmetic cost for computing $\{v_i^t g\}_{i=1}^n$ is no greater than $d|S|$ MAPs. The goal is then to find a generator with as few elements as possible.

THEOREM 3.3. [illegible] $g \in \mathbb{R}^d$ [illegible] $\{v_i^t g\}_{i=1}$ [illegible] $d|S| + 2|V - S|$ [illegible] S [illegible] V

We will initially develop these ideas geometrically rather than graph-theoretically, although later we will present the essential problem over a graph. However, our eventual goal for code generation is to find some sort of a [illegible] that tells us which vectors are constructed from which. The leaves (having no out-neighbors) are the members of S. All nodes not in S will have two out-neighbors. If v has two out-neighbors w, x, then $\{v, w, x\}$ must be 2-dependent; the computation of $v^t g$ will be performed using this linear relation rather than explicitly. A topological sorting of this graph such that each vertex follows its out-neighbors will allow all the dot products to be performed in an appropriate sequence. Straightline code for each dot product (corresponding to an entry of the element stiffness matrix) can then be generated. An example dependence graph is shown in Figure 3.2; this is for the vectors in Table 2.2 for the cubic Lagrange basis on triangles.

Without loss of generality, we will assume that every element of V is contained in at least one $P \in \mathcal{P}$. If some $v \in V$ does not lie in a plane with any two other items, then $v^t g$ must always be computed explicitly anyway. Such vertices are isolated points in a dependence graph, having no in- or out-neighbors.

We may refer to the above process of recursively taking linear combinations of S as a geometric [illegible].

DEFINITION 3.4. [illegible] $S \subset V$ [illegible] geometric 2-closure [illegible] S [illegible] $\bar{S}$ [illegible]

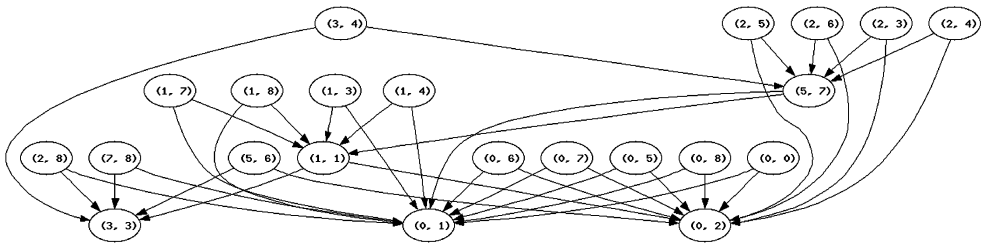


FIG. 3.2. *Generation graph for a* 1*-independent subset of vectors for the Laplacian using cubic Lagrange elements.*

$v \in S$ $v \in \bar{S}$ $v \in V$ $w, x \in \bar{S}$ $\{v, w, x\}$ 2 $v \in \bar{S}$

1. The term is appropriate, as the operation mapping S to its geometric 2-closure is an abstract closure operation. That is, the operation is increasing ($S \subseteq \bar{S}$), idempotent ($\bar{\bar{S}} = \bar{S}$), and monotone ($S \subseteq T \to \bar{S} \subseteq \bar{T}$).

DEFINITION 3.5. $\bar{S} = V$ S generates V generator $S' \subset S$ V S minimal generator S minimum generator

Our terminology here is intentionally analogous to the maximal and maximum independent set problems from algorithmic graph theory. An is a subset of a graph's vertices so that no two vertices are connected by an edge. A independent set is an independent set not contained in any larger independent set. A independent set is a maximal independent set with cardinality no less than any other maximal independent set. While simple efficient algorithms exist for finding a maximal independent set, finding a maximum independent set is NP-hard [9]. The algorithm we study below may have similar properties; while it might not always find a minimum generator, it finds a minimal one. The hardness of finding a minimum generator and the difference in size between the minimal generator that our algorithm finds and a minimum generator are all open questions; we will make an additional remark about this later.

Our algorithm is based on the of the 2-maximal sets.

DEFINITION 3.6. $\mathcal{P}$ 2 $\{v_i\}_{i=1}^n$ plane graph $(\mathcal{P}, E)$ 2 2 P_1 P_2 $|P_1 \cap P_2| = 1$

Equivalently, we may consider the plane graph to be a complete but weighted graph, where the edge weights indicate the number of items shared between two 2-maximal sets. This perspective will be more helpful in the next section, where we consider higher-dimensional geometric dependencies.

In certain simple circumstances, it is possible to infer or at least bound the size of a minimum generator based on properties of the plane graph. We present these results here.

THEOREM 3.7. $\mathcal{P}' \subseteq \mathcal{P}$ V $v \in V$ $P \in \mathcal{P}'$ V $2|\mathcal{P}'|$

. Let S contain two elements from each $P \in \mathcal{P}'$. Then clearly $\bar{S} = V$, since $\mathcal{P}'$ covers V. □

THEOREM 3.8. $\mathcal{P}' \subseteq \mathcal{P}$ V

V $1 + |\mathcal{P}'|$. Choose two elements $u, v \in P \in \mathcal{P}'$. Let $u, v \in S$. The closure of $\{a, b\}$ will contain all of P and one member of each neighbor of P. Select one additional member of each neighbor to add to S. Continue breadth first through the whole (connected) graph. □

These bounds are pessimistic; frequently much smaller generators can be found. It seems that the exact size depends not only on the connectivity of the plane graph, but also on how vectors are shared between planes. We leave this question aside and turn to algorithms for finding a generator.

3.3. Finding a minimal generator. Algorithm 2 is a simple greedy algorithm for finding a minimal generator. This works by assigning priorities to each 2-maximal set in $\mathcal{P}$. These priorities indicate how many items should be selected from that set to "close" it. Initially, these priorities will all be 2. When the priority of some P is two, then no vectors in it have been found by the process yet. If the priority is 1, then one vector has been found; selecting another allows the remainder of P to be added to the closure. At each step of our algorithm, we select a 2-maximal set P with minimal priority. Then, we add elements from P to build the generator set G. After updating the generator and what has been found, the priorities of neighboring planes are updated. Note that a simple modification of this algorithm builds a dependence graph. When items of a plane are selected as generators, they are added as roots of the dependence graph. The rest of the items of a plane are inserted with appropriate arrows.

ALGORITHM 2. Finding a minimal generator.

Given $V, \mathcal{P}$, and the plane graph
Let $G = \{\}$ be the (initially empty) generator.
Let Q be an empty priority queue.
Let $D = \{\}$ be the set of points generated by G.
for $P \in \mathcal{P}$ **do**
  $Q[P] \leftarrow 2$
end for
while Q not empty **do**
  Let P be an item from Q with minimal priority m.
  Remove P from Q.
  Select $\{v_{i_j}\}_{j=1}^{m}$ new generator elements from $P - D$.
  $G \leftarrow G \cup \{v_{i_j}\}_{j=1}^{m}$
  $D \leftarrow D \cup P$
  for P' adjacent to P in the plane graph **do**
    $Q[P'] \leftarrow \max(0, 2 - |P' \cap D|)$
  end for
end while

Algorithm 2 bears some similarity to some well-known algorithms. Like Dijkstra's [5] and Prim's [18] algorithms, it relies on a priority queue to make a locally optimal choice. However, it is not the vertices of the graph (the 2-maximal sets) but the members of those vertices that are of primary interest. Still, the similarity suggests that an optimality proof of this algorithm may be possible. On the other hand, we recall our remarks above regarding the similarity to maximum independent sets.

3.4. Exploiting higher-dimensional dependencies. The discussion of the earlier subsection readily generalizes to higher-dimensional dependencies. Interestingly, both Algorithms 1 and 2 are simple to extend. We explain both of these here.

To handle general k-dependencies, we assume as input a $(k-1)$-independent set. (The generator for a $(k-2)$-independent set is $(k-1)$-independent.) In practice, this means an iterative process by which we find a minimal generator for one order of linear dependence, remove the generated elements, and then proceed to the next higher order. Rather than pairs of vectors, we consider sets of k vectors, each of which spans a k-dimensional Euclidean space. We let $\Pi : \mathbb{R}^d \to \mathbb{R}^{k+1}$. Then we compute the generalized cross product that maps k elements of $\mathbb{R}^{k+1}$ to the unique (up to scaling) vector normal to the inputs. Equivalently, the full singular value decomposition could be used to find a normal vector. (It produces an orthonormal basis both for the column space of a matrix and its orthogonal complement.) Then these $O(n^k)$ normals are searched for colinearity via a hash table, and matches are verified and recurred upon as necessary.

The analogue of Algorithm 2 is likewise very similar. Instead of a plane graph, we make use of a "hyperplane graph" with edges weighted based on the length of intersection (between 0 and $k-1$, inclusive). The initial priorities are set to k rather than 2. The rest of the algorithm is exactly the same.

4. Empirical results. We present a study of using our geometric techniques for the Laplacian on tetrahedra, showing how many dependencies of each order we find. In Table 4.1, we report the number of vectors that are found to be equal or colinear to some other vector, and the number of vectors in the generator for each order

TABLE 4.1

Results of optimizing the evaluation of the Laplacian on tetrahedra using Lagrange elements of degrees two through four. For each degree k, the top portion of the table indicates the number of vectors (n), the length of each vector (d), the number of vectors that are zero (num zero), and the number of vectors removed from the search space because they were equal or colinear to another vector. The remaining three portions of the table describe the results of searching successively for linear dependencies of degrees two, three, and four. Each portion states the number of vectors with which the search process began, the number of vectors that have that order of dependency, the size of the generator (including independent elements), and the number of MAPs in the optimized algorithm. The final row indicates the number of MAPs in the algorithm generated using complexity-reducing relations, as described in [13].

	2	3	4
n	55	210	630
d	6	6	6
Num zero	0	0	0
Num equal	6	42	150
Num colinear	0	44	73
Size for 2-search	49	146	432
Num with 2-dependency	49	145	372
Generator size	6	8	67
MAPs	105	327	1112
Size for 3-search	6	8	67
Num with 3-dependency	0	0	41
Generator size	6	8	47
MAPs	105	327	1072
Size for 4-search	6	8	47
Num with 4-dependency	0	5	44
Generator size	6	7	18
MAPs	105	330	1045
CRR MAPs	101	370	1118

TABLE 4.2

Results of optimizing the first contraction stage for the weighted Laplacian on triangles. See Table 4.1 *for more details about the row and column labels. Blanks indicate computations that could not be completed.*

	2	3	4
n	63	165	360
d	6	10	15
Num zero	14	22	32
Num equal	15	23	33
Num colinear	2	2	2
Size for 2-search	33	119	294
Num with 2-dependency	32	67	112
Generator size	11	92	252
MAPs	110	910	3808
Size for 3-search	11	92	252
Num with 3-dependency	7	39	86
Generator size	9	78	217
MAPs	104	818	3394
Size for 4-search	9	79	
Num with 4-dependency	9	53	
Generator size	6	59	
MAPs	98	717	
CRR MAPs	115	683	3726

of dependency. We also report the number of MAPs in the optimized algorithm. Comparing to the results in [13], we see that geometry can be at least as effective at reducing arithmetic as complexity-reducing relations. On triangles, however, the vectors are already so short that reducing the cost by geometric relations is much less practical.

We remark that, going from third- to fourth-order dependencies, the number of MAPs increases. This is due to the fact that a vector with one nonzero entry was written as a linear combination of four other vectors; greater care in a production-level implementation would prevent this.

We also consider the weighted Laplacian on triangles as in [13],

$$a_w(v,u) = \int_\Omega w(x)\nabla v(x)\cdot\nabla u(x)\,\mathrm{d}x, \tag{4.1}$$

with reference and geometric tensors given by

$$A^0_{i\alpha} = \int_E \Phi_{\alpha_1}(X)\frac{\partial\Phi_{i_1}(X)}{\partial X_{\alpha_2}}\frac{\partial\Phi_{i_2}(X)}{\partial X_{\alpha_3}}\,\mathrm{d}X \tag{4.2}$$

and

$$G^\alpha_e = w_{\alpha_1}\det F'_e\frac{\partial X_{\alpha_2}}{\partial x_\beta}\frac{\partial X_{\alpha_3}}{\partial x_\beta}, \tag{4.3}$$

respectively. In [13], we remarked that G_e is the outer product of the coefficient with the geometric tensor for the standard Laplacian ($(G^L)_e$). With this in mind, we could perform all of the contractions (of larger tensors) as in the constant coefficient case and search for geometric relations. However, in this case, , , geometric relations between vectors were discovered. This suggests that geometric optimization is likely not as good a default optimization strategy as complexity-reducing relations. On the other hand, we may also perform the computations in stages as in [13]. Here, we

consider the case

$$A_i^e = \left(A^0_{i,(\alpha_1,\alpha_2,\alpha_3)} w_{\alpha_1}\right) \left(G^L\right)_e^{(\alpha_2,\alpha_3)}, \tag{4.4}$$

in which case we consider contractions with the coefficient first. This may be optimized, but the subsequent stage of contracting with G^L is performed at full cost. Note that we obtain some benefit, finding vectors that are zero, equal, or colinear. We also compare to Table 4.8 in [13] to see that geometric optimization with second order dependencies leads to a slightly better result than complexity-reducing relations for quadratics and almost as good for cubics. We also see that substantial additional improvements are made by searching for third- and fourth-order dependencies for cubics. Third-order dependencies are also significant for quartics, but even with our efficient search algorithms, fourth-order search was beyond the capacities of our prototype Python implementation.

5. Combined optimization. In [13], we reduced the construction of an optimized algorithm based on binary relations to a well-known graph construction. In the geometric context, the problem is equivalent to finding a generator. It is natural to look for an effective way of using both binary and geometric relations to give a greater overall reduction of arithmetic than using either separately. Rather than describing an algorithm for finding the exact optimum, we describe first a combinatorial structure, which, if it were found, would be the optimum. Then we describe a heuristic modification of Prim's algorithm that outperforms binary relations alone. Finally, we pose the problem in terms of an optimization problem over all permutations of V.

Essentially, we want to construct a rooted, weighted tree or forest with all the vectors as nodes. Each node either will be a leaf node, if the corresponding dot product is performed explicitly, or will have one or more neighbors, depending on whether the corresponding computation is performed via a complexity-reducing relation or a linear combination. Each node is also assigned a weight based on the cost of using either explicit computation, a complexity-reducing relation, or a linear dependence. Of all possible spanning forests, we desire to find one with minimal weight.

While such a structure probably cannot be found efficiently, the idea appropriately generalizes both the dependence graph and the minimum spanning tree. If we neglect all the linear relations and focus only on a complexity-reducing relation, a (rooted) minimum spanning tree exactly solves the problem. On the other hand, if we were to neglect the complexity-reducing relation, we would wind up with a dependence graph (augmented slightly if we had found equality or colinearity among vectors). We have implemented a straightforward generalization of Prim's algorithm in which each time a vector v is added to the tree, if it and another member of the tree are coplanar with a vector w not already added in the tree, a labeled edge is drawn from v and w. While this approach finds (provably and in practice) more efficient algorithms than using complexity-reducing relations alone, the current naive implementation and search through coplanar relations at each step makes the code impractical for anything beyond very simple problems at the present time.

Finally, we may pose the optimal computation as a combinatorial optimization problem over all permutations of V. In effect, the exact solution to this problem allows any previous computation to be employed at each step. To each ordering $\{v_i\}_{i=1}^{|V|}$ of V we can assign a weight. This weight is a sum over weights for each v_i. The weight

of v_1 is always $w(v_1) = m$. For $i > 1$, we define

$$w_R(v_i) = \begin{cases} 2, & \exists j, k < i : R(\{v_i, v_j, v_k\}), \\ m, & \text{otherwise}, \end{cases} \tag{5.1}$$

to indicate whether v_i is linearly dependent on two of its predecessors, and also define

$$w_\rho(v_i) = \min_{j<i} \rho(v_i, v_j) \tag{5.2}$$

to indicate how expensive it is to compute $v_i^t g$ using the complexity-reducing relation ρ. Finally, define

$$w(v_i) = \min(w_{R,i}, w_{\rho,i}). \tag{5.3}$$

Let $\mathcal{V}$ be the set of all permutations of V. Our goal is to solve the minimization problem

$$\min_{\mathcal{V}} \sum_{i=1}^{|V|} w(v_i). \tag{5.4}$$

6. Related mathematics. Our present context seems to be related to many other mathematical structures, especially when we consider the 2-maximal sets on a 1-independent set of Euclidean vectors. We describe these relations here, restricting ourselves to this case.

6.1. Finite linear spaces. Most strikingly, the set of vectors V and the set of all planes spanned by them are an instance of a [illegible] [1]. Finite linear spaces are a logical precursor to finite geometries; they specify an incidence relation between [illegible] and [illegible] such that any two points lie on a unique line, and each line contains at least two points. In our situation, the vectors in V are the points, and the 2-maximal sets are lines. (Formally, we must also include with the set 2-maximal sets all pairs of vectors that do not already lie in some 2-maximal set.) The research literature on finite linear spaces seems focused on fundamental questions of structure and relationships, such as whether spaces with particular properties exist, or when they may be embedded into finite affine or projective planes; algorithms over finite linear spaces seem less studied. We have been unable to locate any notion of geometric closures or generators such as we have described above; the idea may be new.

On the other hand, we have found reference to the plane graph used in section 3. Finite linear spaces, and the weaker notion of [illegible] (in which any two points may or may not be colinear) are naturally interpreted as hypergraphs. In [14], Klein and Margraf restate the Erdős–Faber–Lovász conjecture (see [7]) that the chromatic index of a linear hypergraph on v points is at most v, in terms of properties of the [illegible] of the hypergraph. This graph is defined exactly in the same way as our plane graph.

6.2. Ternary relations. We may consider coplanarity in Euclidean space as an unordered ternary relation, R. Given a set of three vectors, $\{x, y, z\}$, R gives truth if they are linearly dependent and falsity otherwise. Other work has made use of [illegible] ternary relations. The theory of ordered ternary relations has been used to provide a notion of [illegible] for some time [15]. Recent applications have also appeared in [17], which is based on a technical report [16] that provides a historical

survey of the use of ordered ternary relations as a type of geometry, as does [15]. In [15], ternary relations associated with a metric d are studied. More precisely, one can define $T(x, y, z)$ to hold if and only if $d(x, y) + d(y, z) = d(x, z)$. If we take $d(x, y)$ to be the acute angle between the lines x and y, then $T(x, y, z)$ is the coplanarity relation we consider here. However, the fact that y can be viewed as [illegible] x and z plays no significant role for us, so we have not further explored the theory of ordered ternary relations.

6.3. Topology. Our notion of geometric closure is, at least on the surface, different from topological closure. The union of two 2-maximal sets need not be geometrically closed (indeed, this is even beneficial, as it makes the generator much smaller and hence the algorithm we find more efficient). Therefore, the geometrically closed sets (and hence geometric closure) cannot be used to define a topology through closed sets. It may be possible to define some kind of filter or neighborhood base, but we have not pursued this further.

6.4. Matroids. As we deal heavily with linear independence (and more particularly linear dependence) among a set of vectors, it seems that there should be some connection to matroids [21]. Matroids also provide a notion of [illegible]. For vector matroids, the closure involves anything that is linearly dependent on something in the set. On the other hand, our notion of closure requires a low-dimensional linear dependence; only linear dependence on two vectors in the set is allowed (recursively) rather than linear dependence on any number of items. It may be possible in the future to interpret our context as a nonstandard matroid. Doing so would likely open up new algorithmic perspectives on the issue.

7. Conclusions and future work. Searching for linear dependence among the entries of the reference tensor can be a powerful technique for optimizing the evaluation of finite element stiffness matrices. The potential for improving our current implementation, integrating them with tools such as FFC [12, 11], and combining geometric and binary optimization is great. On the other hand, we have left several open issues, both practical and theoretical.

On the practical side, a more efficient implementation of the geometric search algorithms in C++ would allow us to study more detailed forms. Providing an interface to FFC would allow these optimizations to be used in practice, much as complexity-reducing relations are now available. While these issues are fairly straightforward, finding ways of tackling (or reformulating) the combinatorial optimization problem discussed above will be much more challenging.

On the theoretical side, we lack understanding about the structure of the minimum generator problem. If our algorithm does not always find a minimum, is finding the minimum truly difficult? How well does our greedy algorithm approximate the minimum? Perhaps more light will be shed on this problem by better understanding exactly what mathematical structures we are dealing with. As we saw in section 6, there is tantalizing similarity to finite linear spaces and matroids, among other structures. Making a more precise connection to an appropriate mathematical structure would clarify optimal generators greatly.

REFERENCES

[1] L. M. Batten and A. Beutelspacher, *The Theory of Finite Linear Spaces: Combinatorics of Points and Lines*, Cambridge University Press, Cambridge, 1993.

[2] G. Baumgartner, A. Auer, D. E. Bernholdt, A. Bibireata, V. Choppella, D. Cociorva, X. Gao, R. J. Harrison, S. Hirata, S. Krishnamoorthy, S. Krishnan, C.-C. Lam, Q. Lu, M. Nooijen, R. M. Pitzer, J. Ramanujam, P. Sadayappan, and A. Sibiryakov, *Synthesis of high-performance parallel programs for a class of ab initio quantum chemistry models*, Proc. IEEE, 93 (2005), pp. 267–292.

[3] P. Bientinesi, E. S. Quintana-Ortí, and R. A. van de Geijn, *Representing linear algebra algorithms in code: The FLAME application programming interfaces*, ACM Trans. Math. Software, 31 (2005), pp. 27–59.

[4] J. W. Cooley and J. W. Tukey, *An algorithm for the machine calculation of complex Fourier series*, Math. Comp., 19 (1965), pp. 297–301.

[5] E. W. Dijkstra, *A note on two problems in connexion with graphs*, Numer. Math., 1 (1959), pp. 269–271.

[6] S. Egner and M. Püschel, *Automatic generation of fast discrete signal transforms*, IEEE Trans. Signal Process., 49 (2001), pp. 1992–2002.

[7] P. Erdős, *On the combinatorial problems which I would most like to see solved*, Combinatorica, 1 (1981), pp. 25–42.

[8] J. A. Gunnels, F. G. Gustavson, G. M. Henry, and R. A. van de Geijn, *FLAME: Formal linear algebra methods environment*, ACM Trans. Math. Software, 27 (2001), pp. 422–455.

[9] R. M. Karp, *Reducibility among combinatorial problems*, in Complexity of Computer Computations, R. Miller and J. W. Thatcher, eds., Plenum Press, New York, 1972, pp. 85–104.

[10] R. C. Kirby, M. Knepley, A. Logg, and L. R. Scott, *Optimizing the evaluation of finite element matrices*, SIAM J. Sci. Comput., 27 (2005), pp. 741–758.

[11] R. C. Kirby and A. Logg, *A compiler for variational forms*, ACM Trans. Math. Software, 32 (2006), pp. 417–444.

[12] R. C. Kirby and A. Logg, *Efficient compilation of a class of variational forms*, ACM Trans. Math. Software, 33 (2007), to appear.

[13] R. C. Kirby, A. Logg, L. R. Scott, and A. R. Terrel, *Topological optimization of the evaluation of finite element matrices*, SIAM J. Sci. Comput., 28 (2006), pp. 224–240.

[14] H. Klein and M. Margraf, *On the linear intersection number of graphs*, preprint, 2003, http://arXiv.org/PS_cache/math/pdf/0305/0305073.pdf.

[15] R. Mendris and P. Zlatoš, *Axiomatization and undecidability results for metrizable betweenness relations*, Proc. Amer. Math. Soc., 123 (1995), pp. 873–882.

[16] K. Nehring, *A Theory of Qualitative Similarity*, Technical Report 10, U.C. Davis, Davis, CA, 1997; available online at http://www.econ.ucdavis.edu/working_papers/97-10.pdf.

[17] K. Nehring and C. Puppe, *Diversity and dissimilarity in lines and hierarchies*, Math. Social Sci., 45 (2003), pp. 167–183.

[18] R. Prim, *Shortest connection networks and some generalizations*, Bell System Tech. J., 36 (1957), pp. 1389–1401.

[19] M. Püschel, J. M. F. Moura, J. Johnson, D. Padua, M. Veloso, B. W. Singer, J. Xiong, F. Franchetti, A. Gačić, Y. Voronenko, K. Chen, R. W. Johnson, and N. Rizzolo, *SPIRAL: Code generation for DSP transforms*, Proc. IEEE, 93 (2005), pp. 232–275 (special issue on "Program Generation, Optimization, and Adaptation").

[20] R. C. Whaley and J. Dongarra, *Automatically tuned linear algebra software*, in Proceedings of SuperComputing 1998: High Performance Networking and Computing, 1998 (CD-ROM); available online at http://www.cs.utsa.edu/~whaley/papers/atlas_sc98.ps.

[21] H. Whitney, *On the abstract properties of linear dependence*, Amer. J. Math., 57 (1935), pp. 509–533.

SIAM J. SCI. COMPUT.
Vol. 29, No. 2, pp. 842–863
© 2007 Society for Industrial and Applied Mathematics

EFFICIENT MOMENT COMPUTATION OVER POLYGONAL DOMAINS WITH AN APPLICATION TO RAPID WEDGELET APPROXIMATION*

F. FRIEDRICH†, L. DEMARET‡, H. FÜHR‡, AND K. WICKER‡

Abstract. Many algorithms in image processing rely on the computation of sums of pixel values over a large variety of subsets of the image domain. This includes the computation of image moments for pattern recognition purposes, or adaptive smoothing and regression methods, such as wedgelets. In the first part of the paper, we present a general method which allows the fast computation of sums over a large class of polygonal domains. The approach relies on the idea of considering polygonal domains with a fixed angular resolution, combined with an efficient implementation of a discrete version of Green's theorem. The second part deals with the application of the new methodology to a particular computational problem, namely wedgelet approximation. Our technique results in a speedup of $O(10^3)$ by comparison to preexisting implementations. A further attractive feature of our implementation is the instantaneous access to the full scale of wedgelet minimizers. We introduce a new scheme that replaces the locally constant regression underlying wedgelets by basically arbitrary local regression models. Due to the speedup obtained by the techniques explained in the first part, this scheme is computationally efficient and at the same time much more flexible than previously suggested methods such as wedgelets or platelets. In the final section we present numerical experiments showing the increase in speed and flexibility.

Key words. wedgelets, platelets, image approximation, image moments, polygonal domains, discrete Green's theorem, digital lines

AMS subject classifications. 68U10, 65K10, 26B20, 52C99

DOI. 10.1137/050646597

1. Introduction.

This paper presents a new technique for the rapid computation of sums over varying image domains. The algorithms are particularly useful whenever sums of greyvalues over a large number of different image domains are needed for a fixed image. Techniques for this type of problem have been considered in the past mainly for the computation of image moments; see, e.g., [9, 10, 16, 13, 8]. These papers mostly deal with specific moments (such as geometric, Legendre, or Zernike moments), and often only for binary images; hence the algorithms usually depend on specific properties of the moments and cannot be directly adapted to our problem. However, a recurring theme in these papers which also plays a role here is the observation that the complexity of the algorithm may be reduced by appealing to a discrete version of Green's theorem. Recall that Green's theorem relates integration of a given function on a two-dimensional (suitably regular, connected, and simply connected) domain to the integration of associated functions over the boundary of the domain. In the discrete setting, one expects the reduction in dimension to result in a reduction of computational complexity.

*Received by the editors December 2, 2005; accepted for publication (in revised form) October 30, 2006; published electronically April 24, 2007. The first and second authors were funded in part by the European Research and Training Network HASSIP.
http://www.siam.org/journals/sisc/29-2/64659.html

†Computer Systems Institute, ETH Zentrum, CH-8092 Zürich, Switzerland (felix.friedrich@inf.ethz.ch).

‡Institute of Biomathematics and Biometry, GSF-National Research Center for Environment and Health, Ingolstädter Landstraße 1, 85764 Neuherberg, Germany (laurent@gsf.de, demaret@gsf.de, fuehr@gsf.de).

Our approach is based on the simple observation that for [illegible], the integration step boils down to a summation over the vertices of the polygon, once certain auxiliary functions are known. The discretization of this scheme relies on precomputed functions containing integrals over suitably chosen domains. Basically, each auxiliary function corresponds to a possible angle of an edge with the standard coordinate axis. This way, the [illegible] of the polygons which may be treated after the precomputation step can be prescribed in a direct and convenient way, at linear cost, for both computational and memory requirements. As an example, discussed briefly in Remark 2.5 below, after computing the auxiliary functions for the angles 0 and $\pi/2$, integrals over arbitrary rectangles are computable in constant time.

Remark 2.5 illustrates that the use of Green's theorem for summation purposes is based on cancellation. To make this effect prevail in the discrete context, a careful treatment of discrete lines and polygons is necessary. The design and efficient treatment of these discrete lines turns out to be the keystone of our approach.

As a somewhat related collection of techniques, one could also mention the boundary element method (BEM; see e.g. [11]), in which the problem of discretization of the Green function is addressed. The main goal of BEM is to obtain numerical solutions for some classes of boundary integral equations, arising, for example, in the modeling of the mechanical behavior of materials in bounded domains. Observe that this method consists of approximating exact solutions of the considered problems by using discretizations at the boundary, thanks to the Green theorem in a continuous setting. In contrast to this, our method is designed in a fully discrete setting: our "discrete Green formula" is exact and valid for the specific case of discrete summations.

The second half of our paper deals with the application of our techniques to wedgelet approximations. We present an implementation of an algorithm suggested by Donoho [3], and demonstrate that our techniques result in a drastic speedup, by comparison to the only publicly available implementation as part of the package BeamLab [4]. In a sense we solve a problem that had been left open in [3]: Wedgelet approximations are piecewise constant approximations of an image, where the pieces—the [illegible]—are obtained by splitting dyadic squares along straight lines. Approximations which minimize a certain functional are of particular interest (see section 3 below for details). Donoho proposed an algorithm for the rapid computation of the minimizers, resting on an assumption that was not further addressed in [3]—nor, to our knowledge, anywhere else in the literature. The assumption was that constant regression over arbitrary wedges could be efficiently implemented. Since constant regression depends on image moments of order zero, it is obvious that our techniques can be brought to bear on this problem. Hence we obtain a very efficient and transparent implementation of Donoho's algorithm, again with a conveniently prescribable angular resolution at linear cost. This implementation is available at [6].

Moreover, our implementation allows us to discuss regression models other than piecewise constant functions. The observation that Donoho's model can be extended to include piecewise approximation by (e.g.) affine functions has already been made by Willett and Nowak [15]. Our discussion below includes a quite general discussion of such schemes. The reason we present this generalization, which in itself is quite straightforward and may be considered by some as being of purely theoretical interest, is that we want to point out the potential of wedgelet methodology combined with our techniques. While with BeamLab even piecewise constant approximation of moderate size images is rather time-consuming, the speedup due to our techniques allows us to design more involved models, which can also be implemented efficiently.

2. Discrete Green's theorem and rapid summation over polygonal domains. The use of discrete versions of Green's theorem for image analysis purposes can be traced back to Tang [14]. We first recall this theorem, and its application to our central problem, for the continuous setting. Mathematically speaking, the following sequence is not necessary to understand the rest of the paper.

2.1. The continuous case. We restrict the discussion to bounded domains contained in the positive quadrant $\mathcal{Q}^+ = \mathbb{R}^+ \times \mathbb{R}^+$; its boundary $\partial\mathcal{Q}^+ = (\{0\} \times \mathbb{R}^+) \cup (\mathbb{R}^+ \times \{0\})$ will also play a prominent role. Given a measurable function $F : \mathbb{R}^2 \to \mathbb{R}$, and continuously differentiable $\phi : [a, b] \to \mathbb{R}^2$ written as $\phi(t) = (\phi_1(t), \phi_2(t))$, let

$$\int_\phi F dx = \int_a^b F(\phi(t))\phi_1'(t)dt$$

and

$$\int_\phi F dy = \int_a^b F(\phi(t))\phi_2'(t)dt.$$

The definition extends to piecewise differentiable curves in the obvious way. The following version of Green's theorem (cf. [2]) is relevant to our purposes.

THEOREM 2.1. $\Omega \subset \mathcal{Q}^+$ $\partial\Omega$ $F \in \mathcal{C}^0(\overline{\Omega})$ $Q : \mathcal{Q}^+ \to \mathbb{R}$

$$Q(x, y) = \int_0^y F(x, t)dt. \tag{2.1}$$

$$\int_\Omega F(x, y)dxdy = \int_{\partial\Omega} Q dx. \tag{2.2}$$

For purposes of rapid integration methods, the key feature of Green's formula is that it reduces the dimension of the integration domain by one. Let us describe how the formula simplifies further in the case of polygonal domains. Assume from now on that $\partial\Omega$ is a polygon, with nodes $z_1, \ldots, z_n, z_{n+1} = z_1 \in \mathbb{R}^2$. As Ω is simply connected, the polygon does not intersect itself. Given $z, \widetilde{z} \in \mathbb{R}^2$, we let $[z, \widetilde{z}]$ denote the straight line segment from z to $\widetilde{z}$, parameterized by $t \mapsto z + t(\widetilde{z} - z), t \in [0, 1]$.

For an angle $\vartheta \in]-\pi/2, \pi/2]$, define $v_\vartheta = (\cos(\vartheta), \sin(\vartheta))$. The convention $-\pi/2 < \vartheta \leq \pi/2$ entails that v_ϑ points to the right. Given $z \in \mathcal{Q}^+$, let $I_\vartheta(z)$ denote the intersection of the halfline $z + v_\vartheta \cdot \mathbb{R}^- := \{z + \lambda \cdot v_\vartheta : \lambda \leq 0\}$ with $\partial\mathcal{Q}^+$.

We define an auxiliary function K_ϑ on $\mathcal{Q}^+$ by

$$K_\vartheta(z) = \int_{[I_\vartheta(z), z]} Q dx. \tag{2.3}$$

The following lemma is then easily verified.

LEMMA 2.2. $z, \widetilde{z} \in \mathcal{Q}^+$ $z - \widetilde{z} \in \mathbb{R} \cdot v_\vartheta$

$$\int_{[z,\widetilde{z}]} Q dx = \operatorname{sign}(\langle \widetilde{z} - z, v_\vartheta \rangle) \cdot (K_\vartheta(\widetilde{z}) - K_\vartheta(z)). \tag{2.4}$$

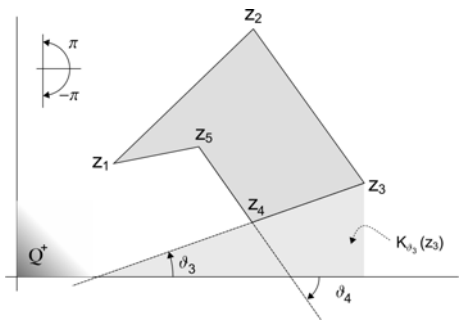


FIG. 2.1. *Illustration of polygon as used in Corollary* 2.3.

We note that for the degenerate case $\vartheta = \pi/2$, $K_\vartheta = 0$.

Plugging our definitions into Green's theorem yields the following result (observe the illustration in Figure 2.1).

COROLLARY 2.3. [illegible] Ω [illegible] $z_1, \ldots, z_n, z_{n+1} = z_1$ [illegible] $\vartheta \in]-\pi/2, \pi/2]$ [illegible] K_ϑ [illegible] (2.3). [illegible] ϑ_i [illegible] x [illegible] z_{i+1} [illegible] z_i [illegible]

$$\int\int_\Omega F(x,y)dxdy = \sum_{i=1}^{n} \operatorname{sign}(\langle z_{i+1} - z_i, v_{\vartheta_i} \rangle) \cdot (K_{\vartheta_i}(z_{i+1}) - K_{\vartheta_i}(z_i)) . \tag{2.5}$$

Note that the evaluation of the right-hand side involves $O(n)$ operations, [illegible] K_{ϑ_i} [illegible] This simple observation makes little sense in the continuous setting, but it contains the key to the algorithm which we will derive for the discrete setting.

A geometrically intuitive understanding of the corollary can be obtained by use of the following, easily verified, observation.

LEMMA 2.4. [illegible] $z \in \mathcal{Q}^+$ [illegible] $\vartheta \in]-\pi/2, \pi/2]$ [illegible] $\Omega_\vartheta(z)$ [illegible] $\partial\mathcal{Q}^+$ [illegible] $[I_\vartheta(z), z]$ [illegible] z [illegible]

$$K_\vartheta(z) = \int_{\Omega_\vartheta(z)} f(x,y)dxdy. \tag{2.6}$$

The lemma also covers the degenerate case $\vartheta = \pi/2$, as $\Omega_\vartheta(z)$ is then a line segment, a set of measure zero.

[illegible] 2.5. Using the lemma, we can recognize Green's theorem as an elegant and efficient way of handling cancellation: For illustration purposes, let us consider rectangles. Denote the four corners of the rectangular domain Ω by $p_1, \ldots, p_4$, numbered as in Figure 2.2. We claim that by the previous lemma,

$$\int_\Omega f(x,y)dxdy = K_0(p_2) - K_0(p_1) - K_0(p_3) + K_0(p_4). \tag{2.7}$$

To see this observe that Ω is obtainable by subtracting $\Omega_0(p_2) \cup \Omega_0(p_3)$ from $\Omega_0(p_1)$. However, by subtracting the integrals accordingly, the integral over the intersection $\Omega_0(p_2) \cap \Omega_0(p_3) = \Omega_0(p_4)$ is subtracted twice, and hence needs to be added again. On the other hand, observing that $K_{\pi/2} = 0$, we see that (2.7) is a special case of (2.5).

Naïvely one might expect that more complex—simply connected—domains could entail increasingly more complicated forms of cancellation. However, Corollary 2.3

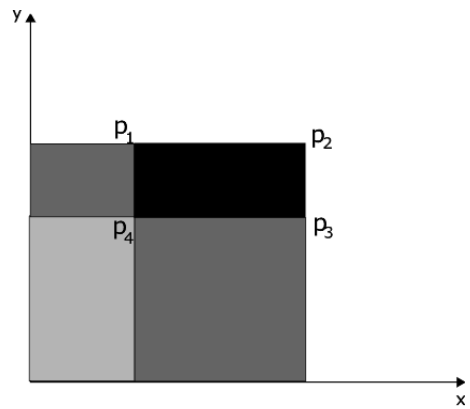


FIG. 2.2. *Illustration of the cancellation principle for the black rectangle, with use of* (2.7).

offers a simple and general way of handling all of them; in particular, the complexity is linear in the number of the vertices, regardless of the shape. The following section is dedicated to the description of a discrete analogue.

2.2. The discrete case. In this section we assume that $f : \mathcal{Q}^+ \cap \mathbb{Z}^2 \to \mathbb{R}$ is given, and describe versions of Green's theorem that allow the quick summation of function values over certain finite discrete domains $\Omega \subset \mathcal{Q}^+ \cap \mathbb{Z}^2$. In the following we restrict the presentation to [illegible] domains, which are defined in Definition 2.11. This restriction considerably simplifies the arguments and notation, while still covering the wedgelet case treated below. Note, however, that similar results can be formulated and proved for general simply connected domains (see [5]) and should also be useful in the general setting.

The key to an efficient discretization of the scheme described in the previous subsection consists of a suitable definition of straight lines, as follows.

DEFINITION 2.6. [illegible] $\alpha \in]-\pi/4, 3\pi/4]$ [illegible] $d_x := \cos\alpha$ [illegible] $d_y := \sin\alpha$ [illegible]

$$\delta_\alpha := \max\{|d_x|, |d_y|\} \qquad v_\alpha^\perp =: \begin{cases} (-d_y, d_x) & \text{[illegible] } |d_x| \geq |d_y|, \\ (d_y, -d_x) & \text{[illegible]} \end{cases}$$

[illegible] digital line through the origin in direction α [illegible]

$$L_\alpha^0 := \left\{ p \in \mathbb{Z}^2 : -\frac{\delta_\alpha}{2} < \langle p, v_\alpha^\perp \rangle \leq \frac{\delta_\alpha}{2} \right\}. \tag{2.8}$$

[illegible] L_α^n [illegible] $n \in \mathbb{Z}$ [illegible]

$$L_\alpha^n := \left\{ p \in \mathbb{Z}^2 : \left(n - \frac{1}{2}\right)\delta_\alpha < \langle p, v_\alpha^\perp \rangle \leq \left(n + \frac{1}{2}\right)\delta_\alpha \right\}. \tag{2.9}$$

[illegible] n [illegible] line number [illegible] L_α^n

In the following let the mapping $\text{Round} : \mathbb{R} \to \mathbb{Z}$ be defined by

$$\text{Round}(x) = \max\left\{ i \in \mathbb{Z} : i \leq x + \frac{1}{2} \right\}.$$

The particular choice $\alpha \in]-\pi/4, 3\pi/4]$ corresponds to all lines with positive horizontal or vertical main direction and leads to the following result.

LEMMA 2.7. [illegible] L_α^n [illegible]

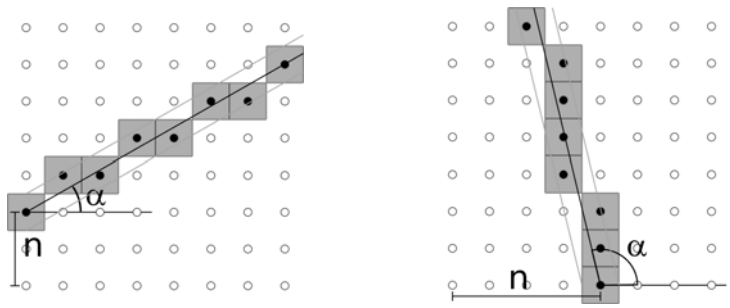


FIG. 2.3. *A flat and a steep discrete line.*

(a) flat line $d_x \geq d_y$ $y_\alpha(x) := \mathrm{Round}(x \cdot d_y/d_x)$

$$L_\alpha^n = (0,n) + \{(x, y_\alpha(x)) : x \in \mathbb{Z}\}, \tag{2.10}$$

(b) steep line $d_y > d_x$ $x_\alpha(y) := \mathrm{Round}(y \cdot d_x/d_y)$

$$L_\alpha^n = (n,0) + \{(x_\alpha(y), y) : y \in \mathbb{Z}\}. \tag{2.11}$$

. We prove (a) first. If $d_x \geq d_y$, then by $\alpha \in]-\pi/4, 3\pi/4]$ it holds that $|d_x| \geq |d_y|$, $d_x > 0$, and

$$\begin{aligned} L_\alpha^n &= \left\{(x,y) : \left(n - \frac{1}{2}\right) d_x < -d_y \cdot x + d_x \cdot y \leq \left(n + \frac{1}{2}\right) d_x\right\} \\ &= \left\{(x,y) : y \leq x \cdot \frac{d_y}{d_x} - n + \frac{1}{2} < y + 1\right\} \\ &= (0,n) + \left\{(x,y) : y \leq x \cdot \frac{d_y}{d_x} + \frac{1}{2} < y + 1\right\}. \end{aligned}$$

For (b) the proof can be done the same way. □

In other words, L_α^n is obtained by shifting L_α^0 by n in the vertical direction for flat lines and by shifts in the horizontal direction for steep line (see Figure 2.3). We remark that this definition of discrete lines is not new; it arises as output of the Bresenham algorithm used in computer graphics [1].

The following lemma notes the central property of digital lines: The digital lines corresponding to a fixed angle provide a partition of the image domain. In view of the observation, made in Remark 2.5, that Green's theorem—at least in the form discussed here—is primarily about cancellation, this simple fact will be crucial.

LEMMA 2.8. $\alpha \in]-\pi/4, 3\pi/4]$ $(L_\alpha^n)_{n\in\mathbb{Z}}$ $\mathbb{Z}^2$ $\mathbb{Z}^2 = \bigcup_{n\in\mathbb{Z}}^{\bullet} L_\alpha^n$

. The statement follows from (2.9), equivalently $L_\alpha^n = \{p \in \mathbb{Z}^2 : \langle p, v_\alpha^\perp\rangle \in](n-1/2)\delta_\alpha, (n+1/2)\delta_\alpha)]\}$, since the intervals $(](n-1/2)\delta_\alpha, (n+1/2)\delta_\alpha)])_{n\in\mathbb{Z}}$ form a partition of $\mathbb{R}$. □

We next want to define the discrete analogues of the auxiliary functions K_ϑ. For this purpose we number the points on a digital line L_α^n in such a way that following the points in ascending order, starting at the points numbered by zero, corresponds to going through L_α^n in direction (d_x, d_y), starting at a suitable element of $L_\alpha^n \cap \partial\mathcal{Q}^+$.

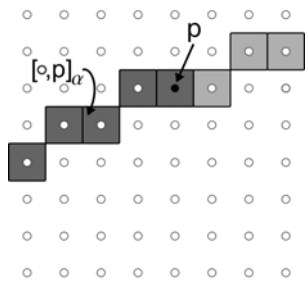


FIG. 2.4. *A line segment* $[\circ, p]_\alpha$.

Let $(x_0, y_0) \in \mathbb{Z}^2$, and let for flat lines the line number be given by $n = y_0 - y_\alpha(x_0)$ and for steep lines by $n = x_0 - x_\alpha(y_0)$; then $(x_0, y_0) \in L^n_\alpha$. Therefore the set of pixels on the line starting at $x = 0$ for flat lines and at $y = 0$ for steep lines and ending in $p = (x_0, y_0)$ is given by

$$[\circ, p]_\alpha := \begin{cases} (0, y_0 - y_\alpha(x_0)) + \bigcup_{x=0}^{x_0} \{(x, y_\alpha(x))\} & \text{if } d_x \geq d_y, \\ (x_0 - x_\alpha(y_0), 0) + \bigcup_{y=0}^{y_0} \{(x_\alpha(y), y)\} & \text{otherwise.} \end{cases}$$

In the following let the set of pixels below some pixel $p = (x_0, y_0)$ be given by $\mathcal{C}(p) := \{(x_0, y)\}_{y=0,\ldots,y_0}$. Moreover, consider the set of points between the line segment $[\circ, p]_\alpha$ (see Figure 2.4) and the x-axis defined by

$$\mathcal{W}_\alpha(p) := \bigcup_{q \in [\circ, p]_\alpha} \mathcal{C}(p).$$

Now for rectangular subsets $S = \{0, \ldots, w\} \times \{0, \ldots, h\}$ $(w, h \in \mathbb{N})$ of $\mathbb{Z}^2$, let [illegible] $f \in \mathbb{R}^S$, $f = (f(p))_{p \in S}$ be given. We denote the corresponding sum of values in columns below the point $p = (x_0, y_0)$ by $C(p) := \sum_{y=0}^{y_0} f((x_0, y))$, in rectangles left and below $p = (x_0, y_0)$ by $R(p) := \sum_{x=0}^{x_0} C((x, y_0))$, and below the piece of line $\mathcal{W}_\alpha(p)$ by

$$W_\alpha(p) := \sum_{q \in \mathcal{W}_\alpha(p)} f(q).$$

The following computation rules hold for columns and rectangles:

$$C((x, y+1)) = C((x, y)) + f((x, y+1)), \tag{2.12}$$

$$R((x+1, y)) = R((x, y)) + C((x+1, y)). \tag{2.13}$$

For the sums below a piece of line we obtain the following results. Let $p = (x, y) \in S$. For flat lines, i.e., $d_x \geq d_y$ with $p^+ = (x + 1, y - y_\alpha(x) + y_\alpha(x+1))$, it holds that

$$\mathcal{W}_\alpha(p^+) = \mathcal{W}_\alpha(p) \,\dot\cup\, \mathcal{C}(p^+),$$

and therefore, if $p^+ \in S$,

$$W_\alpha(p^+) = W_\alpha(p) + C(p^+). \tag{2.14}$$

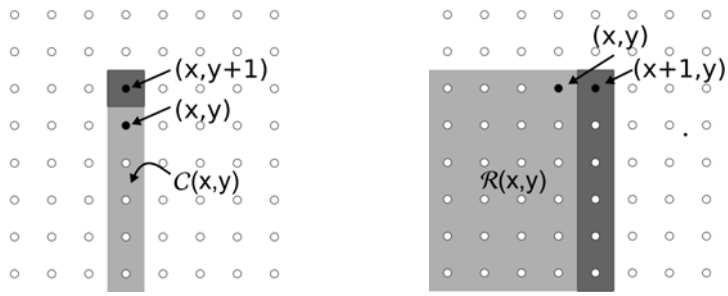


FIG. 2.5. *Summation rule for columns and rectangles.*

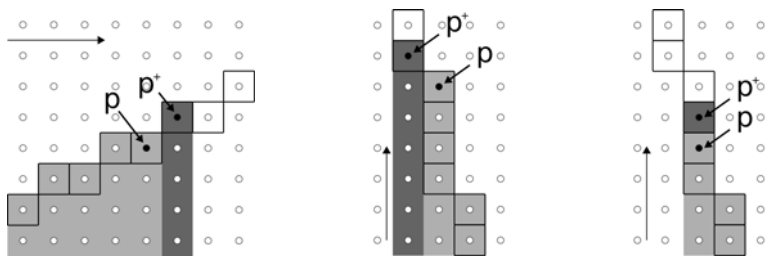


FIG. 2.6. *Summation rule for digital trapezes.*

For steep lines, i.e., $d_y > d_x$ with $p^+ = (x - x_\alpha(y) + x_\alpha(y+1), y+1)$, it holds that

$$\mathcal{W}_\alpha(p^+) = \mathcal{W}_\alpha(p) \,\dot\cup\, \begin{cases} \mathcal{C}(p^+) & \text{if } x_\alpha(y) \neq x_\alpha(y^+), \\ \{p^+\} & \text{if } x_\alpha(y) = x_\alpha(y^+), \end{cases}$$

and thus, if $p^+ \in S$,

$$W_\alpha(p^+) = W_\alpha(p) + \begin{cases} C(p^+) & \text{if } x_\alpha(y) \neq x_\alpha(y^+), \\ f(p^+) & \text{if } x_\alpha(y) = x_\alpha(y^+). \end{cases} \tag{2.15}$$

Note that the computation rules for the sums C and R are special cases of the one for W_α, namely the ones with $\alpha = \pi/2$ and $\alpha = 0$.

The computation rules for C, R, and W_α immediately lead to the following result.

LEMMA 2.9. [illegible] C R [illegible] W_α [illegible] $O(|S|)$ [illegible]. For the computation of C note that $C(p) = f(p)$ for all points p with $p^y = 0$. As illustrated by Figure 2.5, the rest of C can be computed using $C(x,y) = C(x,y-1) + f(x,y)$ in $O(|S|)$. Each line L^n_α that intersects S has at least one point on the lower or left boundary of S. Therefore there are at most $w + h$ such lines. For each line, the left or lower intersecting point p can be computed in $O(1)$; then $W_\alpha(p) = C(p)$. Using the rules (2.14) or (2.15), the values of W_α can then for each point on each line successively be computed in $O(1)$, as illustrated by Figure 2.6. Since the lines partition S this operation has to be performed $|S|$ times, and thus the computation of $W_\alpha(p)$ can be done in $O((w+h) + |S|) = O(|S|)$ steps. □

LEMMA 2.10. [illegible] $p, q \in \mathbb{Z}^2$ [illegible] $n \in \mathbb{Z}$ [illegible] $\alpha \in]-\pi/4, 3\pi/4]$ [illegible] $p, q \in L^n_\alpha$ [illegible]. For instance, we consider the angle α corresponding to $(d_x, d_y) = (q^x - p^x, q^y - p^y)$. We can restrict ourselves to the case of a flat line, the steep case being

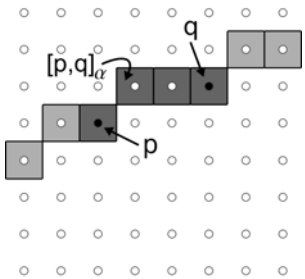


FIG. 2.7. *Segment* $[p,q]_\alpha$.

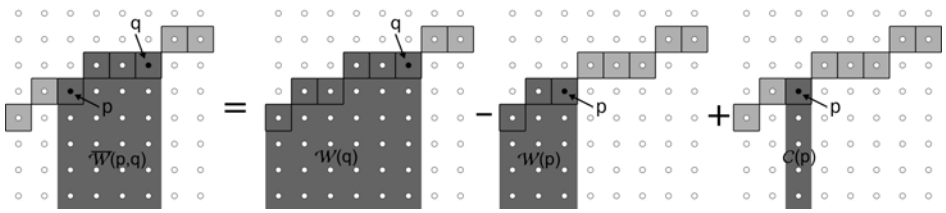


FIG. 2.8. *Summation over wedges.*

similar. Let r denote the point $(0, r^y) \in \mathbb{R}^2$ such that r lies on the continuous line through p and q. If we consider $n = Round(r_y)$, we have $n - r^y = y_\alpha(p^x) - p^x = y_\alpha(q^x) - q^x$; then $p, q \in L^n_\alpha$. □

Note that the discrete line L^n_α through two points p and q is in general not unique.

Let $p, q \in \mathbb{Z}^2$, $\alpha \in [-\pi/4, 3\pi/4[$ and $n \in \mathbb{Z}$ be given such that $p, q \in L^n_\alpha$. Assume that p is "before" q on the line; i.e., assume that $p^x \le q^x$ if $\alpha \in [-\pi/4, \pi/4[$, and that $p^y \le q^y$ if $\alpha \in]\pi/4, 3\pi/4]$. The discrete line segment between p and q, i.e., the set of points between p and q (including p and q) on the line L^n_α, is then given by

$$[p,q]_\alpha := [\circ, q]_\alpha \setminus [\circ, p]_\alpha \cup \{p\}.$$

(See Figure 2.7.) Accordingly, by $\overline{\mathcal{W}}_\alpha(p,q)$ we denote the [illegible] under $[p,q]_\alpha$, i.e., the set of points below this piece of line,

$$\overline{\mathcal{W}}_\alpha(p,q) = \bigcup_{r \in [p,q]_\alpha} \mathcal{C}(r).$$

As illustrated by Figure 2.8, for the sum over values of points below the piece $\overline{\mathcal{L}}_\alpha(p,q)$ it holds that

$$\sum_{r \in \overline{\mathcal{W}}_\alpha(p,q)} f(r) = W_\alpha(q) - W_\alpha(p) + C(p).$$

Now let two arbitrary points p, q on a line L^n_α, $\alpha \in]-\pi/4, 3\pi/4]$, be given. We denote the sum of values below the piece of line between p and q by

$$\overline{W}_\alpha(p,q) := \begin{cases} W_\alpha(q) - W_\alpha(p) + C(p) & \text{if } p^x \le q^x \text{ and } \alpha \in]-\pi/4, \pi/4], \\ W_\alpha(q) - W_\alpha(p) + C(p) & \text{if } p^y \le q^y \text{ and } \alpha \in]\pi/4, 3\pi/4], \\ W_\alpha(p) - W_\alpha(q) + C(q) & \text{otherwise.} \end{cases} \tag{2.16}$$

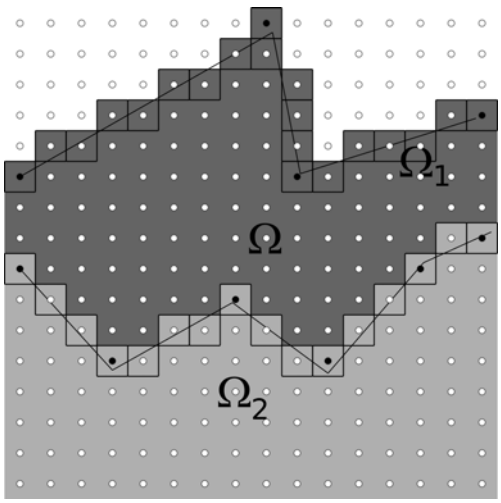


FIG. 2.9. *A simple polygonal domain.*

The next definition describes the class of domains to which we apply the summation techniques. In a first step, this is done by prescribing the upper boundary as a union of line segments and using the x-axis as lower boundary. More general lower boundaries can then be introduced by considering the set-theoretic difference of two such sets. See Figure 2.9 for an example.

DEFINITION 2.11. $p_0 \dots, p_n \in \mathcal{Q} \cap \mathbb{Z}^2$ $\alpha_1, \dots, \alpha_n \in]-\pi/4, 3\pi/4]$

1. $p_0^x < p_1^x < \cdots < p_n^x$
2. $1 \le i \le n$ $k_i \in \mathbb{Z}$ $p_{i-1}, p_i \in L_{\alpha_i}^{k_i}$
3. $[p_{i-1}, p_i]_{\alpha_i} \cap [p_i, p_{i+1}]_{\alpha_{i+1}} = \{p_i\}$ $1 \le i < n$

$\mathcal{D}(p_i, \alpha_i; i = 0, \dots, n)$

$$\mathcal{D}(p_i, \alpha_i; i = 0, \dots, n) = \bigcup_{i=1}^{n} \overline{\mathcal{W}}_{\alpha_i}(p_{i-1}, p_i). \tag{2.17}$$

$\Omega \subset \mathbb{N}^2$ simple polygonal domain of order n $\Omega_1 = \mathcal{D}(p_i, \alpha_i; i = 1, \dots, m)$ $\Omega_2 = \mathcal{D}(q_i, \vartheta_i; i = 1, \dots, k)$ $n = k+m$ $p_0^x = q_0^x$ $p_m^x = q_k^x$ $\Omega_2 \subset \Omega_1$ $\Omega = \Omega_1 \setminus \Omega_2$

Now the following discrete Green's theorem is easily proved.

THEOREM 2.12. Ω n $\Omega = \Omega_1(p_i, \alpha_i, i = 1, \dots, m) \setminus \Omega_2(q_i, \vartheta_i, i = 1, \dots, k)$

$$\sum_{r \in \Omega} f(r) = \sum_{i=1}^{m} \overline{W}_{\alpha_i}(p_{i-1}, p_i) - \sum_{i=1}^{m-1} C(p_i) - \sum_{i=1}^{k} \overline{W}_{\vartheta_i}(q_{i-1}, q_i) + \sum_{i=1}^{k-1} C(q_i). \tag{2.18}$$

. First, since $\Omega = \Omega_1 \setminus \Omega_2$ and $\Omega_2 \subset \Omega_1$,

$$\sum_{r \in \Omega} f(r) = \sum_{r \in \Omega_1} f(r) - \sum_{r \in \Omega_2} f(r).$$

Second, $\Omega_1 = \bigcup_{i=1}^{m} \overline{\mathcal{W}}_{\alpha_i}(p_{i-1}, p_i)$. By $p_i^x < p_{i+1}^x$ only neighboring line segments

intersect. It holds that

$$\overline{\mathcal{W}}_{\alpha_i}(p_{i-1},p_i) \cap \overline{\mathcal{W}}_{\alpha_{i+1}}(p_i,p_{i+1}) = \bigcup_{r\in[p_{i-1},p_i]_{\alpha_i}} \mathcal{C}(r) \cap \bigcup_{s\in[p_i,p_{i+1}]_{\alpha_{i+1}}} \mathcal{C}(s) = \mathcal{C}(p_i).$$

To see this observe that $[p_{i-1},p_i]_{\alpha_i} \cap [p_i,p_{i+1}]_{\alpha_{i+1}} = \{p_i\}$ for all $1 \le i < n$, and therefore $\mathcal{C}(p_i)$ is clearly contained in the intersection. If this inclusion were strict, then there would exist some $v \in \mathbb{Z}^2$ with $v^y > p_i^y$, $v^x = p_i^x$ with $v \in [p_{i-1},p_i]_{\alpha_i} \cap [p_i,p_{i+1}]_{\alpha_{i+1}}$, a contradiction to $[p_{i-1},p_i]_{\alpha_i} \cap [p_i,p_{i+1}]_{\alpha_{i+1}} = \{p_i\}$. Therefore

$$\sum_{r\in\Omega_1} f(r) = \sum_{i=1}^{m} \overline{W}_{\alpha_i}(p_{i-1},p_i) - \sum_{i=1}^{m-1} C(p_i).$$

The equivalent result holds for the sum concerning Ω_2. □

2.3. Implementation. We now summarize the results of the previous section with respect to implementation. Assume that we work on greylevel images $f \in \mathbb{R}^S$ defined over rectangular domains $S = [1,\dots,N_1] \times [1,\dots,N_2]$. We are interested in the computation of sums $\sum_{i\in\Omega} f(i)$ over simple polygonal domain Ω. To utilize Theorem 2.12 for a fast computation, we require the set of angles of Ω to stem from a finite set $\Theta \subset [-\pi/4, 3\pi/4]$. The main required ingredients are

(i) computation and storage of the auxiliary matrices C and $\overline{W}_\vartheta$ for all angles $\vartheta \in \Theta$,
(ii) determination of vectors $(p_{i-1},p_i,\vartheta_i)$, $1 \le i \le n$, defining the edges of the domain Ω,
(iii) use of formula (2.18) for the computation of $\sum_{i\in\Omega} f(i)$.

According to Lemma 2.9, step (i) has complexity $O(|S| \cdot |\Theta|)$, both in running time and memory consumption. Execution of step (ii) is required only if the parameters $(p_{i-1},p_i,\vartheta_i)$ $(1 \le i \le n)$ are not explicit. For example, in our applications of wedge divisions of a rectangle (cf. next section), we specify a wedge $\mathcal{W}$ by an intersection of a line $L_{n,\vartheta}$ with a rectangle R. To apply (iii) to the corresponding simple domain $\mathcal{W}$ the intersection points of R and $L_{n,\vartheta}$ on the boundary of R are required. According to Theorem 2.12, step (iii) requires no more than $2n$ additions, provided that the matrices from step (i) are known.

The following corollary summarizes the findings of this section.

COROLLARY 2.13. [illegible] $f : [1,\dots,N_1] \times [1,\dots,N_2] \to \mathbb{R}$ [illegible] $\Theta \subset]-\pi/4, 3\pi/4]$ [illegible] $O(|S| \cdot |\Theta|)$ [illegible]

$$\sum_{i\in\Omega} f(i)$$

[illegible] $O(n)$ [illegible] Ω [illegible] n [illegible] Θ

2.14. Note again that in practice the results of this section are useful only if the needed cumulative matrices W_{α_i} can be computed beforehand for all occurring polygons. This implies that a certain class of polygons are selected before computation of the integrals take place. In the next section we describe how such a class of polygons is implicitly selected by an arbitrary choice of angular resolution for wedgelets.

We close this section by pointing out that our scheme is [illegible]. Clearly, for large images and/or greyvalue ranges, overflow handling will have to be addressed, but the necessary length of the mantissa can easily be estimated.

3. Wedgelet approximations. In this section we apply the concepts described in section 2 to the specific case of wedgelet approximation. We are concerned first with the case of locally constant functions over dyadic wedgelet partitions. These were first proposed by Donoho [3], who derived approximation rates and showed asymptotic optimality of the wedgelet estimator, for functions of the horizon class and starlike domains. These functions are essentially characteristic functions of image subdomains with smooth boundary. It is also shown in [3] how the hierarchic structure of quad trees allows fast computation of minimizers.

Wedgelets may be viewed as a multiscale geometric dictionary for image representation. In recent years, several such constructions have been proposed (we refer the interested reader to [7]), containing a comparison of wedgelets, wavelets, and curvelets from a nonlinear approximation point of view.

To our knowledge, the only publicly available implementation of a wedgelet approximation algorithm can be found as part of BeamLab, a software package developed at Stanford University (see [4]). A runtime comparison of this implementation with ours can be found in section 5.2. A further source that deals with efficient wedgelet implementation is [12]. Here the central idea is to exploit a multiresolution structure of wedges: Wedges at one dyadic scale are the union of wedges at the next smaller scale, and one can use this in a way that is somewhat similar to our ideas.

However, the exposition in [12] is rather concise, and we have not been able to obtain an implementation of the described algorithms.

3.1. Dyadic square wedge partitions. In this subsection, we present the wedgelet partitions based on dyadic squares. For the sake of simplicity we restrict ourselves to discrete dyadic square domains of the form $S = [1, \dots, N] \times [1, \dots, N]$ with $N = 2^p$.

We adopt the standard definition for a [illegible] of the set S, as a set $\mathcal{P} = \{A_k\}_{k \in K}$, $A_k \subset S$, $\bigcup_{k \in K} A_k = S$, and $A_i \cap A_j = \emptyset$ if $i \neq j$. A subset $A_k \subset \mathcal{P}$ is called the atom of the partition.

The set of dyadic square partitions $\mathfrak{Q}$ can be defined with the following rules:

(1) $\mathcal{P}_0 = \{S\} \in \mathfrak{Q}$,

(2) if $\mathcal{P} \in \mathfrak{Q}$, $A \in \mathcal{P}$ and $|A| > 1$, then $\mathcal{P} \setminus \{A\} \cup_i \{A_i\} \in \mathfrak{Q}$,

where $A_i, i = 1, \dots, 4$, are formed by splitting A into four equally sized squares A_i. Since the image domain is a dyadic square, the atoms of a dyadic partition are also dyadic squares, of the form $[(k-1)2^i + 1, k2^i] \times [(l-1)2^i + 1, l2^i], i \in \{1, \dots, p\}, k, l \in \{1, \dots, 2^{p-i}\}$. Let $\mathcal{S}$ be the set of dyadic squares. Then $\mathcal{Q}$ consists of all partitions $\mathcal{P}$ of S such that $\mathcal{P} \subset \mathcal{S}$.

Now we introduce our [illegible]. This discretization scheme is a keystone for the development of efficient algorithms. The use of nonoverlapping lines allows an efficient implementation of discrete sums.

DEFINITION 3.1. [illegible] $A \in \mathcal{S}$ [illegible] L_ϑ^n [illegible] $(n, \vartheta) \in \mathbb{Z}^2 \times]-\pi/4, 3\pi/4]$ [illegible] $L_\vartheta^n \cap A \neq \emptyset$ $L_\vartheta^{n+1} \cap A \neq \emptyset$ [illegible] A [illegible] wedge [illegible] $\{W_{n,\vartheta}^1(A), W_{n,\vartheta}^2(A)\}$ [illegible]

$$W_{n,\vartheta}^1(A) = \bigcup_{k \leq n} L_\vartheta^k \cap A,$$

$$W_{n,\vartheta}^2(A) = \bigcup_{k > n} L_\vartheta^k \cap A.$$

Note that a wedge is a convex discrete polygon, which is a special case of simple domains in the sense of Definition 2.11, with at most five vertices. Indeed we can distinguish three cases: The wedge either has five vertices, or is a trapeze, or is a triangle. Thus the computation of one moment requires six auxiliary function values in the two first cases, and four function values in the triangular case.

It is now possible to properly define a

DEFINITION 3.2.

$$\{W^1_{n_k,\vartheta_k}(A_k), W^2_{n_k,\vartheta_k}(A_k)\},$$

$\vartheta_k \in]-\pi/4, 3\pi/4]$

3.2. Wedgelet approximation. Like in [3], we are interested in piecewise constant approximations of an image f over a class of dyadic wedge partitions. We follow a variational approach, with a balancing of segmentation complexity versus approximation accuracy. Here, we measure the complexity by the number of atoms in the partition.

More precisely, let $\mathfrak{W}$ be a set of dyadic wedgelet partitions of S. For $\lambda \in \mathbb{R}$ we select a partition $\hat{\mathcal{W}}_\lambda(f)$ from $\mathfrak{W}$ which satisfies

$$\hat{\mathcal{W}}_\lambda \in \operatorname*{argmin}_{\mathcal{W}\in\mathfrak{W}} \{\|f - \hat{f}_{\mathcal{W}}\|_2^2 + \lambda|\mathcal{W}|\}, \tag{3.1}$$

where $\hat{f}_{\mathcal{W}} = \sum_{W\in\mathcal{W}} \overline{f}_W$ and

$$\overline{f}_W = 1_W \frac{1}{|W|} \sum_{i\in W} f(i)$$

denotes the function equal to the average of f over the atom W (which is either a pure wedge or a square), and 0 elsewhere.

Note that the pair $(\hat{\mathcal{W}}_\lambda, f_{\hat{\mathcal{W}}_\lambda})$ minimizes $(\mathcal{W}, g) \mapsto \|f - g\|_2^2 + \lambda|\mathcal{W}|$, where a pair $(\mathcal{W}, g)$ is such that g is piecewise constant over $\mathcal{W}$. It is a particular property of the chosen functional that the minimization can be split up into a minimization over the partition class and (local) minimization on each segment. (Recall that $\overline{f}_W$ is the classical least squares local approximation of $\|f - g\|_2^2$ over W.)

3.3. Algorithm. Let us now describe the algorithm for efficient minimization of (3.1) for a given partition class $\mathfrak{W}$. As we manipulate huge sets $\mathfrak{W}$, finding the optimal partition among this set may lead to high computational costs. These costs are dramatically reduced by exploiting the recursive structure of the dyadic partitions. Nevertheless, still a huge number of sums ("moments") over polygonal domains have to be computed when minimizing over the class of dyadic wedge partitions. In order to handle this, we introduce an algorithm which employs the principles of section 2 for efficient moment computation on the wedge domains.

We work now with a finite set $\Theta \subset]-\pi/4, 3\pi/4]$ of angles. This means that the condition $\vartheta \in]-\pi/4, 3\pi/4]$ is replaced by the condition $\vartheta \in \Theta$. In order to treat the horizontal lines, it is required that $0 \in \Theta$. Since dyadic partitions are obtained by successive division of squares into four squares, each dyadic partition $\mathcal{P}$ can uniquely be mapped to a quad tree T. Each node N of the tree T corresponds to a dyadic square $Q \subset S$, and the four child-nodes of N (if any) correspond to the four squares

generated by a quad split of Q. Each terminal node of the tree T corresponds to a square in $\mathcal{P}$ and vice versa. Each quad tree is obtained as a pruning of the finest quad tree whose leaves correspond to the pixels of S; in implementation the pruning is coded by marking nodes as terminal.

The minimization of the functional consists of two steps: first, computation of the optimal wedge split for each dyadic square, i.e., for each node in the tree, and second, for a given parameter $\lambda \geq 0$ the computation of the minimizer $\hat{\mathcal{W}}_\lambda$ by recursively pruning the quad tree, using the optimal wedge splits obtained in the first step. Here, it is important to note that only the second step depends on λ, and, as will be seen below, its complexity does not depend on the number of angles used in the first step.

The heuristics for the second step can be explained as follows. Given $\lambda \geq 0$, there are three possible cases to consider for the optimal wedge partition $\hat{\mathcal{W}}_\lambda$:

1. $\hat{\mathcal{W}}_\lambda = \{S\}$,
2. $\hat{\mathcal{W}}_\lambda = \{W_1, W_2\}$ for the optimal two wedges W_1, W_2 stored at the root node,
3. $\hat{\mathcal{W}}_\lambda = \bigcup_{i=1,\dots,4} \hat{\mathcal{W}}^i_\lambda$, where each $\hat{\mathcal{W}}^i_\lambda$ is the optimal wedge partition of the dyadic subsquare Q_i, and $Q_1, \dots, Q_4$ are obtained by a quad split of S.

Hence a minimization algorithm consists of comparing the scores induced by the three cases; in the first case the score is local approximation error plus λ, in the second case it is local approximation error plus 2λ, and in the third case it is the sum of the four scores for $\hat{\mathcal{W}}^i_\lambda$, $i = 1, \dots, 4$. Observe that the third case requires a recursive application of the minimization algorithm.

In more precise terms the full minimization algorithm is described as follows:

1. Create empty tree. Initialize each node with approximation error $= \infty$.
2. Computation of optimal wedge splits:
   (a) Compute the vertical sum matrix C using (2.12) and $\overline{W}_0$ using (2.16).
   (b) For each $\vartheta \in \Theta$
      i. compute the auxiliary matrix $\overline{W}_\vartheta$ using (2.16),
      ii. for each dyadic square Q and each n compute the approximation error ε and associated mean values obtained by splitting Q along the line L^n_ϑ. If the induced approximation error is below the stored value at the corresponding node N, then store mean, error, angle ϑ, and line number n at node N.
3. Computation of the minimizer $\hat{\mathcal{W}}_\lambda$ and the associated score, for instance using a straightforward recursive implementation of the above procedure. For a bottom-up version of the algorithm, see [3]. In any case, the algorithm crucially relies on the availability of the optimal wedge splits of all dyadic squares, as provided by step 2.

The first step takes constant time. The complexity of step 2(a) is $O(4^p)$; the same for step 2(b)i, to be performed for each angle. The second step visits every wedge induced by the angles in Θ exactly once, and performs a constant time operation for each. Hence its complexity is linear in the number of wedges induced by Θ. Now, for each angle and each dyadic square q with sidelength 2^j, the number of possible wedge splits of q is less than or equal to $\frac{1}{2}$ the number of boundary pixels. Summing over all 4^{p-j} dyadic squares, all scales, and all angles, we obtain the total of $O(|\Theta| \sum_{j=0}^{p} 4^{p-j} 2^j) = O(|\Theta| 4^p)$ such wedges. The final step visits each dyadic square once, and thus requires $O(4^p)$ operations.

Moreover, we observe that the memory used for storing the lookup tables in step 2(b) can be reallocated; hence the overall memory requirement is only $O(4^p)$.

We have thus obtained the following result.

THEOREM 3.3. [illegible] f [illegible] $2^p \times 2^p$ [illegible] $\Theta \subset]-\pi/4, 3\pi/4]$ [illegible] (4.3) [illegible] $O(4^p \cdot |\Theta|)$ [illegible] $O(4^p)$

[illegible] 3.4. In some ways, our description of wedgelets deviates from Donoho's original exposition. The chief difference is in the parametrization of lines: In Donoho's paper they are parametrized by pairs of points on the boundary, and in our case by one point on the boundary and an angle. For the validity of certain approximation-theoretic results, Donoho requires that the spacing of the boundary points (and thus of the lines) become finer and finer as the resolution increases (see [3, Remark 9.1.1]).

It should be pointed out that our spacing of one pixel between two neighboring lines, as prescribed by Definition 2.6, contradicts this requirement. In principle, our methods can be adapted to this finer spacing, by suitable adjustment of the definition of digital lines; to some extent the finer spacing could also be approximated by simply increasing the set of admissible angles. In any case, it is not altogether clear to us whether this adjustment will actually provide a marked improvement in the approximation rate for digital images of a fixed resolution (of the kind investigated in section 5.2), since the subpixel precision requirement is mainly used to ensure good asymptotic behavior as the resolution tends to infinity.

4. Generalizations. In this section we show how the algorithm described in section 3 can easily be generalized to much larger classes of approximation models for which the computational complexity remains of the same order.

4.1. Wedge partitions with local regression models. The prototype model presented in section 3 was proposed first in a somewhat academic context: The goal was to get optimal approximations for some classes of piecewise constant functions with Hölder regular boundaries. For natural images, however, piecewise constant models lead to undesired block artifacts, especially for low resolutions, although these latter are often the target resolutions. Models based on richer local regression offer a first answer to this problem.

Instead of just considering optimal constant approximation on each atom of the dyadic wedge partition, we consider a finite set $\Phi = (\varphi_j)_{j \in R}$ of r linearly independent functions, $\varphi_j : S \to [0,1]$. Defining the set Φ globally and not locally is motivated by the computation of the moments on many overlapping atoms of several partitions. The functionalities are somewhat separated: The extraction of geometrical information is performed by the partitioning, whereas the local representation of the function is done by the regression on each atom. (We recall that, in our case, an atom consists merely of a wedge or of a square.) There is no theoretical restriction for the choice of Φ. In practice, however, discrete projections of some regular functions should be used. This requirement is connected to the the idea of piecewise smooth approximation.

If $\mathcal{W}$ is a dyadic wedge partition and $A \in \mathcal{W}$ an atom of this partition, we obtain the approximation $\hat{f}_{\Phi,\mathcal{W}}$ of the image as the sum of the projections of the local images f_A onto $\Phi_A = \mathrm{Span}\varphi_{j,A}$. The minimization problem now reads, for an image f and $\lambda \geq 0$,

$$\hat{\mathcal{W}}_{\lambda,\Phi}(f) = \operatorname*{argmin}_{\mathcal{W} \in \mathfrak{W}} \{\|f - \hat{f}_{\Phi,\mathcal{W}}\|_2^2 + \lambda |\mathcal{W}|\}. \tag{4.1}$$

Since $\mathcal{W}$ is a partition, $\hat{f}_{\Phi,\mathcal{W}}$ can be written as the sum of the solutions of the local regression minimization problems over each atom of $\mathcal{W}$,

$$\hat{f}_{\mathcal{W}} = \sum_{A \in \mathcal{W}} \hat{f}_A = \sum_{A \in \mathcal{W}} \sum_{j \in R} \hat{\alpha}_{A,j} \varphi_{A,j}.$$

For a given atom, this is a classical least squares minimization with some functions of the coordinates as regressing variables. The existence and uniqueness of the solution is then ensured, provided that the local family $(\varphi_{j,A})_{j\in R}$ is linearly independent. When it is not the case, uniqueness is forced by choosing a subset $\Phi' \subset \Phi$, so that the family $(\varphi_A)_{\varphi\in\Phi'}$ is linearly independent.

Thus the local minimization problem over A is performed by the solution of a linear system, which essentially requires the computation of the $\frac{r(r+1)}{2}$ local moments $\langle\varphi_i, \varphi_j\rangle_A$, and of the r additional moments $\langle f, \varphi_i\rangle_A$. Finally we also need the moment $\langle f, f\rangle_A$ in order to get the corresponding error measure

$$E_A(f) = \|f_A - \hat{f}_A\|^2. \tag{4.2}$$

The $r \times r$ linear system can be solved in $O(r^3)$ flops with a classical solution method for symmetrical linear systems. In practice, r should be small enough to maintain the advantage of very fast computation of the local regression parameters (typically smaller than 10), provided that the above-mentioned moments are given.

Finally, for step 2(b)ii of the algorithm described in subsection 3.3, it remains just to store the set of optimal parameters $(\alpha_j)_{j\in R}$ instead of the mean.

4.2. Locally adaptive regression models. The model described in the previous subsection allows rich classes of local approximating functions. It remains, however, limited by its global definition. Since a natural image can be viewed as a highly nonstationary signal, a further extension of our schemes consists of combining a finite number of local regression models in the same partition. In each atom an optimal model is selected according to its local relevance; i.e., it is the model for which the according reconstruction error is minimal. Formally, we consider a finite family $\mathcal{M}$ of regression models. If $\mathcal{W} \in \mathfrak{W}$ is a partition, for each atom $A \in \mathcal{W}$, let $\hat{\Phi}(A) \in \mathcal{M}$ denote an optimal local model, given by

$$\hat{f}_{\hat{\Phi}(A),A} = \min_{\Phi\in\mathcal{M}} \|f_A - \hat{f}_{\Phi,A}\|_2^2.$$

We can now define for any partition $\mathcal{W}$ the global approximation $\hat{f}_{W,\mathcal{M}}$, optimal according to the local models

$$\hat{f}_{\mathcal{W},\mathcal{M}} = \sum_{A\in\mathcal{W}} \hat{f}_{\hat{\Phi}(A),A}.$$

In this case, the minimization problem can be read as

$$\hat{\mathcal{W}}_{\lambda,\mathcal{M}} = \operatorname*{argmin}_{\mathcal{W}\in\mathfrak{W}} \|f - \hat{f}_{\mathcal{W},\mathcal{M}}\|_2^2 + \lambda \sum_{A\in\mathcal{W}} |\hat{\Phi}(A)|. \tag{4.3}$$

With regard to the algorithm of subsection 3.3, it should only be remarked that in step 2(b)ii, we now need to store for each dyadic square the index for the selected model in $\mathcal{M}$ and the error defined by

$$E_A = \min_{\Phi\in\mathcal{M}} E_{A,\Phi}.$$

Note that step 3 is unchanged, and hence the computational complexity of this step is independent of the size of the model. This ensures the real time access to every optimal solution to the locally adaptive regression problem, with varying λ, provided that the optimal local models have been computed (step 2).

4.3. General penalization functions. In functionals (3.1) or (4.1), the term $|\mathcal{W}|$ is merely the simplest example for penalizing the complexity of the model. In fact a larger class of suited penalization functions, describing the characteristics of our local approximation model, can be considered, to which our schemes can be applied. To fix the ideas, we treat the most general case of functional (4.3). We consider a function $g_{\mathcal{M}}$ defined for convenience on any subpartition of $\mathcal{W} \in \mathfrak{W}$. We require only that it is additive, which means that it satisfies

$$g_{\mathcal{M}}(\mathcal{W}) = g_{\mathcal{M}}(\mathcal{W}_1) + g_{\mathcal{M}}(\mathcal{W}_2) \text{ for any } (\mathcal{W}_1, \mathcal{W}_2) \quad \text{such that } \mathcal{W} = \mathcal{W}_1 \dot{\cup} \mathcal{W}_2.$$

An important example of such a function is the penalization according to the coding length required for each partition according to some coding scheme (or in practical cases an estimate of this length). This might take into account not only the size of a partition, but also, for instance, its depth or some correlation structure. Such a penalization function would enable rate-distortion optimization, for the purpose of compression schemes of images based on partition regression representations.

5. Experiments and results. This section presents some experimental results. They illustrate the efficiency of our scheme in terms of computational time, and show the flexibility allowed for the choice of the underlying approximation assumptions. The implementation of the algorithms described in the previous sections has been made in the Oberon programming language. Executable files of this implementation can be downloaded at [6].

5.1. Examples with locally adaptive models. In section 4.2, we treated the case where the optimal coefficients are selected locally, as well as the model itself. With the notation of this section, we present here some specific instances for the set of possible models $\mathcal{M}$.

An elementary set of models is formed by the set of bivariate polynomials of degree less than d, represented by their canonical basis $(x^i y^j)_{i+j \le d}$ and denoted by Φ_d. For d respectively equal to $0, 1, 2$, we obtain the so-called [illegible], and [illegible] regression models, with respective sizes $1, 3$, and 6. Consider the quadtree-wedgelet scheme with the set of models Φ_0, Φ_1, Φ_2. These models are nested ($\Phi_0 \subset \Phi_1 \subset \Phi_2$), and thus the computation of the moments reduces to those necessary for Φ_2. If we use the additional multiplicative properties (for example, $\langle x, y^2 \rangle = \langle xy, y \rangle$), we can even reduce our requirements to the computation of 22 moments instead of 28.

The previous models present a nice theoretical framework, leading to what could be called "discontinuous splines" over wedge partitioning. In practice, however, they may suffer from numerical instabilities, especially for big images and for d bigger than 2. This drawback mainly comes from the fact that orthogonality is not conserved by local projection over atoms A, and locally, the angles between some functions can be too small to guarantee numerical stability.

Other constructions, based on sinusoidal functions, present a much more stationary behavior, while still adapting locally to the image contents. Such functions avoid some blocking artifacts occurring, for example, with the use of a mere linear model. Heuristically, the aim of such a model is to combine a model for texture from the locally adaptive models with a pure geometrical method (optimal wedges) to detect the significant contours between patterns in the image. Consider, for instance, the set of parameterized models $\mathcal{M} = \Phi_{\vartheta_i, a_i} \cup \Phi_1$, where $\Phi_{\vartheta, a}$ is the local regression set composed of three functions defined by

$$\Phi_{\vartheta, a}(z) = (1, \cos(\langle z, v_\vartheta \rangle), \sin(\langle z, v_\vartheta \rangle)).$$

FIG. 5.1. *Barbara,* 256×256. *Quad tree with no wedges. Left: Linear model,* 14682 *coefficients, PSNR:* 25.96 *dB. Right: Linear model, combined with a generic sine model,* 14484 *coefficients, PSNR:* 28.97 *dB.*

FIG. 5.2. *Barbara,* 256×256. *Model with eight possible angles. Left: Linear model,* 14730 *coefficients, PSNR:* 27.79 *dB. Right: Linear model, combined with a generic sine model,* 14511 *coefficients, PSNR:* 33.29 *dB.*

Note that in order to keep coherence for nontextured areas, we still keep additionally the mode Φ_1, also consisting of three functions.

We applied this model to the standard image Barbara, which presents many textured areas. We performed this method with the following set of parameters: $(\vartheta, a) \in \{-\pi/4, 0, \pi/4, \pi/2\} \times \{1, 1/2, 1/4\}$. In Figures 5.1, 5.2, and 5.3, we compare, for a similar number of coefficients, an approximation provided by a simple linear model with an approximation provided by the set of models $\mathcal{M} = \Phi_{\vartheta_i, a_i}$. Although the comparison is not completely fair (for each piece an additional bit of information indicating the model selected locally has to be added), this shows the adaptivity of the scheme: With a set of models chosen in a heuristic way, it is possible to improve the approximation dramatically, both in terms of peak signal to noise ratio, PSNR (as displayed in Figure 5.4, and in terms of visual quality (rendering of the details, precision of the textured patterns).

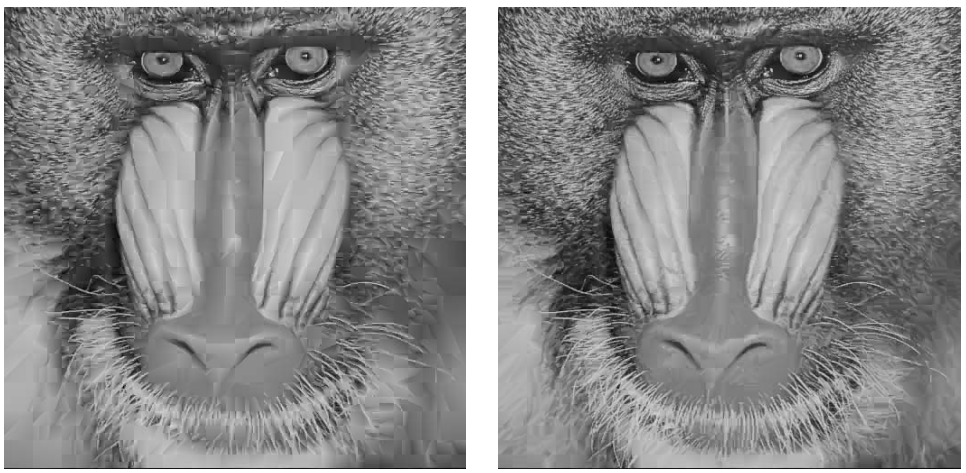


FIG. 5.3. *Baboon,* 512 × 512. *Model with eight possible angles. Left: Linear model,* 27150 *coefficients, PSNR:* 24.44 *dB. Right: Linear model, combined with a generic sine model,* 26574 *coefficients, PSNR:* 27.44 *dB.*

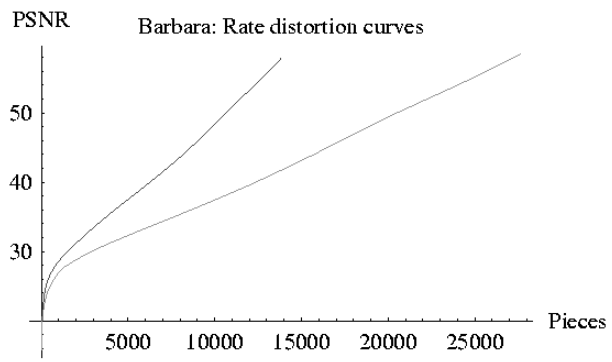


FIG. 5.4. *PSNR versus number of pieces of a linear model (lower line) and the model* $\mathcal{M}$ *(upper line).*

5.2. Influence of angular resolution. As already mentioned in the introduction, a major advantage of our approach is that the angular resolution can be chosen arbitrarily, allowing direct control of the trade-off between computation time and geometric accuracy. In [5, pp. 201–207] the effect of angular resolution is discussed in more detail. In particular, an adaptive choice of angular resolution is motivated there: As in the wedgelets approach in [3], the number of angles in a given dyadic square scales with the number of boundary points. We suggest taking $O(2^j)$ angles for dyadic intervals of size 2^j. PSNR plots in [5] provide evidence that this procedure yields results very similar to those that derive from taking the full set of angles in every dyadic scale. Moreover, the results provided by BeamLab and by our software are very comparable. The major differences between the results of BeamLab and those from our software stem from a different handling of the wedge boundaries. In our discrete approach pixels belong entirely to exactly one wedge, while in BeamLab a subpixel precision approach is used. In Figure 5.5 the effect of angular resolution and differences on BeamLab are illustrated with an example.

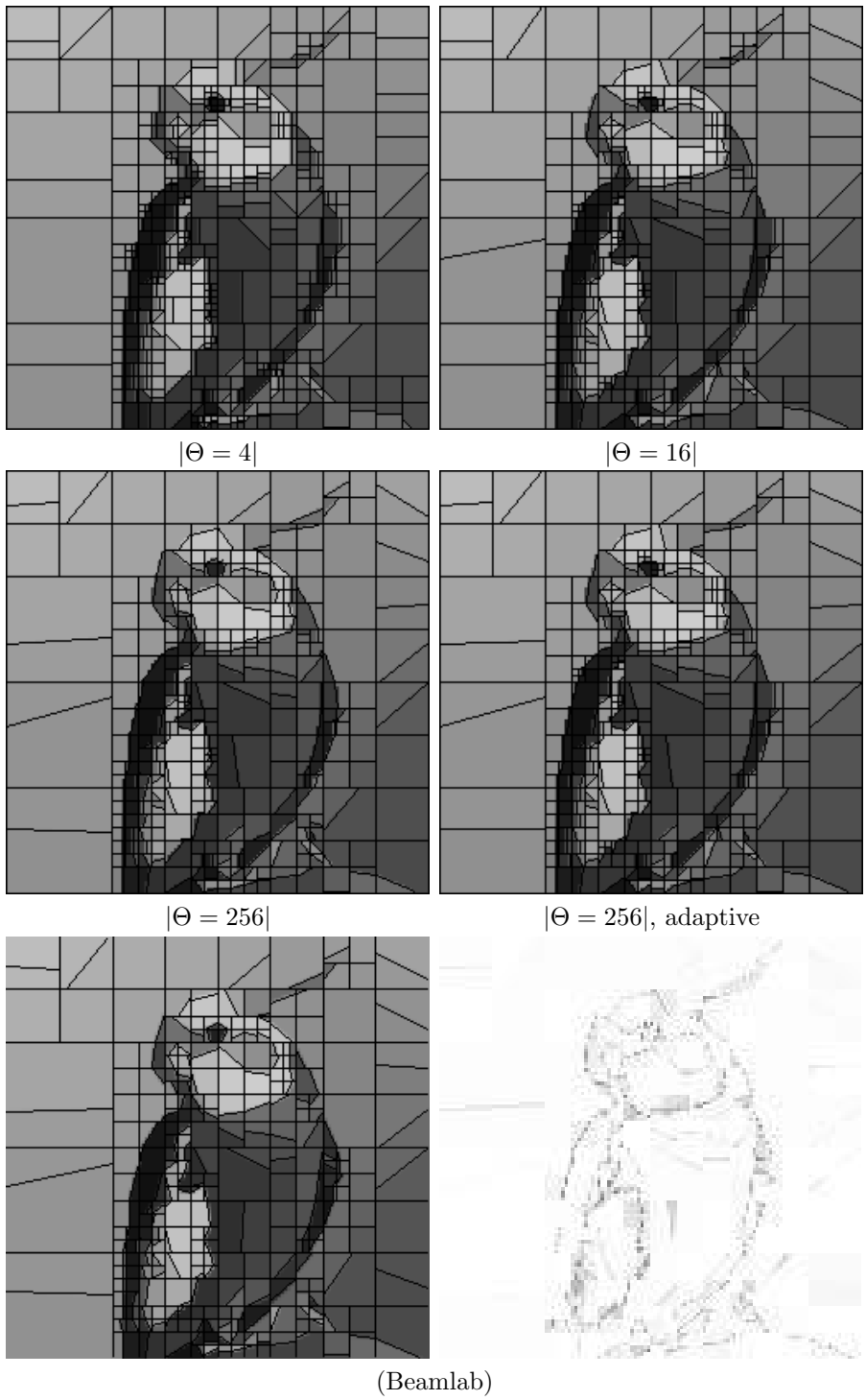


FIG. 5.5. *Wedgelet approximation of image "bird" (*128×128*) with* $\gamma = 0.1 \cdot |S|$ *with* 4, 16, 256 *angles and an adaptive resolution scheme. Lowermost row: wedgelet approximation performed with* BEAMLAB*, and image showing pixelwise* L^1 *distance (white is zero) to the adaptive approximation. Note that the differences mainly stem from different handling of the wedge boundaries.*

5.3. Runtime comparison. The very fast runtimes of our software can be shown by comparison with another library for computation of wedge segmentations, BeamLab [4]. BeamLab is a package running within MATLAB. It contains routines for constant wedgelet segmentations of images with exclusively dyadic dimensions. We used MATLAB version 6 and BeamLab as loaded from the webpage on June 8, 2004; see [4]. For the runtime comparison displayed in the following table, we used a 2.8 GHz Pentium IV machine. The displayed runtimes include all that is necessary to compute a wedgelet approximation of an image: preprocessing (computation of auxiliary matrices and local projection), minimization, and the reconstruction steps. We append approximate runtimes for a wedgelet model in our approach with different numbers of angles. An adaptive model removes half of the angles when entering the next depth in the quad tree. The "equivalent" model is an adaptive model with a number of 1024, 512, ... , 32 angles corresponding to image sizes 512×512, 256×256, ... , 16. This should be roughly equivalent to considering every wedgelet in the image.

Image size	16×16	32×32	64×64	128×128	256×256	512×512
BeamLab	6.72s	45.34s	330.41s $\approx$ 5.5min	2676.3s $\approx$ 44.6min	27918s $\approx$ 7h45min	$>$ 12h
Dyadic wedge equivalent	$<$ 0.02s	0.05s	0.2s	1.2s	8s	59s
Dyadic wedge 1024 angles	1s	2s	8s	37s	140s	722s
Dyadic wedge 180 angles	0.2s	0.3s	1.6s	6s	24s	120s
Dyadic wedge 4 angles	$<$ 0.01s	$<$ 0.01s	0.04s	0.2s	1s	10s
Dyadic square	$<$ 0.01s	$<$ 0.01s	$<$ 0.01s	0.04s	0.5s	6s

6. Conclusion. In this paper, we proposed a new efficient algorithm for rapid moment computation over polygonal domains. It provides a very efficient and flexible tool, enabling comparative experiments between many different bases on arbitrary adaptive local regression functions. A very useful property of our scheme is the ability to compute optimal solutions for many different parameters λ very rapidly: In particular, it enables computation of rate-distortion curves in very reasonable time. It also allows one to easily perform tests with diverse sets of parameterized models.

Acknowledgments. We thank Piyanut Pongpiyapaiboon for help with the implementation. We also thank the referees for their valuable comments.

REFERENCES

[1] J. Bresenham, *Algorithm for computer control of a digital plotter*, IBM Systems J., 4 (1965), pp. 25–30.
[2] R. Dautray and J.-L. Lions, *Mathematical Analysis and Numerical Methods for Science and Technology*, Vol. 1, Springer, New York, 1990, pp. 226–230.
[3] D. Donoho, *Wedgelets: Nearly minimax estimation of edges*, Ann. Statist., 27 (1999), pp. 859–897.
[4] D. Donoho, A. G. Flesia, X. Huo, and O. Levi, *BeamLab* 2001 *website*, http://www-stat.stanford.edu/∼beamlab/.
[5] F. Friedrich, *Complexity Penalized Segmentations in* 2*D*, Ph.D. thesis, Centre for Mathematical Sciences, Technical University of Munich, Munich, Germany, 2005.
[6] F. Friedrich, H. Führ, and L. Demaret, *Wedgelets website*, http://www.wedgelets.de/.
[7] H. Führ, L. Demaret, and F. Friedrich, *Beyond wavelets: New image representation paradigms*, in Document and Image Coding, M. Barni and F. Bartolini, eds., CRC Press/Taylor & Francis, Boca Raton, FL, 2006.
[8] J. Gu, H. Z. Shu, C. Toumoulin, and L. M. Luo, *A novel algorithm for fast computation of Zernike moments*, Pattern Recognition, 35 (2002), pp. 2905–2911.

[9] X. Y. Jiang and H. Bunke, *Simple and fast computation of moments*, Pattern Recognition, 24 (1991), pp. 801–806.

[10] W. Philips, *A new fast algorithm for moment computation*, Pattern Recognition, 26 (1993), pp. 1619–1621.

[11] F. J. Rizzo, P. A. Martin, and R. A.Roberts, *Boundary Elements and a Green's Function Library*, Proceedings of the NIST Workshop on Green's Functions and Boundary Element Analysis, NIST Special Publication SP 910, 1996.

[12] J. Romberg, M. Wakin, and R. Baraniuk, *Multiscale wedgelet image analysis: Fast decompositions and modeling*, in Proceedings of the IEEE International Conference on Image Processing, Vol. 3, Rochester, NY, 2002, IEEE Press, Washington, DC, pp. 585–588.

[13] H. Z. Shu, L. M. Luo, W. X. Yu, and Y. Fu, *A new fast method for computing Legendre moments*, Pattern Recognition, 33 (2000), pp. 341–348.

[14] G. Y. Tang, *A discrete version of Green's theorem*, IEEE Trans. Pattern Anal. Analytic Machine Intell., 4 (1982), pp. 242–249.

[15] R. M. Willett and R. D. Nowak, *Platelets: A multiscale approach for recovering edges and surfaces in photon-limited medical imaging*, IEEE Trans. Medical Imaging, 22 (2003), pp. 332–350.

[16] L. Yang and F. Albregtsen, *Fast and exact computation of Cartesian geometric moments using discrete Green's theorem*, Pattern Recognition, 29 (1996), pp. 1061–1073.

SIAM J. SCI. COMPUT.
Vol. 29, No. 2, pp. 864–889
© 2007 Society for Industrial and Applied Mathematics

MULTI-RESOLUTION-ANALYSIS SCHEME FOR UNCERTAINTY QUANTIFICATION IN CHEMICAL SYSTEMS*

O. P. LE MAÎTRE†, H. N. NAJM‡, P. P. PÉBAY‡, R. G. GHANEM§, AND O. M. KNIO¶

Abstract. This paper presents a multi-resolution approach for the propagation of parametric uncertainty in chemical systems. It is motivated by previous studies where Galerkin formulations of Wiener–Hermite expansions were found to fail in the presence of steep dependences of the species concentrations with regard to the reaction rates. The multi-resolution scheme is based on representation of the uncertain concentration in terms of compact polynomial multi-wavelets, allowing for the control of the convergence in terms of polynomial order and resolution level. The resulting representation is shown to greatly improve the robustness of the Galerkin procedure in presence of steep dependences. However, this improvement comes with a higher computational cost which drastically increases with the number of uncertain reaction rates. To overcome this drawback an adaptive strategy is proposed to control locally (in the parameter space) and in time the resolution level. The efficiency of the method is demonstrated for an uncertain chemical system having eight random parameters.

Key words. uncertainty quantification, chemical systems, polynomial chaos, multi-wavelets

AMS subject classifications. 65C20, 42C40

DOI. 10.1137/050643118

1. Introduction.

Quantification of uncertainty in the modeling of chemical systems is necessary for analysis, design, and optimization purposes. It is also useful in the context of model validation with respect to experimental measurements that are generally imperfect and subject to systematic errors and noise. Numerical models aim to represent real physical processes, but are always approximations of the actual physics, involving both model and parametric uncertainties. Model uncertainty caused by modeling approximations will not be discussed here. Rather, we shall restrict ourselves to the uncertainty associated with the empirical physical parameters appearing in the simulation models (such as the reaction rate constants in chemical models). These physical parameters are usually extracted from (uncertain) experimental measurements, involving noise, by assuming an underlying chemical kinetic model. As a consequence, truly precise values are not available, and measured parameter values can strongly depend on the experimental conditions and the apparatus used for their

*Received by the editors October 19, 2005; accepted for publication (in revised form) October 2, 2006; published electronically May 1, 2007. This work was supported by the Laboratory Directed Research and Development Program at Sandia National Laboratories, funded by the U.S. Department of Energy. Support was also provided by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement F30602-00-2-0612. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights.
http://www.siam.org/journals/sisc/29-2/64311.html

†LMEE and LIMSI-CNRS, Université d'Evry Val d'Essonne, 91020 Evry, France (olm@iup.univ-evry.fr).

‡Combustion Research Facility, Sandia National Laboratories, Livermore, CA 94550 (hnnajm@ca.sandia.gov, pppebay@ca.sandia.gov).

§Aerospace and Mechanical Engineering Department, University of Southern California, Los Angeles, CA 90089 (ghanem@usc.edu).

¶Department of Mechanical Engineering, The Johns Hopkins University, Baltimore, MD 21218 (knio@jhu.edu).

estimation. Thus, a probabilistic characterization (uncertainty bars, likelihood, etc.) of these parameters is more appropriate, and numerical techniques are called for to quantify the resulting uncertainties in the simulation output.

Often, Monte Carlo (MC) techniques are used for analysis of parametric uncertainties. This approach involves performing a (pseudo)random sampling of the parameter distributions, followed by a statistical treatment of the corresponding set of simulations, yielding estimates of the expectation, moments, and any desired statistics of the model output (see, for instance, [11, 15]. However, MC techniques are limited to simple models and/or rough estimates, due to the computational overhead required to solve the deterministic model multiple times. In fact, the large number of model realizations needed comes from the low convergence rate of the statistical estimates with regard to the size of the sample set. This fact strongly supports the use of deterministic techniques, as an alternative to MC sampling strategies. Among deterministic approaches, polynomial chaos (PC) expansions of the solution are attractive candidates, as they allow for a compact and convenient way of representing the functional dependences of the solution on the input random parameters.

PC methods have been the focus of many efforts within different fields of application (solid mechanics, thermal sciences, fluid flow, etc.) over the past decade, starting from the early work of Ghanem and Spanos [10]. The essence of these methods is the projection of the model output onto a subspace spanned by a basis of polynomial functions having for argument a denumerable set of independent random variables parameterizing the uncertain input data. For normalized Gaussian random variables one obtains the "classical" Wiener–Hermite expansion which has been generalized recently to other types of distributions by Xiu and Karniadakis [20]. First attempts to apply these techniques to the propagation of uncertainty in chemical and reacting flow models have been proposed in [7, 17, 18].

There are essentially two types of approaches for the determination of the spectral (projection) coefficients involved in the series expansion of the model output. In the first approach, orthogonality of the PC basis is exploited to perform a nonintrusive projection of the uncertain model solution: a sampling, either stochastic or deterministic, of the uncertainty range provides a set of model output values to be projected, i.e., to compute the correlation between model output and basis polynomials. The second approach makes use of the mathematical model to derive a new set of equations—the spectral problem—through a Galerkin procedure, to be solved for the projection coefficients. The first approach is termed nonintrusive as it requires solving the deterministic model only, but for different values of the uncertain parameters. The second approach, termed intrusive, requires a reformulation of the deterministic model solution algorithm to handle the spectral problem. While more complex than the nonintrusive approach, the intrusive construction is potentially more efficient. However, the intrusive approach does yield a larger (compared to the original one) system of equations and is particularly challenged by the nonlinearities involved in chemical models [8, 17]. In fact, as shown in [17], even for simple reaction mechanisms, the intrusive spectral approach may fail to yield the correct solution because of truncation errors and lack of robustness. These issues are the central focus of the current paper.

The lack of robustness in hitherto intrusive spectral simulations is essentially due to (1) the loss of the "double infinity" limit of the PC expansions because a single stochastic dimension is typically associated with each uncertain parameter and (2) the

decomposition of the solution using a basis of $\Gamma_{\ell}(\cdot,\cdot)$ polynomials, which appears to be inadequate when dealing with uncertain chemical systems. Given the nonlinearity of chemical models, sufficiently large parametric uncertainty levels can lead to solutions with significant uncertainties. Combined with necessary physical constraints (e.g., positivity of concentration values), this can lead to uncertain solutions having highly skewed probability density functions (PDFs). The requisite global PC order necessary for representation of such solutions can be very high, so that truncated spectral computations break down due to numerical instabilities and aliasing errors. For example, the oscillatory nature of the polynomials can result in PC expansions for concentrations that contain negative valued realizations with nonvanishing probability when truncated to finite order. Moreover, the use of high order expansions in spectral computations is not feasible in practice, due, first, to the emergence of numerical instability and, second, to the stiffness of the resulting system of equations whose time integration would require prohibitively small time steps [17]. To address these limitations, we propose here to substitute the PC representation of chemical kinetic models with a multi-resolution scheme based on the multi-wavelet (MW) construction of Alpert [3], following the methodology introduced in [12] for zero order or Wiener–Haar expansions and later extended to arbitrary polynomial order in [13]. These new representations have been designed to deal with steepness and bifurcations triggered by uncertainty; by allowing for discontinuous dependences and introducing the notion of localized decomposition and resolution level (scales), the MWs are expected to be better suited for nonlinear chemical systems than the usual PC expansion.

The paper is organized as follows. Section 2 presents the multi-resolution analysis (MRA) scheme and MW expansion. It starts with a short summary of the spectral stochastic formulation for uncertainty quantification in sections 2.1–2.2. Details on the MRA scheme and the construction method for the MW basis are then provided in sections 2.3–2.5. Section 2.6 offers a short discussion on the Galerkin derivation of the spectral problem and implementation details needed for the examples treated in the remaining sections. In section 3, we focus on the simple test-problem proposed in [17] for which the Wiener–Hermite (WHe) expansion was found ineffective. The causes for the WHe expansion failure are first analyzed in section 3.1. The results obtained using the MW expansion, for the same problem, are then presented in section 3.2, and a numerical inspection of the convergence of the MW representation is given. In the remaining sections, the MRA scheme is applied to a more complex chemical model. This model corresponds to the uncertain hydrogen oxidation mechanism described in [16] and briefly summarized in section 4. In section 5, the situation where only one of the reaction rate constants is uncertain is first considered. This reaction rate constant is selected as the one having the largest impact on the concentrations. Our objective here is to demonstrate the flexibility and robustness provided by the MW representation in adapting the polynomial order and resolution level to the actual solution of the problem. The results shown in section 5 are achieved at the cost of a significant enrichment of the expansion basis, which in turn directly translates into significant computational overhead. As a consequence, an adaptive strategy where a controlled local level of resolution applies is called for. Such a strategy is developed in section 6, based on an extension of the technique proposed in [13] for unsteady solutions. The effectiveness of the overall method (MRA plus adaptive scheme) is demonstrated for up to 8 uncertain reaction rate constants. Finally, major conclusions and proposed future developments and improvements are given in section 7.

2. Multi-resolution analysis and multi-wavelet basis.

2.1. Spectral stochastic representations. Consider a generic equation governing a function $\mathcal{P}$ that depends on space ($\mathbf{x}$), time (t), and an initial random event (θ):

$$\mathcal{O}(\mathcal{P}(\mathbf{x},t,\theta),\mathbf{x},t,\kappa(\theta))=0, \tag{2.1}$$

where $\mathcal{O}$ is a nonlinear operator with random parameters $\kappa(\theta)$. The spectral stochastic representations considered in this paper rely on an orthogonal Fourier-like decomposition of the solution process according to

$$\mathcal{P}(\mathbf{x},t,\theta)=\sum_i \mathcal{P}_i(\mathbf{x},t)\Psi_i(\boldsymbol{\xi}(\theta)), \tag{2.2}$$

where the $\mathcal{P}_i(\mathbf{x},t)$'s are the (yet-to-be-determined) spectral coefficients, $\{\Psi_i\}_{i\in\mathbb{N}}$ is the spectral basis, and $\boldsymbol{\xi}$ is a random vector having independent components. The orthogonality of the basis functions, as well as the equality in (2.2), holds in the mean square sense (for the sake of brevity, the dependence on θ is no longer made explicit):

$$(\forall (i,j)\in\mathbb{N}^2)\quad \langle \Psi_i(\boldsymbol{\xi})\Psi_j(\boldsymbol{\xi})\rangle = \delta_{ij}\left\langle \Psi_i^2(\boldsymbol{\xi})\right\rangle, \tag{2.3}$$

where the brackets denote the mathematical expectation. The expansion (2.2) is an orthogonal basis function representation of $\mathcal{P}$. This approach has lead to the concepts of homogeneous chaos and PC expansions [6, 10, 19] when $\boldsymbol{\xi}$ is a Gaussian vector with independent components, and the Ψ_i are multidimensional Hermite polynomials. It can also be extended to non-Gaussian measures [9, 20]. Due to the orthogonality of the basis functions, the spectral coefficients satisfy

$$(\forall i\in\mathbb{N})\quad \mathcal{P}_i=\frac{\langle \mathcal{P}\Psi_i(\boldsymbol{\xi})\rangle}{\langle \Psi_i(\boldsymbol{\xi})\Psi_i(\boldsymbol{\xi})\rangle}. \tag{2.4}$$

Several approaches can be used to compute the $\mathcal{P}_i$'s that determine the solution $\mathcal{P}$. A simple approach is to use MC sampling (e.g., [18]) or, alternatively, deterministic quadrature formulas (e.g., [14]) to solve (2.4). An alternative to MC and quadrature schemes is the Galerkin approach, where the expansion (2.2) is substituted into the governing equation (2.1) and orthogonal projections onto the basis functions are performed, resulting in the following system for the spectral coefficients:

$$(\forall j\in\mathbb{N})\quad \left\langle \mathcal{O}\left(\sum_i \mathcal{P}_i(\mathbf{x},t)\Psi_i(\boldsymbol{\xi}),\mathbf{x},t,\sum_k \kappa_k\Psi_k(\xi)\right)\Psi_j(\boldsymbol{\xi})\right\rangle = 0, \tag{2.5}$$

where $\sum_k \kappa_k\Psi_k$ is the spectral expansion of the random parameters; this expansion is known provided that the statistics of κ are prescribed. As indicated above, these global expansions can exhibit stability problems. In the following, we introduce an MW basis function expansion which overcomes this limitation. To simplify the presentation, we first focus on the case of a one-dimensional random vector ξ; generalization to the multidimensional case is then addressed in subsection 2.5.

2.2. Change of variable. Let $\mathrm{p}(\eta)$ denote the cumulative density function (CDF), giving the probability that $\xi\le\eta$. As in [12], we assume that $\mathrm{p}(\eta)$ is a continuous, strictly increasing function of η over the interval $[a,b]$, where $a<b$ are two real numbers, possibly infinite, and that $\mathrm{p}(a)=0$ and $\mathrm{p}(b)=1$. Based on these

properties of $\mathrm{p}(\eta)$, it follows that for all $r \in [0,1]$ there is a unique $\eta \in [a,b]$, such that $\mathrm{p}(\eta) = r$. In addition, if x is a uniformly distributed random variable on $[0,1]$, then $p^{-1}(x)$ is a random variable with values in $[a,b]$ having the same distribution as ξ [11]. Consequently, using the mappings $x = \mathrm{p}(\xi)$ and $\xi = \mathrm{p}^{-1}(x)$, we develop a representation of $\mathcal{P}$ in terms of $x = \mathrm{p}(\xi)$. To shorten the notation we shall write $x(\xi)$ (resp., $\xi(x)$) for $x = \mathrm{p}(\xi)$ (resp., $\xi = \mathrm{p}^{-1}(x)$).

2.3. Multi-resolution analysis. In this section, we recall some properties of the multi-wavelet bases introduced by Alpert [3] (see also [2]). The application to the representation of ODEs with random initial conditions is considered in section 2.4.

2.3.1. Vector spaces. For $\mathrm{N_o} = 0, 1, \ldots$ and $k = 0, 1, 2, \ldots$, we define the space $\mathbf{V}_k^{\mathrm{N_o}}$ of piecewise-continuous polynomials according to

$$\mathbf{V}_k^{\mathrm{N_o}} = \left\{ f : \mathrm{Supp}(f) = [0,1] \; \forall l \in \{0, \ldots, 2^k - 1\} \, f_{|_{]2^{-k}l, 2^{-k}(l+1)[}} \in \mathbb{R}_{\mathrm{N_o}}[X] \right\}, \tag{2.6}$$

where $\mathrm{Supp}(f)$ and $\mathbb{R}_{\mathrm{N_o}}[X]$, respectively, denote the support of f and the ring of degree $\mathrm{N_o}$ polynomials with real coefficients. In other words, the restriction of f to any interval $]2^{-k}l, 2^{-k}(l+1)[$ is such a polynomial function, and f vanishes outside of $[0,1]$. Thus, $\mathbf{V}_k^{\mathrm{N_o}}$ has dimension $2^k(\mathrm{N_o}+1)$ and $\mathbf{V}_0^{\mathrm{N_o}} \subset \mathbf{V}_1^{\mathrm{N_o}} \subset \cdots \subset \mathbf{V}_k^{\mathrm{N_o}} \subset \cdots$. Denoting by $\mathbf{V}^{\mathrm{N_o}}$ the union of all spaces $\mathbf{V}_k^{\mathrm{N_o}}$, $\mathbf{V}^{\mathrm{N_o}} = \overline{\bigcup_{k \geq 0} \mathbf{V}_k^{\mathrm{N_o}}}$ (see [3]), we remark that $\mathbf{V}^{\mathrm{N_o}}$ is dense in $L^2([0,1])$ with respect to the norm $\|f\| = \langle f, f \rangle^{1/2}$, where

$$\langle f, g \rangle = \int_0^1 f(x) g(x) \mathrm{d}x \tag{2.7}$$

is the standard inner product in $L^2([0,1])$. In addition, for any integer k, the subspace $\mathbf{W}_k^{\mathrm{N_o}}$, which we employ below as the MW subspace, is defined as the orthogonal complement of $\mathbf{V}_k^{\mathrm{N_o}}$ in $\mathbf{V}_{k+1}^{\mathrm{N_o}}$; we write

$$(\forall k \in \mathbb{N}) \quad \mathbf{V}_k^{\mathrm{N_o}} \oplus \mathbf{W}_k^{\mathrm{N_o}} = \mathbf{V}_{k+1}^{\mathrm{N_o}}, \quad \mathbf{W}_k^{\mathrm{N_o}} \perp \mathbf{V}_k^{\mathrm{N_o}}, \tag{2.8}$$

where $\oplus$ denotes the direct sum of vector spaces. From this construction we have

$$\mathbf{V}_0^{\mathrm{N_o}} \bigoplus_{k \in \mathbb{N}} \mathbf{W}_k^{\mathrm{N_o}} = L^2([0,1]). \tag{2.9}$$

2.3.2. Multi-wavelet basis. Let $\{\psi_0, \psi_1, \ldots, \psi_{\mathrm{N_o}}\}$ be an orthonormal basis of $\mathbf{W}_0^{\mathrm{N_o}}$, where the ψ_i's are piecewise polynomial functions of degree less than or equal to $\mathrm{N_o}$. From the orthonormality condition, we have

$$(\forall (i,j) \in \mathbb{N}^2) \quad \langle \psi_i(x), \psi_j(x) \rangle = \delta_{ij}. \tag{2.10}$$

Since $\mathbf{W}_0^{\mathrm{N_o}} \perp \mathbf{V}_0^{\mathrm{N_o}}$, the first $\mathrm{N_o}+1$ moments of the ψ_i vanish; i.e.,

$$(\forall (i,j) \in \mathbb{N}^2, \, i < j \leq \mathrm{N_o}) \quad \left\langle \psi_j, x^i \right\rangle = 0. \tag{2.11}$$

Equations (2.10) and (2.11) result in a system of polynomial equations which is solved, yielding the $(\mathrm{N_o}+1)$ ψ_i functions (see [3]). The space $\mathbf{W}_k^{\mathrm{N_o}}$, whose dimension is $2^k(\mathrm{N_o}+1)$, is spanned by the multi-wavelets, ψ_{jl}^k, which are images of the ψ_i's by translations and dilations. The ψ_{jl}^k are given by

$$(\forall (j,k,l) \in \mathbb{N}^3, \, j \leq \mathrm{N_o}, \, l < 2^k) \quad \psi_{jl}^k(x) = 2^{k/2} \psi_j(2^k x - l), \tag{2.12}$$

and their support is $\mathrm{Supp}(\psi_{jl}^k) = [2^{-k}l, 2^{-k}(l+1)]$. Due to the orthonormality of the ψ's, we have

$$\left\langle \psi_{im}^k, \psi_{jn}^l \right\rangle = \delta_{ij}\delta_{mn}\delta_{kl}. \tag{2.13}$$

A basis $\{\phi_0, \ldots, \phi_{k-1}\}$ for $\mathbf{V}_0^k$ is then constructed, using rescaled Legendre polynomials. Denoting L_i as the Legendre polynomial [1] of degree i, defined over $[-1, 1]$, we set

$$(\forall i \in \mathbb{N}, i \leq \mathrm{N_o}) \quad \phi_i(x) = \frac{L_i(2x-1)}{l_i}, \tag{2.14}$$

where l_i is a normalization factor selected such that

$$(\forall (i,j) \in \mathbb{N}^2,\, i \leq \mathrm{N_o},\, j \leq \mathrm{N_o}) \quad \langle \phi_i(x), \phi_j(x) \rangle = \delta_{ij}. \tag{2.15}$$

The space $\mathbf{V}_k^{\mathrm{N_o}}$, whose dimension is $2^k(\mathrm{N_o}+1)$, is spanned by the polynomials $\phi_{il}^k(x) = 2^{k/2}\phi_i(2^k x - l)$, for $i = 0, \ldots, \mathrm{N_o}$ and $l = 0, \ldots, 2^k - 1$, that are images of the ϕ_i's by translations and dilations.

2.3.3. MW expansion. A function $f \in L^2([0,1])$ can be arbitrarily well approximated using the MRA scheme constructed above. We denote by $f^{\mathrm{N_o},\mathrm{N_r}}$ the projection of f on $\mathbf{V}_{\mathrm{N_r}}^{\mathrm{N_o}}$; we have

$$f^{\mathrm{N_o},\mathrm{N_r}}(x) = \sum_{l=0}^{2^{\mathrm{N_r}}-1} \sum_{i=0}^{\mathrm{N_o}} \left\langle \phi_{il}^{\mathrm{N_r}}(x), f(x) \right\rangle \phi_{il}^{\mathrm{N_r}}(x) = \sum_{l=0}^{2^{\mathrm{N_r}}-1} \sum_{i=0}^{\mathrm{N_o}} \overline{f}_{il}^{\mathrm{N_r}} \phi_{il}^{\mathrm{N_r}}(x). \tag{2.16}$$

An alternative expression for $f^{\mathrm{N_o},\mathrm{N_r}}$, valid for all $\mathrm{N_r} \geq 1$, in terms of multi-wavelets is

$$f^{\mathrm{N_o},\mathrm{N_r}}(x) = f^{\mathrm{N_o},0} + \sum_{k=0}^{\mathrm{N_r}-1} \sum_{l=0}^{2^k-1} \left(\sum_{i=0}^{\mathrm{N_o}} \mathrm{d}f_{il}^k \psi_{il}^k(x) \right). \tag{2.17}$$

This expression, with $f^{\mathrm{N_o},0}(x)$ evaluated from (2.16), provides the sought-after expansion of $f^{\mathrm{N_o},\mathrm{N_r}}(x)$ in terms of Legendre polynomials overlaid with a specified number of MW details. The MW coefficients $\mathrm{d}f_{il}^k$ appearing in (2.17) are given by

$$\mathrm{d}f_{il}^k = \left\langle \left\{ f^{\mathrm{N_o},k+1} - f^{\mathrm{N_o},k} \right\}, \psi_{il}^k \right\rangle. \tag{2.18}$$

Denoting by $\delta_{\mathrm{N_o},\mathrm{N_r}}$ the L^2-norm of the approximation error of f on $\mathbf{V}_{\mathrm{N_r}}^{\mathrm{N_o}}$,

$$\delta_{\mathrm{N_o},\mathrm{N_r}} = \left\langle f - f^{\mathrm{N_o},\mathrm{N_r}}, f - f^{\mathrm{N_o},\mathrm{N_r}} \right\rangle, \tag{2.19}$$

convergence is characterized by $\delta_{\mathrm{N_o},\mathrm{N_r}} \to 0$ when the polynomial order $\mathrm{N_o}$ (p convergence) and the number of resolution levels $\mathrm{N_r}$ (h convergence) tend to ∞.

2.4. Expansion of a process. Consider an ODE governing $\mathcal{P}$ that depends on a random variable ξ satisfying the assumptions of section 2.2. Further, we assume that $\mathcal{P}(\xi)$ has a second order moment. Using the change of variables introduced in section 2.2, we express $\mathcal{P}(\xi)$ in terms of $x = \mathrm{p}(\xi)$ as

$$\mathcal{P}(\xi) = \mathcal{P}(\mathrm{p}^{-1}(x)) = \tilde{\mathcal{P}}(x(\xi)). \tag{2.20}$$

Introducing this change of variable, and taking into account that x is uniformly distributed on $[0,1]$, it is easy to show that $\tilde{\mathcal{P}}(x) \in L^2([0,1])$. Thus, $\tilde{\mathcal{P}}(x)$ can be expanded according to (2.17). This expansion can be recast through convenient indexing as (see [12])

$$\mathcal{P}(\xi) = \tilde{\mathcal{P}}(x(\xi)) = \sum_{\lambda \in \diamondsuit} \tilde{\mathcal{P}}_\lambda W_\lambda(x(\xi)), \quad \diamondsuit \equiv \triangle \cup \triangle_0, \tag{2.21}$$

where

$$\begin{cases} W_\lambda(x) = \psi_\lambda(x) & \text{for } \lambda \in \triangle \equiv \big\{\lambda : \lambda = (\mathrm{N_o}+1)(2^k + l - 1) + i, \\ & \qquad (i,j,k) \in \mathbb{N}^3,\ i \le \mathrm{N_o},\ j < 2^k\big\}, \\ W_\lambda(x) = \phi_{-1-\lambda}(x) & \text{for } \lambda \in \triangle_0 \equiv \{-\mathrm{N_o}-1, -\mathrm{N_o}, \ldots, -1\}. \end{cases} \tag{2.22}$$

The resolution level k for any $\lambda \in \triangle$ will be denoted by $|\lambda|$.

2.5. The multidimensional basis. Extension of the one-dimensional MW expansion to the $\mathrm{N_d}$-dimensional case is now considered. For simplicity, we focus on a vector $\boldsymbol{\xi}$ with random uncorrelated and independent components $\{\xi_1, \ldots, \xi_{\mathrm{N_d}}\}$:

$$\mathrm{pdf}(\boldsymbol{\xi}) = \prod_{d=0}^{\mathrm{N_d}} \mathrm{pdf}_d(\xi_d),$$

where pdf_d denotes the PDF of ξ_d. For each component ξ_d, we define the real numbers $a_d < b_d$ and assume that pdf_d satisfies the assumptions of section 2.2. Then for all $x_d \in [0,1]$ there is a unique $\xi_d \in [a_d, b_d]$, denoted for short as $\xi_d(x_d)$, such that $\mathrm{p}_d(\xi_d) = x_d$. Consistent with the one-dimensional case, we write for short $\boldsymbol{\xi}(\boldsymbol{x}) = (\xi_1(x_1), \ldots, \xi_{\mathrm{N_d}}(x_{\mathrm{N_d}}))$ and $\boldsymbol{x}(\boldsymbol{\xi}) = (x_1(\xi_1), \ldots, x_{\mathrm{N_d}}(\xi_{\mathrm{N_d}}))$. We now consider the multi-index $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_{\mathrm{N_d}})$ and define the sets

$$\Lambda_k = \left\{ \boldsymbol{\lambda} : \sum_{d=1}^{\mathrm{N_d}} |\lambda_d| = k \right\}, \quad \mathcal{W}_k = \left\{ \prod_{d=1}^{\mathrm{N_d}} W_{\lambda_d}(x_d) : \boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_{\mathrm{N_d}}) \in \Lambda_k \right\}. \tag{2.23}$$

$\mathcal{W}_k$ is the set of multidimensional multi-wavelets having resolution level k. Let $M_k = \mathrm{Card}(\Lambda_k)$, such that the MW expansion of $\tilde{\mathcal{P}}(\boldsymbol{x})$ can now be formally written as

$$\tilde{\mathcal{P}}(\boldsymbol{x}) = \sum_{i=1}^{M_0} c_i^0 \Gamma_i^0(\boldsymbol{x}) + \sum_{i=1}^{M_1} c_i^1 \Gamma_i^1(\boldsymbol{x}) + \sum_{i=1}^{M_2} c_i^2 \Gamma_i^2(\boldsymbol{x}) + \cdots, \tag{2.24}$$

where $\Gamma^k(\boldsymbol{x}) \in \mathcal{W}_k$ denotes a multidimensional multi-wavelet of resolution k. In practice, the MW expansion has to be truncated. For instance, we can choose to retain all multi-indices $\boldsymbol{\lambda}$ such that $|\boldsymbol{\lambda}| = \sum_{d=1}^{\mathrm{N_d}} |\lambda_d| \le \mathrm{N_r}$, where $\mathrm{N_r}$ is a prescribed resolution level. After truncation, the finite expansion may be rewritten in a single index form as

$$\mathcal{P}(\boldsymbol{\xi}) = \tilde{\mathcal{P}}(\boldsymbol{x}(\boldsymbol{\xi})) \approx \sum_{i=0}^{\mathrm{Nw}} \tilde{\mathcal{P}}_i \mathcal{M}_i(\boldsymbol{x}(\boldsymbol{\xi})), \tag{2.25}$$

where $\mathrm{Nw}+1 = \sum_{i=0}^{\mathrm{N_r}} M_i$ is the dimension of the truncated basis $\{\mathcal{M}_i, i = 0, \ldots, \mathrm{Nw}\}$. Further, we use the convention that the indexing is performed in such a way that the

first element of the basis is $\mathcal{M}_0 = 1$. With this convention, the mean and variance of $\mathcal{P}(\boldsymbol{\xi})$ are given, respectively, by

$$\langle \mathcal{P} \rangle = \tilde{\mathcal{P}}_0, \quad \sigma^2(\mathcal{P}) = \sum_{i=1}^{\mathrm{Nw}} \tilde{\mathcal{P}}_i^2.$$

2.6. Galerkin projection schemes. When an intrusive approach is used, the spectral coefficients appearing in the MW expansion of the process $\tilde{\mathcal{P}}(\boldsymbol{x})$ are determined using the Galerkin projection technique outlined in section 2.1. However, the projection of the governing equations on the spectral basis, as given by (2.5), may raise several difficulties depending on the nature of the nonlinear operator $\mathcal{O}$. For instance, in chemical systems, one has to deal with both polynomial and nonalgebraic functionals in the determination of the chemical source terms. The general spectral treatment of uncertain chemical source terms was extensively discussed in [17]. The nonlinearities appearing in the computational examples presented in the following, namely binary and ternary products of uncertain quantities, are treated by means of pseudospectral techniques (multiple application of the Galerkin product). For other types of nonlinearities (e.g., nonalgebraic terms such as exp, log, and square root), the reader should consult [7, 8, 17] since the methodologies readily extend to the MW expansions.

3. Comparison of WHe and MW expansions. The purpose of this section is to examine the efficiency and robustness of the MW representation, compared to a global WHe expansion [6, 10, 19]. To this end, a simple problem is used, namely, that proposed in [17]. This simple problem is designed as a "prototype" for the chemical problem discussed below. By construction, it involves nonlinearity and bifurcations, both being significant challenges with uncertainty quantification in chemical systems. This test problem is now introduced. Consider an uncertain system described by a state variable $X \in \mathbb{R}$ obeying

$$\left\{ \begin{array}{l} \dfrac{\mathrm{d}X}{\mathrm{d}t} = -X(X-B)(X-C), \\ X = X_0 \quad \text{for } t = 0, \end{array} \right. \tag{3.1}$$

where $B < 0$ and $C > 0$ are two real constants and the initial condition X_0 is uncertain. The system has three fixed-points, $X = 0$, B, and C, the first one being unstable. In the following, we set $B = -6$ and $C = 1$. While very simple, this problem is challenging for PC expansions whenever the initial condition is such that the set of possible realizations of X_0 overlaps the unstable point. In this case, the set of realizations splits into two parts, depending on the sign of X_0, leading to a discontinuous solution as time advances. This discontinuity in the solution is a source of breakdown of the spectral representation when using a global smooth basis.

3.1. Wiener–Hermite expansion. To start with, we consider the case where $X_0 \sim \mathcal{N}(\mu_0, \sigma_0^2)$; i.e., the distribution of X_0 is Gaussian with mean μ_0 and standard deviation σ_0, where $\mu_0 \geq 0$ and $\sigma_0 > 0$ are two arbitrary real constants. Hence, the parametric representation of X_0 in terms of a normalized Gaussian variable ξ is

$$X_0(\xi) = \mu_0 + \sigma_0 \xi, \quad \xi \sim \mathcal{N}(0,1). \tag{3.2}$$

Also, the solution $X(t,\xi)$ has a WHe PC expansion up to the order $\mathrm{N_o}$ given by

$$X(t,\xi) = \sum_{k=0}^{\mathrm{N_o}} X_k(t) \Psi_k^{(H)}(\xi),$$

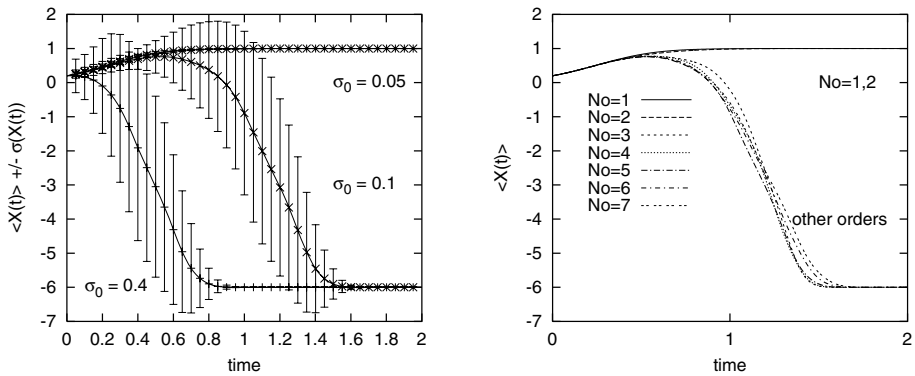


FIG. 3.1. *Time evolution of $X(t)$ for the test problem of section 3, using the WHe expansion and Gaussian initial condition $X_0 \sim \mathcal{N}(\mu_0, \sigma_0^2)$. Left: results for $\mu_0 = 0.2$, fifth order expansion and $\sigma_0 = 0.4$, 0.1, and 0.05; plotted are the mean of $X(t)$ and error bar extending to $\pm\sigma(X(t))$. Right: mean of $X(t)$ for $\mu_0 = 0.2$, $\sigma_0 = 0.1$ and different expansion orders $\mathrm{N_o} = 1, \ldots, 7$. Computations for $\mathrm{N_o} > 7$ are unstable.*

where the basis functions $\Psi_k^{(H)}(\xi)$ are the Hermite polynomials with respective degrees k [1]. To solve this problem, the Galerkin method is employed with a fourth order Runge–Kutta scheme using a time step $\Delta t = 10^{-3}$.

For a fixed order $\mathrm{N_o} = 5$, the numerical experiments presented in Figure 3.1 show that if the probability of $X_0 < 0$ is significant (for instance, for $\sigma_0 = 0.4$ and 0.1), then all the realizations identically (with unit probability) end up on the negative stable fixed-point. In contrast, if the probability of $X_0 < 0$ is small enough (for example, $\sigma_0 = 0.05$), one obtains $\lim_{t\to\infty} X(t) = C$. This result contradicts the exact expected behavior. Further, fixing $\mu_0 = 0.2$ and $\sigma_0 = 0.1$ and increasing the expansion order from $\mathrm{N_o} = 1$ to 7, we observe that the positive stable fixed-point is selected for $\mathrm{N_o} \leq 2$, while the solution goes to the negative fixed-point for $3 \leq \mathrm{N_o} \leq 7$; increasing $\mathrm{N_o}$ further leads to unstable computations.

To enforce a strictly positive initial condition, one often uses a log-normal rather than a Gaussian distribution; thus let us select $X_0(\xi) = \exp(\beta(\xi))$, where $\beta(\xi) = \mu_\beta + \sigma_\beta \xi \sim \mathcal{N}(\mu_\beta, \sigma_\beta^2)$. As a result, the initial condition has the following one-dimensional WHe expansion [9]:

$$X_0(\xi) = \exp\left(\mu_\beta + \frac{\sigma_\beta^2}{2}\right) = \sum_{k=0}^{\infty} \left\{ \frac{\sigma_\beta^k}{\left\langle {\Psi_k^{(H)}}^2 \right\rangle} \Psi_k^{(H)}(\xi) \right\}. \tag{3.3}$$

Even though the infinite series is positive with unit probability, truncation of (3.3) to a finite order $\mathrm{N_o}$ may again introduce a nonvanishing probability of having $X_0 < 0$. This issue was considered in [17]. In fact, if the expansion (3.3) does converge to the exact mapping of the log-normal distribution for increasing $\mathrm{N_o}$, oscillations of the left tail around $X_0 = 0$ can lead to negative realizations for X_0 for low μ_β and large σ_β. These oscillations are damped with increasing frequency as $\mathrm{N_o}$ increases and occur for lower values of ξ, so they are more unlikely. However these oscillations around the unstable point have a dramatic impact on the model robustness: each point where the mapping crosses the axis $X_0 = 0$ induces a discontinuity in the solution $\lim_{t\to\infty} X(t)$, and thus is a source for spectral breakdown. In other words,

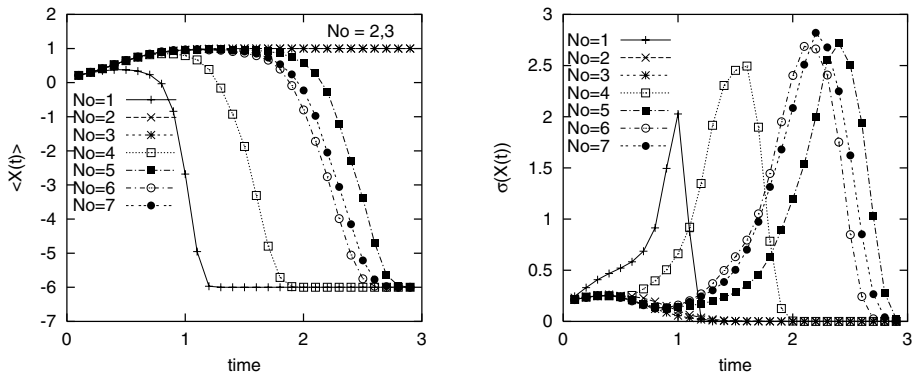


FIG. 3.2. *Time evolution of the computed mean (left) and standard deviation (right) of $X(t)$ for the test problem of section 3, using the WHe expansion with* $\mathrm{N_o} = 1, \ldots, 7$. *Log-normal initial condition with* $\mu_\beta = -2.5$ *and* $\sigma_\beta = 1.1$.

even though the probability of $X_0 < 0$ decreases for increasing $\mathrm{N_o}$, amplification of the higher order coefficients is expected because of the increasing number of zeros in the truncated mapping. This is shown in Figure 3.2, where the evolution of the mean and standard deviation of X with time are plotted, for different orders $\mathrm{N_o} = 1, \ldots, 7$ and $\mu_\beta = -2.5$, $\sigma_\beta = 1.1$. Again, the results contradict the expected behavior and were found unstable for $\mathrm{N_o} > 7$.

3.2. Multi-wavelet expansion. We repeat the experiments of the previous section using the MW expansion. The solution (and initial condition) are now expanded in terms of $x = \mathrm{p}(\xi)$, with $\xi \sim \mathcal{N}(0,1)$:

$$X(t,\xi) = \tilde{X}(t, x = \mathrm{p}(\xi)) = \sum_{i=0}^{\mathrm{Nw}} \tilde{X}_i(t)\mathcal{M}_i(x). \tag{3.4}$$

For these tests, we select the Gaussian distribution of the initial condition $X_0 \sim \mathcal{N}(\mu_0, \sigma_0^2)$ and rely on (3.2) to determine the spectral coefficients of the initial condition:

$$(\tilde{X}_0)_i = \left\langle (\mu_0 + \sigma_0 \mathrm{p}^{-1}(x)), \mathcal{M}_i(x) \right\rangle = \mu_0 \delta_{i0} + \sigma_0 \int_0^1 \mathrm{p}^{-1}(x)\mathcal{M}_i(x)\mathrm{d}x. \tag{3.5}$$

To compute the integral appearing in the right-hand side of (3.5), high order Gauss quadrature formulas [1] are used. However, since $\mathrm{p}^{-1}(x)$ is not a polynomial, $(\tilde{X}_0)_i$ cannot be exactly computed from a Gauss quadrature. To minimize the quadrature error, 32 Gauss-points were used to estimate these integrals.

Then, provided with the mapping of the initial condition, the governing equations for the MW coefficients are time integrated, still using the fourth order Runge–Kutta scheme with $\Delta t = 10^{-3}$, but employing the pseudospectral approximation of the ternary products. In Figures 3.3 and 3.4 we plot the time evolution of the computed mean and standard deviation of X, respectively, for $\mathrm{N_o} = 0, \ldots, 3$ and $\mathrm{N_r} = 1, \ldots, 6$. The results illustrate that, for fixed $\mathrm{N_o}$, both first and second order statistics converge for increasing $\mathrm{N_r}$. In contrast with the WHe computations, the statistics converge to their "exact" values (available by sampling) and not to spurious ones. In fact,

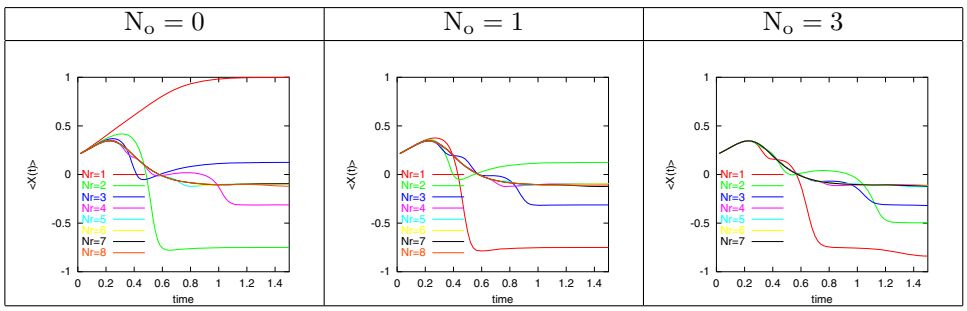


FIG. 3.3. *Time evolution of the computed mean of $X(t)$ for the MW expansions with N_o and N_r as indicated. The initial condition corresponds to the Gaussian distribution with $\mu_0 = 0.2$ and $\sigma_0 = 0.1$.*

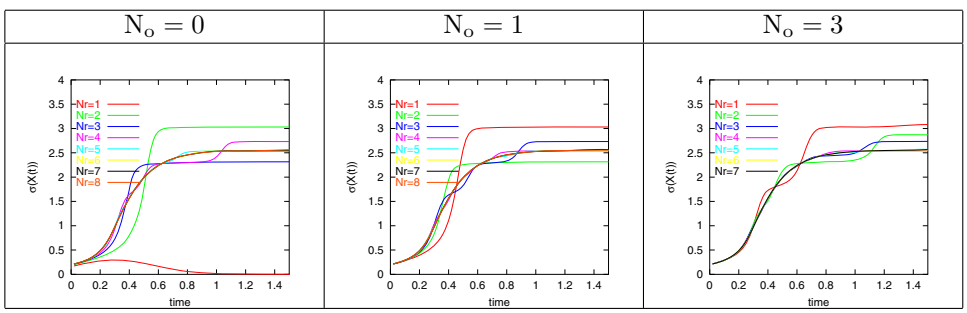


FIG. 3.4. *Time evolution of the computed standard deviation of $X(t)$ for the MW expansions with N_o and N_r as indicated. The initial condition corresponds to the Gaussian distribution with $\mu_0 = 0.2$ and $\sigma_0 = 0.1$.*

provided that the discretization order and resolution level are large enough, the co-existence of two stable fixed-points with respective attraction basins is captured by the MW expansion and appropriately measured with regard to the initial condition distribution. This behavior is clearly an improvement compared to the WHe case where computations, when stable, always predict vanishing standard deviation for a long time, meaning that only one stable fixed-point is recognized. Moreover, it is found that the minimal resolution level to be used to satisfactorily capture the dynamics of X is roughly $N_r = 4$ for $N_o = 1, 3$ and $N_r = 5$ when using $N_o = 0, 2$.

To further understand the behavior of the MW expansion, we provide in Figure 3.5 reconstructions of the response-surfaces $\tilde{X}(t, x)$ for $N_o = 0$ and $N_r = 1$, 2, and 6. It is shown that for $N_r = 1$, the model predicts that all realizations end up on the positive stable point. This is consistent with the mapping of the actual initial condition of the simulation which for this low resolution level is positive for all $x \in [0, 1]$. For $N_r = 2$, the probability of realizations having negative initial condition is, on the contrary, overestimated (the initial condition is negative for $x \in [0, 0.25]$ corresponding to a probability of 0.25 compared to the true probability equal to ≈ 0.159): the computed mean of $X(t)$ presented in Figure 3.3 reflects this fact, overpredicting the final negative mean, while the corresponding response-surface in Figure 3.5 illustrates how the MW expansion can actually split the set of realizations consistently with the approximation of the initial condition. When N_r is increased further, the approximation of the initial mapping becomes better and better, the discontinuity point is found with greater and

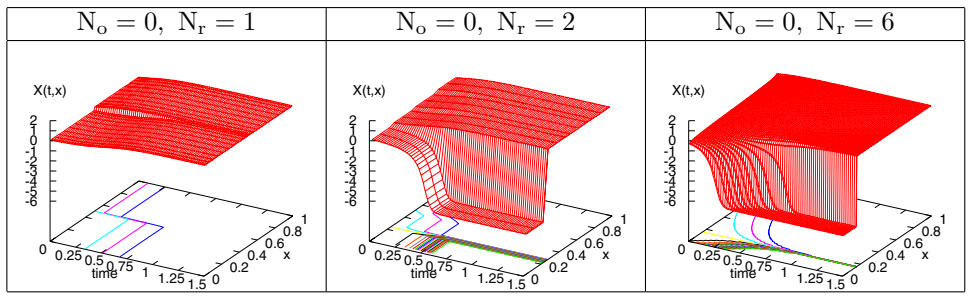


FIG. 3.5. *MW response-surfaces of* $\tilde{X}(t,x)$ *solution of the problem of section* 3 *and initial condition* $X_0 \sim \mathcal{N}(0.2, 0.1^2)$. *Plots were generated for* $N_o = 0$ *and* $N_r = 1, 2, 6$.

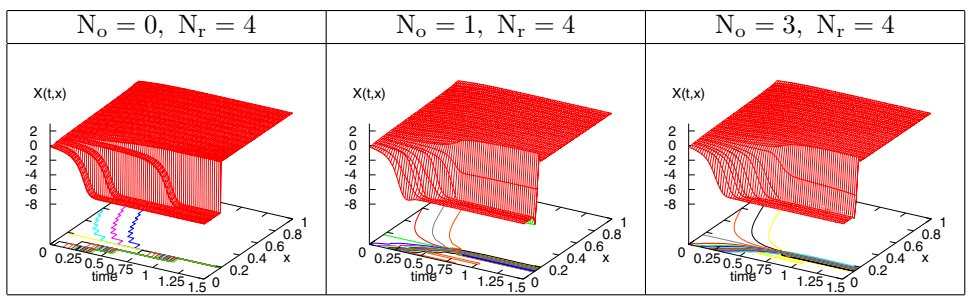


FIG. 3.6. *MW response-surfaces of* $\tilde{X}(t,x)$ *solution of the problem of section* 3 *and initial condition* $X_0 \sim \mathcal{N}(0.2, 0.1^2)$. *Plots were generated for* $N_o = 0$, 1, *and* 3 *with* $N_r = 4$.

greater precision, and the transient dynamics of individual realizations improve (see Figure 3.5).

Figure 3.6 depicts the response-surfaces computed for $N_r = 4$ and $N_o = 0$, 1, and 3. As explained above, this level of resolution provides quite an accurate estimate of the discontinuity location, but spurious oscillations in the solution are observed in the neighborhood of the discontinuity for $N_o > 0$, whose magnitudes increase with the polynomial order. These oscillations are in fact essentially due to Gibbs phenomena, characteristic of the representation of discontinuous solution using (even local) smooth polynomials. However, these oscillations are limited to the support overlapping the discontinuity and do not contaminate the full uncertainty range. Hence, the contribution of the spurious oscillations to the statistics of the process decreases exponentially when the resolution level N_r increases, explaining the convergence, for $N_o > 0$, of the first and second order moments reported in Figures 3.3 and 3.4. Clearly, these observations suggest that zero order polynomials should be used on supports overlapping discontinuities, and future work will focus on the adaptation of the local polynomial order with regard to the local behavior (smoothness) of the solution.

Repeating the experiment for the log-normal distribution of the previous subsection with the MW expansion (not shown) yields similar conclusions: correct long time solution for any resolution level and order, h and p convergence, and more robustness compared to the WHe expansion. To summarize the findings of the section, we stress that the MW expansion offers an attractive alternative to the WHe expansion in the context of chemical modeling, by providing more robustness and a better efficiency when dealing with steep or discontinuous processes. The MW expansion seems to be superior to WHe expansions when dealing with constrained problems, such as posi-

tivity of the solution, thanks to the local character of the representation. Moreover, by offering two ways of improving the predictions through increasing resolution level and polynomial order, the MW representation provides more flexibility than traditional global polynomial expansions with fixed dimension that can suffer from lack of robustness when increasing the representation order (recall, of course, that the general convergence of the PC representation is guaranteed only for increasing order dimensionality). These claims are further investigated in the following sections.

4. Chemical test problem.

4.1. Uncertainty parameterization. In this section, we present a short description of the chemical model used in the remainder of this paper. This model has been previously used as a test problem for uncertainty quantification in a chemical system in [16, 18], where spectral PC expansions were found to fail. The model is a reduced hydrogen oxidation mechanism involving seven species (OH, H, H_2O, H_2, O_2, HO_2, and H_2O_2) and the eight reactions listed in Table 4.1 (see [16] for more details). The initial temperature and pressure are defined to be at supercritical conditions, and the overall oxidation process is assumed to be isothermal. For this chemical system, uncertainties in the forward reaction rates, $k_{f,j}$, are present: from experimental data, these uncertainties have been characterized in [16] by their median values $\hat{k}_{f,j}$ and respective uncertainty levels UF_j. It is assumed that the actual forward rates are contained, with a 95% confidence level, within the lower and upper bounds $\hat{k}_{f,j}/UF_j$ and $\hat{k}_{f,j} \times UF_j$. Moreover, the forward reaction rates are modeled as independent random variables, and their probabilistic behaviors are modeled as log-normal distributions. It has to be noted that the assumption of independence is fairly strong but justified from the information theory point of view in the absence of any knowledge regarding the joint densities of the reaction rates: it yields a less informative probability law of the chemical model. From these assumptions, the eight forward rates can be parameterized using a set of eight independent normalized Gaussian variables $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_8)$ according to

$$(\forall (i,j) \in \{1,\ldots,8\}^2) \quad \begin{cases} k_{f,j}(\xi_j) = \hat{k}_{f,j} \exp\left[\dfrac{\log(UF_j)}{1.96}\xi_j\right], \\ \xi_j \sim \mathcal{N}(0,1),\ \langle \xi_i \xi_j \rangle = \delta_{ij}, \end{cases} \tag{4.1}$$

where [17,18] the factor 1.96 ensures that the rate $k_{f,j}$ is in the range $[\hat{k}_{f,j}/UF_j, \hat{k}_{f,j} \times UF_j]$ with a probability equal to 0.95. The reverse rates are constructed from the equilibrium constants $K_{c,j}^{-1}$ of each of the eight reactions, which we presume to be known with unit probability. As a consequence, the reverse and forward rates are fully correlated and related by $k_{r,j}(\xi_j) = k_{f,j}(\xi_j) \times K_{c,j}^{-1}$. The equilibrium constants are computed using the expected values of the heats of formation reported in Table 2 of [16], neglecting any uncertainty therein, and are listed in Table 4.1. From this parameterization of the reaction rates using the Gaussian vector $\boldsymbol{\xi}$, we deduce their MW expansions in terms of the random vector $\boldsymbol{x} = (x_1, \ldots, x_8)$ with independent components, all being uniformly distributed on $[0, 1]$.

4.2. Solution method. The stochastic process to be modeled is then fully described by the time evolution of the state vector containing the species concentrations

$$\tilde{\mathcal{P}}(t, \boldsymbol{x}) = ([\mathrm{OH}](t, \boldsymbol{x}), [\mathrm{H}](t, \boldsymbol{x}), \ldots, [\mathrm{HO_2}](t, \boldsymbol{x}), [\mathrm{H_2O_2}](t, \boldsymbol{x}))^{\mathrm{T}}, \tag{4.2}$$

TABLE 4.1

List of reactions and chemical data for the reduced hydrogen oxidation mechanism of [16]. $\hat{k}_{f,j}$ *are the median forward rates, the* $K_{c,j} = k_{f,j}/k_{r,j}$ *are the (certain) equilibrium constants, and the coefficients* UF_j *measure the uncertainty levels in each forward reaction rates as discussed in the text.*

j	Reaction			$\hat{k}_{f,j}$	$K_{c,j}$	UF_j
1	$OH + H$	$\longleftrightarrow$	H_2O	$1.479\ 10^{14}$	0	3.16
2	$H_2 + OH$	$\longleftrightarrow$	$H_2O + H$	$6.295\ 10^{11}$	$4.380\ 10^{-04}$	1.26
3	$H + O_2$	$\longleftrightarrow$	HO_2	$8.314\ 10^{13}$	$9.879\ 10^{-14}$	1.58
4	$HO_2 + HO_2$	$\longleftrightarrow$	$H_2O_2 + O_2$	$7.281\ 10^{11}$	$1.045\ 10^{-09}$	1.41
5	$H_2O_2 + OH$	$\longleftrightarrow$	$H_2O + HO_2$	$3.469\ 10^{12}$	$3.382\ 10^{-09}$	1.58
6	$H_2O_2 + H$	$\longleftrightarrow$	$HO_2 + H_2$	$1.696\ 10^{11}$	$7.723\ 10^{-06}$	2.00
7	H_2O_2	$\longleftrightarrow$	$OH + OH$	$3.993\ 10^{01}$	$1.589\ 10^{+11}$	3.16
8	$OH + H_2O$	$\longleftrightarrow$	$H_2O + O_2$	$3.917\ 10^{13}$	$3.534\ 10^{-18}$	3.16

all having MW expansions. The chemical system is governed by the following system of coupled ODEs:

$$\frac{\mathrm{d}\tilde{\mathcal{P}}(t,\boldsymbol{x})}{\mathrm{d}t} = \mathcal{R}(\tilde{\mathcal{P}}(t,\boldsymbol{x}),\boldsymbol{x}), \tag{4.3}$$

where the right-hand side vector $\mathcal{R}$ is a nonlinear functional of the uncertain species concentrations and reaction rates. Introduction of the MW expansions of the concentrations and reaction rates into the governing equations, followed by projection onto the spectral basis, yields a system of $7(\mathrm{Nw}+1)$ ODEs for the MW coefficients, where we recall that $\mathrm{Nw}+1$ is the dimension of the MW basis. Due to the broad spectrum of time scales involved in this chemical system, the deterministic system of equations resulting from the Galerkin projection is stiff, as for the deterministic case. In the following, we shall make use of the software package DVODE of Brown, Hindmarsh and Byrne, designed for the integration of systems of stiff ODEs [4, 5].

4.3. Initial conditions. To solve the chemical problem, initial conditions are needed. In this work, the initial concentrations are certain and set to

$$\begin{cases} [OH] = [H] = [HO_2] = [H_2O_2] = 0, \\ [H_2O] = 4.281\ 10^{-3}, \quad [H_2] = 2.06\ 10^{-6}, \quad [O_2] = 1.04\ 10^{-6}. \end{cases} \tag{4.4}$$

Since these initial conditions are certain, they correspond to the initial coefficients of the zero modes (associated to $\mathcal{M}_0$) of the solution while all other MW coefficients are initially vanishing.

5. One-dimensional MW modeling of supercritical hydrogen oxidation. We first restrict our attention to the case where only the seventh reaction rate constant is uncertain. Hence, the problem can be modeled with a single stochastic dimension, enabling intensive computations and analysis. Still, the uncertainty in $\tilde{k}_{f,7}$ and $\tilde{k}_{r,7}$ impacts the full set of species concentrations in a nontrivial way, as might be expected given the coupling of different species by the chemical source term. Since the mapping of the reaction rate in terms of the CDF $x(\xi)$ exhibits an infinite growth as $x \to 1$, significant resolution is required to properly represent the distribution for the highest values of $\tilde{k}_{f,7}$. However, while the convergence of the mapping with increasing number of resolution levels is quite slow for the zero order expansion, very few levels are needed when using higher order. These claims are verified in the following results.

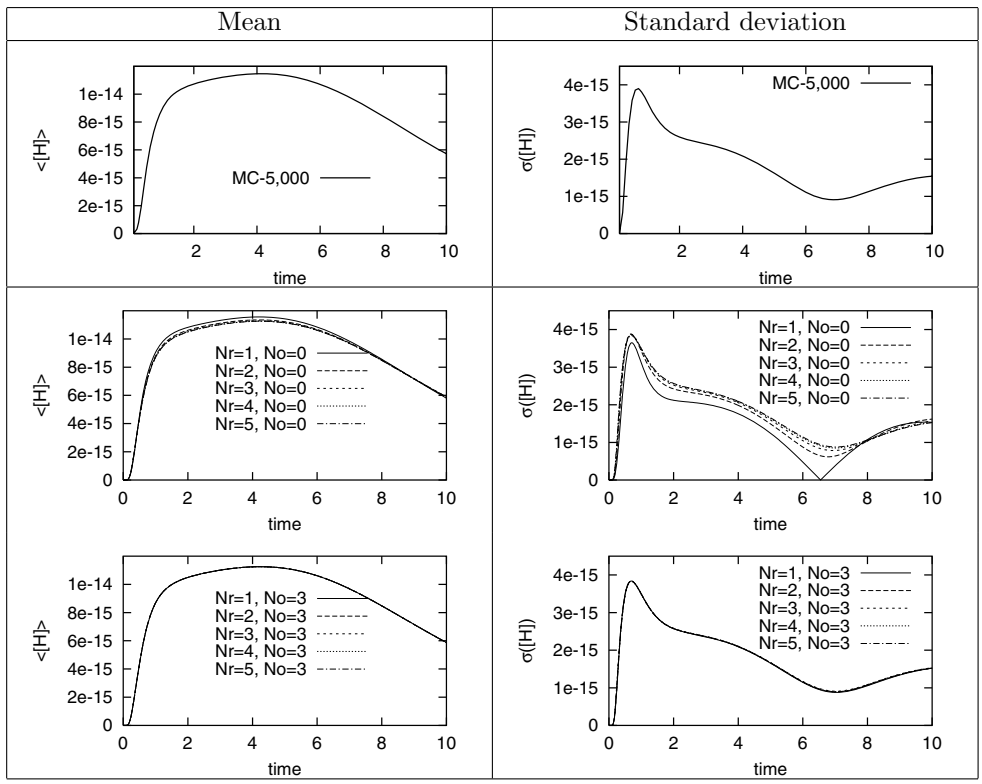


FIG. 5.1. *Mean (left) and standard deviation (right) of* H *concentration as a function of time. Top row: computations with MC sampling using a* 5000*-point sample. Bottom two rows: computations with MW expansions using* N_o *and* N_r *as indicated. Only the uncertainty in the seventh reaction (see Table* 4.1*) is considered, all other forward rates being taken equal to their median values.*

5.1. First order moments. In order to have a baseline against which the MW results can be compared, we provide approximations of the mean and standard deviation of H concentration (as functions of time) that we obtained using straightforward MC sampling, with a 5000-point sample of $\tilde{k}_{f,7}$. Note that the sole purpose of this calculation, whose results are shown in the first row of Figure 5.1, is to illustrate the correctness of the MW results at first and second (statistical) orders. In particular, we do not compare performance in terms of execution speed, since the following hold:

- Brute force MC sampling can be trivially implemented in an embarassingly parallel fashion; on the other hand, despite the fact that the MW algorithm (see section 6.1.2) uses mutually independent subdomains, further analysis must be conducted as to how a parallel implementation would scale.
- In order to achieve good approximations of higher order statistical moments using MC sampling, one typically has to increase the sample size as the order increases. However, this is not the case with the MW approach. Therefore, a fair comparison would require us to compare the advantages of the former method with respect to MC, which is not the purpose of this paper.

In this context, the plots of the last two rows in Figure 5.1 show the results obtained with the intrusive MW approach for $N_o = 0$ and 3 and $N_r = 1, \ldots, 5$.

Focusing first on the zero-order case, i.e., Wiener–Haar expansion, depicted on the top row in the figure, the convergence of the solution when N_r increases is clearly visible. In fact, for the mean of $[H](t)$, the solution obtained with $N_r = 1$ is slightly overestimated, but predictions for $N_r \geq 2$ are hardly distinguishable in the plot and are in excellent agreement with the higher order computation of the last row. For the standard deviation, the convergence appears to be slower with N_r, and the case with $N_r = 1$ significantly underestimates the statistical dispersion of the concentration induced by the uncertainty. It even predicts the absence of any uncertainty in the concentration for $t \approx 6.5$. Nonetheless, the convergence of the second order statistics, albeit slow, is still achieved for $N_o = 0$. Similar trends are reported when using $N_o = 1$ and 2 (not shown), but with more accurate predictions for $N_r = 1$ and a faster convergence rate. For $N_o = 3$, in the bottom row of Figure 5.1, even the prediction for $N_r = 1$ provides very accurate results, as far as these low order moments are concerned. Increasing the level of resolution does not seem to bring additional significant information into the uncertain process, even if the quality of the mapping appears fairly poor for $N_r = 1$ and $N_o = 3$. In fact, a closer analysis (see below) of the supplementary details introduced when increasing the resolution level indicates that they do improve the representation, although their impact on the low order statistics is negligible; a finer analysis of the predicted process distribution is necessary to assess the improvement resulting from increasing resolution. Last, but not least, we see a very good agreement between MC and MW results, agreement that improves as N_o increases.

To support the assertion that the MRA refinement essentially improves the higher order statistics, we show in the left-hand plot of Figure 5.2 the time evolution of the expected concentration in H with the $\pm$ one standard deviation bounding box ($\sigma([H])$). In addition, the deterministic solution corresponding to the median value of the forward rate (solid line) is also reported in the same plot. While one may have expected the deterministic median solution to remain roughly within the neighborhood of the expected solution, it appears that it actually departs significantly from the averaged realization, reaching the $+\sigma$ bound for $t \approx 6.5$. This behavior is characteristic of highly skewed processes, reflecting the nonlinearity of the chemical source term. To better appreciate the skewness of the uncertainty in the H concentration distribution, we provide in the right-hand plot of Figure 5.2 a set of 40 sample paths of $[H](t)$ for random realizations of $\tilde{k}_{f,7}$ following the prescribed log-normal distribution. These sample paths highlight the complex dependence of the concentration on the value of the seventh forward rate: it results in a skewed distribution of [H] with a sharp upper tail at $t > 5$. At this point, it should be clear that first and second order statistical moments are not enough to fully characterize the impact of the uncertainty in $k_{f,7}$ on the solution.

5.2. Analysis of concentration probability density functions. To stress further the need for accurate high order statistics in order to properly represent the impact of uncertainty on the chemical process under study, and to illustrate the improvement in this direction resulting from increasing the level of resolution in the MRA scheme, we now focus on the PDF of the species concentrations. To this end, we use for each species sp the time-dependent [illegible] $\mathcal{R}_{[\mathsf{sp}]}$ of species concentration, defined as follows:

$$(\forall t \in [0, 10]) \quad \mathcal{R}([\mathsf{sp}], t) = \frac{\mathrm{pdf}([\mathsf{sp}](t))}{\max_{[\mathsf{sp}]} \mathrm{pdf}([\mathsf{sp}](t))}. \tag{5.1}$$

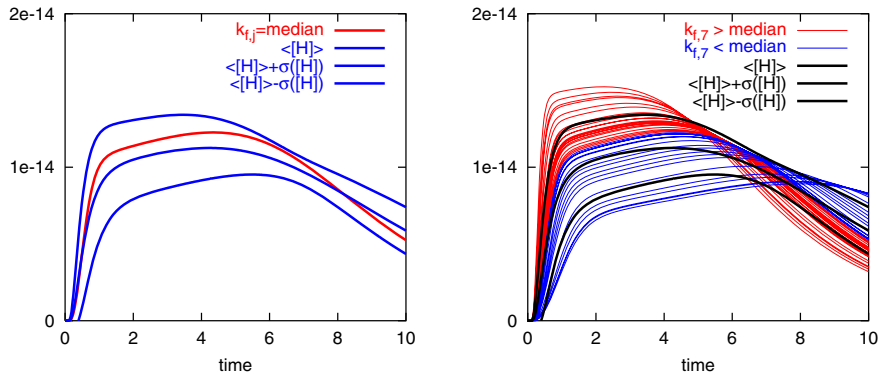


FIG. 5.2. *Left: mean of* $[\mathrm{H}](t)$ *with* $\pm\sigma([H])$ *box (blue lines) and deterministic* $[\mathrm{H}](t)$ *for* $\tilde{k}_{f,7}(x) = \hat{k}_{f,7}$ *(red line). Right:* 40 *deterministic samples of* $[\mathrm{H}](t)$*, corresponding to* $\tilde{k}_{f,7} > \hat{k}_{f,7}$ *(red lines) and* $\tilde{k}_{f,7} < \hat{k}_{f,7}$ *(blue lines), together with the computed expectation of* $[\mathrm{H}](t) \pm \sigma([\mathrm{H}])$ *(black). Mean and standard deviation of* $[\mathrm{H}](t)$ *were obtained using* $\mathrm{N_r} = 5$ *and* $\mathrm{N_o} = 3$.

Hence, one always has $\mathcal{R}([\mathsf{sp}], t) \in [0, 1]$, with equality to 1 if and only if $\mathrm{pdf}([\mathsf{sp}])$ is equal to the highest species PDF at time t; this is in particular the case for the maximum likelihood estimation of $[\mathsf{sp}]$, when it exists—with no guarantee of uniqueness. On the contrary, $\mathcal{R}([\mathsf{sp}], t)$ vanishes if and only if the PDF of $[\mathsf{sp}]$ at time t vanishes as well. The rescaling of the concentration PDF, with its local-time maxima, enables time-tracking of the concentration value with the highest PDF by removing the large fluctuations in magnitude of the absolute PDF. As with the PDF, however, a thin $\mathcal{R}$ for a given observable means a highly predictable concentration, while broad distributions or disperse peaks in $\mathcal{R}$ reflect large variability, possible unpredictability (at least unreliability), and a low confidence level in the predicted concentrations. In Figure 5.3, we have plotted the relative likelihood for two species concentrations: $[\mathrm{H}_2]$ and $[\mathrm{H}]$. The PDFs of the species concentrations were estimated using a set of one million samples for x, uniformly drawn on $[0, 1]$, and used to compute the corresponding concentration samples from their respective MW expansions.

The results in the figure clearly show the convergence of $\mathcal{R}([\mathrm{H}_2], t)$ and $\mathcal{R}([\mathrm{H}], t)$ (and thus of their respective PDFs) with increasing $\mathrm{N_r}$. The dominant features of the $\mathcal{R}([\mathrm{H}_2])$ time evolutions are largely converged for $\mathrm{N_r} = 3$, while low $\mathrm{N_r}$ artifacts in $\mathcal{R}([\mathrm{H}], t)$ are not eliminated until $\mathrm{N_r} = 5$. Further, while $\mathcal{R}([\mathrm{H}_2], t)$ exhibits generally smooth behavior, persistent sharp features are evident in $\mathcal{R}([\mathrm{H}], t)$ for $t > 4$. This suggests the role of implicit equilibrium constraints in the chemical system at late time (as equilibrium is approached) and the high sensitivity of the system to $[\mathrm{H}]$. The result is that a very narrow distribution in $[\mathrm{H}]$ is allowed for $t > 4$. Of course it is true that $[\mathrm{H}_2]$ also exhibits a narrowing distribution at late time, but of a clearly different qualitative nature. Note that the equilibrium state of the system is not uncertain, as it is determined by the thermodynamics of the mixture and not by the rate kinetics. Since uncertainty is only presumed here in one kinetic rate constant, it is expected that, for sufficiently large time, the residual distribution of $\mathcal{R}([\mathrm{H}_2])$ and $\mathcal{R}([\mathrm{H}])$ in the figure will approach peaked functions. Moreover, while $[\mathrm{H}_2]$ tends to zero at equilibrium and must remain positive, $[\mathrm{H}]$ tends to a nonzero limit and is skewed toward an upper limit (highest concentrations).

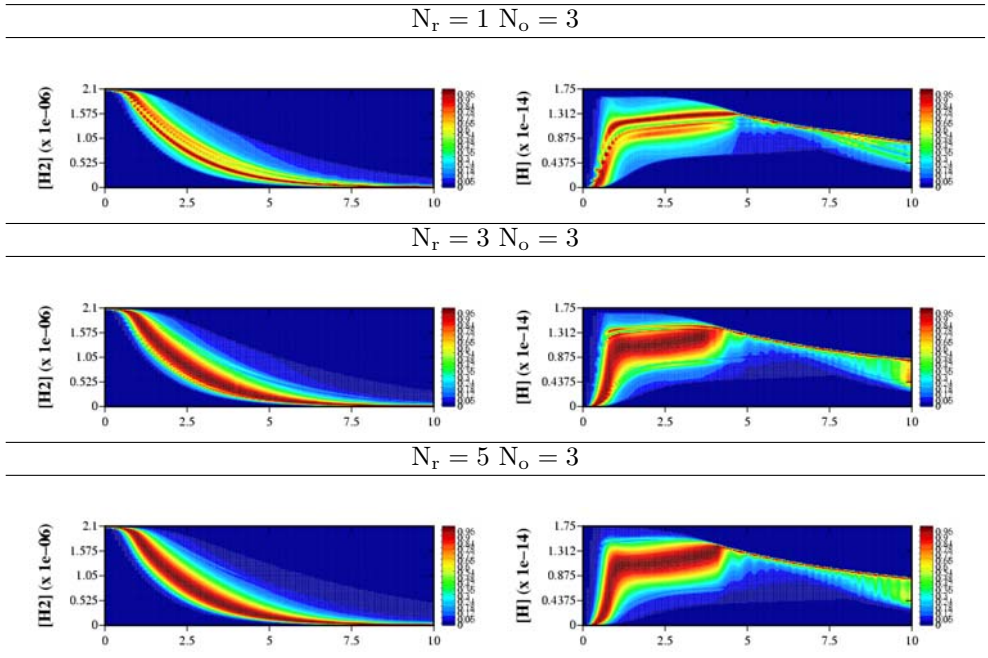


FIG. 5.3. *Color coded normalized PDFs* $\mathcal{R}([\mathrm{H}_2], t)$ *(left) and* $\mathcal{R}([\mathrm{H}], t)$ *(right) computed from their MW expansion using different levels of resolution as indicated, and polynomial order* $\mathrm{N_o} = 3$. *Dark red areas indicate concentrations of highest probability density of observation, while deep blue regions correspond to vanishing probability densities.*

6. Multidimensional process. Direct extension of the methodology of the previous section to the full uncertain problem is not feasible, because the size of the MW basis for given $\mathrm{N_r}$ and $\mathrm{N_o}$ exhibits a prohibitive rate of growth with the number of independent parameters. Also, if the uncertain process to be expanded is essentially smooth with regard to the parameter set, with only localized steep variations along only a few of the stochastic dimensions, one can expect that a large part of the MW coefficients have low magnitude, so that many can actually be neglected. However, the set of MW coefficients that could be removed from the expansion is not known a priori, and an adaptive strategy that dynamically selects the relevant coefficients is required in order to maintain an acceptable computational cost.

Although appealing, adaptive strategies based on coarsening/refinement of the set of MW coefficients are difficult to implement in multidimensional problems and not necessarily optimal from a CPU-load point of view, since any changes in the basis imply changes in the product tensors D_{ijk}, whose evaluation could be expensive if it were to be repeated many times during the simulation. Further, when the number of MW coefficients increases, the size of the set of nonvanishing elements in the binary and ternary products increases dramatically (see [13]) due to the overlapping supports of the MW at different resolution levels. This is a severe limitation for the use of MW expansions for $\mathrm{N_d} > 1$ because the binary product is the key operator in pseudospectral computations [8, 17].

These considerations lead us to the development of an alternative adaptive strategy in [13], which retains the fundamental feature of MRA, specifically the piecewise polynomial character of the expansion, while overcoming the issue of increasing com-

plexity of the binary product when the resolution level increases. The main ingredient of this adaptive scheme is the construction of an "analyzer-basis," involving multi-dimensional Legendre polynomials plus first resolution level one-dimensional details. This analyzer can be rescaled (to an arbitrary resolution level in each of the stochastic dimensions) and displaced (within the parameter space) to compute local expansions of the process. Within this framework, the aim of the adaptive strategy is to decide what is the minimal local level of resolution necessary to properly approximate the solution. In contrast with the algorithm presented in [13], where the resolution was held constant over the simulation time, we construct here a refinement that adaptively evolves with time. Below, we detail the construction of the adaptive scheme, which is then applied to the chemical test-problem.

6.1. Set partition of the parameter space and local basis. Let $\Omega = [a_1, b_1] \times \cdots \times [a_N, b_{\mathrm{N_d}}]$ be the space of random parameters. Let Ω^m, $m = 1, \ldots, \mathrm{N_b}$, be a finite set partition of Ω in $\mathrm{N_b}$ nonoverlapping subdomains:

$$\left\{ \begin{array}{l} \Omega^m = [a_1^m, b_1^m] \times \cdots \times [a_N^m, b_N^m], \\ \Omega = \bigcup_{m=1}^{\mathrm{N_b}} \Omega^m, \\ \Omega^m \cap \Omega^{m'} = \varnothing \quad \text{if } m \neq m'. \end{array} \right. \tag{6.1}$$

We define as follows the local PDF of $\boldsymbol{\xi}$ on each of the sub-domains Ω^m, denoted by $\mathrm{pdf}^m(\boldsymbol{\xi})$, using the independence of the random parameters:

$$\mathrm{pdf}^m(\boldsymbol{\xi}) = \prod_{d=1}^{\mathrm{N_d}} \mathrm{pdf}_d^m(\xi_d), \quad \mathrm{pdf}_d^m(\xi) = \frac{\mathrm{pdf}_d(\xi)}{\mathrm{p}_d(b_d^m) - \mathrm{p}_d(a_d^m)}. \tag{6.2}$$

Clearly, we have $\mathrm{pdf}^m(\boldsymbol{\xi}) > 0 \quad \text{for } \boldsymbol{\xi} \in \Omega^m$ and $\int_{\Omega^m} \mathrm{pdf}^m(\boldsymbol{\xi}) d\boldsymbol{\xi} = 1$. Moreover, defining $\mathrm{p}_d^m(\xi \in [a_d^m, b_d^m]) = \int_{a_d^m}^{\xi} \mathrm{pdf}_d^m(\xi') d\xi' \in [0,1]$, if x_d^m is uniformly distributed over $[0,1]$, then the random variable $(\mathrm{p}_d^m)^{-1}(x_d^m) \in [a_b^m, b_d^m]$ and has the same distribution as ξ_d^m. Thus, a second order stochastic process can be locally expanded on Ω^m, in terms of the random vector $\boldsymbol{x}^m$, having independent components $(x_1^m, \ldots, x_{\mathrm{N_d}}^m) \in [0,1]^{\mathrm{N_d}}$, with uniform distributions. Now let $\boldsymbol{\gamma}$ be the set of multidimensional indices

$$\boldsymbol{\gamma}(\mathrm{N_o}) = \left\{ (\gamma_1, \ldots, \gamma_{\mathrm{N_d}}) : \sum_{d=1}^{\mathrm{N_d}} \gamma_d \leq \mathrm{N_o} \right\}.$$

For $\boldsymbol{\xi} \in \Omega^m$, we build the local projection basis as

$$\mathcal{B}_p(\Omega^m, \mathrm{N_o}) = \left\{ \Phi^m_{\boldsymbol{\lambda} \in \boldsymbol{\gamma}(\mathrm{N_o})}(\boldsymbol{x}^m(\boldsymbol{\xi}^m)) = \prod_{d=1}^{\mathrm{N_d}} \phi_{\lambda_d}(x_d^m(\xi_d^m)) \right\},$$

and the details directional basis $\mathcal{B}_a^d(\mathrm{N_o})$, $d = 1, \ldots, \mathrm{N_d}$, as

$$\mathcal{B}_a^d(\Omega^m, \mathrm{N_o}) = \{\psi_i(x_d^m), i = 0, \ldots, \mathrm{N_o}\}.$$

The full local expansion basis will be the union of $\mathcal{B}_p$ and $\mathcal{B}_a^d$:

$$\mathcal{B}(\Omega^m, \mathrm{N_o}) = \mathcal{B}_p(\Omega^m, \mathrm{N_o}) + \bigcup_{d=1}^{\mathrm{N_d}} \mathcal{B}_a^d(\Omega^m, \mathrm{N_o}).$$

Finally, the multidimensional process $\mathcal{P}(\boldsymbol{\xi})$ will have for local expansion on Ω^m

$$\mathcal{P}(\boldsymbol{\xi}) = \tilde{\mathcal{P}}(\boldsymbol{x}^m(\boldsymbol{\xi})) \approx \sum_{\boldsymbol{\lambda}\in\boldsymbol{\gamma}} \tilde{\mathcal{P}}^m_{\lambda_1,\ldots,\lambda_N} \Phi_{\lambda_1,\ldots,\lambda_N}(\boldsymbol{x}^m(\boldsymbol{\xi})) + \sum_{d=1}^{\mathrm{N_d}} \sum_{i=0}^{\mathrm{N_o}} \tilde{\mathcal{P}}^m_{d,i} \psi_i(x^m_d), \tag{6.3}$$

where again, the equality holds in the mean-square sense. Note that the local basis $\mathcal{B}(\Omega^m, \mathrm{N_o})$, spanning the local expansion of $\mathcal{P}$ according to (6.3), is in fact the rescaled Legendre polynomial basis $(\mathcal{B}_p(\Omega^m, \mathrm{N_o}))$ augmented with the first-level detail basis, $\mathcal{B}^d_a$, $d = 1, \ldots, \mathrm{N_d}$. Thus, $\tilde{\mathcal{P}}(\boldsymbol{x})$ for $\boldsymbol{\xi}(\boldsymbol{x}) \in \Omega^m$, approximated by (6.3), is the local Wiener–Legendre projection of order $\mathrm{N_o}$, plus one-dimensional details. The local expectation of $\mathcal{P}$ is given by $\langle\mathcal{P}\rangle_{\Omega^m} = \mathcal{P}^m_{0,\ldots,0}$, and its local variance is

$$\sigma^2_{\Omega^m}(\mathcal{P}) \approx \left(\hat{\sigma}_{\Omega^m}\right)^2 + \sum_{d=1}^{\mathrm{N_d}} \left(\sigma^d_{\Omega^m}\right)^2,$$

where $\left(\hat{\sigma}_{\Omega^m}\right)^2 = \sum_{\boldsymbol{\lambda}\in\boldsymbol{\gamma}_p} (\mathcal{P}^m_{\boldsymbol{\lambda}})^2$, $\left(\sigma^d_{\Omega^m}\right)^2 = \sum_{i=0}^{\mathrm{N_o}} \left(\mathcal{P}^m_{d,i}\right)^2$, and we have denoted $\boldsymbol{\gamma}_p = \boldsymbol{\gamma} - \{(0,\ldots,0)\}$. The total expectation of the process is given by the volume-weighted summation of the local expectations:

$$\langle\mathcal{P}\rangle = \sum_{m=1}^{\mathrm{N_b}} \langle\mathcal{P}\rangle_{\Omega^m} \mathrm{Vol}^m, \tag{6.4}$$

where Vol^m is the Euclidean volume of Ω^m: $\mathrm{Vol}^m = \prod_{d=1}^{\mathrm{N_d}} (\mathrm{p}_d(b^m_d) - \mathrm{p}_d(a^m_d))$. Finally the total variance of the process is given by

$$\sigma^2(\mathcal{P}) = \sum_{m=1}^{\mathrm{N_b}} \left[\sigma^2_{\Omega^m}(\mathcal{P}) + \left(\mathcal{P}^m_{0,\ldots,0} - \langle\mathcal{P}\rangle\right)^2\right] \mathrm{Vol}^m. \tag{6.5}$$

6.1.1. Adaptive strategy. Assume that the current set partition of Ω involves $\mathrm{N_b}$ blocks; i.e., $\Omega = \bigcup_{m=1}^{\mathrm{N_b}} \Omega^m$. On each subdomain Ω^m, the process is expanded on the local basis $\mathcal{B}(\Omega^m, \mathrm{N_o})$, the spectral coefficients being computed through Galerkin projection methods as previously. To decide if a given block m needs more refinement and to determine which stochastic directions need such refinement, we consider the criterion

$$C_m(d) = \frac{\sigma^d_{\Omega_m}}{\sigma_{\Omega_m}} \geq \varepsilon_r, \tag{6.6}$$

and we refine the subdomain along the dth dimension if this inequality is satisfied. Here $\varepsilon_r < 1$ is a prescribed threshold. The test compares the "energy" of the one-dimensional details along the dth stochastic direction with the local variance of the solution. In other words, the one-dimensional detail coefficients are used as indicators of the quality of the representation along their respective stochastic directions. A new set partition of Ω is then constructed by splitting Ω^m into smaller subdomains. Specifically, if we assume the inequality (6.6) is satisfied for a single dimension d, then refinement of $\Omega^m = [a^m_1, b^m_1] \times \cdots \times [a^m_N, b^m_N]$ will give birth to two new subdomains $\Omega^{m'}$ and $\Omega^{m''}$ of equal volume, with common boundary at $(a^m_d + b^m_d)/2$. Then, local expansions of the process on the newly created subdomains are computed, before

being analyzed to determine whether additional refinement is needed. This sequence of analysis and refinement steps is repeated up to convergence. It is emphasized that, during refinement, computations are performed in newly created subdomains only, since the local solutions over other subdomains are unaffected. Note also that this methodology is well suited for parallel implementation, since local computations are independent of each other.

6.1.2. Application to chemical systems. We now provide the algorithm used to integrate the system of stochastic ODEs, with dynamical adaptation of the parameter space set partition. In order to minimize the computational load, we do not recompute the solution, starting from the initial time, for each refined subdomain, as in [13]. Instead, we advance the solution in time using equally spaced stages, where the correctness of the set partition with regard to the threshold function ϵ_r is verified. If the set partition at the end of a given stage k needs further refinement, then it is refined, and the solution over the created subdomains is computed from the initial conditions corresponding to the solution at the end of the stage $k-1$. A procedure is required to provide initial conditions for the new subdomains at the beginning of the current stage k. This is performed efficiently and without approximation using polynomial interpolations of the solution of the "parent" subdomain that has been split.

Suppose that we wish to integrate the system for $t \in [0, T_f]$. We denote by ΔT the time interval between successive analyses of the set partition. With this notation, the time-marching algorithm has the following structure:

1. **Initialization:**
   (a) Set times $t = 0$, $t_0 = 0$, and $t' = \Delta T$;
   (b) Initialize the set partition of Ω to $\Omega^1 = [0,1]^{\mathrm{N_d}}$, $\mathrm{N_b} = 1$;
   (c) Project the initial condition of the concentration on Ω^1;
   (d) Set subdomain flag $I(m = 1)$ to 1.
2. For each subdomain Ω^m of the set partition, $m = 1, \ldots, \mathrm{N_b}$, with $I(m) = 1$:
   (a) Set the local expansion of the forward and reverse reaction rates;
   (b) Set the local expansions for the concentrations at time t_0;
   (c) Perform the time integration between t_0 and t' of the local system of stochastic ODEs.
3. Set $\mathrm{N_b}' = \mathrm{N_b}$.
4. For each subdomain Ω^m of the set partition, $m = 1, \ldots, \mathrm{N_b}'$, with flag $I(m) = 1$:
   (a) Set subdomain flag to $I(m) = 2$;
   (b) Compute $C_m(d)$ given by (6.6), and retain highest value over the set of species concentrations;
   (c) If for all $d = 1, \ldots, \mathrm{N_d}$: $C_m(d) < \varepsilon_r$, then do:
      i. The subdomain is sufficiently refined, update flag to $I(m) = 0$;
      ii. Continue with the next subdomain in the list (step number (4)).
   (d) Else, for $d = 1, \ldots, \mathrm{N_d}$ do:
   If $C_m(d) \geq \varepsilon_r$, then do:
      i. For every subdomain $\Omega^{m' \leq \mathrm{N_b}}$ with $I(m') = 2$, do:
         - Store the solution S_0 at time t_0 over $\Omega^{m'}$;
         - Split (along the d-dimension) the subdomain $\Omega^{m'}$ into two subdomains with index m' and $m'' = \mathrm{N_b} + 1$: $\Omega^{m'} \to \Omega^{m'} \cup \Omega^{m''}$;
         - Set $I(m')$ and $I(m'')$ to 3 to make sure they are refined only once;

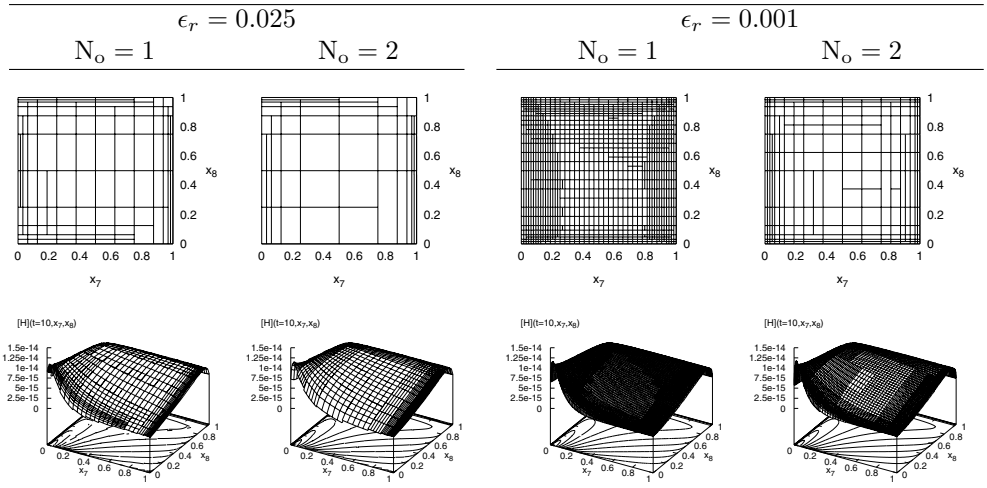


FIG. 6.1. *Results for the 2D problem considering uncertainty in the rates of reactions* 7 *and* 8. *Parameters are* $\epsilon_r = 0.025, 0.001$ *and* $\mathrm{N_o} = 1, 2$ *as indicated. The top row shows the partitioning of the* (x_7, x_8) *plane and its dependence on* ϵ_r *and* $\mathrm{N_o}$. *The bottom row shows the dependence of the* [H] *at* $t = 10$ *on* x_7 *and* x_8.

- Interpolate S_0 on $\Omega^{m'}$ and $\Omega^{m''}$;
- Update the set partition: $\mathrm{N_b} \leftarrow \mathrm{N_b} + 1$.

ii. Shift all subdomain flags with value 3 to $I = 2$, and proceed with the next dimension d in step (d).

(e) Refinement of the subdomain has been completed; set all flags $I = 2$ to $I = 1$ and proceed with the next subdomain in step (4).

5. If $\mathrm{N_b} > \mathrm{N_b}'$, new subdomains have been generated that need to be computed and further analyzed: continue from step (2).
6. The set partition satisfies the refinement criteria; then:
   (a) Record the set partition and the solutions over all subdomains at time $t = t'$, to be used as initial conditions in the next time interval;
   (b) If $t' < T_f$, then:
      i. Set $t_0 \leftarrow t'$, $t' \leftarrow t' + \Delta T$;
      ii. For $m = 1, \ldots, \mathrm{N_b}$ set $I(m) = 1$;
      iii. Continue from step (2).
7. End.

6.2. Test cases. We now present some test cases in order to illustrate the utility of our approach.

6.2.1. Two-dimensional tests. We consider uncertainty in the rates of reactions 7 and 8, and apply the adaptive scheme for three threshold values $\epsilon_r = 0.025$, 0.01, and 0.001. Tests are considered for the first and second order expansion.

In Figure 6.1, we plot the final ($t = 10$) set partitions and response-surfaces for $\epsilon_r = 0.025$, 0.001 and $\mathrm{N_o} = 1, 2$. The results highlight the dependence of the MW discretization and domain decomposition on ϵ_r and $\mathrm{N_o}$. The top row shows the partitioning of the (x_7, x_8) plane and its dependence on ϵ_r and $\mathrm{N_o}$. We note the more refined partitioning with decreasing ϵ_r and $\mathrm{N_o}$. For higher $\mathrm{N_o}$, less refinement of the (x_7, x_8) space is necessary for a given allowed directional block variance controlled by ϵ_r. The localization of the highly refined domain blocks at the $(0, 1)$ limits of x_7

and x_8 reflect both the variability of the rate constant mappings and the dependence of various elements of the solution on the (x_7, x_8) space. The dependence of [H] (at $t = 10$) on (x_7, x_8) is shown in the bottom row in Figure 6.1, again for the indicated ranges of ϵ_r and $\mathrm{N_o}$. The [H] response-surfaces exhibit strong variability with (x_7, x_8) in the vicinity of the origin, with generally smoother behavior elsewhere.

As it can be appreciated from the top row of Figure 6.1, the number of bins is significantly higher for $\mathrm{N_o} = 1$ than for $\mathrm{N_o} = 2$ to achieve the same threshold value. An optimal strategy ultimately requires suitable choices of $\mathrm{N_o}$ and ϵ_r for the problem at hand.

We note finally the relative insensitivity of the [H] response-surface shape to the choice of ϵ_r and $\mathrm{N_o}$ in the ranges considered in Figure 6.1. This is also evident in the mean and standard deviation profiles of [H] versus time (not shown). Any changes in these two moments over the specified range of ϵ_r and $\mathrm{N_o}$ is simply not observable in the profiles for all time.

6.2.2. High-dimensional tests. In this section the adaptive scheme is applied, considering all the reaction rates as uncertain. The simulation runs up to $t = 10$ as previously. The left-hand plot of Figure 6.2 displays the numbers of bins involved in the partitions as a function of time for different orders $\mathrm{N_o}$ and refinement threshold ϵ_r. It indicates that the number of bins quickly increases at the early stages of the simulation and then tends to level off. It may also be noticed that with $\epsilon_r = 0.05$ and $\mathrm{N_o} = 3$ the computations do not reach $t = 10$ because of numerical instabilities (even though the computations are stable for a longer period than with WHe expansions).

Clearly, the efficiency of the partitioning strategy depends on the capability of the algorithm to divide the parameter space into an optimal number of subdomains. To minimize this number of subdomains, the successive divisions have to be limited to the directions along which steep dependences are present, while avoiding division along smooth dimensions. To appreciate the capability of the algorithm to discriminate directions (uncertain reaction rates) requiring divisions from the smooth ones, we present in the right-hand plot of Figure 6.2 the frequency of successive divisions along the different dimensions of the parameter space for $\mathrm{N_o} = 1$ and $\epsilon_r = 0.05$. It is shown that no divisions are done along dimensions 1–4 and 6, denoting a smooth dependence of the concentrations with regard to the uncertainties in the corresponding reaction rates. On the contrary, reactions 5, 7, and 8 do require divisions, up to 3 successive ones in some areas of the parameter space. When ϵ_r is lowered to 0.005, more divisions are necessary along the previously divided directions to meet the refinement criteria, while some subdomains have now also been cut along directions 6 and 2, but with a significantly lower frequency (not shown).

It is seen from the results above that using a higher expansion order requires a lower number of subdomains to satisfy a fixed refinement criteria. However, the dimension of the local basis quickly increases with the order $\mathrm{N_o}$ such that a lower number of subdomains does not necessarily imply a lower computational cost. In fact, for given ϵ_r, the selection of the optimal order in terms of CPU cost remains an open question. Moreover, because the solutions over subdomains are fully independent and can be computed in parallel, the trade-off between number of subdomains and expansion order should account for memory requirement and data flow as well. Also, the optimal trade-off is certainly problem dependent so no general guidelines can be established. Another aspect that needs to be underlined is that different orders used with the same threshold value will not generally lead exactly to the same accuracy, although the same order of error is expected. This claim has been verified through

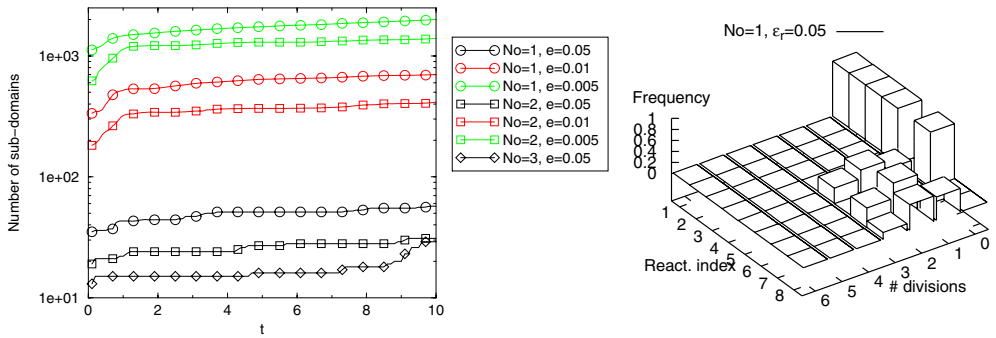


FIG. 6.2. *Left: evolution of the number of bins in the partitions as a function of time and for different expansion orders* N_o *and refinement threshold value* ϵ_r *as indicated. Right: relative frequency along each dimension of the parameter space (reaction indices) of the number of successive divisions. The graphs correspond to the partition at time* $t = 10$ *for* $N_o = 1$ *and threshold value* $\epsilon_r = 0.05$.

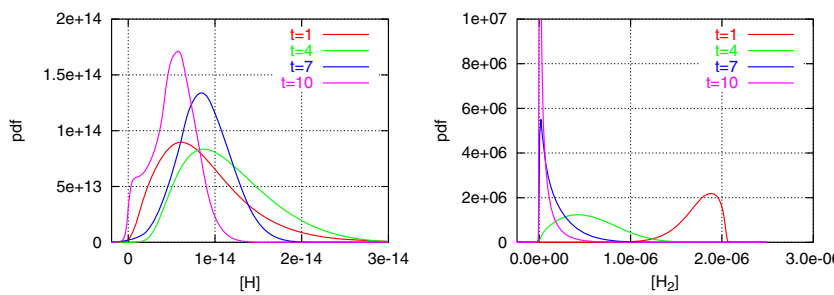


FIG. 6.3. *Pdfs of* [H] *and* [H_2] *at different times, computed with* $N_o = 1$ *and* $\epsilon_r = 0.005$.

inspection of the time evolution of the expectations and standard deviations of the species concentrations when computed with different orders and threshold values. It is found that all predictions are in good agreement and appear to converge to the most refined and highest order solution.

Again, the tests suggest a fast convergence of the first moment with both the threshold value and the expansion order. However, if one is interested in a more complete statistical description of the solution, then the improvement in the prediction with decreasing ϵ_r and increasing expansion order are more visible. To illustrate this fact, we provide a quick analysis of the resulting pdfs of some species concentrations. Provided in Figure 6.3 are pdfs of [H] and [H_2] at different times, computed with $N_o = 1$ and $\epsilon_r = 0.005$ (other species exhibit similar pdfs). The plots indicate that the pdfs are smooth (except at the tails), denoting no significant discontinuities of the solution across subdomain boundaries. For less stringent refinement thresholds ($\epsilon_r = 0.1$) discontinuities are present (not shown). Also, the plots demonstrate the need for an adaptation of the partition, as the shapes of the pdfs drastically evolve with time. It is clear that such a complex uncertainty impact would have required a large order expansion if a global expansion is to be used (i.e., only one domain), but such a high order global expansion is numerically unstable, as shown in [17].

Finally, the numerical instabilities in the global spectral simulations reported in [17] were attributed to the spurious development of tails with negative concentrations in the solution expansion. Although the present computations are stable, it is

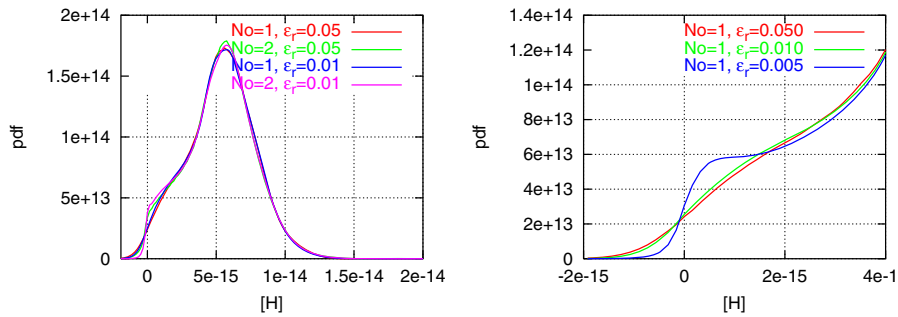


FIG. 6.4. *Left: pdfs of* [H] *for* $N_o = 1$, 2 *and* $\epsilon_r = 0.05$ *and* 0.01. *Right: details of the computed pdfs of* [H], *in the neighborhood of* [H] = 0, $N_o = 1$ *and decreasing refinement threshold as indicated.*

interesting to note that our simulations do not totally satisfy the positivity of the concentrations: the probability of all concentrations being positive is not exactly 0. This is illustrated in Figure 6.4, where pdfs of [H] are plotted at time $t = 10$ for different orders and threshold values. These plots show that the probability of [H] < 0 decreases with the order and the refinement criteria (similar behavior is observed for the other species). It can be concluded that the automatic partitioning of the parameter space stabilizes the computation by maintaining a low order expansion, valid locally, and dividing a subdomain whenever its local solution develops instability. This effect is further illustrated in the right-hand plot of Figure 6.4, which compares the computed pdfs of [H] in the neighborhood of [H] $= 0$ at $t = 10$ for a fixed order $N_o = 1$ and different refinement criteria.

7. Conclusions. In this paper, a multi-resolution analysis scheme for the quantification of parametric uncertainties in chemical systems has been developed. The impact of uncertainties is tackled using an orthogonal projection of the model solution on spectral bases composed of piecewise-continuous polynomial functions (multi-wavelets) of a set of random variables that represent the variability in the system parameters. This approach is designed to allow for local refinement of the representation in areas of the uncertain parameter space where it is needed. Such local refinements are required when the system dynamics exhibit complex and steep dependences on the random parameters (here the reaction rates). In fact, instead of using unnecessarily high order global PC expansions, a series of local low order expansions at controlled resolution levels are employed.

The main improvement of the proposed MRA scheme comes from the elimination of the numerical instability issues encountered with the original PC construction (using a global basis with noncompact support), particularly at higher PC orders. Thus, maintaining a low to moderate expansion order is crucial for the robustness of the computations, while convergence can still be controlled through the selection of an appropriate level of resolution. However, the fast increase of the basis dimension with the resolution level prevents the straightforward application of the MRA scheme to situations involving multiple independent random parameters, and calls for adaptive techniques. Consequently, an attempt toward the local adaptation of the resolution level is proposed. This adaptive technique makes use of the energy contained in the local correction at the next resolution level to decide if more resolution is needed. It naturally leads to an iterative partition of the parameter space, up to a prescribed tolerance, with the property that its computational cost is proportional to

the resulting number of subdomains in the partition and not a function of the basis dimension. Numerical experiments have demonstrated the validity of the proposed adaptive scheme but also highlight the need for an improvement of the methodology to include a coarsening strategy and ultimately a local adaptation of the expansion order too. In fact, the optimal trade-off, between refinement of the partition into smaller subdomains and/or an increase of the local expansion order to reduce the approximation error is problem dependent and needs further investigation; a possible way to determine such an optimal strategy could be based on a posteriori error estimators. Future developments will focus on this issue.

REFERENCES

[1] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions*, Dover, New York, 1970.

[2] B. Alpert, G. Beylkin, D. Gines, and L. Vozovoi, *Adaptive solution of partial differential equations in multiwavelet bases*, J. Comput. Phys., 182 (2002), pp. 149–190.

[3] B. K. Alpert, *A class of bases in L^2 for the sparse representation of integral operators*, SIAM J. Math. Anal., 24 (1993), pp. 246–262.

[4] P. N. Brown, G. D. Byrne, and A. C. Hindmarsh, *VODE: A variable-coefficient ODE solver*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 1038–1051.

[5] G. D. Byrne and A. C. Hindmarsh, *A polyalgorithm for the numerical solution of ordinary differential equations*, ACM Trans. Math. Software, 1 (1975), pp. 71–96.

[6] R. H. Cameron and W. T. Martin, *The orthogonal development of nonlinear functionals in series of Fourier-Hermite functionals*, Ann. Math. (2), 48 (1947), pp. 385–392.

[7] B. Debusschere, H. Najm, A. Matta, R. Ghanem, and O. Le Maître, *Protein labeling reactions in electrochemical flow: Numerical simulation and uncertainty propagation*, Phys. Fluids, 15 (2003), pp. 2238–2250.

[8] B. J. Debusschere, H. N. Najm, P. P. Pébay, O. M. Knio, R. G. Ghanem, and O. P. Le Maître, *Numerical challenges in the use of polynomial chaos representations for stochastic processes*, SIAM J. Sci. Comput., 26 (2004), pp. 698–719.

[9] R. Ghanem, *The nonlinear Gaussian spectrum of lognormal stochastic processes and variables*, ASME J. Appl. Mech., 66 (1999), pp. 964–973.

[10] R. G. Ghanem and P. D. Spanos, *Stochastic Finite Elements: A Spectral Approach*, Springer-Verlag, New York, 1991.

[11] M. Grigoriu, *Stochastic Calculus. Applications in Science and Engineering*, Birkhäuser Boston, Boston, 2002.

[12] O. P. Le Maître, O. M. Knio, H. M. Najm, and R. G. Ghanem, *Wiener-Haar expansion of stochastic processes*, J. Comput. Phys., 197 (2004), pp. 28–57.

[13] O. P. Le Maître, H. M. Najm, R. G. Ghanem, and O. M. Knio, *Multi-resolution analysis of Wiener-type uncertainty propagation schemes*, J. Comput. Phys., 197 (2004), pp. 502–531.

[14] O. P. Le Maître, M. T. Reagan, O. M. Knio, H. N. Najm, and R. G. Ghanem, *A stochastic projection method for fluid flow.* II. *Random process*, J. Comput. Phys., 181 (2002), pp. 9–44.

[15] J. S. Liu, *Monte Carlo Strategies in Scientific Computing*, Springer-Verlag, New York, 2001.

[16] B. D. Phenix, J. L. Dinaro, M. A. Tatang, J. W. Tester, J. B. Howard, and G. J. McRae, *Incorporation of parametric uncertainty into complex kinetic mechanisms: Application to hydrogen oxidation in supercritical water*, Combust. Flame, 112 (1998), pp. 132–146.

[17] M. T. Reagan, H. N. Najm, B. J. Debusschere, O. P. Le Maître, O. M. Knio, and R. G. Ghanem, *Spectral stochastic uncertainty quantification in chemical systems*, Combust. Theory Model., 8 (2004), pp. 607–632.

[18] M. T. Reagan, H. N. Najm, R. G. Ghanem, and O. M. Knio, *Uncertainty quantification in reacting flow simulations through non-intrusive spectral projection*, Combust. Flame, 132 (2003), pp. 545–555.

[19] N. Wiener, *The homogeneous chaos*, Amer. J. Math., 60 (1938), pp. 897–936.

[20] D. Xiu and G. E. Karniadakis, *The Wiener–Askey polynomial chaos for stochastic differential equations*, SIAM J. Sci. Comput., 24 (2002), pp. 619–644.

SIAM J. SCI. COMPUT.
Vol. 29, No. 2, pp. 890–911
© 2007 Society for Industrial and Applied Mathematics

THEORETICAL AND COMPUTATIONAL ISSUES OF OPTIMAL CONTROL FOR DISTRIBUTED HOPFIELD NEURAL NETWORK EQUATIONS WITH DIFFUSION TERM*

QUAN-FANG WANG†

Abstract. This work investigates theoretical and computational issues of optimal control for distributed Hopfield neural network equations with diffusion term. Due to introducing a diffusion term in our model, a theoretical conclusion is deduced for the control of the nonlinear parabolic partial differential equation in the framework of variational methods (cf. [J. L. Lions, *Optimal Control of Systems Governed by Partial Differential Equations*, Springer-Verlag, Berlin, Heidelberg, New York, 1971]) by means of distributed control, boundary control, and initial control. Experimental demonstration is implemented via a semidiscrete numerical algorithm based on the finite element method. The theoretical results are verified using the proposed scheme.

Key words. Hopfield neural network equations, distributed control, boundary control, initial control, numerical approximation, finite element methods

AMS subject classifications. 35B37, 35A40, 35A15, 35K50, 35K55

DOI. 10.1137/050647943

1. Introduction.

The Hopfield neural networks (HNN) model has been extensively considered in various fields, such as the natural sciences (e.g., physics, chemistry), engineering (e.g., artificial intelligence), industry (e.g., signal processing), medicine (e.g., CT detection), and other areas. For related references see [7], [8], [9], [10], [11], [12], [13], [16], and [23]. An interesting topic arises at the viewpoint of biological science: How can we realize control for neural network systems (including about 100 hundred million neurons)? The control input may be language, laser pulse, electric current, or drug simulations, etc. In particular, how can we simulate the control process through experimental demonstration, and what results will be meaningful for the physics problem? That is the reason why we do this research. For more rational meaning in physics, a diffusion term is added to our model.

The goal of this work is to consider the theoretical and computational problems of optimal control for the HNN system with diffusion term by means of distributed, boundary, and initial controls. Physically, boundary control and initial control can be realized by practical methods, but how to realize distributed control (a kind of effect coming from another "neuron," or a kind of perturbation) is still a mystery. The spatial dimensional is restricted to the one-dimensional case in theoretical study for corresponding to numerical study.

Let $\Omega = (0, l)$ be an open bounded domain of $\mathbf{R}^1$ (the theoretical results for $\mathbf{R}^n$ refer to [17]), $T > 0$, and $Q = \Omega \times (0, T)$. Let $(x, t) \in Q$, $y_i = y_i(x, t)$ denote the activation potential of the ith neuron. The distributed HNN model with diffusion term of coupled n-numbers of neurons is described by

*Received by the editors December 18, 2005; accepted for publication (in revised form) November 26, 2006; published electronically May 4, 2007. Related work was presented at the Annual Conference of Japan Society for Industrial and Applied Mathematics (Chuo University, Tokyo, 2003) and the Second International Conference of Applied Mathematics (Plovdiv, Bulgaria, 2005).
http://www.siam.org/journals/sisc/29-2/64794.html
†Department of Automation and Computer-Aided Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong (qfwang@acae.cuhk.edu.hk).

$$
(1.1)\qquad \begin{cases} \dfrac{\partial y_i}{\partial t} - d_i \Delta y_i = -a_i y_i + \displaystyle\sum_{j=1}^{n} c_{ij} F_j(y_j) + u_i \quad \text{in } (0,l)\times(0,T), \\ \dfrac{\partial y_i}{\partial \eta}(0,t) = v_i(0,t), \qquad \dfrac{\partial y_i}{\partial \eta}(1,t) = v_i(1,t), \\ y_i(0,x) = w_i(x) \text{ in } (0,l), \qquad i = 1,2,\dots,n. \end{cases}
$$

Here $d_i > 0$ are diffusion coefficients, $a_i > 0$ and c_{ij} are physical coefficients as explained in [13], u_i are respective distributed control variables, and $v_i(0,t)$ and $v_i(1,t)$ are boundary control variables at $x = 0$ and $x = 1$, respectively. w_i are initial control variables. $F_j : \mathbf{R} = (-\infty, \infty) \to (-1,1)$ are sigmoidal functions representing the effect coming from other neurons, e.g., $F_j(s) = \tanh s$.

Let $\mathcal{U}_i$, $\mathcal{V}_i^0$, $\mathcal{V}_i^1$, and $\mathcal{W}_i$ be the Hilbert spaces of control variables u_i, $v_i(0,t)$, $v_i(1,t)$, and w_i, respectively. Their product spaces are denoted by

$$
\mathcal{U} = \prod_{i=1}^{n} \mathcal{U}_i, \quad \mathcal{V}^0 = \prod_{i=1}^{n} \mathcal{V}_i^0, \quad \mathcal{V}^1 = \prod_{i=1}^{n} \mathcal{V}_i^1, \quad \mathcal{W} = \prod_{i=1}^{n} \mathcal{W}_i
$$

and $\mathbf{u} = (u_1, u_2, \dots, u_n)$, $\mathbf{v}(0) = (v_1(0,t), v_2(0,t), \dots, v_n(0,t))$, $\mathbf{v}(1) = (v_1(1,t), v_2(1,t), \dots, v_n(1,t))$, and $\mathbf{w} = (w_1, w_2, \dots, w_n)$. Set the product Hilbert space $\mathcal{C} = \mathcal{U} \times \mathcal{V}^0 \times \mathcal{V}^1 \times \mathcal{W}$ of control variables $\mathbf{c} = (\mathbf{u}, \mathbf{v}(0), \mathbf{v}(1), \mathbf{w}) \in \mathcal{C}$. Let $\mathbf{y} = \mathbf{y}(\mathbf{c}) = (y_1(\mathbf{c}), y_2(\mathbf{c}), \dots, y_n(\mathbf{c}))^T$ be the solution of the control system for a given $\mathbf{c} \in \mathcal{C}$. The attached cost function to system (1.1) is described by the terminal observation

$$
\begin{aligned} J(\mathbf{c}) = {} & \sum_{i=1}^{n} \int_{\Omega} |y_i(\mathbf{c},T) - z_{id}^T|^2 dx + \sum_{i=1}^{n} (u_i, u_i)_{\mathcal{U}_i} \\ & + \sum_{i=1}^{n} \|v_i(0,t)\|_{\mathcal{V}_i^0}^2 + \sum_{i=1}^{n} \|v_i(1,t)\|_{\mathcal{V}_i^1}^2 + \sum_{i=1}^{n} (w_i, w_i)_{\mathcal{W}_i} \quad \forall \mathbf{c} \in \mathcal{C}, \end{aligned} \tag{1.2}
$$

where $z_{id}^T \in L^2(\Omega)$ are desired values for $i = 1, 2, \dots, n$. Let $\mathcal{C}_{ad}$ be an admissible subset of $\mathcal{C}$. Without loss of generality, $\mathcal{C}_{ad} = L^2(Q)^n \times L^2(0,T)^n \times L^2(0,T)^n \times L^2(\Omega)^n$. The optimal control problem is to find and characterize an element $\mathbf{c}^* = (\mathbf{u}^*, \mathbf{v}^*(0), \mathbf{v}^*(1), \mathbf{w}^*) \in \mathcal{C}_{ad}$, called optimal control, such that

$$
\inf_{\mathbf{c} \in \mathcal{C}_{ad}} J(\mathbf{c}) = J(\mathbf{c}^*),
$$

where $\mathbf{u}^* = (u_1^*, u_2^*, \dots, u_n^*)$, $\mathbf{v}^*(0) = (v_1^*(0,t), v_2^*(0,t), \dots, v_n^*(0,t))$, $\mathbf{v}^*(1) = (v_1^*(1,t), v_2^*(1,t), \dots, v_n^*(1,t))$, and $\mathbf{w}^* = (w_1^*, w_2^*, \dots, w_n^*)$.

This paper is organized as follows. Section 2 states the basic results on the existence and uniqueness of weak solutions for state system (1.1) in the variational framework (cf. [6]). Section 3 proves the existence of an optimal control (distributed, boundary, and initial optimal control) for (1.2). In section 4, a semidiscrete numerical algorithm is proposed, based upon the finite element method. In section 5, numerical experiments are carried out to verify the obtained theoretical results and the efficiency of our proposed algorithm (cf. [2], [3], [4], [17], [19], [20], [21], [22]).

2. Existence and uniqueness of a weak solution. Assume that nonlinear function F_j satisfies the uniform Lipschitz continuity

$$
(2.1)\qquad \exists K > 0 : \quad |F_j(s) - F_j(r)| \le K|s - r|, \quad j = 1, 2, \dots, n.
$$

Introduce two Hilbert spaces $H = L^2(\Omega)$ and $V = H^1(\Omega)$ (cf. [1]) according to the Neumann boundary condition in (1.1). Those spaces are endowed with the usual inner products and norms denoted by $(\cdot,\cdot), |\cdot|, (\!(\cdot,\cdot)\!), \|\cdot\|$, respectively. The product Hilbert spaces $\mathcal{V} = H^1(\Omega)^n$, $\mathcal{H} = L^2(\Omega)^n$ have the inner products defined by

$$(\boldsymbol{\phi},\boldsymbol{\psi})_{\mathcal{H}} = \sum_{i=1}^{n}(\phi_i,\psi_i), \quad \boldsymbol{\phi} = (\phi_1,\phi_2,\dots,\phi_n)^T, \quad \boldsymbol{\psi} = (\psi_1,\psi_2,\dots,\psi_n)^T \in \mathcal{H},$$

$$(\boldsymbol{\phi},\boldsymbol{\psi})_{\mathcal{V}} = \sum_{i=1}^{n}(\!(\phi_i,\psi_i)\!), \quad \boldsymbol{\phi} = (\phi_1,\phi_2,\dots,\phi_n)^T, \quad \boldsymbol{\psi} = (\psi_1,\psi_2,\dots,\psi_n)^T \in \mathcal{V}.$$

Then $\mathcal{V}' = (V^n)'$ is the dual space of $\mathcal{V}$, and the dual pairing between $\mathcal{V}'$ and $\mathcal{V}$ is given by

$$\langle\boldsymbol{\phi},\boldsymbol{\psi}\rangle_{\mathcal{V},\mathcal{V}'} = \sum_{i=1}^{n}\langle\phi_i,\psi_i\rangle, \quad \boldsymbol{\phi} = (\phi_1,\phi_2,\dots,\phi_n)^T \in \mathcal{V}, \quad \boldsymbol{\psi} = (\psi_1,\psi_2,\dots,\psi_n)^T \in \mathcal{V}',$$

where $\langle\phi_i,\psi_i\rangle$ denotes the dual pairing between V and V'. The norms of $\mathcal{V}$ and $\mathcal{H}$ are denoted by $\|\boldsymbol{\psi}\|_{\mathcal{V}}$ and $|\boldsymbol{\psi}|_{\mathcal{H}}$, respectively. For the sake of simplicity, we introduce vectors and matrix representations:

$$\boldsymbol{\Delta}\mathbf{y} = \begin{bmatrix} \Delta y_1 \\ \Delta y_2 \\ \vdots \\ \Delta y_n \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{bmatrix}, \quad \mathbf{F}(\mathbf{y}) = \begin{bmatrix} F_1(y_1) \\ F_2(y_2) \\ \vdots \\ F_n(y_n) \end{bmatrix},$$

$\mathbf{D} = \text{diag}\,\{d_1,d_2,\dots,d_n\}, \mathbf{A} = \text{diag}\,\{a_1,a_2,\dots,a_n\}$. Then the uncontrolled system is rewritten simply by

$$\begin{cases} \dfrac{\partial\mathbf{y}}{\partial t} - \mathbf{D}\boldsymbol{\Delta}\mathbf{y} = -\mathbf{A}\mathbf{y} + \mathbf{C}\mathbf{F}(\mathbf{y}) = \mathbf{g} \quad \text{in } Q, \\ \dfrac{\partial\mathbf{y}}{\partial\eta}(0,t) = \mathbf{k}(0,t), \quad \dfrac{\partial\mathbf{y}}{\partial\eta}(1,t) = \mathbf{k}(1,t), \\ \mathbf{y}(x,0) = \mathbf{y}_0(x) \quad \text{in } \Omega. \end{cases} \tag{2.2}$$

Here $\mathbf{g} = (g_1,g_2,\dots,g_n)^T$, $\mathbf{k}(0,t) = (k_1(0,t), k_2(0,t),\dots,k_n(0,t))^T$, $\mathbf{k}(1,t) = (k_1(1,t), k_2(1,t),\dots,k_n(1,t))^T$, and $\mathbf{y}_0 = (y_0^1, y_0^2,\dots,y_0^n)^T$, corresponding to distributed, boundary, and initial controls, respectively. The Hilbert space $W(0,T;\mathcal{V},\mathcal{V}')$ is called the solution space and is defined by

$$W(0,T;\mathcal{V},\mathcal{V}') = \{\phi \mid \phi \in L^2(0,T;\mathcal{V}),\ \phi' \in L^2(0,T;\mathcal{V}')\}.$$

Here ϕ' denotes the weak/distributed derivative of ϕ with respect to the time variable t. The inner product and norm in $W(0,T;\mathcal{V},\mathcal{V}')$ are defined respectively by

$$(\phi_1,\phi_2)_{W(0,T;\mathcal{V},\mathcal{V}')} = \int_0^T \{(\phi_1(t),\phi_2(t))_{\mathcal{V}} + (\phi_1'(t),\phi_2'(t))_{\mathcal{V}'}\}dt,$$

$$\|\phi\|_{W(0,T;\mathcal{V},\mathcal{V}')} = \left(\|\phi\|^2_{L^2(0,T;\mathcal{V})} + \|\phi'\|^2_{L^2(0,T;\mathcal{V}')}\right)^{\frac{1}{2}}.$$

The space $W(0,T;\mathcal{V},\mathcal{V}')$ is denoted by $\mathbf{W}(0,T)$ for brevity. Since $F_j : \mathbf{R} \to (-1,1)$, $j = 1,2,\ldots,n$, then $\mathbf{F} : \mathcal{H} \to \mathcal{H}$ and

$$\|\mathbf{F}(\boldsymbol{\psi})\|_{\mathcal{H}}^2 = \sum_{j=1}^{n} \int_0^l |F_j(\psi_j(x))|^2 dx \le nl \quad \forall \boldsymbol{\psi} \in \mathcal{H},$$

and by (2.1), $\mathbf{F}(\mathbf{y})$ satisfies the uniform Lipschitz continuity

$$\|\mathbf{F}(\boldsymbol{\phi}) - \mathbf{F}(\boldsymbol{\psi})\|_{\mathcal{H}} \le K\|\boldsymbol{\phi} - \boldsymbol{\psi}\|_{\mathcal{H}} \quad \forall \boldsymbol{\phi}, \boldsymbol{\psi} \in \mathcal{H}. \tag{2.3}$$

DEFINITION 2.1. [illegible] $\mathbf{y}$ [illegible] (2.2), $\mathbf{y} \in \mathbf{W}(0,T)$ [illegible]

$$\begin{cases} \langle \mathbf{y}', \upsilon\rangle_{\mathcal{V}',\mathcal{V}} + (\mathbf{D}\nabla\mathbf{y}, \nabla\upsilon)_{\mathcal{H}} = -(\mathbf{A}\mathbf{y}, \upsilon)_{\mathcal{H}} + (\mathbf{C}\mathbf{F}(\mathbf{y}), \upsilon)_{\mathcal{H}} + \langle \mathbf{g}, \upsilon\rangle_{\mathcal{V}',\mathcal{V}} \\ \qquad\qquad + (\mathbf{D}\mathbf{k}(0), \upsilon(0))_{\mathcal{V}^0} + (\mathbf{D}\mathbf{k}(1), \upsilon(1))_{\mathcal{V}^1} \\ \qquad\qquad \forall\ \upsilon \in \mathcal{V} \ \text{ in the sense of } \ \mathcal{D}'(0,T), \\ \mathbf{y}(0) = \mathbf{y_0} \in \mathcal{H}. \end{cases} \tag{2.4}$$

Here $\mathcal{D}'(0,T)$ denotes the space of distributions on $(0,T)$ (cf.[6]).

THEOREM 2.2. [illegible] $\mathbf{g} \in L^2(0,T;\mathcal{V}')$ $\mathbf{y_0} \in \mathcal{H}$, $\mathbf{k}(0) \in \mathcal{V}^0$ [illegible] $\mathbf{k}(1) \in \mathcal{V}^1$ [illegible] (2.2) [illegible] $\mathbf{y}$, $\mathbf{W}(0,T)$ [illegible] $C([0,T];\mathcal{H})$ [illegible]

$$\begin{aligned} &\|\mathbf{y}\|^2_{L^\infty(0,T;\mathcal{H})},\ \|\mathbf{y}\|^2_{\mathbf{W}(0,T)} \\ &\le C\big(1 + \|\mathbf{y_0}\|^2_{\mathcal{H}} + \|\mathbf{g}\|^2_{L^2(0,T;\mathcal{V}')} + \|\mathbf{k}(0)\|^2_{L^2(0,T)^n} + \|\mathbf{k}(1)\|^2_{L^2(0,T)^n}\big), \end{aligned} \tag{2.5}$$

[illegible] $C > 0$ [illegible] a_i c_{ij} [illegible] d_i [illegible] The proof of existence and uniqueness cites [17]. The regularity $\mathbf{y} \in C([0,T];\mathcal{H})$ is evident from $\mathbf{W}(0,T) \subset C([0,T];\mathcal{H})$.

3. Optimal control problem. The control system described by an HNN has a diffusion term in the form of

$$\begin{cases} \dfrac{\partial \mathbf{y}}{\partial t} - \mathbf{D}\boldsymbol{\Delta}\mathbf{y} = -\mathbf{A}\mathbf{y} + \mathbf{C}\mathbf{F}(\mathbf{y}) + \mathbf{u} \quad \text{in } Q, \\ \dfrac{\partial \mathbf{y}}{\partial \eta}(0,t) = \mathbf{v}(0,t), \quad \dfrac{\partial \mathbf{y}}{\partial \eta}(1,t) = \mathbf{v}(1,t), \\ \mathbf{y}(x,0) = \mathbf{w}(x) \ \text{ in } \Omega. \end{cases} \tag{3.1}$$

For any $\mathbf{u} \in L^2(0,T;\mathcal{V}')$, $\mathbf{v}(0) \in L^2(0,T)^n$, $\mathbf{v}(1) \in L^2(0,T)^n$, and $\mathbf{w} \in \mathcal{H}$, by virtue of Theorem 2.2, there exists a unique weak solution $\mathbf{y} = \mathbf{y}(\mathbf{c})$ of (3.1) in $\mathbf{W}(0,T)$. Hence a solution map $\mathbf{c} \to \mathbf{y}(\mathbf{c})$ can be defined from $\mathcal{C}$ into $\mathbf{W}(0,T)$, and $\mathbf{y}(\mathbf{c})$ will be called the state of control system (3.1).

The quadratic cost function associated with the control system (3.1) is given by (1.2) and can be written in compact form as

$$\begin{aligned} J(\mathbf{c}) = \|\mathbf{y}(\mathbf{c},T) - \mathbf{z}_d^T\|^2_{\mathcal{H}} + \|\mathbf{u}\|^2_{\mathcal{U}} + \|\mathbf{v}(0)\|^2_{\mathcal{V}^0} + \|\mathbf{v}(1)\|^2_{\mathcal{V}^1} + \|\mathbf{w}\|^2_{\mathcal{W}} \\ \forall \mathbf{c} = (\mathbf{u}, \mathbf{v}(0), \mathbf{v}(1), \mathbf{w}) \in \mathcal{C}, \end{aligned} \tag{3.2}$$

where $\mathbf{z}_d^T = (z_{1d}^T, z_{2d}^T, \ldots, z_{nd}^T)^t \in L^2(\Omega)^n$ are desired values.

3.1. Existence of optimal control. This subsection gives the existence theorem of the optimal control solution.

THEOREM 3.1. [illegible] $\mathcal{C}_{ad}$ [illegible] $\mathcal{C}$ [illegible] $\mathbf{c}^* = (\mathbf{u}^*, \mathbf{v}^*(0), \mathbf{v}^*(1), \mathbf{w}^*)$ [illegible] (3.1) [illegible] (3.2)

Set $J = \inf_{\mathbf{c}\in\mathcal{C}_{ad}} J(\mathbf{c})$. Since $\mathcal{C}_{ad}$ is nonempty, there is a sequence $\{\mathbf{c}_n\} = \{\mathbf{u}_n, \mathbf{v}_n(0), \mathbf{v}_n(1), \mathbf{w}_n\}$ in $\mathcal{C}_{ad}$ such that

$$\inf_{\mathbf{u}\in\mathcal{C}_{ad}} J(\mathbf{c}) = \lim_{n\to\infty} J(\mathbf{c}_n) = J.$$

Obviously, $\{J(\mathbf{c}_n)\}$ is bounded in $\mathbf{R}^+$. Since $\mathcal{U}_{ad}$ is bounded closed and convex, the subsequences $\{\mathbf{u}_m\}$, $\{\mathbf{v}_m(0,t)\}$, $\{\mathbf{v}_m(1,t)\}$, and $\{\mathbf{w}_m\}$ from $\{\mathbf{u}_n\}$, $\{\mathbf{v}_n(0,t)\}$, $\{\mathbf{v}_n(1,t)\}$, and $\{\mathbf{w}_n\}$ can be chosen, respectively, as

$$\begin{aligned}
&\mathbf{u}_m = \{u_{1m}, u_{2m}, \dots, u_{nm}\},\\
&\mathbf{v}_m(0,t) = \{v_{1m}(0,t), v_{2m}(0,t), \dots, v_{nm}(0,t)\},\\
&\mathbf{v}_m(1,t) = \{v_{1m}(1,t), v_{2m}(1,t), \dots, v_{nm}(1,t)\},\\
&\mathbf{w}_m = \{w_{1m}, w_{2m}, \dots, w_{nm}\}.
\end{aligned}$$

There exist $\mathbf{c}^* = \{\mathbf{u}^*, \mathbf{v}^*(0), \mathbf{v}^*(1), \mathbf{w}^*\} \in \mathcal{U}_{ad} \times \mathcal{V}^0_{ad} \times \mathcal{V}^1_{ad} \times \mathcal{W}_{ad}$ such that

$$\mathbf{u}_m \to \mathbf{u}^* \quad \text{weakly in } \mathcal{U} \text{ as } m \to \infty, \tag{3.3}$$

$$\mathbf{v}_m(0,t) \to \mathbf{v}^*(0,t) \quad \text{weakly in } \mathcal{V} \text{ as } m \to \infty, \tag{3.4}$$

$$\mathbf{v}_m(1,t) \to \mathbf{v}^*(1,t) \quad \text{weakly in } \mathcal{V} \text{ as } m \to \infty, \tag{3.5}$$

$$\mathbf{w}_m \to \mathbf{w}^* \quad \text{weakly in } \mathcal{W} \text{ as } m \to \infty. \tag{3.6}$$

For $\mathbf{y} = \mathbf{y}(\mathbf{c})$, the estimate (2.5) in Theorem 2.2 implies that

$$\|\mathbf{y}\|^2_{L^\infty(0,T;\mathcal{H})},\ \|\mathbf{y}\|^2_{\mathbf{W}(0,T)} \le C(1 + \|\mathbf{u}\|^2_{\mathcal{U}} + \|\mathbf{v}(0)\|^2_{\mathcal{V}^0} + \|\mathbf{v}(1)\|^2_{\mathcal{V}^1} + \|\mathbf{w}\|^2_{\mathcal{W}}). \tag{3.7}$$

The boundedness of $\mathcal{C}_{ad}$ and (3.7) show that $\{\mathbf{y}(\mathbf{c}_m)\}, \mathbf{c}_m = \{\mathbf{u}_m, \mathbf{v}_m(0), \mathbf{v}_m(1), \mathbf{w}_m\}$ is bounded in $\mathbf{W}(0,T)$. Hence a subsequence $\{\mathbf{y}(\mathbf{c}_{m_k})\}$ of $\{\mathbf{y}(\mathbf{c}_m)\}$ and a $\mathbf{z} \in \mathbf{W}(0,T)$ are extracted such that

$$\mathbf{y}(\mathbf{c}_{m_k}) \to \mathbf{z} \quad \text{weakly in } \mathbf{W}(0,T). \tag{3.8}$$

Denote $\mathbf{y}_{m_k} = \mathbf{y}(\mathbf{c}_{m_k})$ for simplicity. Since the embedding $\mathcal{V} \hookrightarrow \mathcal{H}$ is compact, we deduce from the Aubin–Lions–Temam compactness embedding theorem (cf. [18]) that

$$\mathbf{y}_{m_k} \to \mathbf{z} \quad \text{strongly in } L^2(0,T;\mathcal{H}). \tag{3.9}$$

The uniformly Lipschitz continuity (2.3) and (3.9) implies that

$$\mathbf{F}(\mathbf{y}_{m_k}) \to \mathbf{F}(\mathbf{z}) \quad \text{strongly in } L^2(0,T;\mathcal{H}). \tag{3.10}$$

Then (3.8) implies that

$$\begin{aligned}
\mathbf{y}'_{m_k} &\to \mathbf{z}' \quad \text{weakly in } L^2(0,T;\mathcal{V}'),\\
\nabla\mathbf{y}_{m_k} &\to \nabla\mathbf{z} \quad \text{weakly in } L^2(0,T;\mathcal{H}).
\end{aligned}$$

Hence by the definition of weak solutions we have

$$
\begin{aligned}
(3.11)\qquad & -\int_0^T \{\langle \mathbf{y}_{m_k}, \boldsymbol{\phi}'\rangle_{\mathcal{V}',\mathcal{V}} + (\mathbf{w}_{mk}, \boldsymbol{\phi})_{\mathcal{H}} + (\mathbf{D}\nabla \mathbf{y}_{m_k}, \nabla\boldsymbol{\phi})_{\mathcal{H}}\}dt \\
&= -\int_0^T \{(\mathbf{A}\mathbf{y}_{m_k}, \boldsymbol{\phi})_{\mathcal{H}} + (\mathbf{C}\mathbf{F}(\mathbf{y}_{m_k}), \boldsymbol{\phi})_{\mathcal{H}}\}dt + \int_0^T \langle \mathbf{u}_{mk}, \boldsymbol{\phi}\rangle_{\mathcal{V},\mathcal{V}'}dt \\
&\quad + (\mathbf{D}\mathbf{v}_{mk}(0), \boldsymbol{\phi}(0))_{\mathcal{V}^0} + (\mathbf{D}\mathbf{v}_{mk}(1), \boldsymbol{\phi}(1))_{\mathcal{V}^1} \quad \forall \boldsymbol{\phi} \in L^2(0,T;\mathcal{V}).
\end{aligned}
$$

Therefore, using (3.3)–(3.6) and (3.10), by taking $m_k \to \infty$ in (3.11), we obtain that

$$
\begin{aligned}
& -\int_0^T \{\langle \mathbf{z}, \boldsymbol{\phi}'\rangle_{\mathcal{V},\mathcal{V}'} + (\mathbf{w}^*, \boldsymbol{\phi})_{\mathcal{H}} + (\mathbf{D}\nabla \mathbf{z}, \nabla\boldsymbol{\phi})_{\mathcal{H}}\}dt \\
&= -\int_0^T \{(\mathbf{A}\mathbf{z}, \boldsymbol{\phi})_{\mathcal{H}} + (\mathbf{C}\mathbf{F}(\mathbf{z}), \boldsymbol{\phi})_{\mathcal{H}}\}dt + \int_0^T \langle \mathbf{u}^*, \boldsymbol{\phi}\rangle_{\mathcal{V},\mathcal{V}'}dt \\
&\quad + \int_0^T \mathbf{D}\mathbf{v}^*(0,t)\boldsymbol{\phi}(0) + \mathbf{D}\mathbf{v}^*(1,t)\boldsymbol{\phi}(1)dt,
\end{aligned}
$$

for all $\boldsymbol{\phi} \in L^2(0,T;\mathcal{V})$. Thus by standard manipulation we gain that the limit $\mathbf{z}$ satisfies

$$
\begin{aligned}
\langle \mathbf{z}, \boldsymbol{\phi}'\rangle_{\mathcal{V}',\mathcal{V}} + (\mathbf{D}\nabla \mathbf{z}, \nabla\boldsymbol{\phi})_{\mathcal{H}} &= -(\mathbf{w}^*, \boldsymbol{\phi})_{\mathcal{H}} + (\mathbf{A}\mathbf{z}, \boldsymbol{\phi})_{\mathcal{H}} + (\mathbf{C}\mathbf{F}(\mathbf{z}), \boldsymbol{\phi})_{\mathcal{H}} \\
&\quad + \langle \mathbf{u}^*, \boldsymbol{\phi}\rangle_{\mathcal{V},\mathcal{V}'} + (\mathbf{D}\mathbf{v}^*(0), \boldsymbol{\phi}(0))_{\mathcal{V}^0} + (\mathbf{D}\mathbf{v}^*(1), \boldsymbol{\phi}(1))_{\mathcal{V}^1}
\end{aligned}
$$

for all $\boldsymbol{\phi} \in \mathcal{V}$ in $\mathcal{D}'(0,T)$. Hence from the uniqueness of the weak solution for system (3.1), we have $\mathbf{z} = \mathbf{y}(\mathbf{c}^*)$. Then (3.8) and (3.9) imply that

$$
\begin{aligned}
\mathbf{y}(\mathbf{c}_{m_k}) &\to \mathbf{y}(\mathbf{c}^*) \quad \text{strongly in } L^2(Q)^n, \\
(3.12)\qquad \mathbf{y}(\mathbf{c}_{m_k}, T) &\to \mathbf{y}(\mathbf{c}^*, T) \quad \text{weakly in } \mathcal{H} = L^2(\Omega)^n.
\end{aligned}
$$

Since the norm $\|\cdot\|_{L^2(\Omega)^n}$ is lower semicontinuous in weak topology of $\mathcal{H} = L^2(\Omega)^n$, and using (3.12), we show that

$$
\liminf_{m_k\to\infty} \|\mathbf{y}(\mathbf{c}_{m_k}, T) - \mathbf{z}_d^T\|_{L^2(\Omega)^n} \geq \|\mathbf{y}(\mathbf{c}^*, T) - \mathbf{z}_d^T\|_{L^2(\Omega)^n}.
$$

On the other hand, the weak convergence (3.3)–(3.6) implies

$$
\begin{aligned}
&\liminf_{m_k\to\infty} (\mathbf{u}_{m_k}, \mathbf{u}_{m_k})_{L^2(Q)^n} \geq (\mathbf{u}^*, \mathbf{u}^*), \\
&\liminf_{m_k\to\infty} (\mathbf{v}_{m_k}(0,t), \mathbf{v}_{m_k}(0,t))_{L^2(0,T)^n} \geq (\mathbf{v}^*(0,t), \mathbf{v}^*(0,t)), \\
&\liminf_{m_k\to\infty} (\mathbf{v}_{m_k}(1,t), \mathbf{v}_{m_k}(1,t))_{L^2(0,T)^n} \geq (\mathbf{v}^*(1,t), \mathbf{v}^*(1,t)), \\
&\liminf_{m_k\to\infty} (\mathbf{w}_{m_k}, \mathbf{w}_{m_k})_{L^2(\Omega)^n} \geq (\mathbf{w}^*, \mathbf{w}^*).
\end{aligned}
$$

Since J is weakly lower semicontinuous, we have $J = \liminf_{m_k\to\infty} J(\mathbf{c}_{m_k}) \geq J(\mathbf{c}^*)$, and hence $J(\mathbf{c}^*) = \inf_{\mathbf{c}\in\mathcal{C}_{ad}} J(\mathbf{c})$. This proves that $\mathbf{c}^*(t) = \{\mathbf{u}^*, \mathbf{v}^*(0,t), \mathbf{v}^*(1,t), \mathbf{w}^*\}$ is an optimal control for cost (3.2), and completes the proof of Theorem 3.1.

3.2. Necessary optimality conditions. A first order necessary optimality condition for $\mathbf{c}^*$ is given by the variational inequality

$$
(3.13)\qquad J'(\mathbf{c}^*)(\mathbf{c} - \mathbf{c}^*) \geq 0 \quad \forall\ \mathbf{c} \in \mathcal{C}_{ad},
$$

where $J'(\mathbf{c}^*)$ denotes the Gâteaux derivative of $J(\mathbf{c})$ in (3.2) at $\mathbf{c}^*$.

PROPOSITION 3.2. [illegible] $\mathbf{y}_1$ [illegible] $\mathbf{y}_2$ [illegible] (3.1) [illegible] $\mathbf{c}_1 = \{\mathbf{u}_1, \mathbf{v}_1(0), \mathbf{v}_1(1), \mathbf{w}_1\}$ [illegible] $\mathbf{c}_2 = \{\mathbf{u}_2, \mathbf{v}_2(0), \mathbf{v}_2(1), \mathbf{w}_2\}$ [illegible] $t \in [0,T]$ $\mathbf{z} = \mathbf{y}_1 - \mathbf{y}_2$ [illegible]

$$
\begin{aligned}
&-\frac{1}{2}|\mathbf{z}|^2_{\mathcal{H}} + \frac{1}{2}|\mathbf{w}_1 - \mathbf{w}_2|^2_{\mathcal{H}} + \int_0^t (\mathbf{D}\nabla\mathbf{z}, \mathbf{z})|^2_{\mathcal{H}} dt \\
&= -\int_0^t (\mathbf{A}\mathbf{z}, \mathbf{z})_{\mathcal{H}} dt + \int_0^t (\mathbf{CF}(\mathbf{y}_1) - \mathbf{CF}(\mathbf{y}_2), \mathbf{z})_{\mathcal{H}} dt + \int_0^t ((\mathbf{u}_1 - \mathbf{u}_2), \mathbf{z})_{\mathcal{V}} dt \\
&\quad + \int_0^t (\mathbf{v}_1(0,t) - \mathbf{v}_2(0,t))\mathbf{z}(0) dt + \int_0^t (\mathbf{v}_1(1,t) - \mathbf{v}_2(1,t))\mathbf{z}(1) dt.
\end{aligned}
$$

PROPOSITION 3.3. [illegible] $\mathbf{c_0} \in \mathcal{C}$ [illegible]

$$
\lim_{\lambda \to 0} (\mathbf{y}(\mathbf{c} + \lambda \mathbf{c_0}) - \mathbf{y}(\mathbf{c})) = 0 \quad \text{strongly in} \quad C([0,T]; \mathcal{H}) \quad \text{and} \quad L^2(0,T; \mathcal{V}).
$$

The Gâteaux differentiability of solution map $\mathbf{c} \to \mathbf{y}(\mathbf{c})$ needs to assume $F_j \in C^1(\mathbf{R})$ and

$$
\exists K > 0 : \quad |F_j'(s)| \le K, \quad j = 1, 2, \ldots, n. \tag{3.14}
$$

Then for any $\mathbf{y} \in \mathcal{H}$, $\mathbf{F}$ is Gâteaux differentiable (cf. [17]).

THEOREM 3.4. [illegible] (3.14) [illegible] $\mathcal{C}$ [illegible] $\mathbf{W}(0,T)$ [illegible] $\mathbf{c} \to \mathbf{y}(\mathbf{c})$ [illegible] [illegible] $\hat{a}$ [illegible] $\mathbf{c}^* = \{\mathbf{u}^*, \mathbf{v}^*(0,t), \mathbf{v}^*(1,t), \mathbf{w}^*\}$ [illegible] [illegible] $\hat{a}$ [illegible] $\mathbf{z} = D\mathbf{y}(\mathbf{c}^*)(\mathbf{c} - \mathbf{c}^*) \in \mathbf{W}(0,T)$ [illegible] [illegible]

$$
\begin{cases}
\dfrac{\partial \mathbf{z}}{\partial t} - \mathbf{D}\boldsymbol{\Delta}\mathbf{z} = -\mathbf{A}\mathbf{z} + \mathbf{C}\partial_{\mathbf{y}}\mathbf{F}(\mathbf{y}(\mathbf{c}^*))\mathbf{z} + (\mathbf{u} - \mathbf{u}^*) \quad \text{in } Q, \\
\dfrac{\partial \mathbf{z}(0,t)}{\partial \eta} = \mathbf{v}(0,t) - \mathbf{v}^*(0,t), \quad \dfrac{\partial \mathbf{z}(1,t)}{\partial \eta} = \mathbf{v}(1,t) - \mathbf{v}^*(1,t), \\
\mathbf{z}(x,0) = \mathbf{w}(x) - \mathbf{w}^*(x) \quad \text{in } \Omega.
\end{cases} \tag{3.15}
$$

By Theorem 3.4, cost $J(\mathbf{u})$ is weakly Gâteaux differentiable, and optimality condition (3.13) is rewritten by

$$
\begin{aligned}
&J'(\mathbf{c}^*)(\mathbf{c} - \mathbf{c}^*) = (\mathbf{y}(\mathbf{c}^*, T) - \mathbf{z}_d^T, \mathbf{z}(T))_{\mathcal{H}} + (\mathbf{u}^*, \mathbf{u} - \mathbf{u}^*)_{\mathcal{U}} \\
&+ (\mathbf{v}^*(0), \mathbf{v}(0) - \mathbf{v}^*(0))_{\mathcal{V}^0} + (\mathbf{v}^*(1), \mathbf{v}(1) - \mathbf{v}^*(1))_{\mathcal{V}^1} + (\mathbf{w}^*, \mathbf{w} - \mathbf{w}^*)_{\mathcal{W}} \ge 0
\end{aligned}
$$

for all $\mathbf{c} = \{\mathbf{u}, \mathbf{v}(0), \mathbf{v}(1), \mathbf{w}\} \in \mathcal{C}_{ad}$ and $\mathbf{z}_d^T \in \mathcal{H} = L^2(\Omega)^n$.

THEOREM 3.5. [illegible] $\mathbf{c}^* = \{\mathbf{u}^*, \mathbf{v}^*(0), \mathbf{v}^*(1), \mathbf{w}^*\} \in \mathcal{C}_{ad}$ [illegible] (3.2) [illegible]

$$
\begin{cases}
\dfrac{\partial \mathbf{y}}{\partial t} - \mathbf{D}\boldsymbol{\Delta}\mathbf{y} = -\mathbf{A}\mathbf{y} + \mathbf{CF}(\mathbf{y}) + \mathbf{u}^* \quad \text{[illegible]} \quad Q, \\
\dfrac{\partial \mathbf{y}}{\partial \eta}(0,t) = \mathbf{v}^*(0,t) \ \dfrac{\partial \mathbf{y}}{\partial \eta}(1,t) = \mathbf{v}^*(1,t) \quad \text{[illegible]} \quad (0,T), \\
\mathbf{y}(x,0) = \mathbf{w}^*(x) \quad \text{[illegible]} \quad \Omega,
\end{cases}
$$

$$\begin{cases} -\dfrac{\partial \mathbf{p}(\mathbf{c}^*)}{\partial t} - \mathbf{D}\boldsymbol{\Delta}\mathbf{p}(\mathbf{c}^*) = -\mathbf{A}\mathbf{p}(\mathbf{c}^*) + \partial_{\mathbf{y}}\mathbf{F}(\mathbf{y}(\mathbf{c}^*))^*\mathbf{C}^*\mathbf{p}(\mathbf{c}^*) & \text{in } Q, \\ \dfrac{\partial \mathbf{p}(\mathbf{c}^*)}{\partial \eta}(0,t) = \dfrac{\partial \mathbf{p}(\mathbf{c}^*)}{\partial \eta}(1,t) = 0 & \text{in } (0,T), \\ \mathbf{p}(\mathbf{c}^*,T) = \mathbf{y}(\mathbf{c}^*,T) - \mathbf{z}_d^1 & \text{in } \Omega, \end{cases}$$

$$\begin{aligned} &\int_0^T\int_0^l \Big(\mathbf{D}^{-1}\mathbf{p}(\mathbf{c}^*(x,t)) + \mathbf{u}^*(x,t)\Big)(\mathbf{u}(x,t) - \mathbf{u}^*(x,t))dxdt \\ &+ \int_0^T \Big(\mathbf{D}^{-1}\mathbf{p}(\mathbf{c}^*(0,t)) + \mathbf{v}^*(0,t)\Big)(\mathbf{v}(0,t) - \mathbf{v}^*(0,t))dt \\ &+ \int_0^T \Big(\mathbf{D}^{-1}\mathbf{p}(\mathbf{c}^*(1,t)) + \mathbf{v}^*(1,t)\Big)(\mathbf{v}(1,t) - \mathbf{v}^*(1,t))dt \\ &+ \int_0^l \Big(\mathbf{D}^{-1}\mathbf{p}(\mathbf{c}^*(x,0)) + \mathbf{w}^*(x)\Big)(\mathbf{w}(x) - \mathbf{w}^*(x))dx \geq 0 \\ &\qquad\qquad \forall \mathbf{c} = (\mathbf{u}, \mathbf{v}(0), \mathbf{v}(1), \mathbf{w}) \in \mathcal{C}_{ad}. \end{aligned} \tag{3.16}$$

4. Numerical approach based on finite element methods. Let $0 = x_0 < x_1 < \cdots < x_N < x_{N+1} = l$ be a partition of interval $[0, l]$ into subinterval $I_e = [x_{e-1}, x_e]$ of length $h^e = x_e - x_{e-1}$ for $e = 1, 2, \ldots, N+1$. Let the following hold:

$$V_h = \left\{ b_i^e \;\middle|\; b_i^e \text{ is a quadratic function on } I_e \text{ and continuous on } [0,l], i = 1,2,3 \right\}.$$

Then $V_h \subset V$. The basis functions $b_i^e \in V_h$ are defined by quadratic interpolation functions

$$\begin{cases} b_1^e(x) = \left(1 - \dfrac{x - x_e}{h^e}\right)\left(1 - \dfrac{2(x - x_e)}{h^e}\right), \\ b_2^e(x) = \dfrac{4(x - x_e)}{h^e}\left(1 - \dfrac{x - x_e}{h^e}\right), \\ b_3^e(x) = -\dfrac{(x - x_e)}{h^e}\left(1 - \dfrac{2(x - x_e)}{h^e}\right). \end{cases}$$

Here x_e is the endpoint of the subinterval, and h^e is the length of the eth subinterval. Using quadratic basis functions, we construct the approximate solutions $y_{1h}, y_{2h}, \ldots, y_{nh}$ of $y_1, y_2, \ldots, y_n$ in the form of

$$\begin{cases} y_{1h}(t,x) = \displaystyle\sum_{e=1}^{N} y_{1h}^e(t,x) = \sum_{e=1}^{N}\sum_{i=1}^{3} \xi_{1i}^e(t) b_i^e(x), \\ y_{2h}(t,x) = \displaystyle\sum_{e=1}^{N} y_{2h}^e(t,x) = \sum_{e=1}^{N}\sum_{i=1}^{3} \xi_{2i}^e(t) b_i^e(x), \\ \qquad\qquad \vdots \\ y_{nh}(t,x) = \displaystyle\sum_{e=1}^{N} y_{nh}^e(t,x) = \sum_{e=1}^{N}\sum_{i=1}^{3} \xi_{ni}^e(t) b_i^e(x). \end{cases}$$

For each element, we set

$$\begin{aligned}
&\Psi^e = [(\psi^e_{ij})] = (b^e_i, b^e_j)^{j=1,2,3}_{i=1,2,3} \in M_{3\times 3}(\mathbf{R}),\\
&\Phi^e = [(\phi^e_{ij})] = (\bigtriangledown b^e_i, \bigtriangledown b^e_j)^{j=1,2,3}_{i=1,2,3} \in M_{3\times 3}(\mathbf{R}),\\
&\Xi^e_j(t) = [(\xi^e_j)] = [\xi^e_{j1}(t), \xi^e_{j2}(t), \xi^e_{j3}(t)]^T \in M_{3\times 1}(\mathbf{R}),\\
&F^e_j(t) = [(F^e_j)] = \begin{bmatrix} \left(F_j\left(\sum_{i=1}^{3}\xi^e_i(t)b^e_i(x)\right), b^e_1\right)\\ \left(F_j\left(\sum_{i=1}^{3}\xi^e_i(t)b^e_i(x)\right), b^e_2\right)\\ \left(F_j\left(\sum_{i=1}^{3}\xi^e_i(t)b^e_i(x)\right), b^e_3\right)\end{bmatrix} \in M_{3\times 1}(\mathbf{R}),\\
&U^e_j(t) = [(u^e_j)] = [(u_j(t), b^e_1), (u_j(t), b^e_2), (u_j(t), b^e_3)]^T \in M_{3\times 1}(\mathbf{R}),\\
&V^e_{0j}(t) = [(v^e_{0j})] = [(v_j(0,t), b^e_1), (v_j(0,t), b^e_2), (v_j(0,t), b^e_3)]^T \in M_{3\times 1}(\mathbf{R}),\\
&V^e_{1j}(t) = [(u^e_{1j})] = [(v_j(1,t), b^e_1), (v_j(1,t), b^e_2), (v_j(1,t), b^e_3)]^T \in M_{3\times 1}(\mathbf{R}),\\
&W^e_j = [(w^e_j)] = [(w_j, b^e_1), (w_j, b^e_2), (w_j, b^e_3)]^T \in M_{3\times 1}(\mathbf{R})
\end{aligned}$$

for $e = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, n$. Referring to [21], Gauss–Legendre quadrature for six partitions is used to calculate numerical integration of the nonlinear term. $F^e_1(t)$, $F^e_2(t), \ldots, F^e_n(t)$ are calculated at $p^e_1, p^e_2, \ldots, p^e_6$ with weights $w^e_1, w^e_2, \ldots, w^e_6$ on $I_e = (x_e, x_{e+1})$. Then the approximations $\hat{F}^e_j(t)$ of $F^e_j(t)$ can be calculated as follows:

$$\hat{F}^e_j(t) = \begin{bmatrix} \hat{F}^e_{1j}\\ \hat{F}^e_{2j}\\ \hat{F}^e_{3j}\end{bmatrix} = \begin{bmatrix} \sum_{m=1}^{6} w^e_m F_j\left(\sum_{i=1}^{3}\xi^e_i\psi^e_i(p^e_m)\right)\psi^e_1(p^e_m)\\ \sum_{m=1}^{6} w^e_m F_j\left(\sum_{i=1}^{3}\xi^e_i\psi^e_i(p^e_m)\right)\psi^e_2(p^e_m)\\ \sum_{m=1}^{6} w^e_m F_j\left(\sum_{i=1}^{3}\xi^e_i\psi^e_i(p^e_m)\right)\psi^e_3(p^e_m)\end{bmatrix}$$

for $j = 1, 2, \ldots, n$. Therefore, (2.4) implies that

$$\begin{cases}
\Psi^e\Xi^{e\prime}_1(t) + d_1\Phi^e\Xi^e_1(t) = -a_1\Psi^e\Xi^e_1(t) + \sum_{j=1}^{n} c_{1j}F^e_j(t) + U^e_1(t) + V^e_{01}(t) + V^e_{11}(t),\\
\Psi^e\Xi^{e\prime}_2(t) + d_2\Phi^e\Xi^e_2(t) = -a_2\Psi^e\Xi^e_2(t) + \sum_{j=1}^{n} c_{2j}F^e_j(t) + U^e_2(t) + V^e_{02}(t) + V^e_{12}(t),\\
\qquad\qquad\qquad\qquad\vdots\\
\Psi^e\Xi^{e\prime}_n(t) + d_n\Phi^e\Xi^e_n(t) = -a_n\Psi^e\Xi^e_n(t) + \sum_{j=1}^{n} c_{nj}F^e_j(t) + U^e_n(t) + V^e_{0n}(t) + V^e_{1n}(t)
\end{cases} \tag{4.1}$$

with $\Psi^e\Xi^e_1(0) = W^e_1$, $\Psi^e\Xi^e_2(0) = W^e_2, \ldots, \Psi^e\Xi^e_n(0) = W^e_n$ for $e = 1, 2, \ldots, N$.

Taking into account $\xi^i_{j3} = \xi^{i+1}_{j1}$ for $j = 1, 2, \ldots, n$ and $i = 1, 2, \ldots, N$, the matrices and vectors globally are in the form of

$$\Psi_{2N+1} = \begin{bmatrix} \psi^1_{11} & \psi^1_{12} & \psi^1_{13} & & & & & & \\ \psi^1_{21} & \psi^1_{22} & \psi^1_{23} & & & & & & \\ \psi^1_{31} & \psi^1_{32} & \psi^1_{33}+\psi^2_{11} & \psi^2_{12} & \psi^2_{13} & & 0 & & \\ & & \psi^2_{21} & \psi^2_{22} & \psi^2_{23} & & & & \\ & & \psi^2_{31} & \psi^2_{32} & \psi^2_{33}+\psi^3_{11} & & & & \\ & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \\ & & 0 & & & & \psi^{N-1}_{33}+\psi^N_{11} & \psi^N_{12} & \psi^N_{13} \\ & & & & & & \psi^N_{21} & \psi^N_{22} & \psi^N_{23} \\ & & & & & & \psi^N_{31} & \psi^N_{32} & \psi^N_{33} \end{bmatrix},$$

and Φ_{2N+1} has the same structure instead of ψ^e_{ij} with ϕ^e_{ij}. Denote that

$$\Xi^i_{2N+1} = \begin{bmatrix} \xi_{i1} \\ \xi_{i2} \\ \xi_{i3} \\ \xi_{i4} \\ \xi_{i5} \\ \vdots \\ \xi_{i,2N-1} \\ \xi_{i,2N} \\ \xi_{i,2N+1} \end{bmatrix} = \begin{bmatrix} \xi^1_{i1} \\ \xi^1_{i2} \\ \xi^1_{i3}(=\xi^2_{i1}) \\ \xi^2_{i2} \\ \xi^2_{i3}(=\xi^3_{i1}) \\ \vdots \\ \xi^{N-1}_{i3}(=\xi^N_{i1}) \\ \xi^N_{i2} \\ \xi^N_{i3} \end{bmatrix}, \quad \hat{F}^i_{2N+1} = \begin{bmatrix} \hat{F}_{i1} \\ \hat{F}_{i2} \\ \hat{F}_{i3} \\ \hat{F}_{i4} \\ \hat{F}_{i5} \\ \vdots \\ \hat{F}_{i,2N-1} \\ \hat{F}_{i,2N} \\ \hat{F}_{i,2N+1} \end{bmatrix} = \begin{bmatrix} \hat{F}^1_{i1} \\ \hat{F}^1_{i2} \\ \hat{F}^1_{i3}+\hat{F}^2_{i1} \\ \hat{F}^2_{i2} \\ \hat{F}^2_{i3}+\hat{F}^3_{i1} \\ \vdots \\ \hat{F}^{N-1}_{i3}+\hat{F}^N_{i1} \\ \hat{F}^N_{i2} \\ \hat{F}^N_{i3} \end{bmatrix},$$

$$U^i_{2N+1} = \begin{bmatrix} U_{i1} \\ U_{i2} \\ U_{i3} \\ U_{i4} \\ U_{i5} \\ \vdots \\ U_{i,2N-1} \\ U_{i,2N} \\ U_{i,2N+1} \end{bmatrix} = \begin{bmatrix} u^1_{i1} \\ u^1_{i2} \\ u^1_{i3}+u^2_{i1} \\ u^2_{i2} \\ u^2_{i3}+u^3_{i1} \\ \vdots \\ u^{N-1}_{i3}+u^N_{i1} \\ u^N_{i2} \\ u^N_{i3} \end{bmatrix},$$

$$V^0_{i,2N+1} = \begin{bmatrix} V^0_{i1} \\ V^0_{i2} \\ V^0_{i3} \\ V^0_{i4} \\ V^0_{i5} \\ \vdots \\ V^0_{i,2N-1} \\ V^0_{i,2N} \\ V^0_{i,2N+1} \end{bmatrix} = \begin{bmatrix} v^{01}_{i1} \\ v^{01}_{i2} \\ v^{01}_{i3}+v^{02}_{i1} \\ v^{02}_{i2} \\ v^{02}_{i3}+v^{03}_{i1} \\ \vdots \\ v^{0,N-1}_{i3}+v^{0,N}_{i1} \\ v^{0,N}_{i2} \\ v^{0,N}_{i3} \end{bmatrix},$$

$$V^1_{i,2N+1}=\begin{bmatrix} V^1_{i1}\\ V^1_{i2}\\ V^1_{i3}\\ V^1_{i4}\\ V^1_{i5}\\ \vdots\\ V^1_{i,2N-1}\\ V^1_{i,2N}\\ V^1_{i,2N+1}\end{bmatrix}=\begin{bmatrix} v^{11}_{i1}\\ v^{11}_{i2}\\ v^{11}_{i3}+v^{12}_{i1}\\ v^{12}_{i2}\\ v^{12}_{i3}+v^{13}_{i1}\\ \vdots\\ v^{1,N-1}_{i3}+v^{1N}_{i1}\\ v^{1,N}_{i2}\\ v^{1,N}_{i3}\end{bmatrix},$$

$$W^i_{2N+1}=\begin{bmatrix} W_{i1}\\ W_{i2}\\ W_{i3}\\ W_{i4}\\ W_{i5}\\ \vdots\\ W_{i,2N-1}\\ W_{i,2N}\\ W_{i,2N+1}\end{bmatrix}=\begin{bmatrix} w^1_{i1}\\ w^1_{i2}\\ w^1_{i3}+w^2_{i1}\\ w^2_{i2}\\ w^2_{i3}+w^3_{i1}\\ \vdots\\ w^{N-1}_{i3}+w^N_{i1}\\ w^N_{i2}\\ w^N_{i3}\end{bmatrix}$$

for $i=1,2,\dots,n$. Hence by (4.1), the whole system on $(0,l)$ can be rewritten as

$$\text{(4.2)}\quad\begin{cases}\Psi_{2N+1}\Xi^{1\prime}_{2N+1}+d_1\Phi_{2N+1}\Xi^1_{2N+1}\\ \qquad=-a_1\Psi_{2N+1}\Xi^1_{2N+1}+\sum\limits_{j=1}^{n}c_{1j}\hat F^j_{2N+1}+U^1_{2N+1}+V^0_{1,2N+1}+V^1_{1,2N+1},\\ \Psi_{2N+1}\Xi^{2\prime}_{2N+1}+d_2\Phi_{2N+1}\Xi^2_{2N+1}\\ \qquad=-a_2\Psi_{2N+1}\Xi^2_{2N+1}+\sum\limits_{j=1}^{n}c_{2j}\hat F^j_{2N+1}+U^2_{2N+1}+V^0_{2,2N+1}+V^1_{2,2N+1},\\ \qquad\qquad\qquad\qquad\vdots\\ \Psi_{2N+1}\Xi^{n\prime}_{2N+1}+d_n\Phi_{2N+1}\Xi^n_{2N+1}\\ \qquad=-a_n\Psi_{2N+1}\Xi^n_{2N+1}+\sum\limits_{j=1}^{n}c_{nj}\hat F^j_{2N+1}+U^n_{2N+1}+V^0_{n,2N+1}+V^1_{n,2N+1}.\end{cases}$$

Using the notations of matrix and vectors, then

$$\Psi_{n(2N+1)}=\operatorname{diag}\{\Psi_{2N+1},\Psi_{2N+1},\dots,\Psi_{2N+1}\},$$
$$\Phi_{n(2N+1)}=\operatorname{diag}\{\Phi_{2N+1},\Phi_{2N+1},\dots,\Phi_{2N+1}\},$$
$$\Xi_{n(2N+1)}=\begin{bmatrix}\Xi^1_{2N+1}\\ \vdots\\ \Xi^n_{2N+1}\end{bmatrix},\qquad \hat F_{n(2N+1)}=\begin{bmatrix}\hat F^1_{2N+1}\\ \vdots\\ \hat F^n_{2N+1}\end{bmatrix},$$
$$U_{n(2N+1)}=\begin{bmatrix}U_{1,2N+1}\\ \vdots\\ U_{n,2N+1}\end{bmatrix},\qquad V^0_{n(2N+1)}=\begin{bmatrix}V^0_{1,2N+1}\\ \vdots\\ V^0_{n,,2N+1}\end{bmatrix},$$
$$V^1_{n(2N+1)}=\begin{bmatrix}V^1_{1,2N+1}\\ \vdots\\ V^1_{n,2N+1}\end{bmatrix},\quad W_{n(2N+1)}=\begin{bmatrix}W_{1,2N+1}\\ \vdots\\ W_{n,2N+1}\end{bmatrix}.$$

Let $D^i_{2N+1} = \operatorname{diag}\{d_i, d_i, \ldots, d_i\}$, $A^i_{2N+1} = \operatorname{diag}\{a_i, a_i, \ldots, a_i\}$ for $i = 1, 2, \ldots, n$; then

$$D_{n(2N+1)} = \operatorname{diag}\Big\{D^1_{2N+1}, D^2_{2N+1}, \ldots, D^n_{2N+1}\Big\},$$

$$A_{n(2N+1)} = \operatorname{diag}\Big\{A^1_{2N+1}, A^2_{2N+1}, \cdots, A^n_{2N+1}\Big\},$$

$$C_{n(2N+1)} = \begin{bmatrix} c_{11} & \cdots & c_{11} & & c_{1n} & \cdots & c_{1n} \\ \vdots & \cdots & \vdots & & \vdots & \cdots & \vdots \\ c_{11} & \cdots & c_{11} & & c_{1n} & \cdots & c_{1n} \\ \vdots & \cdots & \vdots & \ddots & \vdots & \cdots & \vdots \\ c_{n1} & \cdots & c_{n1} & & c_{nn} & \cdots & c_{nn} \\ \vdots & \cdots & \vdots & & \vdots & \cdots & \vdots \\ c_{n1} & \cdots & c_{n1} & & c_{nn} & \cdots & c_{nn} \end{bmatrix}.$$

Since $\Psi_{n(2N+1)}$ is invertible, one can convert (4.2) into a set of $n(2N+1)$-dimensional vector equations

$$\begin{aligned} (4.3)\quad & \Xi'_{n(2N+1)} + D_{n(2N+1)}\Psi^{-1}_{n(2N+1)}\Phi_{n(2N+1)}\Xi_{n(2N+1)} + A_{n(2N+1)}\Xi_{n(2N+1)} \\ & = C_{n(2N+1)}\Psi^{-1}_{n(2N+1)}\hat{F}_{n(2N+1)} + \Psi^{-1}_{n(2N+1)}(U_{n(2N+1)} + V^0_{n(2N+1)} + V^1_{n(2N+1)}). \end{aligned}$$

Here coefficient matrices $\Psi_{n(2N+1)}$ and $\Phi_{n(2N+1)}$ are given by (assume $h^e = h$),

$$\frac{h}{30}\begin{bmatrix} 16 & 2 & 0 & 0 & \cdots & 0 & 0 & & & & & & & \\ 2 & 8 & 2 & -1 & \cdots & 0 & 0 & & & & & & & \\ 0 & 2 & 16 & 2 & \cdots & 0 & 0 & & & 0 & & & & \\ 0 & -1 & 2 & 8 & \cdots & 0 & 0 & & & & & & & \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & & & & & & & \\ 0 & 0 & 0 & 0 & \cdots & 8 & 2 & & & & & & & \\ 0 & 0 & 0 & 0 & \cdots & 2 & 16 & & & & & & & \\ & & & & & & & \ddots & & & & & & \\ & & & & & & & & 16 & 2 & 0 & 0 & \cdots & 0 & 0 \\ & & & & & & & & 2 & 8 & 2 & -1 & \cdots & 0 & 0 \\ & & & & & & & & 0 & 2 & 16 & 2 & \cdots & 0 & 0 \\ & & & & 0 & & & & 0 & -1 & 2 & 8 & \cdots & 0 & 0 \\ & & & & & & & & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ & & & & & & & & 0 & 0 & 0 & 0 & \cdots & 8 & 2 \\ & & & & & & & & 0 & 0 & 0 & 0 & \cdots & 2 & 16 \end{bmatrix}$$

and

$$\frac{1}{3h}\begin{bmatrix}
16 & -8 & 0 & 0 & \cdots & 0 & 0 & & & & & & & \\
-8 & 14 & -8 & 1 & \cdots & 0 & 0 & & & & & & & \\
0 & -8 & 16 & -8 & \cdots & 0 & 0 & & & & & & & \\
0 & 1 & -8 & 14 & \cdots & 0 & 0 & & & 0 & & & & \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & & & & & & & \\
0 & 0 & 0 & 0 & \cdots & 14 & -8 & & & & & & & \\
0 & 0 & 0 & 0 & \cdots & -8 & 16 & & & & & & & \\
 & & & & & & & \ddots & & & & & & \\
 & & & & & & & & 16 & -8 & 0 & 0 & \cdots & 0 & 0 \\
 & & & & & & & & -8 & 14 & -8 & 1 & \cdots & 0 & 0 \\
 & & & & & 0 & & & 0 & -8 & 16 & -8 & \cdots & 0 & 0 \\
 & & & & & & & & 0 & 1 & -8 & 14 & \cdots & 0 & 0 \\
 & & & & & & & & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
 & & & & & & & & 0 & 0 & 0 & 0 & \cdots & 14 & -8 \\
 & & & & & & & & 0 & 0 & 0 & 0 & \cdots & -8 & 16
\end{bmatrix}.$$

We follow [21] to solve (4.3) by a fourth order Runge–Kutta method. Therefore, the approximate solution $y_{1h}, y_{2h}, \ldots, y_{nh}$ can be obtained on $(0,l)\times[0,T]$ numerically.

5. Numerical approach for optimal control.

5.1. Systems discretization. Let $\mathbf{c}_h = (\mathbf{c}_{1h}, \mathbf{c}_{2h}, \ldots, \mathbf{c}_{nh})$, $\mathbf{c}_{ih} = \{u_{ih}, v_{ih}(0), v_{ih}(1), w_{ih}\}$. The minimization problem for finite element approximation is formulated as follows: minimize the approximate cost functions

$$\begin{aligned}(5.1)\quad J(\mathbf{c}_h) &= \sum_{i=1}^{n}\int_0^l (y_{ih}(T)-z_{id}^T)^2dx + \sum_{i=1}^{n}\int_0^T (u_{ih}-u_i^d)^2dt \\ &+\sum_{i=1}^{n}\int_0^T (v_{ih}(0)-v_{0i}^d)^2dt + \sum_{i=1}^{n}\int_0^T (v_{ih}(1)-v_{1i}^d)^2dt + \sum_{i=1}^{n}\int_0^l (w_{ih}-w_i^d)dx\end{aligned}$$

with $y_{ih}\equiv y_i(\mathbf{c}_h)\in V_h$ satisfying

$$\begin{aligned}(5.2)\quad &\int_0^l \frac{dy_{ih}}{dt}\phi_h dx + d_i\int_0^l \nabla y_{ih}\nabla\phi_h dx = -\int_0^l a_i y_{ih}\phi_h dx + \sum_{j=1}^{n}\int_0^l F_j(y_{jh})\phi_h dx \\ &+\int_0^T\int_0^l u_{ih}\phi_h dxdt + \int_0^T v_{ih}(0)\phi_h dt + \int_0^T v_{ih}(1)\phi_h dt \qquad \forall\phi_h\in V_h.\end{aligned}$$

THEOREM 5.1. [illegible] 3.1 [illegible] [24] [illegible] (5.1)–(5.2)

Note that the Gâteaux derivative for the discrete solution y_{ih} at any direction ϕ_h is denoted by $y'_{ih}\phi_h$, satisfying (5.2) for all $\phi_h\in V_h$. For quadratic cost, let $p_{ih}\equiv p_i(\mathbf{c}_h)$, so that

$$\begin{aligned}J'(\mathbf{c}_h)\phi_h &= 2\sum_{i=1}^{n}\int_0^l p_i(\mathbf{c}_h)(x,T)\phi_h(x,T)dx + 2\sum_{i=1}^{n}\int_0^T\int_0^l u_{ih}(x,t)\phi_h(x,t)dxdt \\ &+2\sum_{i=1}^{n}\int_0^T v_{ih}(0,t)\phi_h(0,t)dt + 2\sum_{i=1}^{n}\int_0^T v_{ih}(1,t)\phi_h(1,t)dt + 2\sum_{i=1}^{n}\int_0^l w_{ih}(x)\phi_h(x,0)dx.\end{aligned}$$

Then necessary optimality condition (3.16) for optimal control pairing $u^*_{ih} \in U^i_{ad}$, $v^*_{ih}(0,t) \in V^0_{i,ad}, v^*_{ih}(1,t) \in V^1_{i,ad}, w^*_{ih} \in W^i_{ad}$ can be rewritten as

$$\begin{aligned}
&\sum_{i=1}^{n} \int_0^T \int_0^l (p_i(\mathbf{c}^*_h(x,t)) + u^*_{ih}(x,t))(u_{ih}(x,t) - u^*_{ih}(x,t))dxdt \\
&+ \sum_{i=1}^{n} \int_0^T (p_i(\mathbf{c}^*_h(0,t)) + v^*_{ih}(0,t))(v_{ih}(0,t) - v^*_{ih}(0,t))dt \\
&+ \sum_{i=1}^{n} \int_0^T (p_i(\mathbf{c}^*_h(1,t)) + v^*_{ih}(1,t))(v_{ih}(1,t) - v^*_{ih}(1,t))dt \\
&+ \sum_{i=1}^{n} \int_0^l (p_i(\mathbf{c}^*_h(x,0)) + w^*_{ih}(x))(w_{ih}(x) - w^*_{ih}(x))dx \geq 0,
\end{aligned}$$

for all $u_{ih} \in U^i_{ad}, v_{ih}(0) \in V^0_{i,ad}, v_{ih}(1) \in V^1_{i,ad}, w_{ih} \in W^i_{ad}$, where $\mathcal{U}^i_{ad}, V^0_{i,ad}, V^1_{i,ad}, W^i_{ad}$ are the admissible set of $\mathcal{U}^i, V^0_i, V^1_i, W_i$ for $i = 1, 2, \ldots, N$.

THEOREM 5.2. $\{\mathbf{c}^m_{ih}\} = \{u^m_{ih}, v^m_{ih}(0), v^m_{ih}(1), w^m_{ih}\}$ (5.1)–(5.2) $\mathbf{c}^m_h = \{\mathbf{c}^m_{1h}, \mathbf{c}^m_{2h}, \ldots, \mathbf{c}^m_{nh}\}$ $\mathbf{c}^{m_k}_h = \{\mathbf{c}^{m_k}_{1h}, \mathbf{c}^{m_k}_{2h}, \ldots, \mathbf{c}^{m_k}_{nh}\}$ $\{\mathbf{c}^m_i\}$ $\{\mathbf{c}^{m_k}_i\}$ $L^2(Q)^n \times L^2(0,T)^n \times L^2(0,T)^n \times L^2(0,l)^n$ (5.1)

The proof can be found in [24] and elsewhere.

5.2. Updated conjugate gradient method for minimization. Let m be the iteration times, and $J^m_{ih} = J_{ih}(\mathbf{c}^m_{ih})$; then

$$\mathbf{c}^{m+1}_{ih}(t) = \mathbf{c}^m_{ih}(t) - \beta^m_i \mathbf{P}^m_i(t) \quad \text{for} \quad i = 1, 2, \ldots, N \quad \text{and} \quad m = 1, 2, \ldots,$$

where $\beta^m_i(t)$ are searching step sizes in going from iteration m to iteration $m+1$, and $\mathbf{P}^m_i(t)$ are the directions of descent given by

$$\mathbf{P}^m_i(t) = J'^m_{ih}(t) + \gamma^m_i \mathbf{P}^{m-1}_i(t), \tag{5.3}$$

which are a conjugation of the gradient directions $J'^m_{ih}(t)$ at iteration m and the directions of descent $\mathbf{P}^{m-1}_i(t)$ at iteration $m-1$. The conjugate coefficients are determined from

$$\gamma^m_i = \frac{\varepsilon_1 \int_0^T (J'^m_{ih})^2 dt}{\varepsilon_2 \int_0^T (J'^{(m-1)}_{ih})^2 dt}, \quad \text{with} \quad \gamma^0_i = 0 \quad \text{and} \quad i = 1, \ldots, N. \tag{5.4}$$

Note that we can adjust $\varepsilon_1, \varepsilon_2$ to get proper γ^m_i. The stopping criterion is $J(\mathbf{c}^{m+1}_{ih}(t)) < \varepsilon$, where ε is a small specified number. The convergence of the above iteration procedure in minimizing the functional J_{ih} is guaranteed (cf. [14]).

5.3. The nonlinear minimization problem (5.1)–(5.2) can be solved by the nonlinear conjugate gradient method (CGM) consisting of (5.3) and (5.4). The updated CGM is a kind of efficient modification of the usual CGM.

5.3. Computation procedure. Suppose that $\mathbf{c}^k_{ih}(t)$ is available at iteration k, and $\mathbf{c}^k_{ih} = \{c^k_{ih}\}_{k=1}^{\infty}$ are minimizing sequences of $\{\mathbf{c}^k_{ih}\}$ such that the cost function (5.1) achieves minimization.

Step 0. Given $\mathbf{z}_d$ and $\mathbf{c}^k_{ih}(0) \in \mathcal{C}_{ad}$, use the approximate solution $\mathbf{y}_h = \sum_{e=1}^{N} \sum_{j=1}^{3} \xi^e_j(t) b^e_j(x)$ to solve the direct problems (5.2) for $\mathbf{y}_h$. Let $\mathbf{P}^k(t) := -\nabla J(\mathbf{c}(0))$; $k := 0$;

Step 1. By calculating the initial inputs $\mathbf{y}(\mathbf{c}_h, T) - \mathbf{z}_d^T$ of approximate adjoint solution $\mathbf{p}_h = \sum_{e=1}^{N} \sum_{j=1}^{3} \zeta_j^e(t) b_j^e(x)$, we can solve simultaneously equations consisting of (3.1) as in Step 0.

Step 2. Compute the search step size $\beta^k = (\beta_1^k, \beta_2^k, \ldots, \beta_n^k)$ such that

$$J_h(\mathbf{c}_h^k + \beta^k \mathbf{c}_h^k) = \min\{J_h(\mathbf{c}_h^k + \beta^k \mathbf{c}_h^k); \beta \geq 0\}.$$

Given $0 < \xi < \dfrac{1}{2}$, $0 < \tau < 1$. Let $\rho_i^0 = 1$, for $m = 0, 1, 2, \ldots$. If

$$J_h(\mathbf{c}_h^k - \rho_i^m \mathbf{P}^k) \leq J_h(\mathbf{c}_h^k) - \xi \rho_i^m J_h'(\mathbf{c}_h^k)^2,$$

then $\beta^n = \rho^m$; else $\rho^{m+1} = \tau \rho^m$.

Step 3. $\mathbf{c}_h^{k+1} := \mathbf{c}_h^k + \beta^k \mathbf{P}^k$;

Step 4. Computing the gradient of the functional J_h', one can obtain

$$\begin{aligned} J_h'(\mathbf{y}_h, \mathbf{c}_h^k)\boldsymbol{\phi}_h = {} & 2\varepsilon_3 \int_0^T \int_0^l (\mathbf{p}_h(x,t) + \mathbf{u}_h^k(x,t))\boldsymbol{\phi}_h(x,t)dxdt \\ & + 2\varepsilon_4 \int_0^T \Big(\mathbf{p}_h(0,t) + \mathbf{v}_h^k(0,t)\Big)\boldsymbol{\phi}_h(0,t)dt \\ & + 2\varepsilon_4 \int_0^T \Big(\mathbf{p}_h(1,t) + \mathbf{v}_h^k(1,t)\Big)\boldsymbol{\phi}_h(1,t)dt \\ & + 2\varepsilon_5 \int_0^T \Big(\mathbf{p}_h(x,0) + \mathbf{w}_h^k(x,0)\Big)\boldsymbol{\phi}_h(x)dx, \end{aligned} \tag{5.5}$$

for all $\boldsymbol{\phi}_h = (\phi_h^1, \phi_h^2, \ldots, \phi_h^n) \in V_h^n$. Citing the thought of Carleman weighted functions (cf. [5]), $\varepsilon_3, \varepsilon_4, \varepsilon_5$ will be considered to obtain available solutions in gradient calculation (5.5).

Step 5. Compute the conjugate coefficients $\gamma^k = (\gamma_1^k, \gamma_2^k, \ldots, \gamma_n^k)$ in (5.4);

Step 6. Compute direction of descent $\mathbf{P}^k(t) := J_h^{'k}(t) + \gamma^k \mathbf{P}^{k-1}(t)$ in (5.3); $k := k + 1$; return to Step 1, and so on.

5.4. Convergence of the nonlinear algorithm. We just need to prove the sequence $\{\mathbf{c}_h^k\}_{k=1}^{\infty}$ satisfying

$$J_h(\mathbf{c}_h^k - \beta^k \mathbf{P}^k) \leq J_h(\mathbf{c}_h^k) - \xi \rho^k J_h'(\mathbf{c}_h^k)^2$$

and $\mathbf{c}_h^{k+1} = \mathbf{c}_h^k - \beta^k \mathbf{P}^k$ such that cost function $J_h(\mathbf{c}_h^k) \to$ minimize. In fact, then

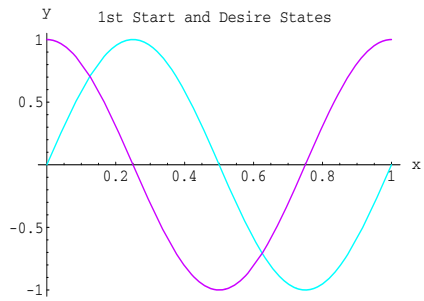
$$0 \leq \xi\{\rho^1 J_h'(\mathbf{c}_h^k)^2 + \rho^2 J_h'(\mathbf{c}_h^k)^2 + \cdots + \rho^n J_h'(\mathbf{c}_h^k)^2\} \leq J(\mathbf{c}_h^k) - J(\mathbf{c}_h^{k+1}) \leq J(\mathbf{c}_h^1).$$

Therefore, the series

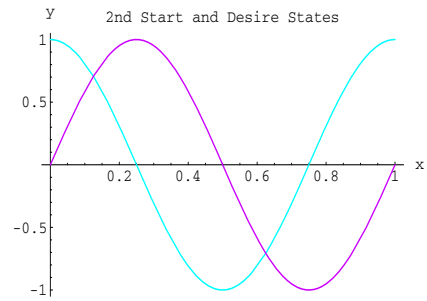
$$\rho^1 J_h'(\mathbf{c}_h^k)^2 + \rho^2 J_h'(\mathbf{c}_h^k)^2 + \cdots + \rho^n J_h'(\mathbf{c}_h^k)^2 + \cdots \to 0 \quad \text{as } k \to \infty.$$

For arbitrary k, there exists a ρ^* such that $\rho^k \geq \rho^* > 0$, and $J_h(\mathbf{c}_h)$ can be minimized by $\{\mathbf{c}_h^k\}$.

The convergence order can be easily obtained for approximate pairing $(\mathbf{y}_h, \mathbf{c}_h)$; it is clear that $\mathbf{c}_h$ moves towards $\mathbf{c}^*$ in the order of $O(h)$ as $h \to 0$.

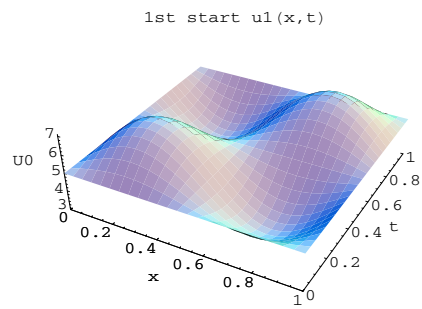


$y_1(x,0), y_1(x,T)$

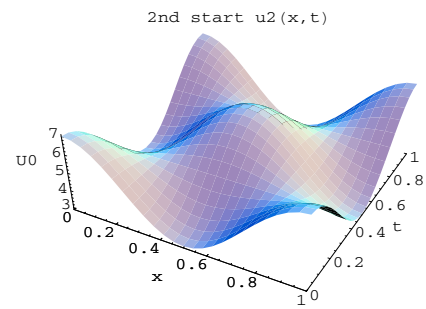


$y_2(x,0), y_2(x,T)$

FIG. 6.1.



$u_1(x,t)$ at iteration $k=0$



$u_2(x,t)$ at iteration $k=0$

FIG. 6.2.

6. Experimental demonstration. Consider distributed, boundary, and initial controls of two simultaneous systems:

$$\begin{cases} \dfrac{\partial y_1}{\partial t} - d_1\Delta y_1 = -y_1 + \beta_1(\tanh y_1 + \tanh y_2) + u_1(t,x) \ \text{ on } (0,1)\times[0,T], \\ \dfrac{\partial y_2}{\partial t} - d_2\Delta y_2 = -y_2 - \beta_2(100\tanh y_1 - 2\tanh y_2) + u_2(t,x) \ \text{ on } (0,1)\times[0,T], \\ y_{1x}(t,0) = v_1(0,t), y_{1x}(t,1) = v_1(1,t), y_{2x}(t,0) = v_2(0,t), y_{2x}(t,1) = v_2(1,t) \text{ in } [0,T], \\ y_1(0,x) = w_1(x), \quad y_2(0,x) = w_2(x) \ \text{ on } [0,1]. \end{cases}$$

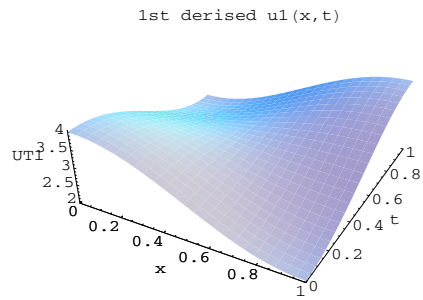
Let $h = 1/16$, $d_1 = d_2 = 0.0001$, and $\beta_1 = \beta_2 = 0.5$. The initial and desired state are given as follows.

$$\begin{aligned} y_1(x,0) &= \sin(2\pi x), \quad & z_{1d}^T &= 0.00001\sin(\pi x), \\ y_2(x,0) &= \cos(2\pi x), \quad & z_{2d}^T &= 0.00001\cos(\pi x). \end{aligned}$$

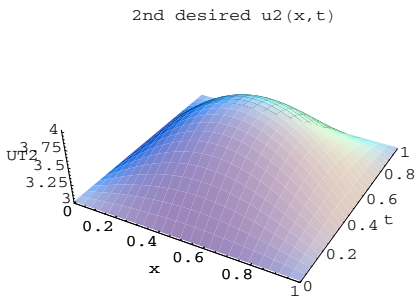
Figure 6.1 shows their graphs. The start and desired distributed controls:

$$\begin{aligned} u_1(x,t) &= 3 + 2\sin(2\pi t)\sin(2\pi x), \quad & u_1^d &= 3 + \cos(\pi t)\cos(\pi x), \\ u_2(x,t) &= 3 + 2\cos(2\pi t)\cos(2\pi x), \quad & u_2^d &= 3 + 2\sin(\pi t)\sin(\pi x). \end{aligned}$$

The graphs of the start controls are shown in Figure 6.2. The desired distributed controls are shown in Figure 6.3.

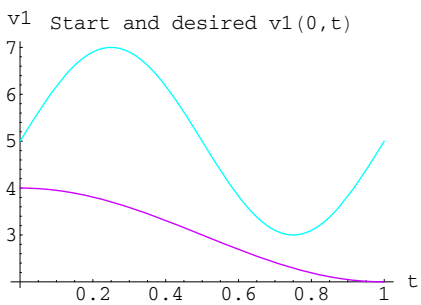


$u_1^d(x,t)$

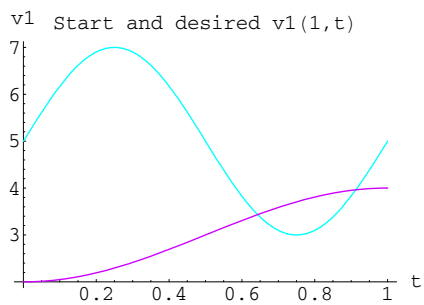


$u_2^d(x,t)$

FIG. 6.3.

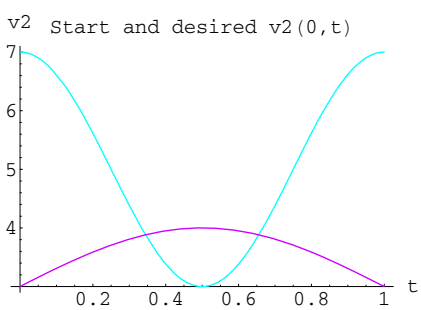


$v_1(0,t)$ at iteration $k=0$

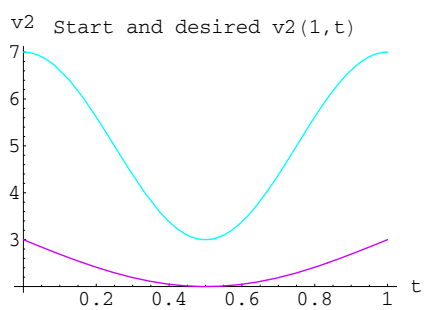


$v_1(1,t)$ at iteration $k=0$

FIG. 6.4.



$v_2(0,t)$ at iteration $k=0$



$v_2(1,t)$ at iteration $k=0$

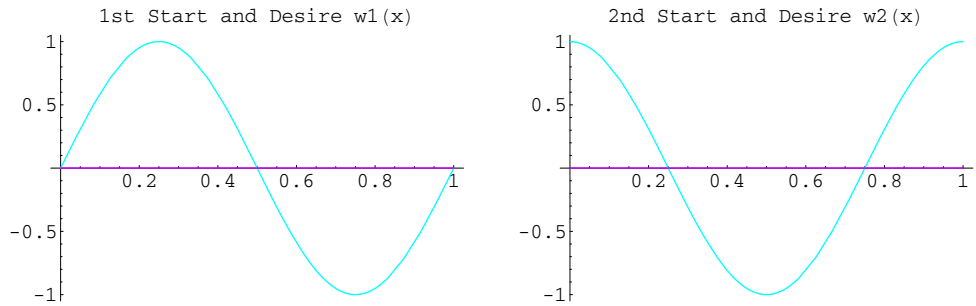
FIG. 6.5.

The first boundary controls are given by

$$v_1(0,t) = 5 + 2\sin(2\pi t), \quad v_{10}^d = 3 + 2\cos(\pi t),$$
$$v_1(1,t) = 5 + 2\sin(2\pi t), \quad v_{11}^d = 3 + 2\cos(\pi t).$$

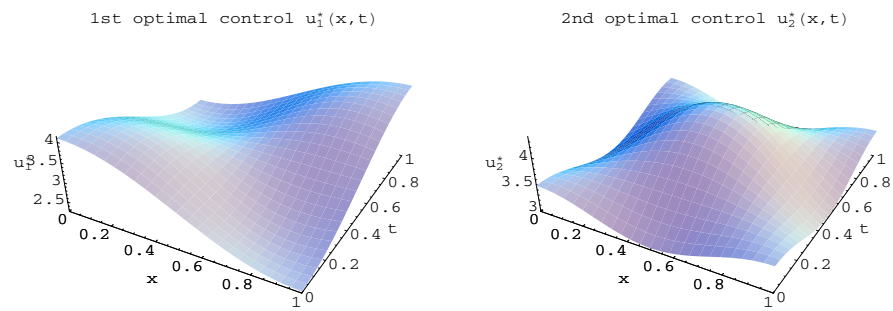
Their graphs are shown in Figure 6.4. The second boundary controls are given as follows, and plotted in Figure 6.5:

$$v_2(0,t) = 5 + 2\cos(2\pi t), \quad v_{20}^d = 3 + 2\sin(\pi t),$$
$$v_2(1,t) = 5 + 2\cos(2\pi t), \quad v_{21}^d = 3 + 2\sin(\pi t).$$



$w_1(x)$ at iteration $k = 0$ $\qquad$ $w_2(x)$ at iteration $k = 0$

FIG. 6.6.



$u_1^*(x,t)$ $\qquad$ $u_2^*(x,t)$

FIG. 6.7.

The initial controls are provided in the following form, and shown in Figure 6.6:

$$w_1(x) = \sin(2\pi x), \quad w_1^d = 0.00001\cos(\pi x),$$
$$w_2(x) = \cos(2\pi x), \quad w_2^d = 0.00001\sin(\pi x).$$

Optimal distributed control functions of two neurons can be obtained as follows:

$$u_1^*(t,x) = 3.19707 + 0.901467\cos[\pi t]\cos[\pi x] + 0.197066\sin[2\pi t]\sin[2\pi x];$$
$$u_2^*(t,x) = 3.23974 + 0.239738\cos[2\pi t]\cos[2\pi x] + 0.880131\sin[\pi t]\sin[\pi x].$$

Optimal distributed control graphs are displayed in Figure 6.7. The first optimal boundary controls are given by

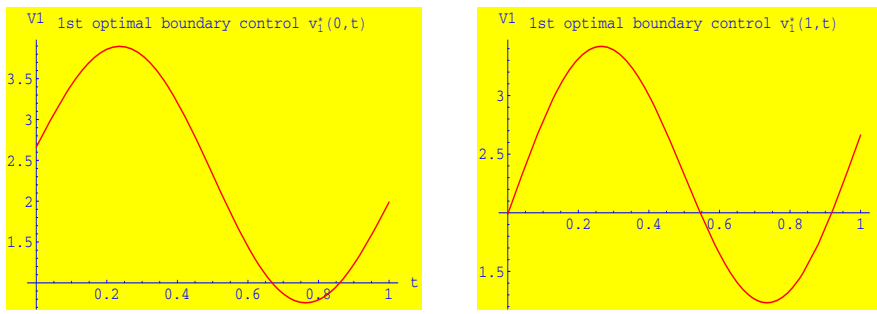
$$v_1^*(0,t) = 2.32593 + 0.33703\cos[\pi t] + 1.32593\sin[2\pi t];$$
$$v_1^*(1,t) = 2.32593 + 0.33703\cos[\pi(1+t)] + 1.32593\sin[2\pi(1+t)].$$

The graphs of first optimal boundary controls are given in Figure 6.8. The second optimal boundary controls are

$$v_2^*(0,t) = 2.26570 + 1.26561\cos[2\pi t] + 0.36719\sin[\pi t];$$
$$v_2^*(1,t) = 2.26570 + 1.26561\cos[2\pi(1+t)] + 0.36719\sin[\pi(1+t)].$$

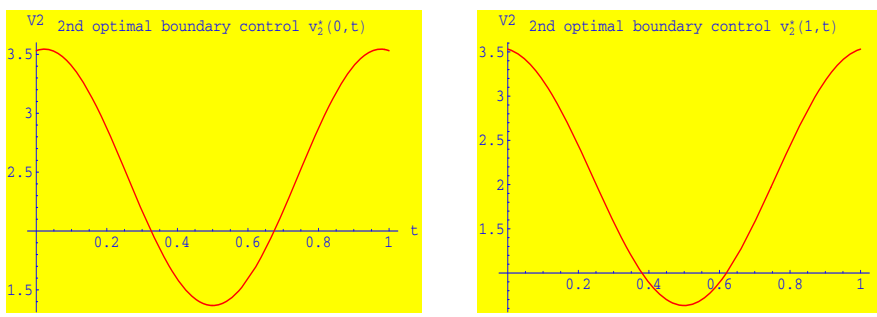
The graphs of second optimal boundary controls are shown in Figure 6.9. The optimal initial controls are given by the following, and graphed in Figure 6.10:

$$w_1^*(x) = -4.23433 \times 10^{-7} + 3.3704110^{-6}\cos[\pi x] + 0.66296\sin[2\pi x];$$
$$w_2^*(x) = 0.00009 + 0.27130\cos[2\pi x] + 7.28697 \times 10^{-6}\sin[\pi x].$$



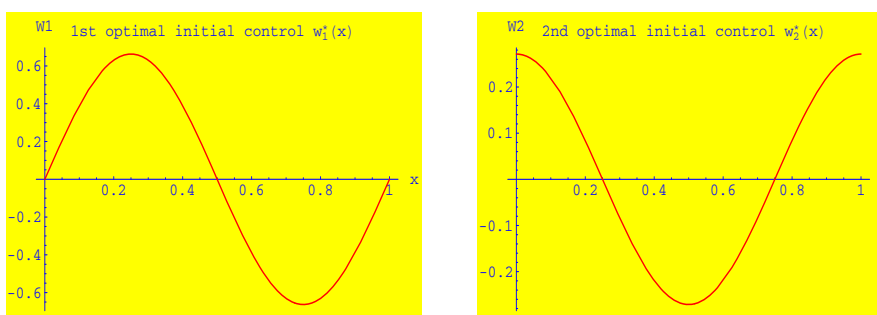
$v_1^*(0,t)$ $v_1^*(1,t)$

FIG. 6.8.



$v_2^*(0,t)$ $v_2^*(1,t)$

FIG. 6.9.

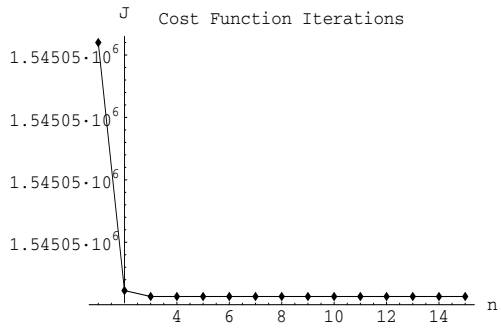


$w_1^*(x)$ $w_2^*(x)$

FIG. 6.10.

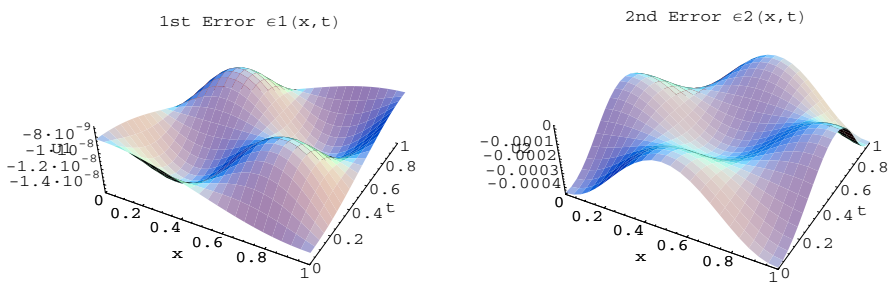
The optimal cost value $J[\mathbf{u}^*] = 1.54505 \times 10^6$; for the cost iterations, see Figure 6.11. Error estimates for cost function can be calculated step by step as

$$
\begin{array}{lll}
eJ[1] = - - -, & eJ[2] = 29158.8, & eJ[3] = 192.595, \\
eJ[4] = 0.459979, & eJ[5] = 0.517135, & eJ[6] = 0.587047, \\
eJ[7] = 0.663581, & eJ[8] = 0.747305, & eJ[9] = 0.838851, \\
eJ[10] = 0.938926, & eJ[11] = 1.04832, & eJ[12] = 1.16791, \\
eJ[13] = 1.29869, & eJ[14] = 1.44176, & eJ[15] = 1.59836.
\end{array}
$$



$J(\mathbf{y}, \mathbf{c})$

FIG. 6.11.



$\varepsilon_{u_1(x,t)}$ $\varepsilon_{u_2(x,t)}$

FIG. 6.12.

The L^2 error between optimal control and desired control is as follows:

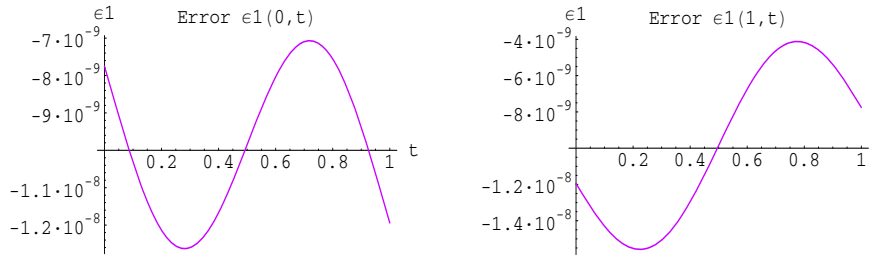
$$\begin{aligned}
&\|u_1 - u_1^T\|_{L^2(Q)} = 0, \qquad \|u_2 - u_2^T\|_{L^2(Q)} = 0,\\
&\|v_1(0) - v_1^T\|_{L^2(0,T)} = 0, \qquad \|v_1(1) - v_1^T\|_{L^2(0,T)} = 0,\\
&\|v_2(0) - v_2^T\|_{L^2(0,T)} = 0, \qquad \|v_2(1) - v_2^T\|_{L^2(0,T)} = 0,\\
&\|w_1 - w_1^T\|_{L^2(0,l)} = 0, \qquad \|w_2 - w_2^T\|_{L^2(0,l)} = 0.009977802.
\end{aligned}$$

For the error of the distributed control functions at the final iteration, see Figure 6.12. For the error of the boundary controls at the final iteration, refer to Figures 6.13–6.14. The error of the initial controls at the final iteration is shown in Figure 6.15.

Necessary optimality conditions can be checked numerically for control variables in each iteration step. Let N_C represent the left-hand side of the inequality in (3.16); then

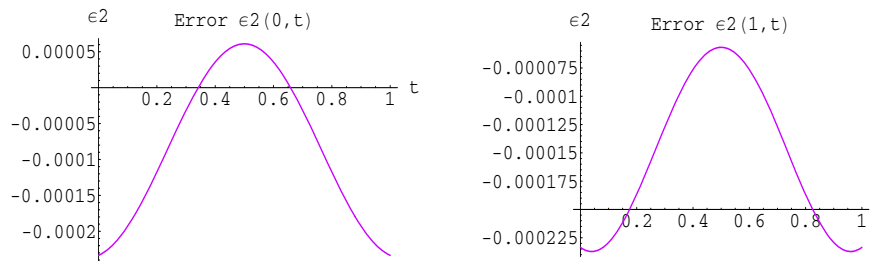
$$\begin{aligned}
&N_C[1] = 8.99137 \times 10^9, \quad N_C[2] = 7.41451 \times 10^8, \quad N_C[3] = 1.73174 \times 10^7,\\
&N_C[4] = 1.58551 \times 10^7, \quad N_C[5] = 1.44051 \times 10^7, \quad N_C[6] = 1.29565 \times 10^7,\\
&N_C[7] = 1.15096 \times 10^7, \quad N_C[8] = 1.0064 \times 10^7, \quad N_C[9] = 8.62095 \times 10^6,\\
&N_C[10] = 7.17927 \times 10^6, \quad N_C[11] = 5.73944 \times 10^6, \quad N_C[12] = 4.30154 \times 10^6,\\
&N_C[13] = 2.86562 \times 10^6, \quad N_C[14] = 1.43175 \times 10^6, \quad N_C[15] = 0.
\end{aligned}$$

Clearly, the optimality necessary condition is valid.



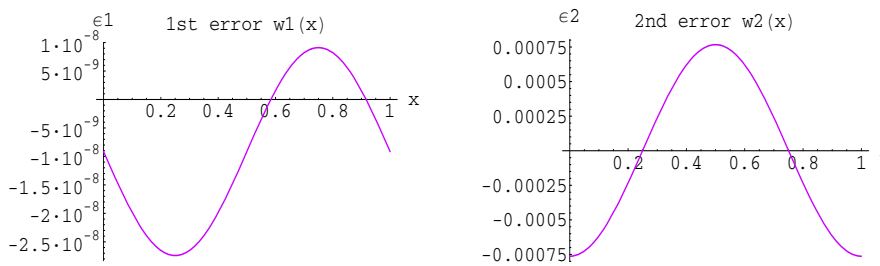
$\varepsilon_{v_1(0,t)}$ $\varepsilon_{v_1(1,t)}$

FIG. 6.13.



$\varepsilon_{v_2(0,t)}$ $\varepsilon_{v_2(1,t)}$

FIG. 6.14.



$\varepsilon_{w_1(x)}$ $\varepsilon_{w_2(x)}$

FIG. 6.15.

Observing the experimental results on the control of neural network systems, in the viewpoint of biological science and medicine science, the methodology can be considered for controlling a large number of neurons by laser pulse input, language, or medicine drug technology. It would be useful to combine this approach with interdisciplinary research.

7. Conclusions. The optimal control for diffusion Hopfield neural networks was studied both in theoretical and numerical aspects. Based on the development of a variational method to the diffusion model, the optimal control problem has been investigated by the means of distributed, boundary, and initial control simultaneously. The minimization of quadratic criteria with terminal observation was solved by con-

structing a semidiscrete algorithm with a finite element approach. The study of the higher-dimensional case is interesting and will be considered in upcoming research.

Acknowledgments. The author thanks the Japan Society of Industry and Applied Mathematics and the Second International Conference of Applied Mathematics for presenting the previous work.

REFERENCES

[1] R. A. Adams, *Sobolev Spaces*, Academic Press, New York, 1975.
[2] N. Arada and J. P. Raymond, *Dirichlet Boundary Control of Semilinear Parabolic Equations, Part* 1: *Problem With No State Constraints*, Report 98-05, UMR-CNRS 5640, Toulouse, France.
[3] V. Barbu, *Analysis and Control of Nonlinear Infinite Dimensional Systems,* Academic Press, New York, 1993.
[4] E. Casas, *Pontryagin's principle for state-constrained boundary control problems of semilinear parabolic equations*, SIAM J. Control Optim., 35 (1997), pp. 1297–1327.
[5] T. Carleman, *Sur un probléme d'unicité pour les systémes déquations aux dérivées partielles á deux variables indépendantes*, Ark. Mat. Astr. Fys., 26, 17 (1939), pp. 1–9.
[6] R. Dautray and J, L. Lions, *Mathematical Analysis and Numerical Methods for Science and Technology, Vol.* 5*, Evolution Problems* I, Springer-Verlag, Berlin, Heidelberg, New York, 1992.
[7] P. Van den Driessche and X. Zou, *Global attractivity in delayed Hopfield neural network models*, SIAM J. Appl. Math., 58 (1998), pp. 1878–1890.
[8] R. Fitz-Hugh, *Mathematical models of threshold phenomena in the nerve membrane*, Bull. Math. Biophys, 17 (1955), p. 257.
[9] M. Forti and A. Tesi, *New conditions for global stability of neural networks with applications to linear and quadratic programming problems*, IEEE Trans. Circuits Systems I Fund. Theory Appl., 42 (1995), pp. 354–366.
[10] A. L. Hodgkin and A. F. Huxley, *A qualitative description of membrane current and its application to conduction and excitation in nerve*, J. Physiol., 117 (1952), pp. 500–544.
[11] J. J. Hopfield, *Neural network and physics system with collective computational abilities*, Proc. Natl. Acad. Sci. USA, 79 (1982), pp. 2554–2558.
[12] J. J. Hopfield, *Neurons with graded response have collective properties like those of two-state neurons*, Proc. Natl. Acad. Sci. USA, 81 (1984), pp. 3088–3092.
[13] J. J. Hopfield and D. W. Tank, *Computing with neural circuits: A Model*, Science, 233 (1986), pp. 625–632.
[14] L. S. Lasdon, S. K. Mitter, and A. D. Warren, *The conjugate gradient method for optimal control problems*, IEEE Trans. Automat. Control, 12 (1967), pp. 132–138.
[15] J. L. Lions, *Optimal Control of Systems Governed by Partial Differential Equations*, Springer-Verlag, Berlin, Heidelberg, New York, 1971.
[16] J. Nagumo, S. Arimoto, and S. Yoshizawa, *An active pulse transmission line simulating nerve axon*, Proc. Inst. Radio Engineers, 50 (1962), pp. 2061–2070.
[17] S. Nakagiri and Q. F. Wang, *Optimal control problems for distributed Hopfield-type neural networks*, Nonlinear Funct. Anal. Appl., 7 (2002) pp. 167–186.
[18] R. Temam, *Navier Stokes Equation*, North-Holland, Amsterdam, 1984.
[19] Q. F. Wang, *Numerical analysis for optimal control problems of Hopfield-type neural networks equations*, Poster, International Symposium on Intervention and Adaptation in Complex Systems, Beijing, China, 2002.
[20] Q. F. Wang, D. Feng, and D. Cheng, *Parameter identification problems for Hopfield-type neural network equations*, Appl. Math. Comput., 152 (2004), pp. 535–550.
[21] Q. F. Wang, *Numerical approximation of optimal control for distributed diffusion Hopfield neural network*, in Proceedings of the 2004 Annual Conference of the Japan Society for Industrial and Applied Mathematics, 2004, pp. 178–179.
[22] Q. F. Wang and S. Nakagiri, *Numerical solution of distributed diffusion Hopfield neural network equations*, Appl. Math. Comput., 174 (2006), pp. 1343–1362.
[23] P. De. Wilde, *Neural Network Models*, 2nd ed., Springer-Verlag, London, 1997.
[24] R. Winther, *Initial value methods for parabolic control problems*, Math. Comp., 34 (1980), pp. 115–125.

SIAM J. SCI. COMPUT.
Vol. 29, No. 3, pp. 913–940
© 2007 Society for Industrial and Applied Mathematics

AN INTERNAL STRUCTURE DEPENDENT RIEMANN SOLVER FOR REGULARIZATION-SENSITIVE SHOCK WAVES AND ITS APPLICATION TO FRONT TRACKING*

HYUN-CHEOL HWANG†

Abstract. We present an algorithm for the iterative Riemann solver for regularization-sensitive shock waves which constitute a codimension-one submanifold of the wave manifold. The algorithm uses the computation of the saddle-to-saddle type of connections in the state space using the multiple shooting method. This Riemann solver can be combined with front tracking methods to track these shock waves correctly if the physical internal structure is provided. Several numerical examples, such as transitional shock waves in the planar system and weak deflagration waves in combustion, are illustrated.

Key words. Riemann problem, hyperbolic conservation laws, front tracking, regularization-sensitive shock wave

AMS subject classifications. 35L65, 76L05, 65M06

DOI. 10.1137/040621715

1. Introduction. It is well known that certain types of shock waves are very sensitive to the regularization procedure. In the physical sense, the regularization procedure provides a discontinuous wave with a continuous internal structure on a small length scale [16], and mathematically it is a selection procedure for solutions satisfying certain mathematical properties. Usually internal structure of shock waves on a small length scale can be ignored; however, certain types of shock waves are highly dependent on the regularization procedure, not only in their internal structure, but also in their propagation speeds and limiting states. These waves, called [illegible], have been observed in numerical or real experiments, and occur in applications such as combustion [9], three-phase flow through a porous medium [3, 44], and magneto-hydrodynamics [54]. However, their exact physical meaning is not understood clearly, and the correct numerical method to handle these waves has rarely been studied.

In the mathematical structure, a regularization-sensitive shock wave differs from a classical shock wave in that fewer characteristic families impinge on the wave. Therefore the dynamical information carried into the wave along the characteristics is insufficient to determine the evolution of the wave. The missing information comes from the internal structure of the shock wave, implying that the internal structure strongly influences the wave propagation. Most numerical methods for shock waves rely on numerical diffusion for stability, and thus provide discontinuous waves with an artificial internal structure. Such methods are successful only if they are applied to waves that are insensitive to this numerical regularization. Difficulties will arise when they are used for regularization-sensitive waves: for example, excess diffusion can cause significant errors in detonation wave speeds [7], while the limiting states of a deflagration wave are sensitive to heat conduction and viscosity [9]. In order to

*Received by the editors December 29, 2004; accepted for publication (in revised form) October 12, 2006; published electronically May 7, 2007.
http://www.siam.org/journals/sisc/29-3/62171.html
†Department of Mathematics and Information, Kyungwon University, Songnam 461-701, Korea (hchwang@kyungwon.ac.kr).

circumvent these difficulties, numerical methods must account for the true internal structure of discontinuities.

Our Riemann solver is conceived for this purpose. We consider the shock waves which constitute a codimension-one submanifold of the wave manifold, i.e., the case in which one piece of characteristic information is missing compared with classical shock waves. The Riemann solver for these waves utilizes the multiple shooting method to compute the viscous profile of the corresponding dynamical system; i.e., it constructs the exact internal structure of the wave corresponding to its physically correct viscous regularization. Equipped with this Riemann solver, the front tracking method can be used to accurately locate the position of the regularization-sensitive wave. Our method has been implemented for two different applications: quadratic models and reactive gas dynamics.

The paper consists of the following. In section 2, we briefly introduce mathematical preliminaries for the general system of hyperbolic conservation laws. We discuss the construction of Riemann solutions and regularization-sensitive waves. Our main algorithm for the Riemann solver is presented in the next section 3. The algorithm for solutions of Riemann problems near a regularization-sensitive shock wave is proposed for general $m \times m$ systems of conservation laws. This Riemann solver, which incorporates the information coming from the internal structure of a shock wave, is an essential component of the front tracking method. The complete front tracking algorithm is also presented.

In sections 4 and 5, the algorithm proposed in section 3 is applied to two applications: quadratic models, which are 2×2 systems of conservation laws having quadratic flux functions ($m = 2$), and weak deflagration waves in combustion ($m = 4$). Transitional waves, regularization-sensitive waves appearing in quadratic systems, are explained, and the Riemann solution near a transitional shock wave is obtained by applying our algorithm. Computational results using front tracking are also presented and compared with those using standard shock capturing methods. For the application to weak deflagration waves, the dynamical system for the internal structure of combustion waves is derived first, and then it is shown that deflagration waves are regularization-sensitive. The algorithm of the Riemann solver for solutions near a deflagration wave is developed, and computational results are presented. We conclude this paper in section 6.

2. Mathematical preliminaries. We consider solutions of general systems of conservation laws in one spatial dimension:

$$U_t + F(U)_x = 0, \tag{2.1}$$

$$U(x, 0) = U_0(x). \tag{2.2}$$

Here $U(x, t) \in \mathbf{R}^m$ is an m-dimensional vector of conserved quantities, $F : \mathbf{R}^m \to \mathbf{R}^m$ is a smooth function called the flux function, and U_0 is the initial data. It is well known that the uniqueness of solutions depends on the admissibility condition for selecting the physically correct solution from many possible weak solutions. The correct formulation of admissibility conditions is still a crucial issue, especially for a system of more than one equation. There are many different admissibility criteria, with their own domains of application. Throughout this work, we utilize the admissibility condition called the viscous profile criterion [14, 8], since we believe it to be most physically relevant.

The viscous profile criterion requires that a shock wave arise as the limit, as $\epsilon \to 0$, of a traveling wave solution to a parabolic equation associated with (2.1),

$$U_t + F(U)_x = \epsilon[D(U)U_x]_x. \tag{2.3}$$

Here $D(U)$ is the viscosity matrix, which models physical effects that have been neglected in (2.1). The viscosity matrix $D(U)$ is required to be positive definite. A traveling wave solution takes the form

$$U(x,t) = \hat{U}(\xi), \quad \xi := \frac{x - st}{\epsilon}, \tag{2.4}$$

where $\hat{U}(\xi) \to U_\pm$ as $\xi \to \pm\infty$ and s is the propagation speed. In order for U to satisfy (2.3), $\hat{U}$ must be a solution of the dynamical system

$$-s[\hat{U}(\xi) - U_-] + F(\hat{U}(\xi)) - F(U_-) = D(\hat{U}(\xi))\hat{U}'(\xi). \tag{2.5}$$

The limiting states U_- and U_+ are equilibria of this dynamical system, satisfying the Rankine–Hugoniot condition. If a solution of system (2.5) leads from U_- to U_+, then we say that the shock wave has a viscous profile. Only those shock waves having viscous profiles are accepted as admissible solutions; this is the viscosity profile criterion.

When this criterion is used, the admissibility of a shock wave might depend on the viscosity matrix $D(U)$. Certain shock waves admissible under the Lax criterion are also admissible under the viscous profile criterion, but there are other kinds of shock waves that are admissible under the viscous profile criterion but do not satisfy the Lax criterion. Such shock waves, undercompressive shock waves which have fewer entering characteristic curves than Lax shock waves and overcompressive shock waves which have more entering characteristics, play an important role in solving Riemann problems specifically in a system of nonstrictly hyperbolic or mixed type of conservation laws [38, 44, 46, 23, 11]. Overcompressive shock waves can be regarded as a combination of classical shock waves [38]. Such shock waves are the main concern of this work.

2.1. Construction of Riemann solution and regularization-sensitive shock waves. A Riemann problem is a particular Cauchy problem having piecewise constant initial data with a single discontinuity, i.e.,

$$U(x,0) = U_0(x) = \begin{cases} U_L & \text{if } x < 0, \\ U_R & \text{if } x > 0. \end{cases} \tag{2.6}$$

The solution of Riemann problems is important both analytically and numerically. A scale invariant solution of a Riemann problem, i.e., one depending only on x/t, describes the large-time asymptotic behavior of a stable wave pattern. Lax [28, 29] constructed the solution of Riemann problems when U_L is close to U_R under the assumptions of strict hyperbolicity (distinctness of the characteristic speeds) and of genuine nonlinearity within each mode. Under these hypotheses, at a given state U_L, there is a unique half of the shock curve stable under forward time evolution and a unique half of the rarefaction curve in other direction for the wave speed to increase along a curve. Hence two half curves join smoothly and define a [illegible] through U_R. There is one such curve for each characteristic mode, and the solution of a Riemann problem is constructed by connecting two initial states with the series of wave curves.

However, using only classical wave groups is not always sufficient for solving all Riemann problems for a systems of conservation laws. For example, when the strict hyperbolicity fails, the system has portions of state space that cannot be reached using classical waves. A nonclassical wave group, which does not correspond to a particular characteristic wave mode and is regularization-sensitive, fits between classical wave groups and fills these gaps.

The shock waves which we consider are different from classical Lax shock waves in that fewer characteristic families impinge on the wave. This difference has two main consequences. First, a regularization-sensitive wave is not associated with a unique characteristic family. It could appear as a wave family between two distinct wave families, say i and $(i+1)$. If we admit only Lax shock waves, then any wave has an associated family. Moreover, no wave can be preceded by a wave of a faster family, and two waves of the same family must belong to the same wave group. It follows that the solution of a Riemann problem contains at most m shock or rarefaction waves, where each is associated with a distinct family. However, if we allow nonclassical shock waves, then it is possible for the Riemann solution to contain more than m wave groups.

The other consequence is that less information enters into the wave from the surrounding flow. To compensate for this lack of information, information must come from the internal structure of the shock wave. Thus the internal structure of the shock wave strongly influences the propagation of the wave. From the dynamical systems viewpoint, its phase portrait is generally unstable. In the case of $m = 2$, this connection corresponds to a saddle-to-saddle connection. The structural instability of this connection requires that only special discontinuities, those in a codimension-one set, have viscous profiles [23, 39]. Moreover, the set of such shock waves generally varies with the viscosity matrix $D(U)$ in (2.3). In other words, not only is the internal structure sensitive to the precise form of the viscosity matrix, but the limiting states between the shock wave and the speed are also. Therefore, we say that these waves are regularization-sensitive.

3. Algorithm for Riemann solutions near a regularization-sensitive shock wave. We solve the Riemann problem defined by the states on either side. In general, solving a Riemann problem is extremely complicated. However, the Riemann problems that occur in numerical methods like front tracking usually have a simple feature: except for the tracked wave, all waves in the solution are weak. Indeed, the Riemann problem at a particular time level arises from the interaction, during the previous time step, of the tracked wave and the (usually) weak waves that have impinged on it. We say that a Riemann problem is near a certain wave type if its solution consists of a wave of this type together with small amplitude waves of other types. The algorithm we now describe works for a Riemann problem near a regularization-sensitive shock wave.

We consider the system of hyperbolic conservation laws (2.1) and its corresponding parabolic regularized equation (2.3). As we have shown in section 2, since the shock wave we consider is the limit of a traveling wave solution of (2.3) as $\epsilon \to 0$, U should be a solution of the dynamical system

$$U' = D(U)^{-1}[-s(U - U_M) + F(U) - F(U_M)] =: G(U, U_M, s), \tag{3.1}$$

where $U_M = U(-\infty)$ and $U_N = U(+\infty)$.

We shall solve the Riemann problem by finding a sequence of states $U_1, U_2, \ldots, U_m$ such that U_L is connected to U_1 by a 1-wave, U_1 is connected to U_2 by a 2-wave, ...,

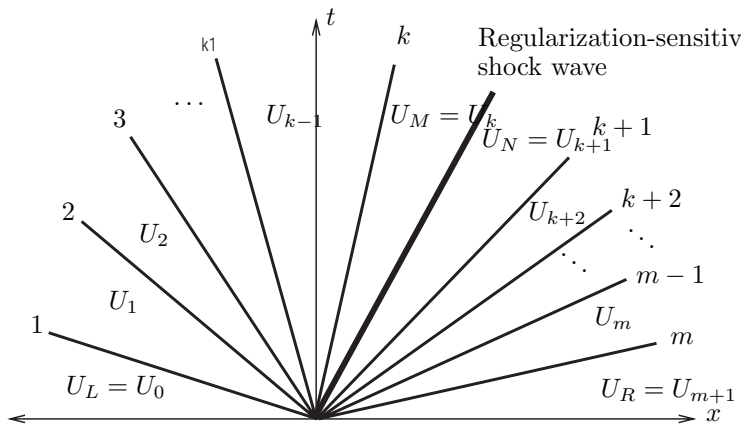


FIG. 1. *The solution of a Riemann problem near a regularization-sensitive wave which lies between k and $k+1$ waves as drawn in the (x,t)-plane.*

$U_k =: U_M$ is connected to $U_{k+1} =: U_N$ by a regularization-sensitive shock, and so on, with U_m connected to U_R by an m-wave. (See Figure 1.) Here the connecting wave from U_{i-1} to U_i can be either a shock wave or a rarefaction wave; thus U_i lies on the wave curve (either a shock curve or a rarefaction curve) from U_{i-1} in the state space.

Therefore, from U_L we can reach a one parameter family of states $U_1 = \mathcal{W}_1(U_L, \varepsilon_1)$ through a 1-wave. From U_1 we can reach another one parameter family of states $U_2 = \mathcal{W}_2(U_L, \varepsilon_1, \varepsilon_2)$ through a 2-wave. Continuing, we see that from U_L we can reach a k parameter family of states

$$U_M = \mathcal{W}_M(U_L, \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_k). \tag{3.2}$$

Similarly, we can reach an $m-k$ parameter family of states

$$U_N = \mathcal{W}_N(U_R, \varepsilon_{k+1}, \varepsilon_{k+2}, \ldots, \varepsilon_m) \tag{3.3}$$

from U_R. Moreover, we know that if U_i lies on the ith family of wave curves through U_{i-1}, the wave curve is tangent to the ith eigenvector of $F'(U)$ at U_{i-1}. Thus when U_{i-1} and U_i are close enough, this curve can be approximated by a straight line. It gives the following formulations:

$$U_i = \begin{cases} \mathcal{W}_i(U_{i-1}, \varepsilon_i) \approx U_{i-1} + \varepsilon_i\, r_i(U_{i-1}) & \text{when } i = 1, 2, \ldots, k, \\ \mathcal{W}_i(U_{i+1}, \varepsilon_i) \approx U_{i+1} - \varepsilon_i\, r_i(U_{i+1}) & \text{when } i = k, k+1, \ldots, m. \end{cases} \tag{3.4}$$

Here $r_i(U)$ is the ith eigenvector of $F'(U)$ corresponding to the eigenvalue $\lambda_i(U)$, and U_0 and U_{m+1} are U_L and U_R, respectively.

The states U_M and U_N are also equilibria of dynamical systems (3.1), which satisfies the Rankine–Hugoniot condition for a regularization-sensitive shock wave:

$$G(U_N, U_M, s) = 0. \tag{3.5}$$

To calculate a viscous profile connection from U_M to U_N, we employ the multiple shooting method [49], as follows.

First, the ξ-axis is divided into the intervals $(-\infty, -T_1]$, $[-T_1, T_2]$, and $[T_2, \infty)$, where T_1 and T_2 are suitably large (see below); then the interval $[-T_1, T_2]$ is divided into n subintervals defined by the endpoints $\xi_0 := -T_1 < \xi_1 < \cdots < \xi_{n-1} < \xi_n := T_2$. State variables U_0^*, ..., U_n^* corresponding to points ξ_0, ..., ξ_n are introduced; U_M corresponds to $\xi = -\infty$, and U_N to $\xi = \infty$. The object is to have successive states joined by integral curves of the dynamical system (3.1).

Let the dimension of the unstable manifold at U_M be l. l could be any number between 0 and m, but its sum with the dimension of the stable manifold at U_N must be m since a regularization-sensitive wave constitutes the codimension-one submanifold of the wave manifold. The state U_0^* at $\xi = -T_1$ can then be approximated as

$$U_0^* = U_M + \tilde{\varepsilon}_m \tilde{r}_m(U_M) + \tilde{\varepsilon}_{m-1} \tilde{r}_{m-1}(U_M) + \cdots + \tilde{\varepsilon}_{m-l+1} \tilde{r}_{m-l+1}(U_M), \tag{3.6}$$

for some values of $\tilde{\varepsilon}_m, \ldots, \tilde{\varepsilon}_{m-l+1}$. Here $\tilde{r}_i(U_M)$ denotes an ith eigenvector of $G_U(U_M, s)$ corresponding to the positive eigenvalue $\mu_i(U_M), i = m, m-1, \ldots, m-l+1$; these eigenvectors form a basis for the tangent space of the unstable manifold at U_M. Similarly, the state U_n^* at $\xi = T_2$ is approximated as

$$U_n^* = U_N + \tilde{\varepsilon}_1 \tilde{r}_1(U_N) + \tilde{\varepsilon}_2 \tilde{r}_2(U_N) + \cdots + \tilde{\varepsilon}_{m-l} \tilde{r}_{m-l}(U_N) \tag{3.7}$$

because the dimension of stable manifold at U_N is $m-l$. Here $\tilde{r}_i(U_N)$ is the eigenvector of $G_U(U_N, s)$ corresponding to the negative eigenvalues $\mu_i(U_N), i = 1, 2, \ldots, m-l$. This linearized approximation also motivates choosing the values

$$T_1 = \frac{a}{\min_{m-l+1 \le i \le m} \mu_i(U_M)}, \quad T_2 = \frac{a}{\min_{1 \le i \le m-l} |\mu_i(U_N)|}, \tag{3.8}$$

where a is a dimensionless constant (taken to be 4 or 5 in the numerical experiments of later sections). For $i = 0, \ldots, n-1$, we let $H(U_i^*, s, U_L, \varepsilon_1, \ldots)$ denote the state obtained by numerically integrating the dynamical system (3.1) from $\xi = \xi_i$ to $\xi = \xi_{i+1}$ starting from the state U_i^*.

We have $mn + m + 1$ unknown quantities, namely,

$$\mathbf{x} := (\varepsilon_1, \ldots, \varepsilon_m, \tilde{\varepsilon}_1, \ldots, \tilde{\varepsilon}_m, s, U_1^*, \ldots, U_{n-1}^*)$$

(treating U_M, U_N, U_0^*, and U_n^* as given in these terms by the approximations described above), for which we require $mn+m+1$ equations. The mn matching conditions and the m Rankine–Hugoniot condition constitute $mn + m$ equations:

$$\begin{aligned}
H(U_0^*, s, U_L, \varepsilon_1, \ldots, \varepsilon_m) - U_1^* &= 0, \\
H(U_1^*, s, U_L, \varepsilon_1, \ldots, \varepsilon_m) - U_2^* &= 0, \\
&\vdots \\
H(U_{n-2}^*, s, U_L, \varepsilon_1, \ldots, \varepsilon_m) - U_{n-1}^* &= 0, \\
H(U_{n-1}^*, s, U_L, \varepsilon_1, \ldots, \varepsilon_m) - U_n^* &= 0, \\
G(U_N, U_M, s) &= 0.
\end{aligned} \tag{3.9}$$

There is one remaining degree of freedom, which reflects nonuniqueness of the solutions: since the dynamical system is autonomous, replacing t by $t +$ const. in one

solution yields another. To fix the solution uniquely, we impose the phase condition of [10], which requires that

$$\int_{-\infty}^{\infty} \langle G(U) - G(U_{\text{ref}}), G_U(U) \cdot G(U) \rangle \, d\xi = 0 \tag{3.10}$$

for some reference solution U_{ref}. (Here the arguments of G are suppressed, and the angle brackets denote the inner product.) We choose U_{ref} to be the solution calculated at the previous time level in the front tracking method, and we discretize the integral using the trapezoid rule. Now we have a complete set of equations, which we solve using a damped Newton method [49], with Broyden update of the Jacobian, to obtain the solution of the Riemann problem particularly U_M, U_N, and s.

. . .. 3.1 (derivation of the phase condition). The idea behind the phase condition (3.10) is that the solution should resemble the reference solution, in our case the previous computed orbit, as much as possible. We need to find the minimum distance orbit $U(t)$ among the one parameter family of solutions $\tilde{U}$ satisfying matching conditions and the Rankine–Hugoniot condition. Let $L(\sigma)$ denote the following distance between the orbit U_{ref} and the orbit $\tilde{U}(t+\sigma)$, as shifted by σ from which the current orbit is selected:

$$L(\sigma) = \int_{-\infty}^{\infty} \|\tilde{U}'(t+\sigma) - U'_{\text{ref}}(t)\|^2 dt, \tag{3.11}$$

where U_{ref} is a previous computed orbit. The necessary condition for a minimum distance is $\frac{dL}{d\sigma} = 0$. Let us define $U(t)$ as

$$U(t) := \tilde{U}(t + \sigma^*), \tag{3.12}$$

where σ^* satisfies $\frac{dL}{d\sigma}|_{\sigma=\sigma^*} = 0$. Then, the necessary condition for minimum distance is given as

$$\begin{aligned} \left.\frac{dL}{d\sigma}\right|_{\sigma=\sigma^*} &= \int_{-\infty}^{\infty} \langle U' - U'_{\text{ref}},\, U''(t) \rangle \, dt \\ &= \int_{-\infty}^{\infty} \langle G(U) - G(U_{\text{ref}}),\, G_U(U) \cdot G(U) \rangle \, d\xi \\ &= 0. \end{aligned} \tag{3.13}$$

. . .. 3.2 (damped Newton method with Broyden update). We solve the nonlinear system of equations by using a damped Newton method combined with line searches (to obtain a global convergence and to avoid trouble with U_M and U_N being in an elliptic region during iterations):

$$\mathbf{F}(\mathbf{x}) = 0, \quad \mathbf{x}_{\text{new}} = \mathbf{x}_{\text{old}} + \lambda \delta \mathbf{x}, \tag{3.14}$$

where $\delta \mathbf{x} = -\mathbf{F}'(\mathbf{x}_{\text{old}})^{-1} \cdot \mathbf{F}$ is a Newton step. As a strategy to decide whether to accept the Newton step $\delta \mathbf{x}$, we require that the step decreases $|\mathbf{F}|^2 = \mathbf{F} \cdot \mathbf{F}$. This is the same requirement to minimize

$$f = \frac{1}{2} \mathbf{F} \cdot \mathbf{F}. \tag{3.15}$$

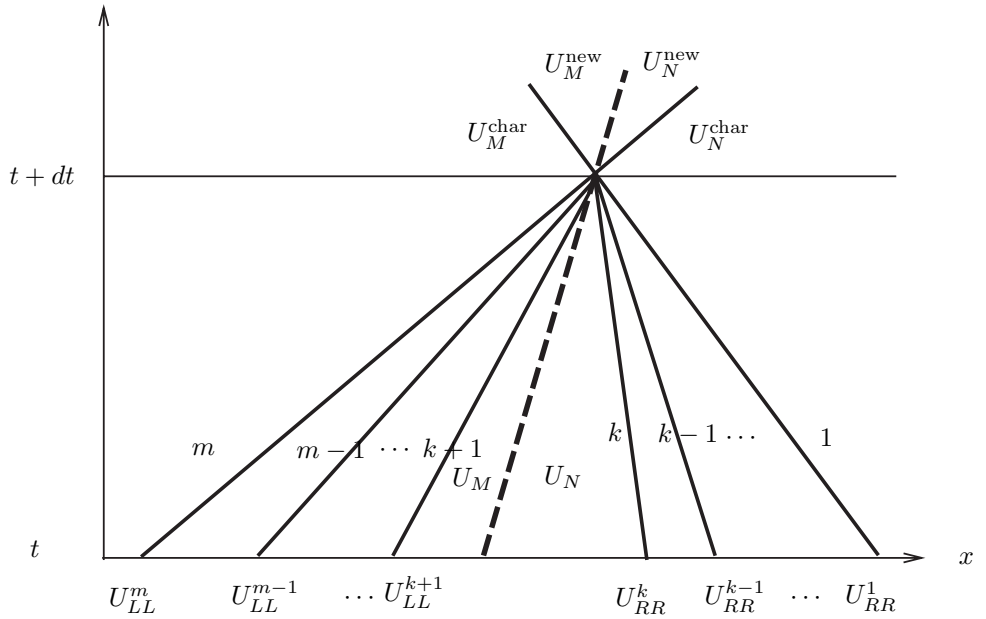


FIG. 2. *This figure shows the characteristic relation between a tracked regularization-sensitive shock wave and surrounding characteristic curves, drawn in the* (x, t) *plane.*

The aim is to find λ so that $f(\mathbf{x}_{\text{old}} + \lambda\delta\mathbf{x})$ has decreased sufficiently. We choose $\lambda = 1$ at first to make it be quadratic convergent when $\mathbf{x}$ is sufficiently close to the solution. However, if $f(\mathbf{x}_{\text{new}})$ does not decrease sufficiently, then choose λ so that $\mathbf{x}_{\text{new}}$ minimizes f in the Newton direction as a smaller value of λ. The criterion for accepting a step is to require the average rate of decrease of f to be at least some fraction α of the initial rate of decrease $\nabla f \cdot \delta\mathbf{x}$:

$$f(\mathbf{x}_{\text{new}}) \le f(\mathbf{x}_{\text{old}}) + \alpha\nabla f \cdot (\mathbf{x}_{\text{new}} - \mathbf{x}_{\text{old}}). \tag{3.16}$$

In computation, we choose α as a very small value: 10^{-4}.

If the step does not satisfy (3.16), we backtrack along the Newton direction by finding a suitable λ to satisfy (3.16). To do this, let

$$g(\lambda) \equiv f(\mathbf{x}_{\text{old}} + \lambda\delta\mathbf{x}). \tag{3.17}$$

Since we know $g(0), g'(0)$, and $g(1)$ which is a Newton step, we can approximate λ as a quadratic polynomial; thus we can have the exact form of λ which gives a minimum of $g(\lambda)$. If U_M and U_N is in an elliptic region in this procedure, then λ is reduced by half until they escape from an elliptic region. For details about Broyden the update, see [49].

3.1. Tracking algorithm. The algorithm to track the shock waves comprises four steps. (For the notation of states surrounding a tracked shock wave, see Figure 2.) Given the states U_L and U_R at time t, we do the following:

1. solve the Riemann problem and find U_M, U_N, and s;
2. find the states $U_{LL}^{k+1}, \ldots, U_{LL}^m$ and $U_{RR}^1, \ldots, U_{RR}^k$ at the feet of characteristics drawn from the new shock position;

3. calculate a characteristic correction, for improved accuracy, by solving the equations

$$l_i(U_M^{\text{char}})(U_M^{\text{char}} - U_M) = l_i(U_{LL}^i)(U_{LL}^i - U_L), \tag{3.18}$$

$$l_i(U_N^{\text{char}})(U_N^{\text{char}} - U_N) = 0, \quad i = k+1, k+2, \ldots, m, \tag{3.19}$$

$$l_j(U_N^{\text{char}})(U_N^{\text{char}} - U_N) = l_j(U_{RR}^j)(U_{RR}^j - U_R), \tag{3.20}$$

$$l_j(U_M^{\text{char}})(U_M^{\text{char}} - U_M) = 0, \quad j = 1, 2, \ldots, k, \tag{3.21}$$

for the states U_M^{char} and U_N^{char} at time $t + dt - 0$ (where $l_i(U)$ is an ith left eigenvector at U);

4. solve the Riemann problem with data U_M^{char} and U_N^{char} to obtain U_M^{new}, U_N^{new}, and s^{new} at time $t + dt + 0$, and revise the shock position using the average shock speed $\frac{1}{2}(s + s^{\text{new}})$.

4. Application to transitional shock waves in quadratic models. Quadratic models are nonlinear 2×2 systems of conservation laws having quadratic flux functions. Since a quadratic flux approximates a general flux near the neighborhood of a certain point, such models have been studied extensively as a prototype for the general 2×2 systems of conservation laws [41, 42, 45, 37, 38, 44, 40, 23, 24, 26, 19] just as the Burger's equation is studied in a scalar conservation law. Quadratic models are important for investigating special points, such as umblic points (isolated points at which the Jacobian is a multiple of the identity matrix). Inhomogeneous quadratic models are important as prototypes of mixed-hyperbolic/elliptic systems. For example, the existence of an umblic point or an elliptic region in some models of three-phase flow in porous media offers physical motivation for the study of Riemann problems for quadratic models [3, 43, 47, 1, 25].

Transitional shock waves are nonclassical shock waves which satisfy the viscosity profile criterion but are not admissible under the Lax criterion. In quadratic models, these waves are used to construct Riemann solutions near the elliptic region, where the Lax criterion fails to give solutions. Transitional waves are regularization-sensitive in the sense of the previous section. In this section, we apply our Riemann solver to these waves and the front tracking method for the correct numerical computation.

4.1. Transitional waves and quadratic models. We consider a general system of two conservation laws with $U = (u, v)^T$ and $F = (f, g)^T$ in (2.1), and for which f, g are quadratic functions of u and v:

$$f(u, v) = \frac{1}{2}\left(a_1 u^2 + 2b_1 uv + c_1 v^2\right) + d_1 u + e_1 v, \tag{4.1}$$

$$g(u, v) = \frac{1}{2}\left(a_2 u^2 + 2b_2 uv + c_2 v^2\right) + d_2 u + e_2 v. \tag{4.2}$$

Generally, such a model fails to be strictly hyperbolic; for example, for homogeneous models (when linear terms are absent), the characteristic speeds coincide at $U = 0$. Furthermore, an elliptic region, where characteristic speeds are not real, can occur. In the case of homogeneous models, Schaeffer and Shearer [37] found the nonsingular linear change of dependent variables that transforms (2.1) into a normal form which has two free parameters. Thus homogeneous quadratic models can be classified into four cases, I–IV, based on the structure of the integral curves through an origin in the

two parameter space. The first Riemann problem of this class, which was the symmetric case I, was solved by Shearer et al. [45]. In succession, other cases were solved [26, 38, 44, 24]. Most of these solutions were obtained under the Lax admissibility criterion. In [20], a Riemann problem is solved using the viscous profile criterion. This solution differs from that obtained using the Lax admissibility criterion, even though solutions exist and are unique for both criteria. With the viscous profile criterion, some nonlocal Lax shock waves are inadmissible; in their place, transitional waves appear in the wave patterns.

Transitional waves appear in wave groups located between the classical 1- and 2-family wave groups. A transitional shock wave from U_- to U_+ with speed s is a [illegible] in that

$$\lambda_1(U_-) < s < \lambda_2(U_-), \tag{4.3}$$

$$\lambda_1(U_+) < s < \lambda_2(U_+). \tag{4.4}$$

Thus all characteristic curves cross through the transitional shock wave. Transitional rarefaction waves appear when there exist special points at which two characteristic speeds coincide (the singularities of the rarefaction foliation). An integral curve from family 2 passes through this point and continues with an integral curve of family 1. This curve defines a transitional rarefaction wave. Transitional composite waves are also possible [23]. All of these kinds of transitional wave groups appear in the wave pattern located between the 1- and 2-wave groups.

4.2. Saddle-to-saddle connections. For a system of two conservation laws, the question of shock wave admissibility reduces to the study of planar dynamical systems (2.5) and their critical points, given a viscosity matrix $D(U)$. Let us first consider the situation where the viscosity matrix is the identity matrix. Then the eigenvalues at a critical point U_c of the dynamical system are

$$\mu_i = \lambda_i(U_c) - s, \quad i = 1, 2. \tag{4.5}$$

Therefore a shock wave satisfying the Lax admissibility condition [28] corresponds either to a connection from a repelling node to a saddle point, in the case of a 1-family wave, or to a connection from a saddle point to an attracting node, in the case of a 2-family wave. When $D(U)$ is not necessarily a multiple of the identity matrix, the node can be a spiral point, but the phase portrait is still generically stable, in that the sum of the dimensions of the unstable manifold of U_- and the stable manifold of U_+ is three.

In contrast, a transitional shock wave is a crossing discontinuity where the inequalities (4.4) hold, so that both characteristic families cross through the wave. Since $D(U)$ has a positive determinant, it follows that the eigenvalues of system (3.3) at U_- and U_+ satisfy $\mu_1(U_-)\,\mu_2(U_-) < 0$ and $\mu_1(U_+)\,\mu_2(U_+) < 0$, respectively. Therefore the viscous profile for a transitional wave connects two saddle points, and the phase portrait is generically unstable. (See Figure 3.)

In the special case of the quadratic model, there are two kinds of saddle-to-saddle connecting orbits possible: straight-line and curved. For dynamical systems that are quadratic gradients, Chicone [4] has shown that every saddle-saddle connection has a straight-line orbit. In [23], analytic formulae for Hugoniot curves and various bifurcation loci were given for quadratic models; in particular, all straight-line saddle-to-saddle connections were identified. This work has been extended [31, 21]. In section 4.4, we compare our numerical results with the answers obtained by this analytic work.

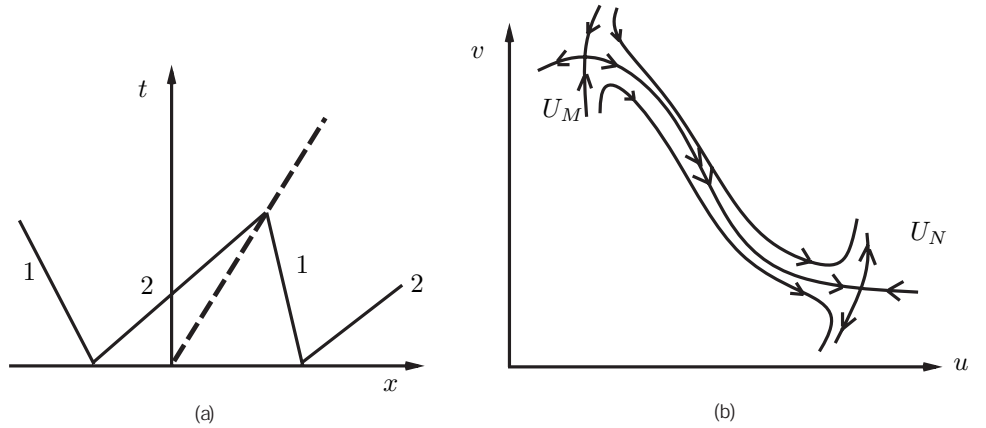


FIG. 3. *This picture shows a transitional shock wave in* (a) *the* (x,t) *plane and* (b) *the* (u,v) *plane.*

4.3. Numerical Riemann solver near a transitional shock wave. In this section we apply the algorithm of section 3 in the case $m = 2$. As illustrated in Figure 3, the solution of such a Riemann problem comprises a 1-wave from U_L to U_M, a transitional shock wave from U_M to U_N, and a 2-wave from U_N to U_R. Given U_L and U_R, we shall calculate U_M, U_N, and the speed s of the transitional wave.

Since U_M and U_N lie on the wave curves from U_L and U_R, they can be expressed as

$$U_M = \mathcal{W}_1^+(U_L, \varepsilon_L) \approx U_L + \varepsilon_L\, r_1(U_L), \tag{4.6}$$

$$U_N = \mathcal{W}_2^-(U_R, \varepsilon_R) \approx U_R + \varepsilon_R\, r_2(U_R), \tag{4.7}$$

where $\mathcal{W}_1^+$ (resp., $\mathcal{W}_2^-$) parameterizes the forward 1-family (resp., backward 2-family) wave curve in terms of the parameter ε_L (resp., ε_R). As indicated, we approximate the wave curves by straight lines, since U_M and U_N are close to U_L and U_R, but this approximation can easily be avoided. The states U_M and U_N are also saddle points of the dynamical system 3.1.

To calculate the saddle-to-saddle connection from U_M to U_N, we employ the multiple shooting method, as in section 3. The states U_0^* at $\xi = -T_1$ and U_n^* at $\xi = T_2$ can be approximated as

$$U_0^* \approx U_M + \varepsilon_M\, \tilde{r}_2(U_M), \tag{4.8}$$

$$U_n^* \approx U_N + \varepsilon_N\, \tilde{r}_1(U_N). \tag{4.9}$$

Then we have $2n + 3$ unknown quantities,$(\varepsilon_L,\ \varepsilon_R,\ \varepsilon_M,\ \varepsilon_N,\ s,\ U_1^*,\ \ldots,\ U_{n-1}^*)$, and we have $2n + 3$ equations to solve: $2n$ matching conditions, two Rankine–Hugoniot conditions, and a phase condition.

Figure 4 shows the convergence of a Riemann solver for examples of Riemann problems for a quadratic model with an elliptic region, whose coefficients are given as $a_1 = -1$, $b_1 = 0$, $c_1 = 1$, $d_1 = 0$, $e_1 = 0.23$, $a_2 = 0$, $b_2 = 1$, $c_2 = 0$, $d_2 = -0.23$, $e_2 = 0$ in (4.2). In this example, the saddle-to-saddle connections are curved connections. As described in section 3, our iterative solver adopts the line search technique to obtain a global convergence for different initial guesses. We use the straight line connection from U_L to U_R as an initial guess; however, a different initial guess could make a faster

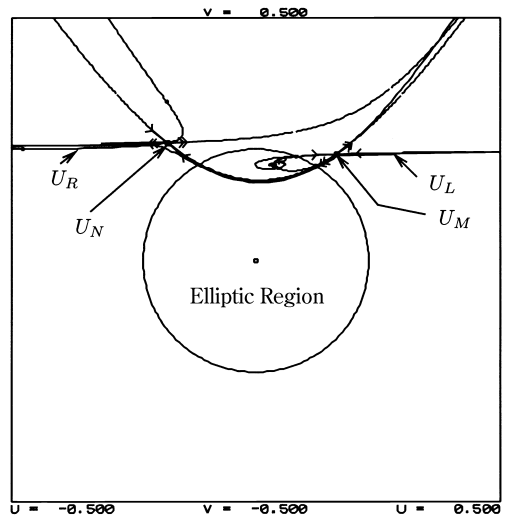


FIG. 4. *The (u, v)-plane. The region inside of the circle is the elliptic region, where the characteristic speeds are complex. Figure* (a) *shows an example of a Riemann solution containing a transitional wave. Figure* (b) *shows several steps in the iteration for finding the solution. The initial guess is a straight line.*

convergence. Figure 4(a) is an exact solution obtained from the Riemann Problem Package of Isaacson, Marchesin, and Plohr [22] based on an analytic formulation for quadratic models, and the (b) is the computed result of the Riemann solver, which is exactly same as that in (a).

4.4. Computational results with front tracking and comparison to other methods. Equipped with the Riemann solver of the previous section, we are in a posi-

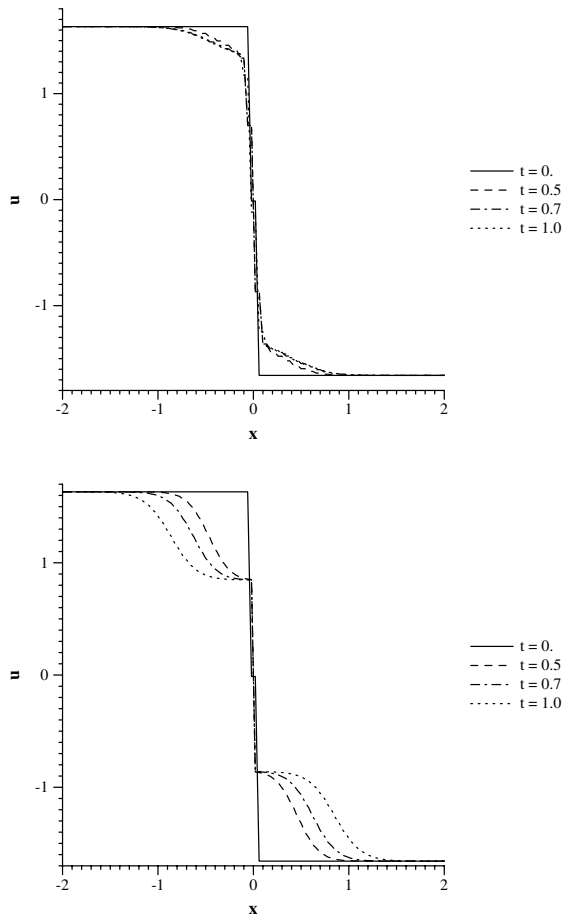


FIG. 5. *The case* II *model solved using the LF method without (top) and with (bottom) front tracking.*

tion to track a transitional shock wave using the front tracking method of section 3.1. We have implemented a front tracking method for pairs of conservation laws with quadratic flux functions of (4.2). In the following we describe some of the results of testing our tracking method. We also make comparisons to some commonly used shock capturing methods. We apply these methods to a homogeneous model in case II and to an inhomogeneous model for which the homogeneous quadratic part belongs to case I. In the first case, transitional shock waves have straight connections; in the latter case, we focus on curved connections.

For the homogeneous case II model, the coefficients are $a_1 = 0.5$, $b_1 = 0.2$, $c_1 = 1$, $d_1 = e_1 = 0$, $a_2 = 0.2$, $b_2 = 1$, $c_2 = d_2 = e_2 = 0$; the viscosity matrix is taken to be constant and diagonal, with diagonal entries 1 and 0.3. We solve the initial-value problem with Riemann data $U_L = (1.632, 1.278)$ and $U_R = (-1.658, -1.269)$. The exact solution comprises a 1-shock wave, a transitional wave, and a 2-shock wave [20]; the connecting orbit for the transitional wave lies along a line through the origin (a viscosity line) that can be determined analytically [23]. This solution is to be contrasted with the solution when $D(U)$ is a multiple of the identity matrix, or, more generally, a function of $F'(U)$: then the transitional wave lies along a different line

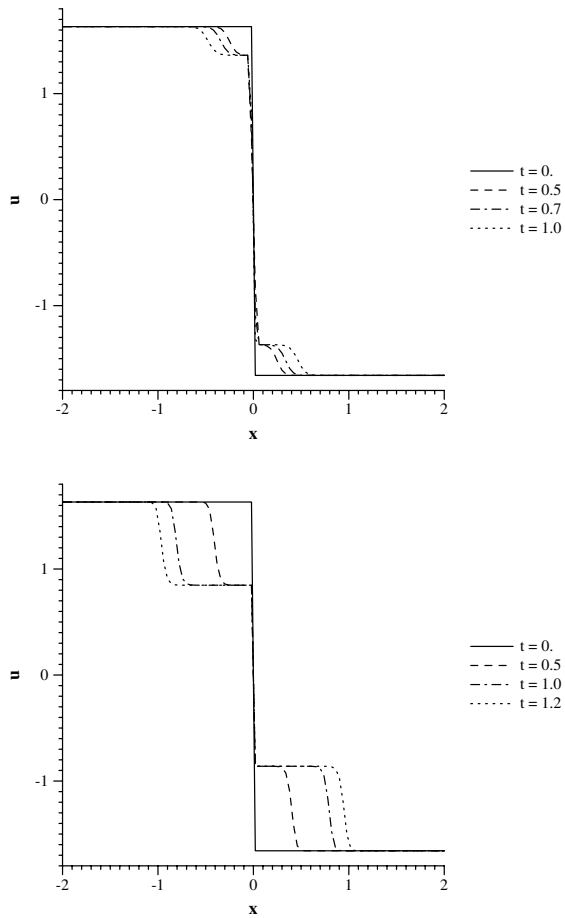


FIG. 6. *The case* II *model solved using the BCT method without (top) and with (bottom) front tracking.*

through the origin (a secondary bifurcation line).

Figures 5 and 6 show the results of using the Lax–Friedrichs (LF) [27] and Bell–Colella–Trangenstein (BCT) [2] methods with and without front tracking for comparison. Graphs of u plotted versus x for several time steps are shown superimposed. Graphs of v are omitted to save space but are similar to those of u. For both methods, the calculated solution shows the 1- and 2-family shock waves moving away from the tracked transitional wave; of course the untracked waves are sharper for the BCT method. Naturally, for these schemes, the transitional wave is less sharply defined, but more significantly, the limiting states of this wave are very different, leading to 1- and 2-family waves that are very different as well.

To compare these results with the exact solution, we have plotted the computed solutions in state space; see Figure 7. In the figure there are two sets of dashed lines: short-dashed lines are secondary bifurcation lines, whereas long-dashed lines are viscosity lines corresponding to the chosen viscosity matrix. These lines have been computed using the formulae in [23]. For the front tracking solutions, the orbit for the transitional wave lies along a viscosity line and thus approximates the exact solution well. For the untracked solutions, however, the orbit lies along a bifurcation

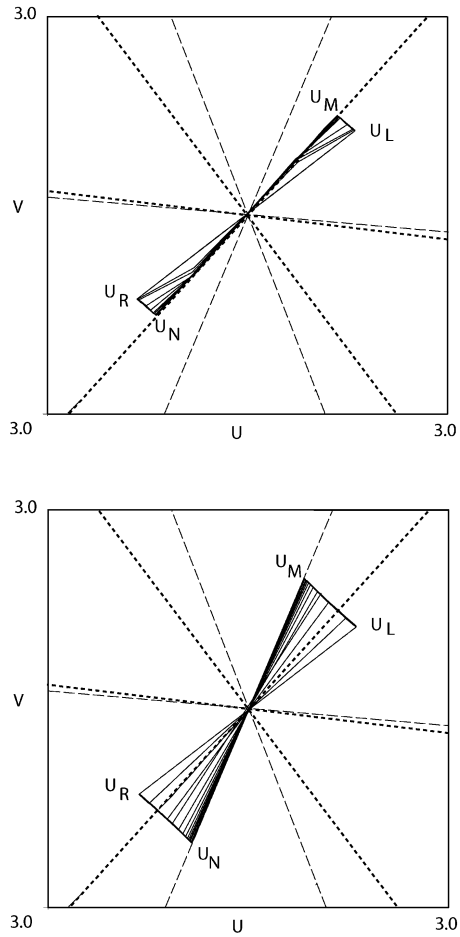


FIG. 7. *The case* II *model solved using the BCT method as shown in the* (u, v)*-plane. The top figure is computed without front tracking, and the bottom with front tracking.*

line; this is consistent with the fact that the numerical viscosity matrices for these schemes (i.e., the viscosity matrices for the corresponding modified equations) are functions of $F'(U)$.

The next set of figures shows the test results for the case I model. The coefficients for this model are $a_1 = -1$, $b_1 = 0$, $c_1 = 1$, $d_1 = 0$, $e_1 = 0.23$, $a_2 = 0$, $b_2 = 1$, $c_2 = 0$, $d_2 = -0.23$, $e_2 = 0$; the viscosity matrix is taken to be the identity matrix. This model has a elliptic region, the boundary of which is a circle shown in the state space Figure 4. The Riemann initial values are $U_L = (0.273, 0.182)$ and $U_R = (-0.246, 0.310)$. The exact solution, as computed using the Riemann Problem Package, comprises a 1-rarefaction wave, a transitional wave with a curved connecting orbit, and a 2-shock wave.

The results of using the front tracking method are shown in Figures 8, and 9. The underlying finite-difference schemes are the LF and the Lax–Wendroff (LW) [30] schemes; the LW scheme is modified by adding artificial viscosity [35]. For comparison, the results without front tracking are shown together. In the state space figures are drawn not only the solutions for several time steps but also the orbits of the dynami-

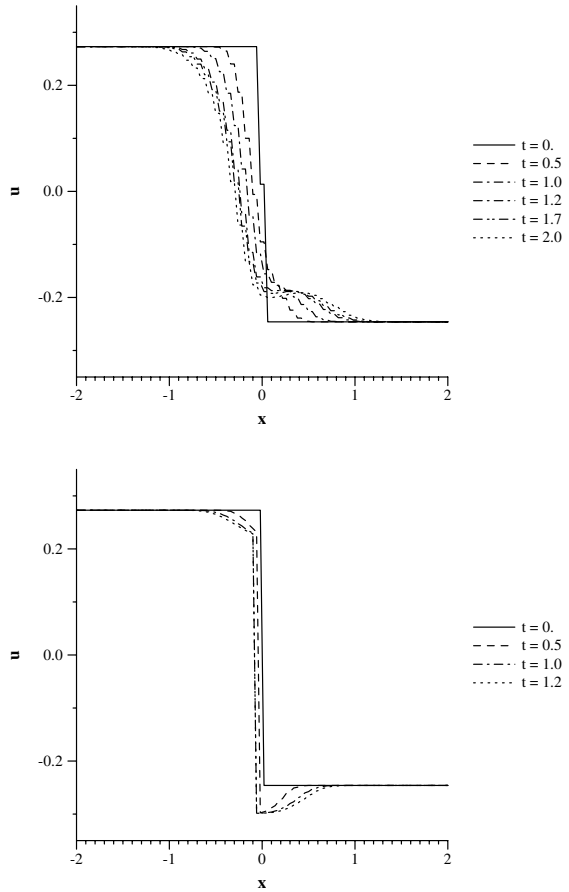


FIG. 8. *The case* I *model solved using the LF method without (top) and with (bottom) front tracking.*

cal system corresponding to the transitional shock wave (particularly the connecting orbit), the Hugoniot locus based at the point U_L, and the circle bounding the elliptic region. As seen in the state-space figures, the front tracking solution approximates the exact solution well, whereas the untracked solutions are significantly different.

5. Application to deflagration waves in combustion. In this section, we apply our algorithm for regularization-sensitive waves to deflagration waves appearing in combustion. Combustion is one of the most complex phenomena in nature, and it is extremely difficult to model numerically. Our standard approach is to assume that the reaction zone is infinitely thin. In this approach, all microscopic dissipative terms, such as viscosity, diffusion, and heat conductivity, which are dominated in the reaction zone are neglected. This assumption is motivated by their small sizes compared to a typical flow size, and by the short time for the chemical reactions compared to a typical flow time. Thus, combustion waves are treated as discontinuities in the resulting governing equations, which comprise a system of hyperbolic conservation laws. These discontinuities should be required to have viscous profiles, as in the previous section. The neglected microscopic dissipative terms play the the role of the vanishing viscosity matrix.

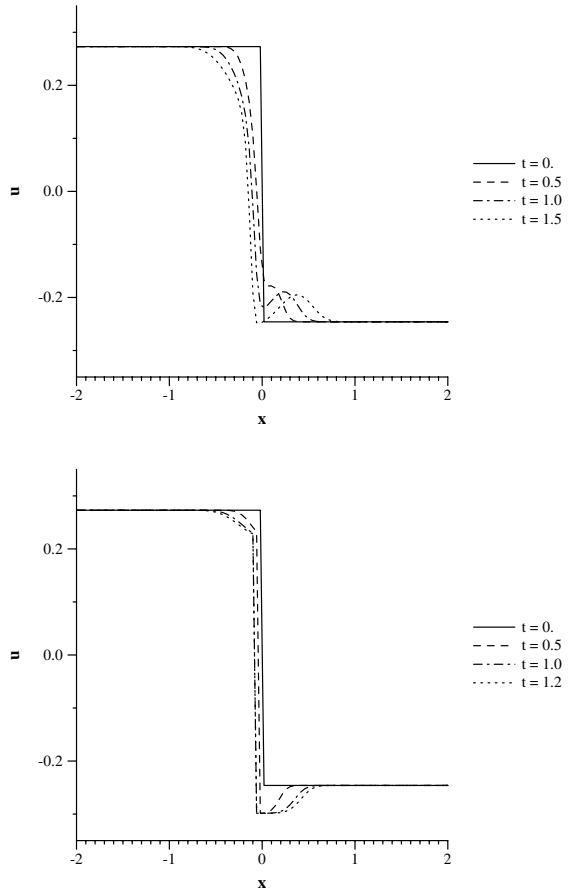


FIG. 9. *The case* I *model solved using the LW method without (top) and with (bottom) front tracking.*

In combustion, there are two kinds of waves in addition to standard gas dynamics waves. One is a detonation wave, and the other is a deflagration wave. Detonation waves have shock-like structures and are admissible under the Lax criterion. In contrast, deflagration waves violate the Lax criterion; i.e., fewer characteristics impinge on the wave than Lax shocks do. Thus its structure and propagation speed are sensitively dependent on the internal wave structure, which is, in turn, influenced by viscosity, diffusion, and heat conductivity.

This internal structure dependency implies difficulties in numerical computations. Excess diffusion can cause significant errors in calculating combustion wave speeds [7], and the naive use of standard numerical methods such as shock capturing schemes produces spurious results in the case of deflagration waves because of artificial viscosities.

In this section, we propose an algorithm for solving Riemann problems near a weak deflagration wave, the side states and speed of which depend on dissipation mechanisms.

5.1. Governing equations and viscous profiles of combustion waves. We consider the Navier–Stokes equations for a reacting gas in one space dimension. We assume the simple chemical reaction of one reactant that is converted to the product by a one-step exothermic reaction: $A \to B + \text{heat}$. Therefore we need only one variable

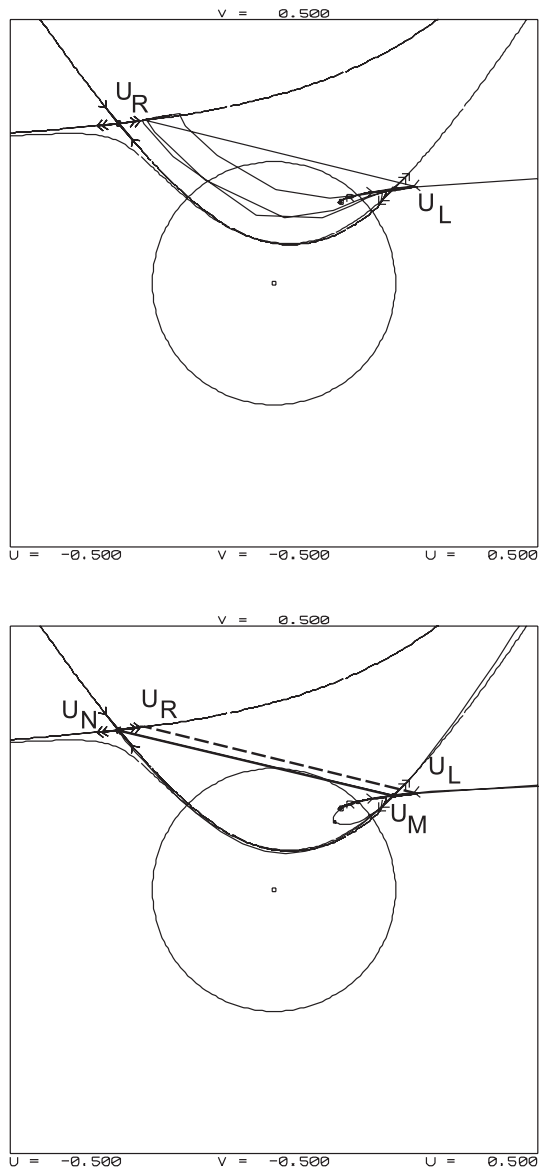


FIG. 10. *The case* I *model solved using the LW methods without (top) and with (bottom) front tracking, as shown in the* (u, v)*-plane.*

Y, which is the reactant mass fraction. More complicated reactions have a similar structure but more variables [13]. The governing equations are

$$\rho_t + (\rho u)_x = 0,$$

$$(\rho u)_t + (\rho u^2 + p)_x = (\mu u_x)_x,$$

$$\left[\rho\left(\frac{u^2}{2} + e\right)\right]_t + \left[\rho u\left(\frac{u^2}{2} + e\right) + pu\right]_x = (\lambda T_x)_x + (q\rho D Y_x)_x + (\mu u u_x)_x,$$

$$(\rho Y)_t + (\rho u Y)_x = (\rho D Y_x)_x - k\rho Y \phi(T). \tag{5.1}$$

Here, the variables ρ, u, p, e, T, and Y are respectively the mass density, velocity, pressure, specific energy, temperature, and reactant mass fraction of the gas. This last variable Y ranges from 0 to 1; $Y = 0$ means that gas is totally burned, and $Y = 1$ corresponds to an unburned gas. The constants μ, λ, and D are the viscosity, heat conductivity, and diffusion constants, respectively. Assuming a polytropic equation state, the specific energy e is given by

$$e = c_v T + qY = \frac{p}{\rho(\gamma - 1)} + qY, \tag{5.2}$$

where c_v is the specific heat at constant volume and q is the heat release parameter. For an exothermic reaction, q is positive, and the gas undergoes a chemical reaction in which energy q is released. In (5.1) we have also assumed a reaction rate

$$r = -k\rho Y \phi(T). \tag{5.3}$$

The function $\phi(T)$ is given by Arrhenius term of the form

$$\phi(T) = \begin{cases} \exp(-E_a/RT) & \text{if } T > T_i, \\ 0 & \text{otherwise,} \end{cases} \tag{5.4}$$

where E_a is an activation energy constant. To avoid the well-known cold boundary difficulty, we adapt the common mathematical remedy such that $\phi(T)$ vanishes identically for T below a certain ignition temperature T_i. The system (5.1) is a standard set of equations for a reactive gas flow [53].

If we neglect all dissipative effects, then (5.1) reduces to a system of hyperbolic conservation laws with a source term of a reaction rate, $-k\rho Y\phi(T)$, which is called the ZND model (named after Zeldovich, Neumann, and Döring).

A thorough analysis shows that the downstream gas velocity, relative to the wave, exceeds the sound velocity for weak detonations and strong deflagrations and is less than the sound velocity for strong detonations and weak deflagrations [9, 6]. These relations are known as [illegible]. From the the relations between characteristic speeds and speeds of deflagration waves, we can see that a weak deflagration wave is underdetermined.

Detonation and deflagration waves are traveling wave solutions of the Navier–Stokes equations (5.1) connecting an unburned state at $x = -\infty$ to burned state at $x = \infty$. To be admissible as physical solutions of corresponding hyperbolic conservation laws (5.5), these traveling waves must have viscous profiles, i.e., be the limits of solutions of (5.1) as the dissipation coefficients go to zero. The existence of combustion waves of system (5.1) has been analyzed by several authors under the simplifying assumptions such as certain fixed Prandtl and Lewis number [18] or very large activation energy [32]. However, the convergence of these solutions to solutions of the ZND model is not discussed by these authors. Recently, Wagner [52, 51] obtained the existence of viscous profiles for detonation and deflagration waves as ZND-limits under the assumption of certain fixed Prandtl and Lewis number. Gasser and Szmolyan [12] proved the same, using a geometric singular perturbation theory. In this section, we will derive the corresponding dynamical system for combustion waves and will show that the weak deflagration wave is regularization-sensitive in the sense described in previous sections.

5.2. Dynamical systems for combustion waves and linearization near an equilibrium point. Let us call m a conserved mass, i.e.,

$$(u - s)\rho = (u_0 - s)\rho_0 = -m, \tag{5.5}$$

and define a new variable

$$Z = Y + \rho D \frac{Y_x}{m}, \tag{5.6}$$

so that the third equation in (5.1) can be written as a first-order ordinary differential equation since Y_x vanishes at $x = \pm\infty$,

$$Z_1 = Y_1, \qquad Z_0 = Y_0. \tag{5.7}$$

Then we could obtain the following ordinary differential equations for the viscous profile of a combustion wave:

$$\begin{aligned}
\mu u_x &= -m(u-u_0) - mR\left(\frac{T}{u-s} - \frac{T_0}{u_0-s}\right),\\
\lambda T_x &= -mc_p(T-T_0) + \frac{m}{2}(u-u_0)^2 - mq(Z-Z_0)\\
&\quad + mR(u-s)\left(\frac{T}{u-s} - \frac{T_0}{u_0-s}\right),\\
DY_x &= (u-s)(Y-Z),\\
Z_x &= -k\frac{Y\phi(T)}{u-s},
\end{aligned} \tag{5.8}$$

after using the relation $c_p = c_v + R$.

To understand the solution of system (5.8), we first examine its equilibria. Let the state variables be denoted by $V = (u, T, Y, Z)$, setting the right-hand sides of (5.8) to zero; we find that the initial state V_0 and new state V_1 are related by the jump condition discussed in section 5.1. The ordinary differential equations for a shock profile in standard gas-dynamics correspond to the first two equations of (5.8) when q is zero, and it is easily shown that the equilibrium point is a saddle for a subsonic flow and a node (repeller or attractor) for a supersonic flow [15, 34]. A fast shock wave corresponds to a profile connecting an attractor to a saddle, and a slow shock wave to a profile from a saddle to a repeller. Recall that we are integrating from $x = \infty$ backwards. Thus the flow in the dynamical system is reversed.

A similar analysis can be performed for combustion waves. Let us linearize (5.8) near an equilibrium point V_c. Then, (5.8) may be expanded to first order to yield

$$\frac{d\tilde{V}}{dx} = \mathbf{J}\tilde{V}, \tag{5.9}$$

where

$$\tilde{V} = (\tilde{u}, \tilde{T}, \tilde{Y}, \tilde{Z}) = (u-u_c, T-T_c, Y-Y_c, Z-Z_c), \tag{5.10}$$

and $\mathbf{A}$ and $\mathbf{B}$ forming the Jacobian matrix $\mathbf{J} = \mathbf{B}^{-1}\mathbf{A}$ are

$$\mathbf{A} = \begin{bmatrix}
-m + m\dfrac{RT}{(u-s)^2} & -m\dfrac{R}{u-s} & 0 & 0\\
m(u-u_0) - m\dfrac{RT_0}{u_0-s} & -mc_v & 0 & -mq\\
Y-Z & 0 & u-s & -(u-s)\\
k\dfrac{Y\phi(T)}{(u-s)^2} & -k\dfrac{Y\phi'(T)}{u-s} & -k\dfrac{\phi(T)}{u-s} & 0
\end{bmatrix}, \tag{5.11}$$

$$\mathbf{B} = \begin{bmatrix} \mu & 0 & 0 & 0 \\ 0 & \lambda & 0 & 0 \\ 0 & 0 & D & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.12}$$

All matrix elements are evaluated at the equilibrium point.

Let us assume that the equilibrium point V_0 is an unburned state where $Y_0 = 1$ and $\phi(T_0) = 0$ since $T_0 < T_i$, and that the other critical point V_1 is a totally burned state, with $Y_1 = 0$. After some algebraic manipulation, we can find the signs of the eigenvalues at V_0 and V_1. At V_1, the eigenvalues μ_1, μ_2, μ_3, and μ_4 of $\mathbf{J}$ satisfy the following relations:

$$\mu_1 \cdot \mu_2 = -\frac{k\phi(T_1)}{D} < 0, \tag{5.13}$$

$$\mu_1 + \mu_2 = \frac{u_1 - s}{D}, \tag{5.14}$$

$$\begin{aligned} \mu_3 \cdot \mu_4 &= \frac{c_v \rho_1^2}{\lambda\mu}[(u-s)^2 - c_1^2] \\ &= \frac{c_v m^2}{\lambda\mu}\left(1 - \frac{1}{M_1^2}\right), \end{aligned} \tag{5.15}$$

$$\begin{aligned} \mu_3 + \mu_4 &= \frac{1}{\mu}\left[-m + \frac{mRT_1}{(u-s)^2}\right] - \frac{mc_v}{\lambda} \\ &= -\frac{m}{\mu}\left(1 - \frac{1}{\gamma M_1^2}\right) - \frac{mc_v}{\lambda}, \end{aligned} \tag{5.16}$$

where $M^2 = u^2/\gamma RT$ is the square of the Mach number. At an unburned state V_0, the analysis is simpler since $\phi(T_0) = 0$, and $\phi'(T_0) = 0$; the eigenvalues satisfy

$$\mu_1 = \frac{u_0 - s}{D}, \quad \mu_2 = 0, \tag{5.17}$$

$$\mu_3 \cdot \mu_4 = \frac{c_v m^2}{\lambda\mu}\left(1 - \frac{1}{M_0^2}\right). \tag{5.18}$$

As shown in section 5.1, the flow is subsonic both behind and ahead of a weak deflagration wave. Therefore two eigenvalues are positive and two eigenvalues are negative at both V_1 and V_0. In contrast, for a strong detonation wave, which is subsonic behind and supersonic ahead, there are three negative eigenvalues at V_0 and two positive eigenvalues at V_1.

To understand better the relative sizes and importance of variables terms in (5.8), we nondimensionalize them using some reference state V^*. Each nondimensional quantity, denoted by a tilde $\sim$, is obtained by dividing the corresponding dimensional quantity by its reference value. For a reference state, we choose a state from the ozone

decomposition reaction data in [7] and [33]. These values are summarized below (all data are CGS units):

$$R = 8.3143 \,\mathrm{x}\, 10^7,$$

$$\gamma = 1.4,$$

$$\rho_{\mathrm{atm}} = 1.29 \times 10^{-3},$$

$$p_{\mathrm{atm}} = 1.0135 \times 10^6,$$

$$\rho^* = 0.931\, \rho_{\mathrm{atm}},$$

$$p^* = 0.821\, p_{\mathrm{atm}},$$

$$T^* = \frac{\alpha p^*}{R\rho^*} \approx 267^\circ\mathrm{K},$$

where $\alpha = 36$ is a mole fraction.

5.3. Numerical Riemann solver near a deflagration wave. In this section, we apply the algorithm described in section 3 to solving a Riemann problem near a deflagration wave. The solution is essential for tracking a deflagration wave. As we have shown in section 5.1, a deflagration wave has fewer characteristics impinging on it; therefore, we need additional information to determine the evolution of the wave. Recently, Hilditch and Colella [17] tracked a deflagration flame, taking an experimental flame speed law for the additional information. The same strategy has been used also by Smiljanovski and Klein [48] except for a different tracking procedure. These authors all utilized the Riemann solution procedure of Teng, Chorin, and Liu [50] and used a phenomenological flame speed law to overcome the underdeterminacy. Our approach differs in that we construct the Riemann solution from the exact information coming from the internal structure of the flame. In particular, we do not use a phenomenological flame speed law.

Solving a Riemann problem near a deflagration wave means finding the flame speed s and the two states U_M and U_N in Figure 11 for given Riemann initial data U_L and U_R. We consider the dynamical system (5.8). For simplicity of notation, we drop the tildes from the scaled quantities. Then U_M and U_N are equilibrium points of a dynamical system

$$U' = G(U, s, U_N) = G(U, s, \varepsilon_R), \tag{5.19}$$

and these equilibria are connected by an orbit as we described in section 5.1.

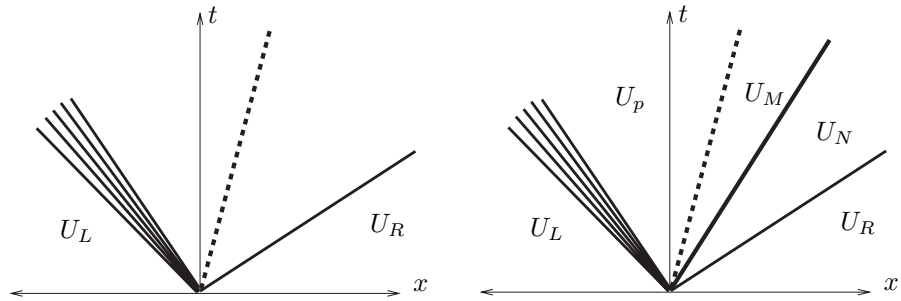
Since U_M is reached from U_L by a combination of weak 1- and 2-waves, it can be expressed as

$$U_M = \mathcal{W}_{1,2}^+(U_L, \varepsilon_L, \varepsilon_M) \approx U_L + \varepsilon_L\, r_1(U_L) + \varepsilon_M\, r_2(U_p). \tag{5.20}$$

Similarly, since U_N is reached from U_R by a weak 3-wave,

$$U_N = \mathcal{W}_3^-(U_R, \varepsilon_R) \approx U_R + \varepsilon_R\, r_3(U_R). \tag{5.21}$$

The algorithm for calculating a flame is the same as for a transitional shock wave except the number of parameters and the dimensions of stable and unstable manifolds.



(a) Gas dynamics (b) Weak deflagration

FIG. 11. *Riemann solution for gas dynamics (a), and for reactive gas with a right-facing weak deflagration (b). The waves between U_L and U_p, and between U_N and U_R, can be shock waves or rarefaction waves.*

The dimensions of the stable manifold at U_N and that of the unstable manifold at U_M are both two, as we have seen in section 5.2. The states U_0^* and U_n^* are approximated here as

$$U_0^* \approx U_M + \varepsilon_3 \tilde{r}_3(U_M) + \varepsilon_4 \tilde{r}_4(U_M), \tag{5.22}$$

$$U_n^* \approx U_N + \varepsilon_1 \tilde{r}_1(U_N) + \varepsilon_2 \tilde{r}_2(U_N), \tag{5.23}$$

where $\tilde{r}_i(U)$ is the ith eigenvector of $G'(U)$. Dividing the interval between U_0^* and U_n^* into n subintervals gives $4n-4$ unknown quantities, which are $U_1, \ldots, U_{n-1}$; thus we have a total of $4n+4$ unknowns, namely,

$$(\varepsilon_L, \varepsilon_M, \varepsilon_R, \varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4, s, U_1^*, \ldots, U_{n-1}^*),$$

which require $4n+4$ equations. There are $4n$ matching conditions in the multiple shooting algorithm, three equations from the Rankine–Hugoniot relations, and one phase condition. Therefore we have a complete system of equations to solve.

5.4. Computational results with the Riemann solver. For testing the algorithm of section 5.3, we simplify (5.8) by neglecting the species diffusion effect; i.e., we set $D = 0$. Then $Y = Z$, and system (5.8) reduces to a 3×3 system of ordinary differential equations. The equations for U_M and U_N are still (5.20) and (5.21), but those for U_0 and U_n, (5.22) and (5.23), simplify to

$$U_0^* \approx U_M + \varepsilon_3 \tilde{r}_3(U_M), \tag{5.24}$$

$$U_n^* \approx U_N + \varepsilon_1 \tilde{r}_1(U_N) + \varepsilon_2 \tilde{r}_2(U_N). \tag{5.25}$$

This simplification leads to the system of $3n+4$ equations to solve for $3n+4$ unknown variables.

Figures 12 and 13 show the computational results for the solutions of Riemann problems near a deflagration wave. The values of the coefficients are taken from the ozone decomposition reaction data in [7] and [33] and are scaled for nondimensionalization using $\gamma = 1.4, T_i = 1.9, q = 2, E_a = 1.41, \mu = 0.4, \lambda = 0.3$. In this example, the initial data is $U_L = (0.26, -0.15, 1.57)$ and $U_R = (1.501, 0.45, 1.7)$, where $U = (\rho, u, p)$. The computed left and right states of a deflagration wave

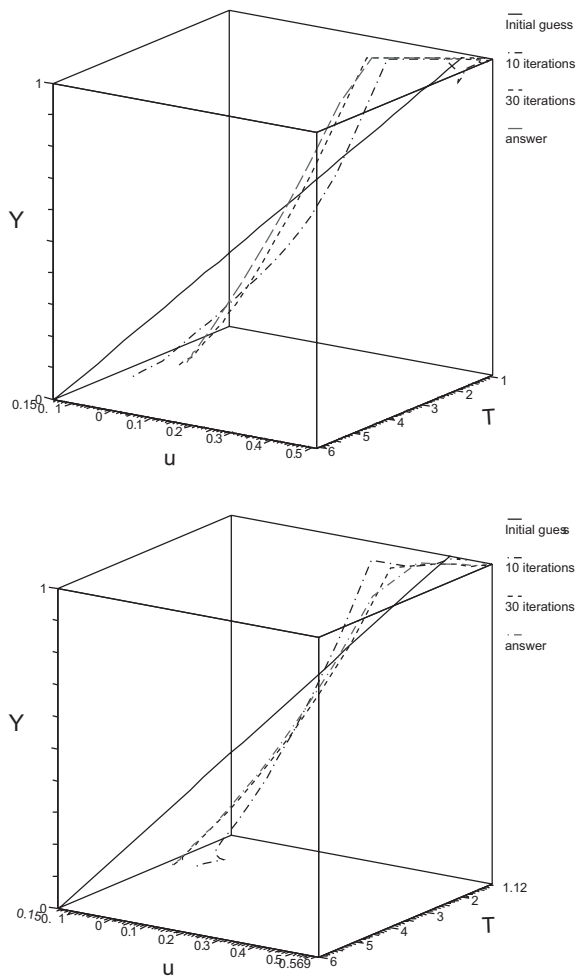


FIG. 12. *The top figure shows the convergence of the Riemann solver when* $\mu = 0.4$, $\lambda = 0.3$. *The Riemann initial data are* $U_L = (0.26, -0.15, 1.57)$ *and* $U_R = (1.501, 0.45, 1.7)$, *and the initial guess is a straight line connecting initial states. The final answer is a viscous profile connecting* U_M *and* U_N. *The bottom figure is when* $\mu = 0.5$, $\lambda = 0.4$; $U_M = (0.514279, -0.146324, 1.567224)$ *and* $U_N = (1.655849, 0.603822, 1.987245)$.

are $U_M = (0.501913, -0.092041, 1.526870)$ and $U_N = (1.538359, 0.489946, 1.770701)$. This result shows that the density and pressure decrease and the temperature increases ($T_{U_M} = 3.042099$) behind a deflagration wave. The temperature at U_N also increases towards temperature T_i. The flame speed obtained is $s = 0.773140$. In contrast, the experimental flame speed law used in the computations of Hilditch and Colella [17], Chorin [5], and Teng, Chorin, and Liu [50] is

$$s = u_0 + a \left(\frac{p_0}{\rho_0} \right)^b, \tag{5.26}$$

where $a = 0.1$ and $b = 0.2$ are determined by fitting experimental data to this equation. This law gives $s = 0.615809$.

In the next example, we change the values of viscosity and heat conductivity to

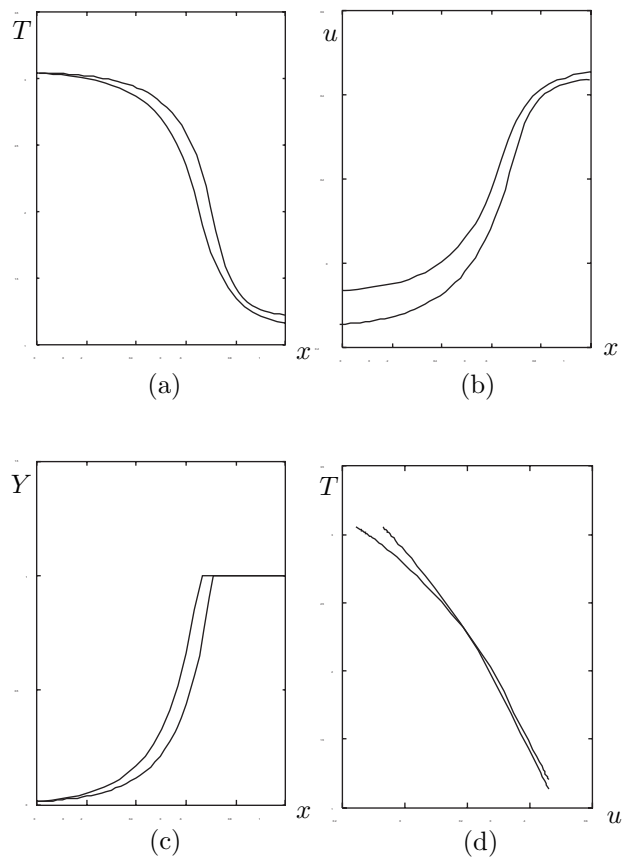


FIG. 13. *These figures show the different structure of the internal layer ($x = 0$ to $x = 1$) of a deflagration wave when μ and λ are given differently: ($\mu = 0.4$, $\lambda = 0.3$) and ($\mu = 0.5$, $\lambda = 0.4$).* (a) *velocity u versus x,* (b) *temperature T versus x,* (c) *mass fraction Y versus x, and* (d) *T versus u.*

$\mu = 0.5$ and $\lambda = 0.4$ and compute the Riemann solution, obtaining $U_M = (0.514279, -0.146324, 1.567224)$, $U_N = (1.655849, 0.603822, 1.987245)$, and $s = 0.941279$. This different solution illustrates the influence of the choice of regularization on wave propagation. Our Riemann solver has found a solution which correctly incorporates the information coming from the internal structure. Figure 12 shows converging approximations and the final computed viscous profiles in (u, T, Y)-space when initial guesses are taken as straight lines between U_L and U_R. Figure 13 illustrates the internal structures of waves resolved in our Riemann solver, drawn together with different viscosity and heat conductivity to demonstrate the effect of these internal coefficients.

6. Conclusion. The main purpose of this project has been to understand the mathematical structure of regularization-sensitive shock waves and to develop a numerical method that accurately captures their behaviors. We have devised a Riemann solver that directly accounts for the internal structure of such discontinuities, and have developed a front tracking algorithm based on this Riemann solver. By considering two applications of this method, we have demonstrated that our method accurately computes regularization-sensitive shock waves, in contrast to commonly used numerical methods, which fail to give correct solutions.

Our future plans include the extension of this algorithm to multiple space dimensions and its application to other physical systems. For the multidimensional algorithm, bifurcation cases must be handled; for example, regularization-sensitive shock waves can merge into classical shock waves. Another physical application is to multiphase flow through porous media, which is known to have regularization-sensitive waves. The physical meaning of these waves in physical applications remains a question that requires deeper understanding. We hope that our Riemann solver and front tracking algorithm will contribute to this understanding.

Acknowledgments. The author is grateful to Prof. Bradley Plohr at Los Alamos National Laboratories for his encouragement and for permisions to use his Riemann Problem Package , in exploring this interesting problem. Thanks also to Prof. James Glimm at Stony Brook University for his support and concern for this work.

REFERENCES

[1] A. Azevedo and D. Marchesin, *Multiple viscous profile Riemann solutions in mixed elliptic-hyperbolic models for flow in porous media*, in Hyperbolic Equations that Change Type, B. Keyfitz and M. Shearer, eds., IMA Vol. Math. Appl. 27, Springer-Verlag, New York, Heidelberg, Berlin, 1990, pp. 1–17.

[2] J. Bell, P. Colella, and J. Trangenstein, *Higher order Godunov methods for general systems of hyperbolic conservation laws*, J. Comput. Phys., 82 (1989), pp. 362–397.

[3] J. B. Bell, J. A. Trangenstein, and G. R. Shubin, *Conservation laws of mixed type describing three-phase flow in porous media*, SIAM J. Appl. Math., 46 (1986), pp. 1000–1017.

[4] C. Chicone, *Quadratic gradients on the plane are generically Morse-Smale*, J. Differential Equations, 33 (1979), pp. 159–166.

[5] A. Chorin, *Random choice methods with applications to reacting gas flow*, J. Comput. Phys., 25 (1977), pp. 253–272.

[6] A. Chorin and J. Marsden, *A Mathematical Introduction to Fluid Mechanics*, Springer-Verlag, New York, Heidelberg, Berlin, 1979.

[7] P. Colella, A. Majda, and V. Roytburd, *Theoretical and numerical structure for reacting shock waves*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 1059–1080.

[8] C. Conley and J. Smoller, *Viscosity matrices for two-dimensional nonlinear hyperbolic systems*, Comm. Pure Appl. Math., 23 (1970), pp. 867–884.

[9] R. Courant and K. Friedrichs, *Supersonic Flow and Shock Waves*, Springer-Verlag, New York, 1976.

[10] M. J. Friedman and E. J. Doedel, *Numerical computation and continuation of invariant manifolds connecting fixed points*, SIAM J. Numer. Anal., 28 (1991), pp. 789–808.

[11] H. Freistuhler and T.-P. Liu, *Nonlinear stability of overcompressive shock waves in a rotationally invariant system of viscous conservation laws*, Comm. Math. Phys., 153 (1993), pp. 147–158.

[12] I. Gasser and P. Szmolyan, *A geometric singular perturbation analysis of detonation and deflagration waves*, SIAM J. Math. Anal., 24 (1993), pp. 968–986.

[13] I. Gasser and P. Szmolyan, *Detonation and deflagration waves with multistep reaction schemes*, SIAM J. Appl. Math., 55 (1995), pp. 175–191.

[14] I. Gelfand, *Some problems in the theory of quasi-linear equations*, Uspekhi Math. Nauk, 14 (1959), pp. 87–158.

[15] D. Gilbarg, *The existence and limit behavior of the one-dimensional shock layer*, Amer. J. Math., 73 (1951), pp. 256–274.

[16] J. Glimm, *The continuous structure of discontinuities*, in PDEs and Continuum Models of Phase Transitions, M. Rascle, D. Serre, and M. Slemrod, eds., Lecture Notes in Phys. 344, Springer-Verlag, New York, Heidelberg, Berlin, 1989, pp. 177–186.

[17] J. Hilditch and P. Colella, *A front tracking method for compressible flames in one dimension*, SIAM J. Sci. Stat. Comput., 16 (1995), pp. 755–772.

[18] J. O. Hirschfelder and C. C. Curtiss, *Theory of detonations*. i. *Irreversible unimolecular reaction*, J. Chem. Phys., 28 (1958), pp. 1130–1147.

[19] H. Holden, *On the Riemann problem for a prototype of a mixed type conservation law*, Comm. Pure Appl. Math., 40 (1987), pp. 229–264.

[20] J. Hurley, *Effects of Viscous Terms on Solutions of Riemann Problems*, Ph.D. thesis, State University of New York at Stony Brook, Stony Brook, NY, 1995.

[21] J. Hurley and B. Plohr, *Some effects of viscous terms on Riemann problem solutions*, Mat. Contemp., 8 (1995), pp. 203–224.

[22] E. Isaacson, D. Marchesin, and B. Plohr, *Riemann Problem Package*, unpublished software, 1988.

[23] E. L. Isaacson, D. Marchesin, and B. J. Plohr, *Transitional waves for conservation laws*, SIAM J. Math. Anal., 21 (1990), pp. 837–866.

[24] E. Isaacson, D. Marchesin, B. Plohr, and B. Temple, *The Riemann problem near a hyperbolic singularity: The classification of solutions of quadratic Riemann problems* I, SIAM J. Appl. Math., 48 (1988), pp. 1009–1032.

[25] E. Isaacson, D. Marchesin, B. Plohr, and J. B. Temple, *Multiphase flow models with singular Riemann problems*, Mat. Aplic. Comput., 11 (1992), pp. 147–166.

[26] E. Isaacson and B. Temple, *The Riemann problem near a hyperbolic singularity* II, SIAM J. Appl. Math., 48 (1988), pp. 1287–1301.

[27] P. Lax, *Weak solutions of nonlinear hyperbolic equations and their numerical computations*, Comm. Pure Appl. Math., 7 (1954), pp. 159–193.

[28] P. D. Lax, *Hyperbolic systems of conservation laws* II, Comm. Pure Appl. Math., 10 (1957), pp. 537–566.

[29] P. D. Lax, *Hyperbolic Systems of Conservation Laws and the Mathematical Theory of Shock Waves*, CBMS-NSF Reg. Conf. Ser. Appl. Math. 11, SIAM, Philadelphia, PA, 1973.

[30] P. Lax and B. Wendroff, *Systems of conservation laws*, Comm. Pure Appl. Math., 13 (1960), pp. 217–237.

[31] S. Čanic and B. Plohr, *Shock wave admissibility for quadratic conservation laws*, J. Differential Equations, 118 (1995), pp. 293–335.

[32] G. C. Lu and G. S. S. Ludford, *Asymptotic analysis of plane steady detonations*, SIAM J. Appl. Math., 42 (1982), pp. 625–635.

[33] S. B. Margolis, *Time-dependent solution of a premixed laminar flame*, J. Comput. Phys., 27 (1978), pp. 410–427.

[34] R. Menikoff and B. Plohr, *The Riemann problem for fluid flow of real materials*, Rev. Mod. Phys., 61 (1989), pp. 75–130.

[35] R. Richtmyer and K. Morton, *Difference Methods for Initial Value Problems*, Interscience, New York, 1967.

[36] D. Schaeffer, *Instability in the evolution equations describing incompressible granular flow*, J. Differential Equations, 66 (1987), pp. 19–50.

[37] D. Schaeffer and M. Shearer, *The classification of* 2×2 *systems of non-strictly hyperbolic conservation laws, with application to oil recovery*, Comm. Pure Appl. Math., 40 (1987), pp. 141–178.

[38] D. Schaeffer and M. Shearer, *Riemann problems for nonstrictly hyperbolic* 2×2 *systems of conservation laws*, Trans. Amer. Math. Soc., 304 (1987), pp. 267–306.

[39] S. Schecter, D. Marchesin, and B. Plohr, *Structurally stable Riemann solutions*, J. Differential Equations, 126 (1996), pp. 303–354.

[40] S. Schecter and M. Shearer, *Undercompressive shocks for nonstrictly hyperbolic conservation laws*, J. Dynam. Differential Equations, 3 (1991), pp. 199–271.

[41] M. Shearer, *The Riemann problem for a class of conservation laws of mixed type*, J. Differential Equations, 46 (1982), pp. 426–443.

[42] M. Shearer, *Nonuniqueness of admissible solutions of Riemann initial value problems for a system of conservation laws of mixed type*, Arch. Ration Mech., 93 (1986), pp. 45–59.

[43] M. Shearer, *Loss of strict hyperbolicity of the Buckley-Leverett equations for three phase flow in a porous medium*, in Numerical Simulation in Oil Recovery, M. Wheeler, ed., IMA Vol. Math. Appl. 11, Springer-Verlag, New York, Heidelberg, Berlin, 1988, pp. 263–283.

[44] M. Shearer, *The Riemann problem for* 2×2 *systems of hyperbolic conservation laws with case* I *quadratic nonlinearities*, J. Differential Equations, 80 (1989), pp. 343–363.

[45] M. Shearer, D. Schaeffer, D. Marchesin, and P. Paes-Leme, *Solution of the Riemann problem for a prototype* 2×2 *system of non-strictly hyperbolic conservation laws*, Arch. Ration. Mech. Anal., 97 (1987), pp. 299–320.

[46] M. Shearer and S. Schecter, *Riemann problems involving undercompressive shocks*, in PDEs and Continuum Models of Phase Transitions, M. Rascle, D. Serre, and M. Slemrod, eds., Lecture Notes in Phys. 344, Springer-Verlag, New York, Heidelberg, Berlin, 1989, pp. 187–200.

[47] M. Shearer and J. Trangenstein, *Loss of real characteristics for models of three-phase flow in a porous medium*, Transp. Porous Media, 4 (1989), pp. 499–525.

[48] V. Smiljanovski and R. Klein, *Flame front tracking via in-cell reconstruction*, Proceedings of the Fifth International Conference on Hyperbolic Problems Theory, Numerics, and Applications (London), J. W. Grove, J. Glimm, M. J. Graham, and B. Plohr, eds., World Scientific Publishers, River Edge, NJ, 1995.
[49] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, Springer-Verlag, New York, Heidelberg, Berlin, 1992.
[50] Z.-H. Teng, A. J. Chorin, and T.-P. Liu, *Riemann problems for reacting gas, with applications to transition*, SIAM J. Appl. Math., 42 (1982), pp. 964–981.
[51] D. Wagner, *Detonation Waves and Deflagration Waves in the One Dimensional ZND-Model for High Mach Number Combustion*, IMA preprint 498, 1989.
[52] D. H. Wagner, *The existence and behavior of viscous structure for plane detonation waves*, SIAM J. Math. Anal., 20 (1989), pp. 1035–1054.
[53] F. Williams, *Combustion Theory*, Addison–Wesley, Reading, MA, 1965.
[54] C. Wu, *On MHD intermediate shocks*, Geophys. Res. Lett., 14 (1987), pp. 668–671.

SIAM J. SCI. COMPUT.
Vol. 29, No. 3, pp. 941–963
© 2007 Society for Industrial and Applied Mathematics

TOWARDS A BLACK BOX ALGORITHM FOR NONLINEAR FUNCTION APPROXIMATION OVER HIGH-DIMENSIONAL DOMAINS*

ARTA A. JAMSHIDI† AND MICHAEL J. KIRBY†

Abstract. We propose an algorithm for constructing nonlinear models from high-dimensional scattered data. The algorithm progresses iteratively, adding a new function at each step to refine the model. The placement of the functions is driven by a statistical hypothesis test that reveals geometric structure when it fails. At each step the added function is fit to data contained in a spatio-temporally defined local region to determine the parameters, particularly the scale of the local model. Unlike currently available techniques for nonlinear function fitting over scattered data, the proposed method requires no ad hoc parameters. Thus, the number of basis functions required for an accurate fit is determined automatically by the algorithm. We illustrate the approach using several illustrative problems including modeling data on manifolds and the prediction of financial time-series. The algorithm is presented in the context of radial basis functions but in principle can be employed with other methods for function approximation, such as multilayer perceptrons.

Key words. algorithms for nonlinear functional approximation, radial basis functions, manifold learning, curve fitting

AMS subject classifications. 65D10, 65D15, 62M10, 94A12

DOI. 10.1137/050646457

1. Introduction.

The problem of extracting nonlinear relationships in large high-dimensional scattered data sets is of central importance across fields of science, engineering, and mathematics. The beginnings of empirical data fitting may be traced to Gauss's work on using least squares to construct linear models. Over the last two decades we have seen a tremendous growth in this area, motivated by new ideas for computing nonlinear models; see, e.g., [10, 44, 39, 36, 37]. Today, diverse areas such as machine learning, optimal control, and mathematical modeling of physical systems often rely significantly on the ability to construct relationships from data. Subsequently there have been a multitude of applications including financial time-series analysis, voice recognition, failure prediction, and artificial intelligence, all of which provide evidence of the importance of nonlinear function approximation algorithms. Our interest in this problem relates to representing data on a manifold as the graph of a function [7, 8] and the reduction of dynamical systems [9].

A common element in empirical data fitting applications is that the complexity of the required model including the number and scale of representation functions is not known a priori and must be determined as efficiently as possible. A variety of approaches have been proposed to determine the number of model functions, i.e., the model order problem. A generally accepted measure of quality of such data

*Received by the editors November 30, 2005; accepted for publication (in revised form) November 7, 2006; published electronically May 7, 2007. This study was partially supported by the National Science Foundation under grants 0530884 and 0434351, by the DOD-USAF-Office of Scientific Research under contract FA9550-04-1-0094, and by a donation by Siemens Corporate Research. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation, the DOD-USAF-Office of Scientific Research, or Siemens Corporate Research.

http://www.siam.org/journals/sisc/29-3/64645.html

†Department of Mathematics, Colorado State University, Fort Collins, CO 80523 (jamshidi@math.colostate.edu, kirby@math.colostate.edu).

fitting algorithms is that the resulting models generalize well to testing data, i.e., data associated with the same process but that was not used to construct the model. This requirement is essentially that the data not be overfit by a model with too many parameters or underfit by a model with too few parameters.

One general approach to this problem is known as [illegible], i.e., fitting a smooth function through the data set using a modified optimization problem that penalizes variation [42]. A standard technique for enforcing regularization constraints is via cross-validation [17, 43]. Such methods involve partitioning the data into subsets of training, validation, and testing data; for details see, e.g., [18].

Additionally, a variety of model growing and pruning algorithms have been suggested, e.g., the upstart algorithm in [14], cascade correlation [13], optimal brain damage [12], and the resource allocating network (RAN) proposed by Platt [35]. Statistical methods have also been proposed that include, e.g., Akaike information criteria (AIC), Bayesian information criteria (BIC), and minimum description length (MDL) [41, 1] and Bayesian model comparison [28]. In [38, 30, 19] the issue of selecting the number of basis functions with growing and pruning algorithms from a Bayesian prospective has been studied. In [4], a hierarchical full Bayesian model for radial basis functions (RBFs) is proposed. The maximum marginal likelihood of the data has also been used to determine RBF parameters [33]. For a more complete list of references, the reader is referred to [21].

In general, model order determination via both regularization and growing and pruning algorithms can be computationally intensive and data hungry. More important, however, is that these algorithms do not explicitly exploit the geometric and statistical structure of the residuals during the training procedure. In addition, many algorithms in the literature require that anywhere from a few to a dozen ad hoc parameters be tuned for each data set under consideration.

This paper presents an approach for the model order determination problem free of ad hoc parameters. The algorithm is based on detecting any structure in the model residuals. As in the previous work [3, 2] on which this algorithm is based, such structure is quantified via a statistical hypothesis test to determine whether the residuals are independent and identically distributed (IID). The main contribution of this paper is the implementation of the algorithm over higher-dimensional domains using a spatio-temporal ball rather than a temporal window for constructing local training sets. This innovation is particularly critical if the data is periodic or quasi-periodic or, more generally, resides on a manifold. Further, a new initialization procedure is adopted that greatly accelerates the optimization algorithms. To illustrate the absence of ad hoc parameters, diverse data sets are fit without making any changes to the algorithm. In particular, no parameters are varied, or tuned, across data sets.

While we use RBFs to demonstrate the algorithms, the methodology holds for other function fitting problems. In particular, these methods may be applied directly to multilayer perceptrons. Also, we restrict the scope of the presentation to the case of batch data and will present extensions to the online algorithm elsewhere.

The organization of this paper is as follows. Section 2 reviews the RBF approach for multiscale approximation of scattered data. Section 3 develops the background for the hypothesis test for IID noise. Section 4 introduces the algorithm including the concept of spatio-temporal windowing and appropriate stopping criteria. Section 5 demonstrates the performance and robustness of the system using different data sets. Finally, section 6 provides some concluding remarks and discusses future work. For the purposes of keeping this paper self-contained we present some background details of the algorithm proposed in [3, 2] and [21].

2. Radial basis functions. RBFs were introduced for the function approximation problem as an alternative to multilayer perceptrons [10]. Part of their appeal is the variety of efficient algorithms available for their construction. In the extreme, the basis functions may be selected randomly (following the distribution of the data) with fixed scales. In this instance the resulting optimization problem is simply an overdetermined least squares problem to determine the expansion coefficients. One may improve on this approach at modest expense by employing a clustering algorithm to determine the basis function centers [31]. Furthermore, RBFs may be adapted with rank one updates or down-dates [32, 34]. Over the years RBFs have been used successfully to solve a wide range of function approximation and pattern classification problems [5, 21]. More recently, RBFs have been proposed as a tool for the simulation of partial differential equations; see, e.g., [48].

An RBF expansion is a linear summation of special nonlinear basis functions. In general, an RBF is a mapping $f : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ that is represented by

$$f(x) = Ax + \alpha_0 + \sum_{k=1}^{K} \alpha_k \phi_k(\|x - c_k\|_W), \tag{2.1}$$

where x is an input pattern, ϕ_k is the kth radial basis function centered at location c_k, α_k denotes the weight for kth RBF, and A is an $m \times n$ matrix. The term W denotes the parameters in the weighted inner product

$$\|x\|_W = \sqrt{x^T W x}.$$

The term $Ax + \alpha_0$ affords an affine transformation of the data and is useful so that the nonlinear terms are not attempting to fit flat regions. More general polynomials may be used for this purpose [40]. As usual, the dimensions of the input n and output m are specified by the dimensions of the input-output pairs.

The general training problem for the RBF is the determination of the unknown parameters: $\{A, \alpha_i, c_k, W, K\}$. The focus of this paper is to determine an optimal value for K, i.e., the model order. Of course optimizing K depends on high quality values for the remaining parameters, and we propose algorithms for this purpose.

The requirements of the RBF expansion are the same as standard data fitting problems; i.e., given a set of L input-output pairs $\{(x_l, y_l)\}_{l=1}^L$, $\mathcal{X} = \{x_l\}_{l=1}^L$, and $\mathcal{Y} = \{y_l\}_{l=1}^L$, the goal is to find the underlying mapping f such that $y_l = f(x_l)$. In the standard situation we have more data than equations, so we cannot expect to satisfy each equation exactly. Thus the problem is to minimize the cost function

$$E(A, \alpha_i, c_k, W, K) = \frac{1}{2} \sum_{l=1}^{L} \| f(x_l) - y_l \|^2,$$

where now the metric $\| \cdot \|$ is generally the Euclidean inner product and is distinct from the W-weighted inner product used to compute the contributions of the basis functions.

One of the attractive features of RBFs is the variety of basis functions available for the expansion. In particular, these functions come in both local and global flavors and include

$$\phi(r) = \{\exp(-r^2), r, r^2 \ln r, r^3\}.$$

These functions satisfy the criteria of RBFs as described in [37], and the associated approximation theorem states that if $\mathcal{D}$ is a compact subset of $\mathbb{R}^d$, then every continu-

ous real valued function on $\mathcal{D}$ can be approximated uniformly by linear combinations of RBFs with centers in $\mathcal{D}$. In this study we restrict our attention to (local) Gaussian RBFs, i.e., $\phi(r) = \exp(-r^2)$, but the algorithm works in principle for any admissible RBF. We have also recently proposed several candidates for compact RBFs that have excellent conditioning properties associated with the overdetermined least squares problem [20].

We note that in function approximation problems one must distinguish whether the data for which a model is desired is available at the outset or whether the data becomes available as the model is being built. In keeping with standard terminology we refer to these problems as [illegible] and [illegible] training, respectively. In this paper we are specifically interested in learning input/output relationships from batch data.

3. Testing for structure in model residuals. As indicated earlier, one can infer essential information about an empirical model by examining its residuals. Following [3, 2], the proposed algorithm functions on the premise that if there is structure remaining in the residuals, then one or more additional basis functions should be added. On the other hand, if there is no discernible structure, then this is tantamount to a stopping criterion.

We seek to model data that may be viewed as the superposition of a signal and IID noise, observing that if data is noise-free, then one may simply add IID noise to employ this algorithm. In view of this, we may expect the model residuals to be IID, while the model itself represents the geometric structure of the data. Hence, an indication that an RBF model is unsatisfactory is the failure of the residuals to satisfy a hypothesis test for IID noise. This observation forms the basic idea for the stopping criterion used in this research work. The primary advantage of such a criterion is that the hypothesis test from which it stems does not involve any ad hoc parameters that require adjustment.

Now we outline the IID test for determining structure in the residuals and its associated algorithm.

3.1. Statistical background of test for IID noise. The model residual for the nth data point is defined as

$$e_n = y_n - f(x_n).$$

Following [2], we denote the set of residuals for a model of order K as

$$R^K = \{e_n\}_{n=1}^{L}, \tag{3.1}$$

where L is the cardinality of the training set. The standard definition for the sample autocorrelation function (ACF), $\widehat{\rho}(h)$, for a set of residuals $e_1, e_2, e_3, \ldots, e_L$ with sample mean $\bar{e}$ and lag h is defined as

$$\widehat{\rho}(h) = \frac{\widehat{\gamma}(h)}{\widehat{\gamma}(0)}, \tag{3.2}$$

where $-L < h < L$ and

$$\widehat{\gamma}(h) = \frac{1}{L} \sum_{i=1}^{L-|h|} \alpha(h, e_i). \tag{3.3}$$

As proposed in [3, 2], we decompose the ACF into its component terms

$$\alpha(h, e_i) = (e_{i+|h|} - \bar{e})(e_i - \bar{e}). \tag{3.4}$$

For a fixed lag h the quantity $\alpha(h, e_i)$ is the contribution of the ith residual to the ACF. Later we focus on this quantity α and will illustrate that it reveals critical information concerning where new basis functions should be placed. Given its importance, we refer to this term as the autocorrelation contribution, or ACC [3, 2].

3.2. IID hypothesis test. As indicated above, we seek to terminate the addition of new basis functions when the residuals appear to have no further structure. As a test for structure, we consider whether the residuals are IID. The relevant theorem from statistics states that for large L the sample autocorrelations of an IID sequence $U_1, U_2, \ldots, U_L$ with finite variance are approximately IID, with normal distribution with mean zero and variance $\frac{1}{L}$, $N(0, \frac{1}{L})$ [6, p. 222]. Hence, if $u_1, u_2, \ldots, u_n$ is a realization of such an IID sequence, then 95% of the sample autocorrelations should fall between the bounds

$$\frac{-1.96}{\sqrt{L}} < \widehat{\rho}(h) < \frac{1.96}{\sqrt{L}}. \tag{3.5}$$

Note that if the underlying model is known, there is a more accurate bound for this test, described in section 9.4 of [6]. Therefore, if one computes the sample autocorrelation up to lag h and finds that more than $0.05h$ of the samples fall outside the bound, or that one value falls far outside the bounds, the IID hypothesis is rejected. This test can equivalently be written in terms of χ^2 distribution. Given

$$Q = L\widehat{\rho}^T\widehat{\rho} = L\sum_{j=1}^{L-1}\widehat{\rho}^2(j),$$

it has been shown in [6] that Q has a χ^2 distribution with $L-1$ degrees of freedom. The adequacy of the model is therefore rejected at level α if

$$Q > \chi^2_{1-\alpha}(L-1).$$

3.3. Additional testing possibilities. To show rigorously that a sequence of random variables is truly IID, higher moments (if they exist) also need to be considered. In particular, if a sequence of random variables is IID, then any function of the random variables is also IID. Thus, the ACF of not only the sequence of random variables but also, e.g., squares, cubes, and the absolute values, must pass the above given test. For simplicity in this paper we only consider the ACF of the raw residuals. Although this limits our ability to conclude that the residuals are strictly IID, the results suggest that this test is already quite powerful. Note that the test as implemented does indeed provide a necessary and sufficient condition for a sequence to be white noise.

Finally, we remark that there are alternatives to the test described above based on the ACF for testing IID or white noise. Other such tests include the difference-sign test, the rank test, and a test based on turning point [6]. These tests might also be applied as stopping criteria individually, or in conjunction with the current test based on the ACF. Another route for improvement may be possible by considering the relationship between the data inputs and the residuals of the outputs [27].

4. RBF algorithm using a spatio-temporal ball. In this section we present the details of the proposed batch algorithm. The question of whether a new basis function should be added is answered by the IID test. We shall see that this test also

indicates where the new basis function should be added. We introduce the concept of a space-time ball for defining local regions.

Initially, the residuals of the model are equal to the original data. A step in this algorithm consists of evaluating the residuals and determining whether they indicate that a new basis function should be added. Pseudocode is provided in Algorithm 1, and we now proceed to describe the details.

Algorithm 1. RBF algorithm using spatio-temporal ball.

$ran_flag = 1$, $K = 0$
while $ran_flag = 1$ **do**
  evaluate the RBF on the training data set $\{f(x_n)\}_{n=1}^{L}$
  compute the model error $\{e_n\}_{n=1}^{L}$
  compute component contributions $\alpha(h, e_i) = (e_{i+|h|} - \bar{e})(e_i - \bar{e})$
  compute ACF for all $0 < h < L$
  if the autocorrelation test is rejected **then**
    compute h^* via equation $h^* = \arg\max \widehat{\gamma}(h), h > 0$, and
    compute $x^* = x_{i^*} = e^{-1}(e_{i^*})$, where $i^* = \arg\max_{i=1,\ldots,n-h^*} \alpha(h^*, e_i)$
    compute the ACC function, $\beta_i = \alpha(h^*, e_i)$, $i = 1, \ldots, n - h^*$
    optional: denoise the ACC function
    find the right and left local minimizers of i^*, i.e., l^* and r^*
    compute $d_l = d(x_{i^*}, x_{l^*})$, $d_r = d(x_{i^*}, x_{r^*})$, and $d_c = \max\{d_l, d_r\}$
    define the local ball as $\mathcal{X}_{local} = \{x \in \mathcal{X} : \|x - x^*\| \leq d_c\}$
    add a new RBF $h(x; v)$ with initial values $v = [c_0, \sigma_0, \alpha_0]^T$
    solve $E(v) = \min_{v} \| h(x; v) - y \|_2^2$, where $x \in \mathcal{X}_{local}$
    $K = K + 1$
  else
    $ran_flag = 0$
  end if
  compute confidence, RMSE, and $\gamma(h^*)$ of the current model on the training set
end while

4.1. Determining optimal locations of new RBFs. If the autocorrelation test indicates that the data is not IID, then the next requirement is to determine where the new basis function should be located to optimally reduce the structure in the model residuals. Following [3, 2], we look for the point in the domain that makes the largest contribution to the ACF. This is accomplished by observing that the residuals are associated with the data in the domain in a one-to-one manner; i.e., there is a mapping, say ψ, from a data point to its residual of the form

$$e_i = \psi(x_i).$$

Thus, by identifying the residual associated with the largest contribution to the ACF we may identify the location in the domain where the basis function should be added. Note that if the maximum contribution value to the ACF function is shared by more than one residual, a single basis function would be added, selected from the residuals at random. To actually find this point we first determine the exact lag for which the ACF $\widehat{\gamma}(h)$ reaches its maximum value h^*, i.e.,

$$h^* = \arg\max_{h>0} \widehat{\gamma}(h). \tag{4.1}$$

Note that we consider only lags $h > 0$. If we allowed $h = 0$, then the term $\alpha(0, e_i)$ would always have the maximum contribution (for some i) to the ACF, and the method would be similar to those that use maximum magnitude of residuals as a criterion to allocate new basis functions, critically losing the spatial component of the diagnostic.

Then, we find the point in the spatial domain that has the maximum contribution to the ACF for lag $h = h^*$ by solving

$$i^* = \arg \max_{i=1,\ldots,n-h} \alpha(h^*, e_i). \tag{4.2}$$

Thus the center for the new basis function is given by

$$x_{i^*} = \psi^{-1}(e_{i^*}),$$

where ψ^{-1} is the inverse of the function ψ. For simplicity, we will refer to this center location as x^*.

4.2. Spatio-temporal windowing. Now that the center of the new basis function has been found as described above, it is necessary to determine what data should be used to determine the scale and weight of the new RBF. Consider the function $\beta_i = \alpha(h^*, e_i)$. The index i is inherited from the data labels and, in the case of a time-series, corresponds to a time ordering. In practice, if we plot β_i as a function of i, we observe that the values of β_i decrease away from i^*, which is what one expects since this index was selected because it corresponded to a local maximum in (4.2). How quickly the values of β_i decrease for both $i > i^*$ and $i < i^*$ is a property of the scales of the data and the model.

For simplicity, we assume that β_i decreases monotonically for both increasing and decreasing values of i until local minima are reached at the indices $l^* < i^*$ and $r^* > i^*$; here we use l, r to indicate left and right, respectively. We now compute the distances

$$d_l = d(x_{i^*}, x_{l^*})$$

and

$$d_r = d(x_{i^*}, x_{r^*}),$$

as these indicate the size of the data ball around the center x^*. The subset of the data employed to update the added basis function is then

$$\mathcal{X}_{local} = \{x \in \mathcal{X} : \|x - x^*\| \leq d_c\},$$

where $\mathcal{X}$ is the entire training set. The distance d_c can be selected in a variety of ways, and here we select

$$d_c = \max\{d_l, d_r\}.$$

Note that $\mathcal{X}_{local}$ now may contain data whose indices have values that are substantially different from i^*, l^*, and r^*. For time-series data it is apparent that spatial neighbors may not be temporal neighbors. Hence, this spatial-temporal windowing has the potential to capture substantial training data that would otherwise be ignored. For periodic or quasi-periodic data we have found that space-time windowing is essential. As observed in [3, 2], if β_i does not decrease monotonically away from the peak, then a small amount of filtering can be employed to recover monotonicity. Note that while no smoothing of β_i was required to accurately determine $\mathcal{X}_{local}$ for the examples in this paper, it may be necessary in practice; see [3, 2, 21] for details.

4.3. Updating the model. We may write the RBF expansion as consisting of K adapted terms and one new term, i.e.,

$$f^{(K+1)}(x) = f^{(K)}(x) + h(x; v),$$

where $v = [c, \sigma, \alpha]^T$ is the vector of parameters to be optimized. The new term $h(x; v)$ is initialized as

$$h(x; v) = \alpha_0 \phi(\|x - c_0\|_W).$$

The initialization of the center c_0 is at the point of most structure according to our test, i.e., $c_0 = x^*$. The vector of widths σ is very effectively initialized using the diagonal elements of the covariance matrix of the local data,

$$\sigma_0 = \sqrt{\text{diag}(cov(\mathcal{X}_{local}))}. \tag{4.3}$$

Note here that $W = \text{diag}(\sigma_0)$, where the diagonal matrix W is the weighting term in the inner product and has its diagonal elements equal to the components in σ. This initialization of the weights has proven to be extremely valuable in accelerating the convergence of the conjugate gradient iteration. The initial value for the weight, α_0, is calculated via least squares using the initial values for center location and widths.

Once the data in $\mathcal{X}_{local}$ has been determined, the scale (width), weight, and center location of the new basis function are optimized using the conjugate gradient method with cost function

$$E(v) = \min_v \sum_{x \in \mathcal{X}_{local}} \| h(x; v) - y \|_2^2 .$$

We optimize only the parameters associated with the new RBF and keep the others fixed.

4.4. Stopping criteria. One of the most critical components of any growing algorithm is the stopping criterion. Once the optimization of the new term $h(x; v)$ is complete, a new set of residuals is computed over the entire training set. For an algorithm based on the IID test of the residuals the most natural stopping criterion to use is the 95% confidence level. Note that it is possible to interpret the level of confidence as a parameter, but we do not vary it. We present several examples in the section on numerical experiments which illustrate the effectiveness of this stopping criterion.

For all applications we have computed the root mean square error (RMSE)

$$RMSE = \sqrt{\frac{1}{T} \sum_{i=1}^{T} e_i^2}, \tag{4.4}$$

where T is the number of test points. To compare our results with other work we also compute two forms of the normalized prediction error (NPE), namely,

$$NPE_1 = \frac{\sum_{i=1}^{T} \mid e_i \mid}{\sum_{i=1}^{T} \mid y_i - \overline{y} \mid} \tag{4.5}$$

and

$$NPE_2 = \frac{\sum_{i=1}^{T} e_i^2}{\sum_{i=1}^{T} (y_i - \overline{y})^2}. \tag{4.6}$$

It is also interesting to note that the quantity $\hat{\gamma}(h^*)$, i.e., the total contribution to the ACF at lag h^*, monotonically decreases with the number of basis functions

and becomes very flat when the algorithm has converged. Hence, this quantity could be used as a sort of statistical no progress criterion. We have compared the usual criterion on the RMSE with the idea of a no progress criterion on $\hat{\gamma}(h^*)$ and have found the latter to be more robust. Although we did not need to use either as stopping criteria for our applications, it is possible that they could be useful for this purpose with other data sets. With this in mind, we will illustrate their behavior in the numerical experiments that follow.

5. Numerical examples. Here we present several applications to demonstrate the performance of the algorithm in higher-dimensional domains. The successful extension of this algorithm from one- to higher-dimensional domains requires the introduction of the notion of a space-time window. Here we illustrate the impact of this idea on several applications. Note that throughout all the following examples the same code was employed; in particular, there were no parameters that were adjusted or tuned to the data set. Applications to one-dimensional domains for a variety of noise levels and types have been explored in [21].

5.1. A simple manifold. In this example we illustrate the representation of data on a manifold as the graph of a function. We employ the data set (shown in Figure 5.1) named for its similarity to the boundary of a potato chip; see also [8, 7]. The task is to construct a mapping from an (x, y) value in the plane to its corresponding z value on the pringle. Thus, we are fitting the graph of a function from $\mathbb{R}^2$ to $\mathbb{R}$. Such graph fitting problems are at the center of the Whitney's manifold embedding theorem, where $2m + 1$ dimensional domains suffice (in general) to write m dimensional manifolds as graphs; see [8, 7] for a discussion.

This data set, as proposed in [9], can be generated as the solution to the following systems of ordinary differential equations:

$$\begin{aligned}
\frac{dx}{dt} &= y,\\
\frac{dy}{dt} &= -x - (x^2 + y^2 - 1)y,\\
\frac{dz}{dt} &= -\lambda z + 2(\lambda xy + \omega(x^2 - y^2)),
\end{aligned}$$

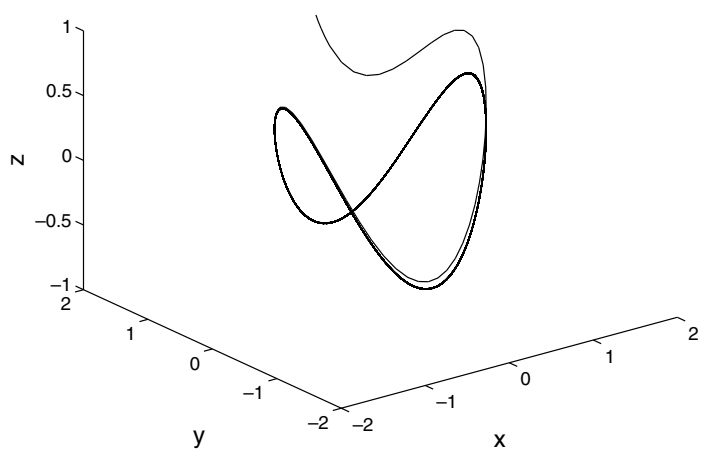


FIG. 5.1. *Plot of a typical pringle set with* $\lambda = 1$ *and* $\omega = 0.5$.

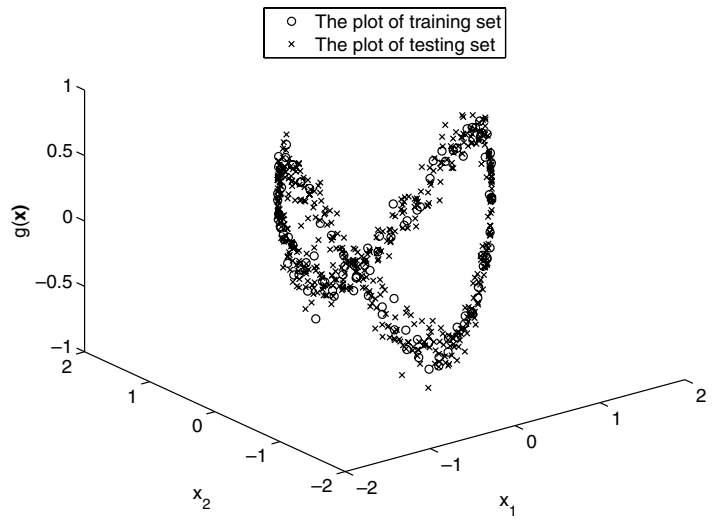
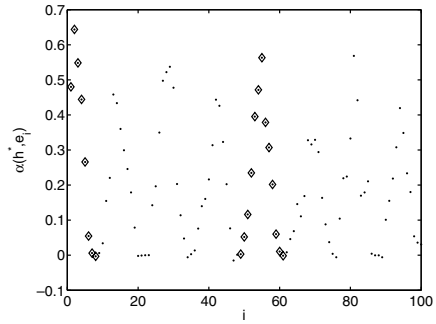


FIG. 5.2. *Plots of the training and testing data sets. The solution to the dynamical system is corrupted with Gaussian noise with standard deviation (STD) of* 0.1. *There are* 54 *data points in one cycle.*

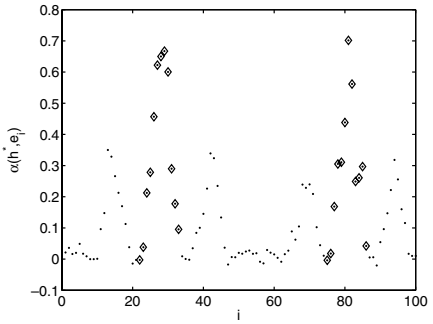
where λ and ω are parameters. In Figure 5.1 a numerically integrated trajectory of an attracting cycle is shown. In this example, we are concerned only with fitting data on the limit cycle and ignore transients. Figure 5.2 shows the training set consisting of 101 points (almost two cycles) and testing data set consisting of 500 points, or almost 9 cycles. The fact that the solution is periodic will clearly illustrate the need for spatial as well as temporal windowing of the data. The system is capable of learning a specific part of the trajectory with a small amount of data, and generalizes well to the data that resides in the same region.

Figure 5.3 shows the ACC functions for the four major RBFs that capture the underlying structure of this data set. Again, the diamonds indicate the points in the ACC function that contribute to the RBF at that stage; i.e., they belong to $\mathcal{X}_{local}$. In Figure 5.3(a) we see that the spatio-temporal window collects data from two peaks, indicating that we have cycled through the data twice in that region. This example clearly illustrates the difference between spatio-temporal windowing and temporal windowing: [illegible]. We see the same effect in Figure 5.3(b), (c), and (d).

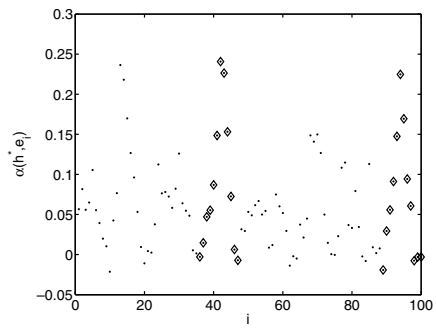
Figure 5.4 shows the location and shape of the four RBFs that are generated by the algorithm to model the data before the IID stopping criteria is satisfied. The training data and the RBFs are displayed together to illustrate how the algorithm has fit the RBFs to the data. Figure 5.5(a) shows the maximum value of the ACC function for each step in the training process. Figure 5.5(b) shows the performance of the model in the RMSE sense as the number of assigned RBFs increases, while Figure 5.5(c) shows the confidence level at each stage of training. We see that the first four basis functions are clearly the most significant. The four major RBFs model the data with RMSE of 0.1187, and 99% of points in the ACF reside in the 95% confidence bands. Note that neither the RMSE nor the values of $\gamma(h^*)$ provides reliable stopping criteria in this example. A plot of the output of the model and target values of the testing set are shown in Figure 5.6. We note that a similar experiment has been carried out for the noise-free pringle data set; see [21] for details.



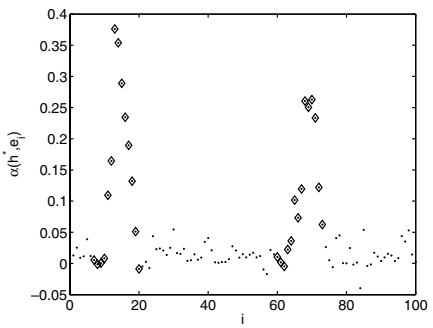
(a) The ACC function and the selected points to construct the first RBF.



(b) The ACC function and the selected points to construct the second RBF.



(c) The ACC function and the selected points to construct the third RBF.



(d) The ACC function and the selected points to construct the fourth RBF.

FIG. 5.3. *The plots of ACC functions for the four major basis functions.*

5.2. Mackey–Glass time series. The Mackey–Glass time-delay equation

$$\frac{ds(t)}{dt} = -bs(t) + a\frac{s(t-\tau)}{1+s(t-\tau)^{10}} \tag{5.1}$$

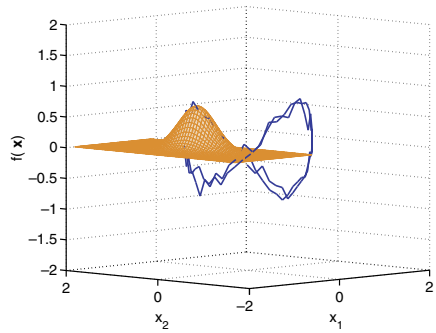
generates a chaotic time series with short-range time coherence, where long time prediction is very difficult; it has become a standard benchmark for testing model fitting algorithms [35, 23, 46].

The time series is generated by integrating the equation with model parameters $a = 0.2$, $b = 0.1$, and $\tau = 17$ using the trapezoidal rule with $\Delta t = 1$, with initial conditions $x(t-\tau) = 0.3$ for $0 \le t \le \tau$ ($\tau = 17$). The initial 1000 data points corresponding to transient behavior are discarded. Then 4000 data points are reserved for the training set. The test set consists of 500 data points starting from point 5001. Note that not all 4000 training points collected were actually used for training the model. (These conditions are very similar to those in Platt [35].)

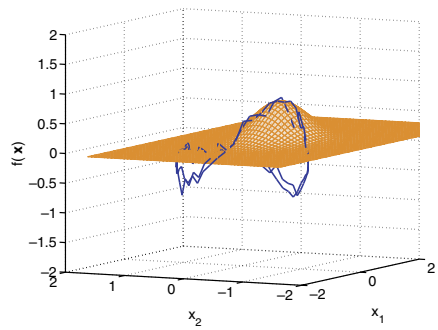
For purposes of comparison [47], the series is predicted with $\upsilon = 50$ samples ahead, using four past samples: s_n, s_{n-6}, s_{n-12}, and s_{n-18}. Hence, the nth input-output data for the network to learn are

$$x_{n+\upsilon} = [s_n, s_{n-6}, s_{n-12}, s_{n-18}]^T$$

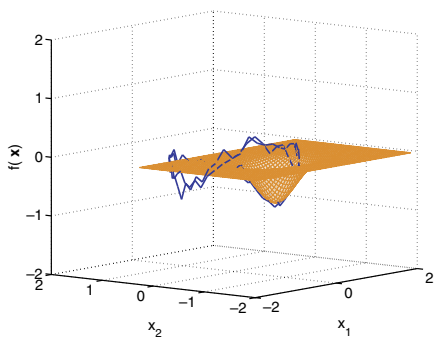
with $y_n = s_n$, whereas the υ step-ahead predicted value at time n is given by $z_{n+\upsilon} = f(x_{n+\upsilon})$, where $f(x_{n+\upsilon})$ is the network output at time n. The υ step-ahead prediction



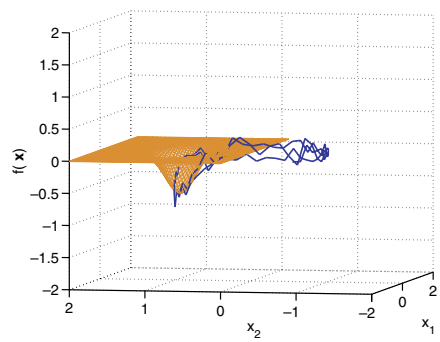
(a) The first RBF in relation with the training data set.



(b) The second RBF in relation with the training data set.



(c) The third RBF in relation with the training data set.



(d) The fourth RBF in relation with the training data set.

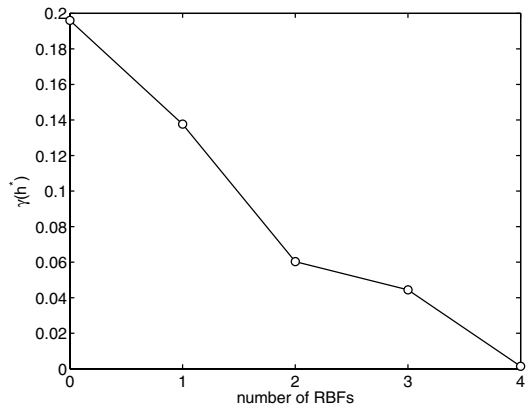
FIG. 5.4. *The primary four RBFs allocated by the algorithm. The residuals of the four mode model pass the IID test.*

error is $\epsilon = s_{n+v} - z_{n+v}$. As such, this time series provides a good example for illustrating the construction of a nontrivial mapping from $\mathbb{R}^4$ to $\mathbb{R}$.

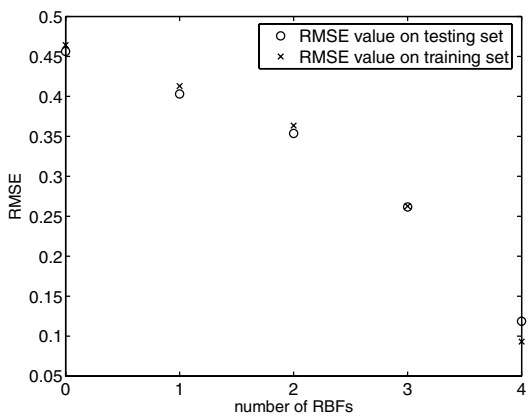
Figure 5.7(a) shows the initial ACF (computed on the training data), while Figure 5.7(b) shows the ACF of the residuals that indicates that the model fitting process should be terminated when a 95% confidence has been achieved. Figures 5.7(c) and (d) show the associated ACC functions, i.e., the point wise values β_i corresponding to the maximum value of the ACF in (a) and (b), respectively. From Figure 5.8 we see that it is sufficient to use only 76 centers to get the 95% confidence fit for the Mackey–Glass data set with a resulting RMSE of 0.0116. (See Figure 5.9.) The output of the 76 mode model for the testing data set appears to fit the target values very well. This example is interesting in that a large number of modes is required to attain the stopping criterion.

Our algorithm based on space-time balls provides a result similar to MRAN [46] (RMSE of 0.035) using 1500 data points with 21 centers. However, at this level of RMSE, both our algorithm (21 modes) and MRAN (24 modes and 4000 data points), produce sporadic but significant overshoots and undershoots of the function in regions of high gradient. These large pointwise errors are hidden to some degree by a relatively small RMSE. The IID test is, of course, point wise and reveals local unmodeled structure in the data, preventing the algorithm from terminating prematurely.

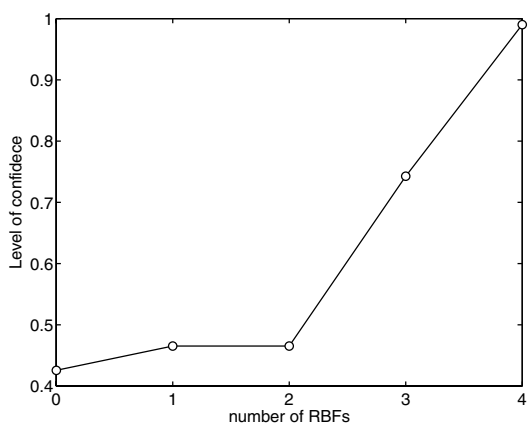
Yet, one might argue that stopping at 95% confidence and 76 modes is still premature stopping, as a slightly improved final RMSE value of 0.0090 on the test data



(a) The plot of $\gamma(h^*)$ as new basis functions are added to the model.



(b) The RMSE plot of the model as new basis functions are added to the model.



(c) The confidence level of the fitted model as the new basis functions are added to the model.

FIG. 5.5. *The performance of the RBF fit on the pringle data set. Note: The confidence level at the end of the process is* 99%.

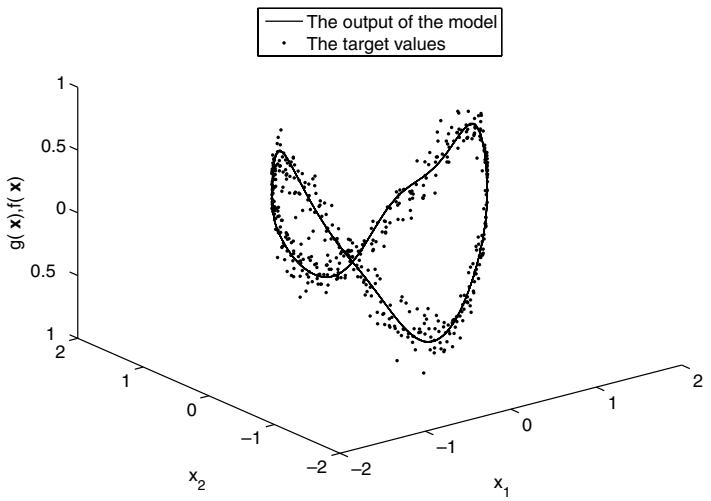
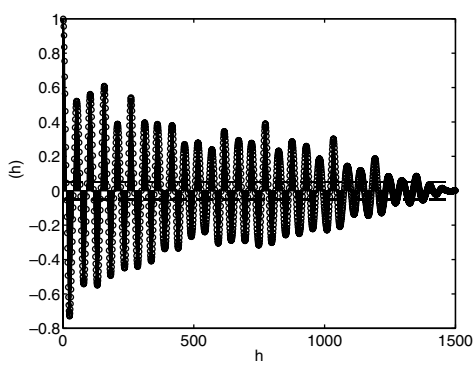
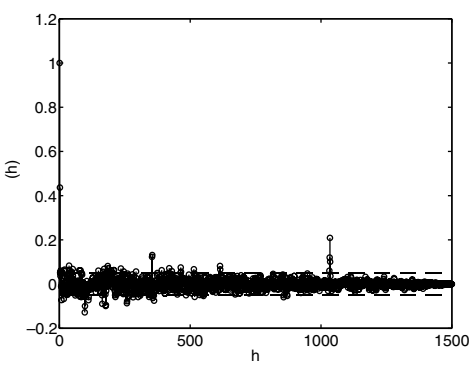


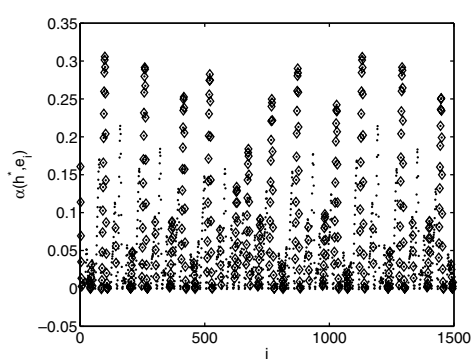
FIG. 5.6. *The testing data set and the output of the four mode model.*



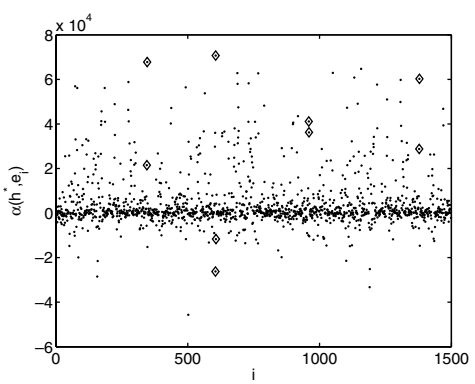
(a) Initial ACF.



(b) Final ACF.

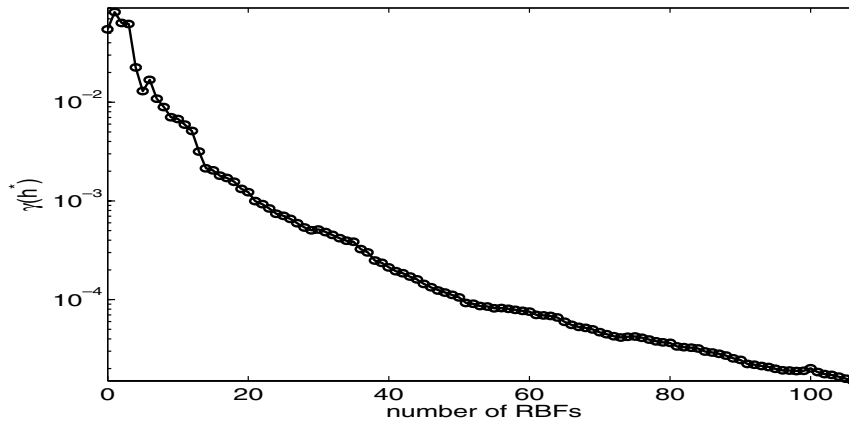


(c) The ACC function associated with (a).

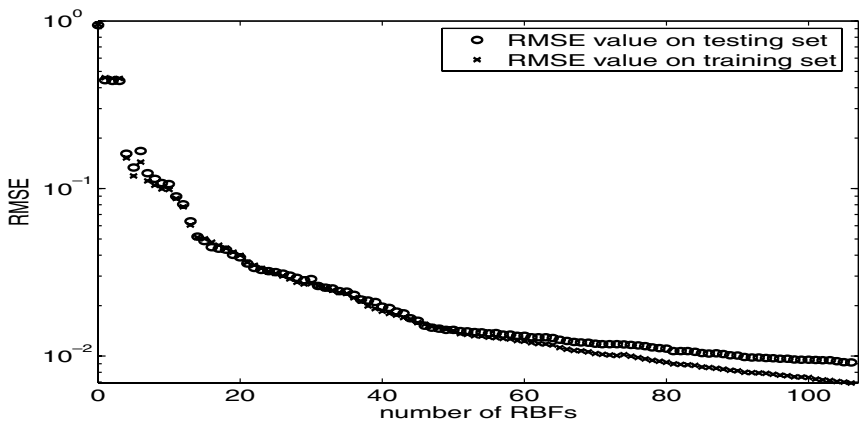


(d) The ACC function associated with (b).

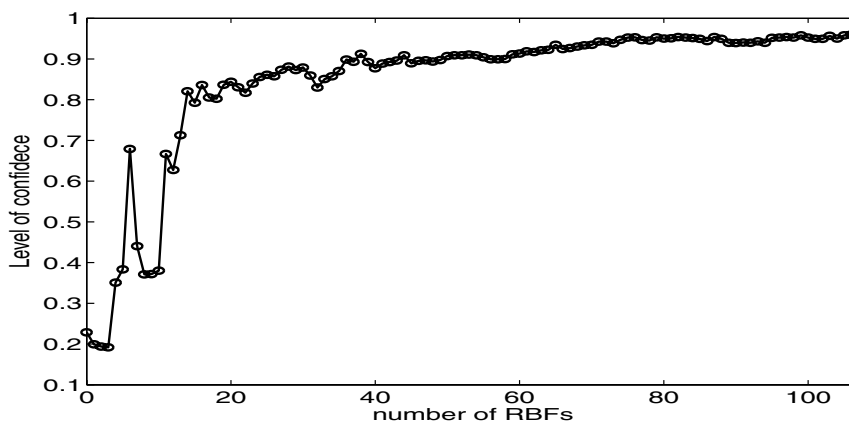
FIG. 5.7. *The first and last ACFs and the associated ACC functions that are used to determine the local balls.*



(a) The log plot of $\gamma(h^*)$ as new basis functions are added to the model.



(b) The RMSE log plot of the model as new basis functions are added to the model.



(c) The confidence level of the fitted model as the new basis functions are added to the model.

FIG. 5.8. *The performance of the RBF fit on the Mackey–Glass data set.*

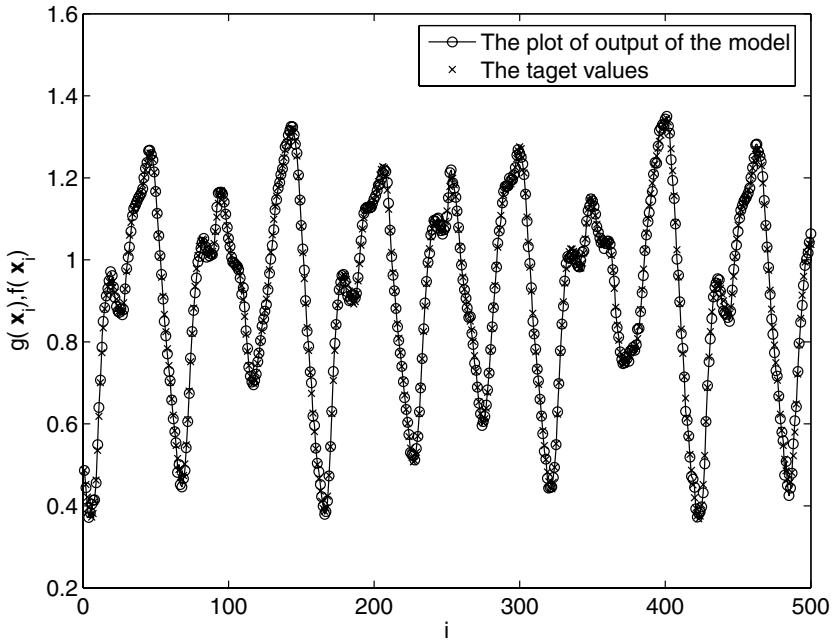


FIG. 5.9. *The output of the* 76 *mode model for the testing set compared to the target values. For this model an RMSE value of* 0.0116 *was obtained, and the* 95% *of confidence stopping criteria was satisfied.*

is achieved with 109 modes (but then does not improve with more). However, this example is for the special case of noise-free data. In such instances we recommend that the IID test be coupled with the RMSE test to draw optimal conclusions, unless, of course, one chooses to add noise artificially to the data. Given how close the RMSE errors are at 76 and 109 modes, one must seriously consider that even in this case the 95% confidence level is arguably superior.

5.3. Time series prediction using the exchange rate data set.[1] This data set consists of daily values of the Deutsche Mark/French Franc exchange rate over 701 days; see Figure 5.10. As mentioned in [29], this data set has irregular nonstationary components due to government intervention in the European exchange rate mechanism. Following [29], as there can be "day of week" effects in such data, a window of five previous values can be used as input, giving a data set of 696 patterns. Hence, this data set forms an interesting example of a mapping from $\mathbb{R}^5$ to $\mathbb{R}$.

The training data for this model was taken to be the first 600 data points. The test data set was taken to be the last 96 data points. Figure 5.11 shows the output of the resulting model (one-step prediction values) and the target (market) values for the test data. The modeling process terminated with a model of order three when the 95% confidence threshold was attained (actually 97.17%). The ACC and RMSE criteria are also in agreement with the model order of three; see Figure 5.12. The three mode model produces the RMSE value of 0.0043, $NPE_1 = 0.2760$, and $NPE_2 = 0.1033$. The model has centers at $(3.4933, 3.9292, 3.2870, 3.8574, 4.0983)$, $(3.2793, 3.3475, 3.3337, 3.18433.2718)$, and $(3.3666, 3.4187, 3.6620, 3.2056, 3.6457)$ with

[1]We would like to thank Dr. D. Lowe at Aston University for providing us with this data set.

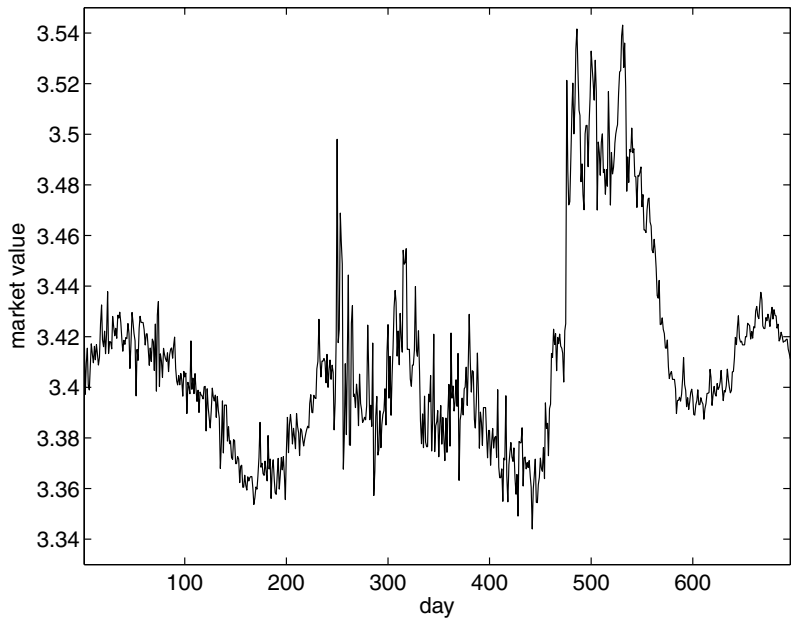


FIG. 5.10. *The plot of exchange rate data set.*

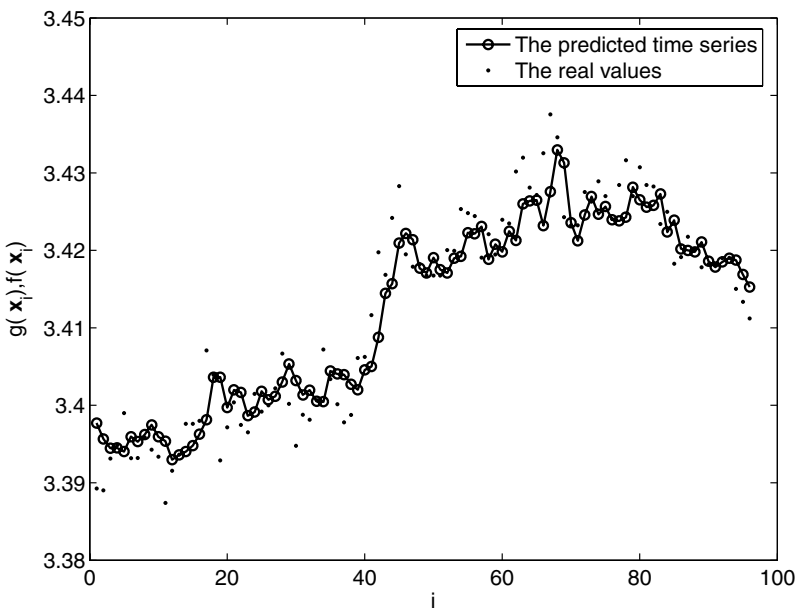
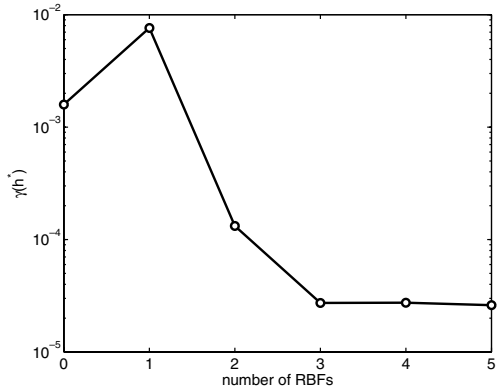
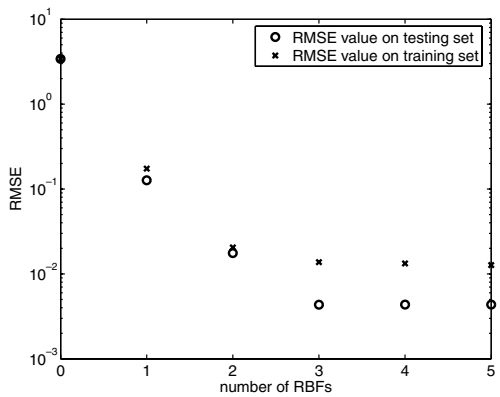


FIG. 5.11. *The one-step prediction of the exchange rate data using a three mode RBF constructed using spatio-temporal balls. The associated errors for this model are:* $RMSE = 0.0043$, $NPE_1 = 0.2760$, $NPE_2 = 0.1033$.

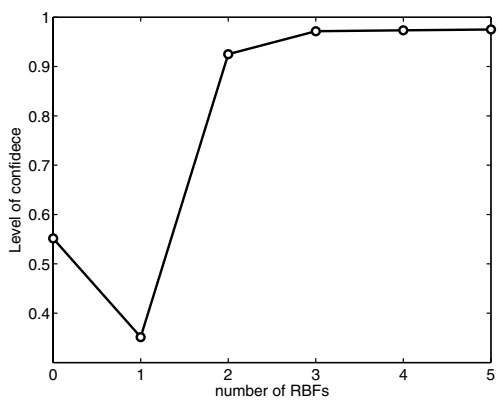
widths $(0.4501, 2.7037, 2.2175, 2.5672, 2.9234)$, $(0.1136, 0.1336, 8.5380, 0.6561, 0.5541)$, and $(0.0555, 0.0358, 0.1740, 0.1939, 0.4015)$, and weights 3.8595, 0.5751, and 1.3805, respectively.



(a) The log plot of $\gamma(h^*)$ as new basis functions are added to the model.



(b) The RMSE log plot of the model as new basis functions are added to the model.



(c) The confidence level of the fitted model as the new basis functions are added to the model.

FIG. 5.12. *The performance of the RBF fit on the exchange rate data set. Note: The confidence level is set at* 97.5% *so one can observe the behavior of the system beyond the* 95% *confidence. Also note that after adding three basis functions, the* 97.17% *of the residuals fall within the confidence bounds.*

The results for this exchange rate data reported in [29] show a model fit with 11 RBFs and the NPE_1 of 0.336. We note that in that study the data was assumed to be online so the results presented here do not compare directly. However, we have observed that the online performance of our algorithm was comparable to the batch mode for the exchange rate data.

Figure 5.13 highlights the patterns in the ACC functions associated with the maximum contributions of the ACFs in the process of adding the first three main RBFs. Note again the need for the spatio-temporal window as evidenced in Figure 5.13(b) and (c). Figure 5.13(b) shows two distinct time regions contributing to the local data., indicating either a periodic or quasi-periodic behavior. Figure 5.13(b) suggests that the remaining structure in the data is distributed across a preponderance of the data. A time-local windowing procedure would not capture this global structure in the data.

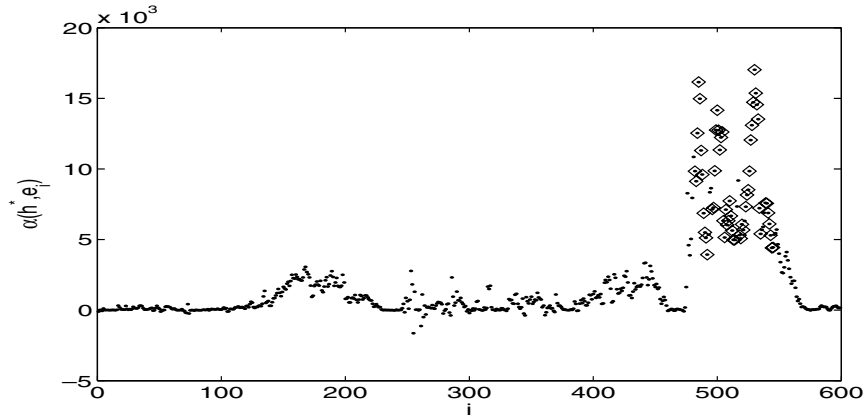
5.4. Overview of related work. In this section we review the work that is most similar in spirit to the algorithm presented here. It appears that the first paper to propose a growing algorithm for RBFs is [35]. In this paper, a new RBF is added to the model when the algorithm detects novelty in the input data. Briefly, the criteria for novelty include model error tolerance and the distance threshold between the input pattern and the nearest center. The new input pattern serves as the center of the new RBF, while the width of the RBF relates to a constant multiple of the distance between the new input and its closest center. If any of the criteria are not satisfied, the current model parameters are adjusted using the Widrow–Hoff LMS algorithm [45]. In attempt to achieve more compact networks, a new algorithm referred to as RAN-EKF has been introduced which uses the extended Kalman filter (EKF) instead of the LMS adaptation procedure [23]. Although RAN-EKF can produce more parsimonious models, it requires initialization of additional ad hoc parameters.

A minimal radial basis function neural network (MRAN) was proposed in [46]. MRAN adds a pruning strategy to RAN-EKF [23] that identifies RBFs that have negligible contribution to the overall model output over a number of consecutive inputs with respect to a threshold parameter. Also, MRAN includes an additional criterion that restrains the premature addition of new modes by smoothing the output error over a sliding window. A modified version of MRAN, called EMRAN, which utilizes an additional winner neuron strategy is proposed in [26]. Similarly, an algorithm that uses accumulated error information as a criterion for adding new RBFs is proposed in [15] and [16]. The diameter of the localized units is chosen based on the mutual distances of the RBFs. This modified method generates smaller networks with less training.

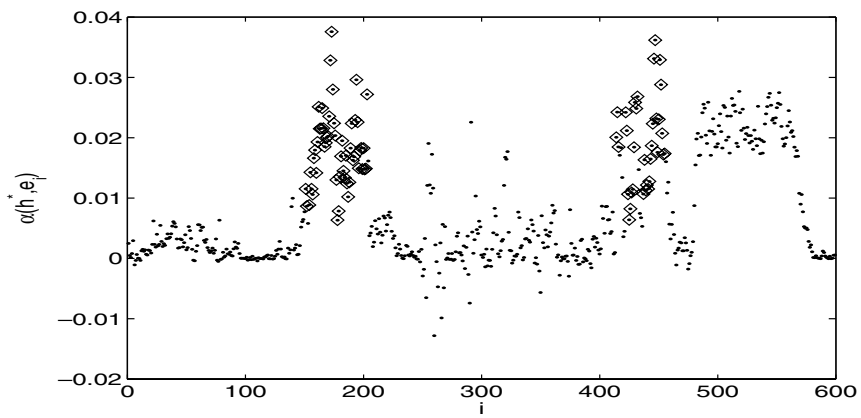
In an effort to reduce the number of required ad hoc parameters, a statistical test for adding new units to the network is proposed in [24]. This method, which uses EKF for training, is called incremental network (IncNet). In this model, if the prediction error does not fall within a certain level of confidence for the Z-statistic hypothesis test, then a new RBF will be added to the model.

More recently, a new RAN-EKF algorithm has been proposed that is applicable to both stationary and slowly varying nonstationary problems [29]. Here the novelty criterion involves testing whether the prediction error sequence corresponds to a zero mean Gaussian sequence at the 95% confidence level. A t-statistic is used to determine whether the sequence has zero mean, while the weighted sum squared residual (WSSR) statistic [11] tests the normality of the sequence.

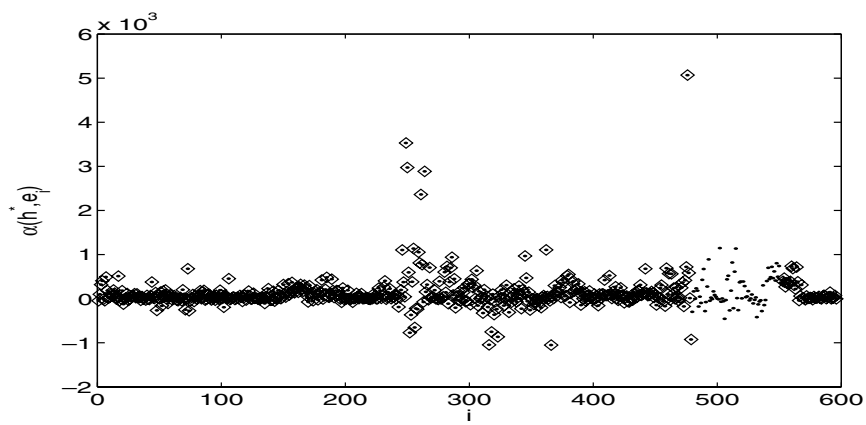
The algorithms described above that employ the results of hypothesis tests to add modes are valid only if the residuals have Gaussian distribution. In addition, perhaps



(a) The selected data points for the first center.



(b) The selected data points for the second center.



(c) The selected data points for the third center.

FIG. 5.13. *The process of extracting three basis functions from the training (exchange rate) data set demonstrated by the ACC functions.*

TABLE 5.1
This table presents a comparison of related RBF algorithms. Note that the proposed algorithm is the only one that applies to the more general case of IID noise.

Algorithm	Noise type	Ad hoc parameters
Platt's RAN	-	7
RAN-EKF	Normal	9
MRAN	Normal	12
IncNet	Normal	5
RAN-WSSR	Normal	6
Full Bayesian	Normal	12
New algorithm	IID	None

more critically, all the algorithms mentioned in this section require careful adjustment of the ad hoc parameters for each data set to which they are applied. See Table 5.1 for a summary of these methods. For further details about the described algorithms and their ad hoc parameters, please see [21].

6. Conclusions. We propose an algorithm for approximating functions from scattered data and compare its performance to the leading algorithms in the literature. To illustrate the absence of ad hoc parameters, all the data sets presented in this paper were fit by exactly the same code. No adjustments were made based on the data set being fit. Hence, we claim that the proposed algorithm is approaching a black-box methodology for nonlinear function approximation. This feature will permit the advancement of a variety of other algorithms, e.g., the representation of data on manifolds as graphs of functions [7, 8], pattern classification [22, 25], as well as the low-dimensional modeling of dynamical systems [9].

It is assumed that the available data represents a functional relationship, or signal, with IID additive noise. Note that if the signal contains multiplicative noise, we can take the natural logarithm of the signal to make it additive. A hypothesis test is applied to the residuals at each step in the algorithm to determine whether a new basis function should be added, and if so, where it should be added. When it has been determined that this test has been passed using the 95% confidence criterion, one may infer that there is no geometric structure left in the residuals, and thus the model order has been found.

We extended previous work by employing a spatio-temporal window, i.e., space-time balls, for determining the local data to be used in updating the model. The examples suggest that this novelty is critical for approximating data over high-dimensional domains and in particular for data generated by dynamical systems. We have observed that significantly more data is located in the space-time balls than the temporal windows previously considered, resulting in the construction of significantly improved models.

In this paper we have assumed that the received signal is composed of additive IID noise, while prior algorithms based on statistical hypotheses are restricted to Gaussian noise. Despite the fact that the IID test provides only a necessary condition, it appears to generate low order models with small RMSE.

We have presented this algorithm for batch data. It is possible to extend this approach for data that arrives in a stream, i.e., online data. Our preliminary results show that this approach does not require the data to be seen repeatedly, as some "online" algorithms require. We will present these results in a companion study. Also, although the algorithm was presented here in the context of growing RBFs, in principle it can be employed with other architectures for fitting nonlinear functions such as feed-forward neural networks.

Acknowledgment. This work benefited from many early discussions with Markus Anderle.

REFERENCES

[1] H. Akaike, *A new look at statistical model identification*, IEEE Trans. Automat. Control, 19 (1974), pp. 716–723.

[2] M. Anderle and M. Kirby, *Correlation feedback resource allocation RBFs*, in Proceedings of the 2001 IEEE International Joint Conference on Neural Networks (IJCNN), Washington, DC, 2001, Vol. 3, pp. 1949–1953.

[3] M. Anderle, *Modeling Geometric Structure in Noisy Data*, Ph.D. dissertation, Department of Mathematics, Colorado State University, Fort Collins, CO, 2001.

[4] C. Andrieu, N. Freitas, and A. Doucet, *Robust full Bayesian learning for radial basis networks*, Neural Comput., 13 (2001), pp. 2359–2407.

[5] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, Oxford, UK, 1995.

[6] P. J. Brockwell and R. A. Davis, *Time Series: Theory and Methods*, 2nd ed., Springer Ser. Statist., Springer, New York, 1991.

[7] D. S. Broomhead and M. Kirby, *A new approach for dimensionality reduction: Theory and algorithms*, SIAM J. Appl. Math., 60 (2000), pp. 2114–2142.

[8] D. S. Broomhead and M. Kirby, *The Whitney reduction network: A method for computing autoassociative graphs*, Neural Comput., 13 (2001), pp. 2595–2616.

[9] D. S. Broomhead and M. Kirby, *Large dimensionality reduction using secant-based projection methods: The induced dynamics in projected systems*, Nonlinear Dynam., 41 (2005), pp. 47–67.

[10] D. S. Broomhead and D. Lowe, *Multivariable functional interpolation and adaptive networks*, Complex Systems, 2 (1988), pp. 321–355.

[11] J. V. Candy, *Signal Processing: The Model-Based Approach*, McGraw–Hill Series in Electrical Engineering, Communications and Signal Processing, McGraw–Hill, New York, 1986.

[12] Y. Le Cun, J. S. Denker, and S. A. Solla, *Optimal brain damage*, in Advances in Neural Information Processing Systems, D. S. Touretzky, ed., Morgan Kaufmann, San Mateo, CA, 1990, Vol. 2, pp. 598–605.

[13] S. E. Fahlman and C. Lebiere, *The cascade-correlation learning architecture*, in Proceedings of the Connectionist Models Summer School, D. S. Touretzky, ed., Morgan Kaufmann, San Mateo, CA, 1988, Vol. 2, pp. 524–432.

[14] M. Frean, *An upstart algorithm: A method for constructing and training feedforward neural networks*, Neural Comput., 2 (1990), pp. 198–209.

[15] B. Fritzke, *Fast learning with incremental RBF networks*, Neural Process. Lett., 1 (1994), pp. 2–5.

[16] B. Fritzke, *Supervised learning with growing cell structures*, in Advances in Neural Information Processing Systems, G. Tesauro J. Cowan, and J. Alspector, eds., Morgan Kaufmann, San Mateo, CA, 1994, Vol. 6, pp. 255–262.

[17] F. Girosi, M. Jones, and T. Poggio, *Regularization theory and neural network architectures*, Neural Comput., 7 (1995), pp. 219–269.

[18] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed., Prentice–Hall, Englewood Cliffs, NJ, 1999.

[19] C. C. Holmes and B. K. Mallick, *Bayesian radial basis functions of variable dimension*, Neural Comput., 10 (1998), pp. 1217–1233.

[20] A. A. Jamshidi and M. J. Kirby, *Examples of compactly supported functions for radial basis approximations*, in Proceedings of the 2006 International Conference on Machine Learning; Models, Technologies and Applications, H. R. Arabnia, E. Kozerenko, and S. Shaumyan, eds., Las Vegas, 2006, CSREA Press, Irvine, CA, 2006, pp. 155–160.

[21] A. A. Jamshidi, *A New Spatio-Temporal Resource Allocation Network (ST-RAN)*, Master's thesis, Department of Mathematics, Colorado State University, Fort Collins, CO, 2004.

[22] A. A. Jamshidi, *An Adaptive Underwater Target Classification System with a Decision Feedback Mechanism*, Master's thesis, Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO, 2002.

[23] K. Kadirkamanathan and M. Niranjan, *A function estimation approach to sequential learning with neural networks*, Neural Comput., 5 (1993), pp. 954–975.

[24] K. Kadirkamanathan, *A statistical inference based growth criterion for RBF network*, in Proceedings of the IEEE Workshop on Neural Networks for Signal Processing IV, Ermioni, Greece, IEEE Press, Piscataway, NJ, 1994, pp. 12–21.

[25] M. Kirby and C. Anderson, *Geometric analysis for the characterization of nonstationary time-series*, in Springer Applied Mathematical Sciences Series Celebratory Volume for the Occasion of the 70th Birthday of Larry Sirovich, E. Kaplan, J. Marsden, and K. R. Sreenivasan, eds., Springer-Verlag, New York, 2003, pp. 263–292.

[26] Y. Li, N. Sundararajan, and P. Saratchandran, *Analysis of minimal radial basis function network algorithm for real-time identification of nonlinear dynamic systems*, IEE Proc. Control Theory Appl., 147 (2000), pp. 476–484.

[27] L. Ljung, *System Identification Theory for the User*, 2nd ed., Prentice–Hall PTR, Englewood Cliffs, NJ, 1999.

[28] D. J. C. Mackay, *A practical Bayesian framework for backpropagation networks*, Neural Comput., 4 (1989), pp. 448–472.

[29] A. MacLachlan, *An improved novelty criterion for resource allocating networks*, in Proceedings of the IEE 5th International Conference on Artificial Neural Networks, London, 1997, Cambridge, UK, 1997, pp. 48–52.

[30] A. D. Marrs, *An application of reversible-jump MCMC to multivariate spherical Gaussian mixtures*, in Advances in Neural Information Processing Systems, M. I. Kearns and S. A. Solla, eds., MIT press, Cambridge, MA, 1998, Vol. 10, pp. 577–583.

[31] J. Moody and C. Darken, *Fast learning in networks of locally tuned processing units*, Neural Comput., 1 (1989), pp. 281–294.

[32] M. J. L. Orr, *Introduction to Radial Basis Function Networks* (1996), Technical report, Center for Cognitive Science, University of Edinburgh, Edinburgh, Scotland, 1996.

[33] M. J. L. Orr, *An EM algorithm for regularized RBF networks*, in International Conference on Neural Networks and Brain, Beijing, China, Publishing House of Electronics Industry, China, 1998.

[34] M. J. L. Orr, *Recent Advances in Radial Basis Function Networks*, Technical report, Institute for Adaptive and Neural Computation, Division of Informatics, Edinburgh University, Edinburgh, Scotland, 1999.

[35] J. Platt, *A resource allocating network for function interpolation*, Neural Comput., 3 (1991), pp. 213–225.

[36] M. J. D. Powell, *Radial basis functions for multivariable interpolation: A review*, in Algorithms for Approximation, J. C. Mason and M. G. Cox, eds., Clarendon Press, Oxford, UK, 1987, pp. 143–167.

[37] M. J. D. Powell, *The theory of radial basis functions in* 1990, in Advances in Numerical Analysis, W. Light, ed., Wavelets, Subdivision Algorithms, and Radial Basis Functions 2, Oxford University Press, Oxford, UK, 1992, pp. 105–210.

[38] D. Insua Rios and P. Muller, *Feedforward neural networks for nonparametric regression*, in Practical Nonparametric and Semiparametric Bayesian Statistics, P. Muller, D. K. Dey, and D. Sinha, eds., Springer-Verlag, New York, 1998, pp. 181–191.

[39] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing*, MIT Press, Cambridge, MA, 1986.

[40] R. Schaback, *Reconstruction of Multivariate Functions from Scattered Data*, preprint, University of Göttingen, Göttingen, Germany, 1997.

[41] P. Smyth, *On stochastic complexity and admissible models for neural network classifiers*, in Proceedings of Neural Information Processing Systems, D. S. Touretzky, R. Lippmann, and J. Moody, eds., Morgan Kaufmann, San Mateo, CA, 1991, Vol. 3, pp. 818–824.

[42] A. N. Tikhonov and V. Y. Arsenin, *Solutions of Ill-Posed Problems*, John Wiley & Sons, New York, 1977.

[43] G. Wahba, *Spline bases, regularization, and generalized cross validation for solving approximation problems with large quantities of data*, in Approximation Theory III, W. Cheney, ed., Academic Press, New York, 1980, pp. 905–912.

[44] P. J. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*, Ph.D. dissertation, Harvard University, Cambridge, MA, 1974.

[45] B. Widrow and M. E. Hoff, *Adaptive switching circuits*, in 1960 WESCON Convention Record Part IV, reprinted in Neurocomputing: Foundations of Research, J. A. Anderson and E. Rosenfeld, eds., MIT Press, Cambridge, MA, 1988, pp. 96–104.

[46] L. Yingwei, N. Sundararajan, and P. Saratchandran, *A sequential scheme for function approximation using minimal radial basis function neural networks*, Neural Comput., 9 (1997), pp. 461–478.

[47] L. Yingwei, N. Sundararajan, and P. Saratchandran, *Performance evaluation of a sequential minimal radial basis function (RBF) neural network learning algorithm*, IEEE Trans. Neural Networks, 9 (1998), pp. 308–318.

[48] In Edward J. Kansa and Yiu-Chung Hon, ed., *Computers and mathematics with applications*, R. Lippmann, J. Moody, D. S. Touretzky, eds., Vol. 43, Elsevier, 2002, pp. 275–619.

SIAM J. SCI. COMPUT.
Vol. 29, No. 3, pp. 964–985
© 2007 Society for Industrial and Applied Mathematics

PARALLEL MULTILEVEL RESTRICTED SCHWARZ PRECONDITIONERS WITH POLLUTION REMOVING FOR PDE-CONSTRAINED OPTIMIZATION*

ERNESTO E. PRUDENCIO† AND XIAO-CHUAN CAI‡

Abstract. We develop a class of V-cycle-type multilevel restricted additive Schwarz (RAS) methods and study the numerical and parallel performance of the new fully coupled methods for solving large sparse Jacobian systems arising from the discretization of some optimization problems constrained by nonlinear partial differential equations. Straightforward extensions of the one-level RAS to multilevel do not work due to the pollution effects of the coarse interpolation. We then introduce, in this paper, a pollution removing coarse-to-fine interpolation scheme for one of the components of the multicomponent linear system and show numerically that the combination of the new interpolation scheme with the RAS smoothed multigrid method provides an effective family of techniques for solving rather difficult PDE-constrained optimization problems. Numerical examples involving the boundary control of incompressible Navier–Stokes flows are presented in detail.

Key words. Schwarz preconditioners, domain decomposition, multilevel methods, parallel computing, PDE-constrained optimization, inexact Newton, flow control

AMS subject classifications. 49K20, 65F10, 65K05, 65K10, 65N22, 65N55, 65Y05, 76N05, 76D55, 90C06, 90C90, 93C20

DOI. 10.1137/050635663

1. Introduction.

There are two major families of Newton techniques for solving nonlinear optimization problems: reduced space methods, characterized by the partition of the problem into smaller ones at each Newton step, and full space ones. As computers become more powerful in processing speed and memory capacity, full space methods become more attractive, as exemplified by the Lagrange–Newton–Krylov–Schur [3, 4] and one-level [illegible] [28] methods.

A key element of any successful full space approach is the Jacobian preconditioner, which needs to be able to simultaneously reduce the condition number and provide good parallel scalability [23]. In this paper, we focus on fully coupled Schwarz-type preconditioners in which all components of the linear system are treated equally; i.e., no variables are eliminated as in some Schur complement-type approaches. Among Schwarz-type preconditioners [32, 33], the recently introduced restricted versions [6, 9] seem to be more robust and computationally more efficient, especially for harder problems such as those indefinite, highly ill-conditioned, multicomponent systems arising from PDE-constrained optimizations. The extension of the one-level restricted additive Schwarz method (RAS) to multilevel using the multigrid V-cycle idea and standard coarse-to-fine interpolations is straightforward but may not work as expected due to the pollution effects of the interpolation. After many experiments with some flow control problems, we identified the source of the pollution at one of the three

*Received by the editors July 10, 2005; accepted for publication (in revised form) January 29, 2007; published electronically May 7, 2007. This research was supported in part by the National Science Foundation, CCR-0219190, ACI-0305666, ACI-0352334, and CCF-0634594, and in part by the Department of Energy, DE-FC02-01ER25479.

http://www.siam.org/journals/sisc/29-3/63566.html

†Advanced Computations Department, Stanford Linear Accelerator Center, Menlo Park, CA 94025 (prudenci@slac.stanford.edu).

‡Department of Computer Science, University of Colorado at Boulder, Boulder, CO 80309 (cai@cs.colorado.edu).

components of the Jacobian system, namely the Lagrange multiplier. Using a pollution removing interpolation scheme we have then been able to restore the robust and fast convergence of RAS. We discuss only linear versions of Schwarz methods even though nonlinear versions can also be obtained [8, 13]. We refer to [2, 21, 31] for the analysis of some preconditioning techniques for optimal control problems and general saddle point problems.

In this paper we consider only optimization problems with equality constraints:

$$\left\{ \begin{array}{ll} \min\limits_{\mathbf{x}\in\mathbf{W}} & \mathcal{F}(\mathbf{x}) \\ \text{subject to} & \mathbf{C}(\mathbf{x}) = \mathbf{0} \in \mathbf{Y}. \end{array} \right. \tag{1.1}$$

Here $\mathbf{W}$ and $\mathbf{Y}$ are normed spaces, $\mathbf{W}$ is the space of optimization variables, $\mathcal{F} : \mathbf{W} \to \mathbb{R}$ is the objective functional, and $\mathbf{C} : \mathbf{W} \to \mathbf{Y}$. The associated Lagrangian functional $\mathcal{L} : \mathbf{W} \times \mathbf{Y}^* \to \mathbb{R}$ is defined as

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) \equiv \mathcal{F}(\mathbf{x}) + \langle \boldsymbol{\lambda}, \mathbf{C}(\mathbf{x}) \rangle_{\mathbf{Y}} \quad \forall\ (\mathbf{x}, \boldsymbol{\lambda}) \in \mathbf{W} \times \mathbf{Y}^*,$$

where $\mathbf{Y}^*$ is the adjoint space of $\mathbf{Y}$, $\langle \cdot, \cdot \rangle_{\mathbf{Y}}$ denotes the duality pairing, and variables $\boldsymbol{\lambda}$ are called Lagrange multipliers or adjoint variables. In many cases it is possible to prove that if $\hat{\mathbf{x}}$ is a (local) solution of (1.1), then there exist Lagrange multipliers $\hat{\boldsymbol{\lambda}}$ such that $(\hat{\mathbf{x}}, \hat{\boldsymbol{\lambda}})$ is a critical point of $\mathcal{L}$. So, under sufficient smoothness assumptions, one proves that a solution has to necessarily solve a system of equations, called the Karush–Kuhn–Tucker (KKT) or optimality system.

For the numerical solution of (1.1) with PDE constraints, generally speaking, there are two families of approaches: [illegible] (OTD) and [illegible] (DTO). In OTD one usually works with constraints in the weak form, derives a KKT system containing both the original constraints and extra PDEs related to the Lagrange multipliers, and only then discretizes it, obtaining a finite dimensional system of nonlinear equations. In DTO one discretizes (1.1) to obtain a finite dimensional optimization problem and then derives the KKT system. The discrete KKT systems from both approaches are [illegible] necessarily the same, and no approach has a clear advantage over the other [18].

In this paper we follow the DTO approach, generating a mesh Ω_h of characteristic size $h > 0$ to obtain a finite dimensional optimization problem with $\mathbf{W} = \mathbb{R}^{n_h}$ and $\mathbf{Y} = \mathbb{R}^{m_h} = \mathbf{Y}^*$. The KKT system becomes $\nabla \mathcal{L}_h(\hat{\mathbf{x}}, \hat{\boldsymbol{\lambda}}) = \mathbf{0} \in \mathbb{R}^{n_h + m_h}$. Omitting symbols "$h$" and "$\hat{\ }$" and denoting $N \equiv n + m$ and $\mathbf{X} \equiv (\mathbf{x}, \boldsymbol{\lambda}) \in \mathbb{R}^N$, one has $\mathcal{L} : \mathbb{R}^N \to \mathbb{R}$ and an optimality system $\nabla \mathcal{L}(\mathbf{X}) = \mathbf{0} \in \mathbb{R}^N$, which we solve by the inexact Newton's method [15, 26] globalized by a line search, according to the heuristic explained in [28].

Let ξ be some real value and $\hat{\mathbf{x}}(\xi)$ denote the solution, if it exists, of the finite dimensional problem

$$\left\{ \begin{array}{lrcl} \min\limits_{\mathbf{x}\in\mathbb{R}^n} & \mathcal{F}(\mathbf{x}) & & \\ \text{subject to} & \mathbf{C}_1(\mathbf{x}) & = & 0, \\ & \vdots & & \\ & \mathbf{C}_{i-1}(\mathbf{x}) & = & 0, \\ & \mathbf{C}_i(\mathbf{x}) & = & \xi, \\ & \mathbf{C}_{i+1}(\mathbf{x}) & = & 0, \\ & \vdots & & \\ & \mathbf{C}_m(\mathbf{x}) & = & 0, \end{array} \right.$$

with arbitrarily fixed $1 \leqslant i \leqslant m$. Under appropriate assumptions, one can prove [16, 29]

$$\frac{\partial \mathcal{F}(\hat{\mathbf{x}}(\xi))}{\partial \xi} \bigg|_{\xi=0} = \lambda_i; \tag{1.2}$$

that is, the Lagrange multiplier λ_i indicates how sensitive the optimal value is to changes on the ith constraint. The module $|\lambda_i|$ gives a measure of how "effective" the corresponding constraint is. In the case of constraints corresponding to the discretized equations of boundary value problems, one might intuitively expect (1) different equations to have different effects on the objective function and (2) the "effectiveness" of a given original PDE constraint to have a "smooth behavior" throughout the domain over which the PDE acts. Such intuitive expectations match result (1.2). One less obvious symptom, however, is the potential discontinuity between those Lagrange multipliers related to PDEs governing the system behavior inside the problem domain and those Lagrange multipliers related to initial and boundary conditions. Indeed, we have observed such discontinuity in our test problems.

The rest of the paper is organized as follows. Section 2 explains multilevel restricted Schwarz preconditioners, and section 3 introduces the pollution removing interpolation. In section 4 we test the combination of both approaches on some boundary flow control problems. Final conclusions are given in section 5.

2. Multilevel restricted Schwarz preconditioners. Schwarz methods can be used in one-level or multilevel variants and, in each case, in combination with additive and/or multiplicative algorithms [33]. They can be also used as linear [14] and nonlinear preconditioners [8, 22].

In this section we introduce a multilevel version of the one-level RAS preconditioner initially studied in [6, 9]. The multilevel preconditioner is applicable to general linear systems arising from the discretized PDEs on a mesh using finite element or finite difference methods. Let $\Omega \subset \mathbb{R}^2$ be a bounded open domain on which a PDE is defined and a discretization is performed with a mesh Ω_h of characteristic size $h > 0$. To obtain the overlapping partition, we first divide Ω_h into nonoverlapping subdomains Ω_j, $j = 1, \ldots, N_S$. We then expand each Ω_j to $\Omega_j^{'}$, i.e., $\Omega_j \subset \Omega_j^{'}$. The overlap $\delta > 0$ is defined as the minimum distance between $\partial\Omega_j^{'}$ and $\partial\Omega_j$, in the interior of Ω. For boundary subdomains we simply cut off the part outside Ω. Let $H > 0$ denote the characteristic diameter of $\{\Omega_j\}$.

Let N and N_j denote the number of degrees of freedom associated to Ω and $\Omega_j^{'}$, respectively. Let $\mathbf{K}$ be a $N \times N$ sparse matrix of a linear system

$$\mathbf{K}\mathbf{p} = \mathbf{b} \tag{2.1}$$

that needs to be solved during the application of an algorithm for the numerical solution of the discretized differential equations. Let d indicate the number of degrees of freedom per mesh point. For simplicity let us assume that d is the same throughout the entire mesh. We define the $N_j \times N$ matrix $\mathbf{R}_j^{\delta}$ as follows: Its $d \times d$ block element $(\mathbf{R}_j^{\delta})_{l_1,l_2}$ is either (a) an identity block, if the integer indices $1 \leqslant l_1 \leqslant N_j/d$ and $1 \leqslant l_2 \leqslant N/d$ are related to the same mesh point and this mesh point belongs to $\Omega_j^{'}$, or (b) a zero block, otherwise. The multiplication of $\mathbf{R}_j^{\delta}$ with a $N \times 1$ vector generates a smaller $N_j \times 1$ vector by discarding all components corresponding to mesh points outside $\Omega_j^{'}$. The $N_j \times N$ matrix $\mathbf{R}_j^{0}$ is similarly defined, with the difference that its

application to a $N \times 1$ vector also zeros out all of those components corresponding to mesh points on $\Omega_j^{'} \setminus \Omega_j$. Let $\tilde{\mathbf{K}}_j$ be defined as

$$\tilde{\mathbf{K}}_j \equiv \mathbf{R}_j^{\delta}\, \mathbf{K}\, \left(\mathbf{R}_j^{\delta}\right)^T,$$

that is, as the $N_j \times N_j$ matrix related to a subdomain problem having zero Dirichlet boundary conditions at regions of $\partial\Omega_j^{'}$ not coinciding with $\partial\Omega$. We assume $\tilde{\mathbf{K}}_j$ to be nonsingular and denote by $\tilde{\mathbf{B}}_j^{-1}$ either the inverse of or a preconditioner for $\tilde{\mathbf{K}}_j$. The one-level classical, right restricted (r-RAS) and left restricted (l-RAS) additive Schwarz preconditioners for $\mathbf{K}$, respectively, are defined as [6, 9, 14]

$$\mathbf{B}_{\delta\delta}^{-1} = \sum_{j=1}^{N_s} \left(\mathbf{R}_j^{\delta}\right)^T \tilde{\mathbf{B}}_j^{-1} \mathbf{R}_j^{\delta}, \quad \mathbf{B}_{\delta 0}^{-1} = \sum_{j=1}^{N_s} \left(\mathbf{R}_j^{\delta}\right)^T \tilde{\mathbf{B}}_j^{-1} \mathbf{R}_j^{0}, \quad \mathbf{B}_{0\delta}^{-1} = \sum_{j=1}^{N_s} \left(\mathbf{R}_j^{0}\right)^T \tilde{\mathbf{B}}_j^{-1} \mathbf{R}_j^{\delta}.$$

When the distinction is not important, we will denote a Schwarz preconditioner simply by $\mathbf{B}^{-1}$. We refer to [17, 24, 25] for further analysis and examples of one-level restricted preconditioning techniques.

For the description of multilevel Schwarz preconditioners [34], let us use the index $i = 0, 1, \ldots, L-1$ to designate any of the $L \geqslant 2$ levels. All previously defined entities using the subindex "j" will now use double subindexes "i, j": $\mathbf{\Omega}_{i,j}$, $\mathbf{\Omega}_{i,j}^{'}$, $N_{i,j}$, $\mathbf{R}_{i,j}^{\delta}$, $\mathbf{R}_{i,j}^{0}$, $\tilde{\mathbf{K}}_{i,j}$, and $\tilde{\mathbf{B}}_{i,j}^{-1}$. All previously defined entities using no subindex will now use the subindex "i": h_i, N_i, $N_{S,i}$, H_i, δ_i, $\mathbf{B}_{i,\delta\delta}^{-1}$, $\mathbf{B}_{i,\delta 0}^{-1}$, $\mathbf{B}_{i,0\delta}^{-1}$, $\mathbf{B}_i^{-1}$, and $\mathbf{K}_i$, with the eventual notation $\mathbf{K}_{L-1} = \mathbf{K}$. The L meshes are not assumed to be either nested or structured. Let $\mathbf{I}_i$ denote the identity operator; for $i > 0$ let

$$\mathcal{I}_i^{i-1} : \mathbb{R}^{N_i} \to \mathbb{R}^{N_{i-1}}$$

denote a linear restriction operator from level i to level $i-1$, and let

$$\mathcal{I}_{i-1}^{i} : \mathbb{R}^{N_{i-1}} \to \mathbb{R}^{N_i}$$

denote a linear interpolation operator from level $i-1$ to level i. Given the iterate used for the computation of $\mathbf{K}_{L-1}$, the computation of coarse matrices $\mathbf{K}_i$ (i.e., $0 \leqslant i \leqslant L-2$) proceeds recursively from the finest coarse level $i = L-2$ until the coarsest level $i = 0$ by simply first restricting or injecting the finer iterate and then computing the Jacobian. Multilevel Schwarz preconditioners are obtained through the combination of one-level Schwarz preconditioners $\mathbf{B}_i^{-1}$ assigned to each level. Here we focus on multilevel preconditioners that can be seen as multigrid (MG) V-cycle algorithms [5] having Schwarz preconditioned Richardson working as the pre- and the postsmoother at each level $i > 0$, with $\mathbf{B}_{i,\mathrm{pre}}^{-1}$ preconditioning the $\mu_i \geqslant 0$ presmoother iterations and $\mathbf{B}_{i,\mathrm{post}}^{-1}$ preconditioning the $\nu_i \geqslant 0$ postsmoother iterations. In a general MG V-cycle algorithm with $L \geqslant 2$ levels, given the current iterate $\mathbf{p}^{(\ell)}$ for the solution of (2.1), the next iterate is computed as $\mathbf{p}^{(\ell+1)} = AlgV(\mathbf{b}, L, \mathbf{p}^{(\ell)})$, where the procedure

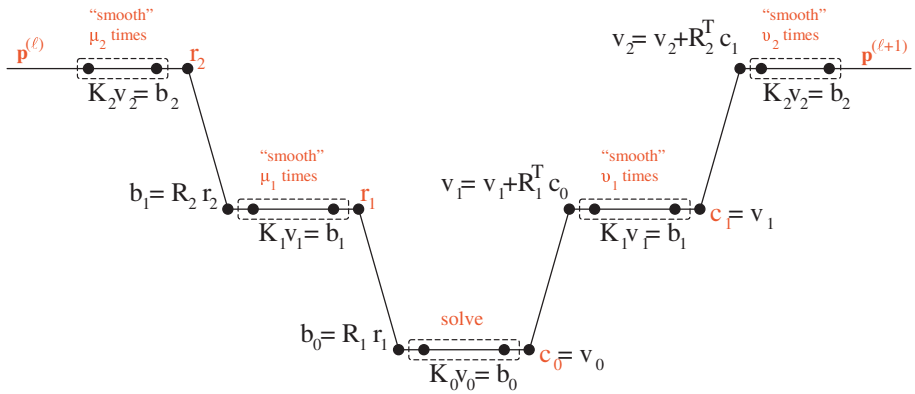


FIG. 2.1. *Graphical representation of a three-level MG V-cycle method:* $\mathbf{r}_1$ *and* $\mathbf{r}_2$ *denote residuals;* $\mathbf{c}_0$ *and* $\mathbf{c}_1$ *denote corrections.*

$\mathbf{v}_i = AlgV(\mathbf{b}_i, i, \mathbf{v}_i)$ consists of the following steps:

if $i = 0$
  Solve $\mathbf{K}_0\mathbf{v}_0 = \mathbf{b}_0$; /* Coarsest correction */
else
  Smooth μ_i times $\mathbf{K}_i\mathbf{v}_i = \mathbf{b}_i$: /* Presmoothing */
  $(\mathbf{b}_i - \mathbf{K}_i\mathbf{v}_i) = (\mathbf{I}_i - \mathbf{K}_i\mathbf{B}_{i,\text{pre}}^{-1})^{\mu_i}(\mathbf{b}_i - \mathbf{K}_i\mathbf{v}_i)$;
  $\mathbf{b}_{i-1} = \mathcal{I}_i^{i-1}(\mathbf{b}_i - \mathbf{K}_i\mathbf{v}_i)$; /* Residual restriction */
  $\mathbf{v}_{i-1} = AlgV(\mathbf{b}_{i-1}, i-1, 0)$; /* Recursivity */
  $\mathbf{v}_i = \mathbf{v}_i + \mathcal{I}_{i-1}^{i}\mathbf{v}_{i-1}$; /* Correction interpolation */
  Smooth ν_i times $\mathbf{K}_i\mathbf{v}_i = \mathbf{b}_i$: /* Postsmoothing */
  $(\mathbf{b}_i - \mathbf{K}_i\mathbf{v}_i) = (\mathbf{I}_i - \mathbf{K}_i\mathbf{B}_{i,\text{post}}^{-1})^{\nu_i}(\mathbf{b}_i - \mathbf{K}_i\mathbf{v}_i)$;
end

In this paper we consider multilevel Schwarz preconditioners with the coarsest correction computed as

$$\mathbf{v}_0 = \mathbf{B}_0^{-1}\mathbf{b}_0,$$

where $\mathbf{B}_0^{-1}$ might denote either a Schwarz preconditioner or an exact solver.

Then, as iterative methods for (2.1), with $\mathbf{r}^{(\ell)}$ denoting the residual at iteration $\ell = 0, 1, 2, \ldots$, such multilevel Schwarz procedures can be described in the case $L = 2$ as

$$\mathbf{r}^{(\ell+1)} = (\mathbf{I}_1 - \mathbf{K}_1\mathbf{B}_{1,\text{post}}^{-1})^{\nu_1}(\mathbf{I}_1 - \mathbf{K}_1\mathcal{I}_0^1\mathbf{B}_0^{-1}\mathcal{I}_1^0)(\mathbf{I}_1 - \mathbf{K}_1\mathbf{B}_{1,\text{pre}}^{-1})^{\mu_1}\mathbf{r}^{(\ell)}. \tag{2.2}$$

In the case of $L = 3$ and the usual choices of $\mathcal{I}_i^{i-1} = \mathbf{R}_i$ and $\mathcal{I}_{i-1}^{i} = \mathbf{R}_i^T$, for some $N_{i-1} \times N_i$ matrix $\mathbf{R}_i$, a graphical representation is given in Figure 2.1.

When classic Schwarz preconditioners are applied to symmetric positive definite systems arising from the discretization of elliptical problems defined in $H_0^1(\Omega)$, the condition number κ of the preconditioned system satisfies $\kappa \leqslant C\,(1 + H/\delta)\,/H^2$ for one-level methods and $\kappa \leqslant C\,(1 + H/\delta)$ for two-level methods, where C is independent of h, H, and δ. The factor $1/H^2$, associated to the number of subdomains on the fine level, relates itself to the increase on the number of iterations (needed for the exchange

of information among distant subdomains) with the increase in the total number of subdomains. The use of a coarse level helps the exchange of information. The necessity of information exchange among distant domain regions can be understood through the expression of the solutions of elliptic problems in terms of Green's functions: Although the solution value at a point strongly depends on surrounding values, there is weaker dependence w.r.t. the entire domain [32]. Regarding the application of two-level methods to indefinite model problems, the study in [10] suggests that the coarse mesh needs to be sufficiently fine for the two-level Schwarz preconditioner to perform well.

Theoretically, however, these results may not be directly applied to the case of symmetric indefinite KKT Jacobians. Let $\overline{\ell}$ be the average number of linear iterations per Newton step. We then look for the following more general properties when applying a two-level preconditioner:

(2.3) For fixed h and H/δ, $\overline{\ell}$ is not very sensitive to H decreasing;

(2.4) for fixed H and δ, $\overline{\ell}$ is not very sensitive to the mesh refinement.

Also, it might happen that a presumed "enhancement" of the preconditioner, aiming to cluster the eigenvalues around 1, eventually results in negative eigenvalues getting too close to 0. Although there is no necessary relationship between the eigenvalue distribution and ill-conditioning for general matrices [27], this process might qualitatively explain an eventual degradation of the linear convergence before it finally gets better [12, p. 199]. So,

(2.5) presumed enhancements of the preconditioner, e.g., an increase of H or δ, eventually result in a worse preconditioned Krylov convergence.

We report such temporary degradation in some of our numerical experiments in section 4.

3. Pollution removing interpolation. The motivation of including, additively or multiplicatively, coarse preconditioners to an existing fine mesh preconditioner is to make the overall preconditioner scalable w.r.t. the number of processors or the number of subdomains. In most cases, the addition of coarse preconditioners reduces the total number of iterations, as expected. However, for some classes of important problems, the unexpected happens: The number of iterations increases when a coarse preconditioner is involved. We refer to this phenomenon as coarse mesh pollution. To figure out the source of the pollution for a general problem is a highly nontrivial task. For the constrained optimization problems being considered we have discovered that the pollution is due to the coarse-to-fine mesh interpolation for one of the three types of variables (state variables, control variables, and Lagrange multipliers) of the problem, namely the Lagrange multipliers [29]. Due to the sharp jumps often encountered on the multiplier values over those regions of Ω where constraints are greatly affecting the behavior of the optimized system, we introduce a new interpolation for these particular components. Such modified interpolation constitutes, indeed, a key procedure in our proposed multilevel preconditioner. Although the evidence of this discontinuity property of Lagrange multipliers is just empirical in this paper, it is consistent with their interpretation (1.2): The value of a Lagrange multiplier at a mesh point gives the rate of change of the optimal objective function value w.r.t. to the respective constraint at that point.

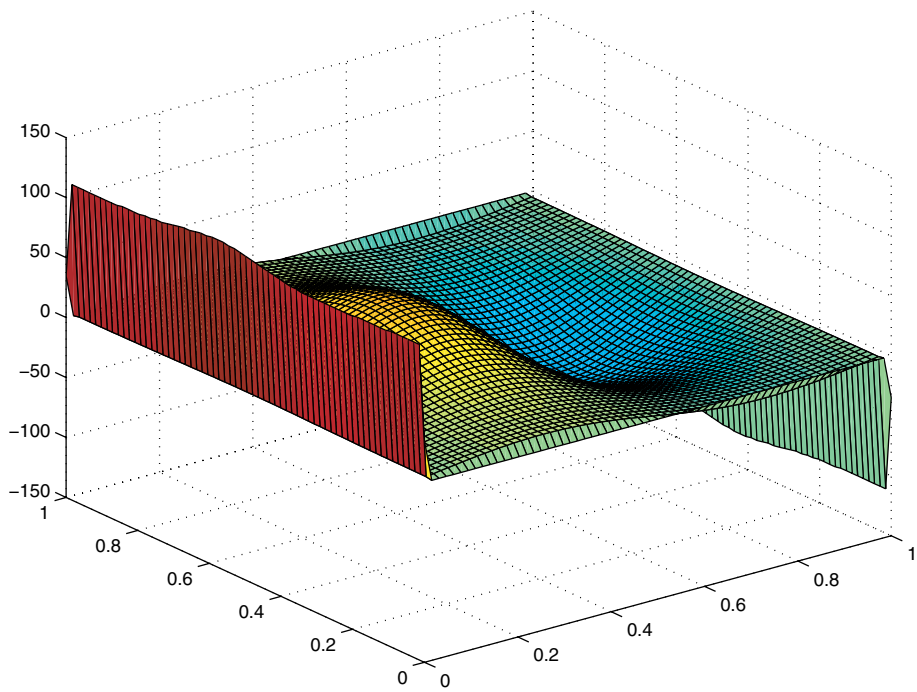


FIG. 3.1. *Lagrange multiplier* $\boldsymbol{\lambda}_1$ *related to the horizontal velocity* $\mathbf{v}_1$ *of a steady-state flow inside a unit square box with rigid slip walls.*

In the case of the problem corresponding to Figure 3.1, for instance, an external force causes the fluid to move clockwise, and the boundary consists of rigid slip walls (see Figure 4.1). The vertical walls greatly affect the overall vorticity throughout the domain, i.e., the value of the objective function, because they completely oppose the horizontal velocity component v_1. The values of λ_1 at the walls then reflect this situation. Similarly, λ_2 develops sharp jumps at the other two walls opposing v_2. In all of our experiments the discontinuities are located only across the boundary and not around it, even for very fine meshes. Common coarse-to-fine interpolation techniques will then smooth the sharp jumps present in coarse solutions, with a more gradual change, from interior mesh points towards boundary mesh points, appearing in those fine cells (elements, volumes) located inside coarse boundary ones, as represented in Figure 3.2(b). That is, the good correction information provided by the coarse solution is lost with a common interpolation. We refer to the smoothed jump as "pollution," in contrast to the "clean" sharp jump that is expected at the fine level as well.

We therefore propose a , , coarse-to-fine interpolation procedure that is based on a general and simple "removal of the pollution." Let $\mathcal{I}_{i-1}^{i}$ denote, as before, any unmodified interpolation operator, while

(3.1) $\boldsymbol{\mathcal{Z}}_i$ is the operator that zeros out, from a vector at level i, the Lagrange multipliers at all of those mesh points with equations that have a greater influence on the objective function.

For the case of PDEs describing physical systems, the number of such points can be expected to be relatively small. Our modified interpolation is then expressed by

$$\mathcal{I}_{i-1,\text{modif}}^{i} = \mathcal{I}_{i-1}^{i} - \boldsymbol{\mathcal{Z}}_i \mathcal{I}_{i-1}^{i} (\mathbf{I}_{i-1} - \boldsymbol{\mathcal{Z}}_{i-1}). \tag{3.2}$$

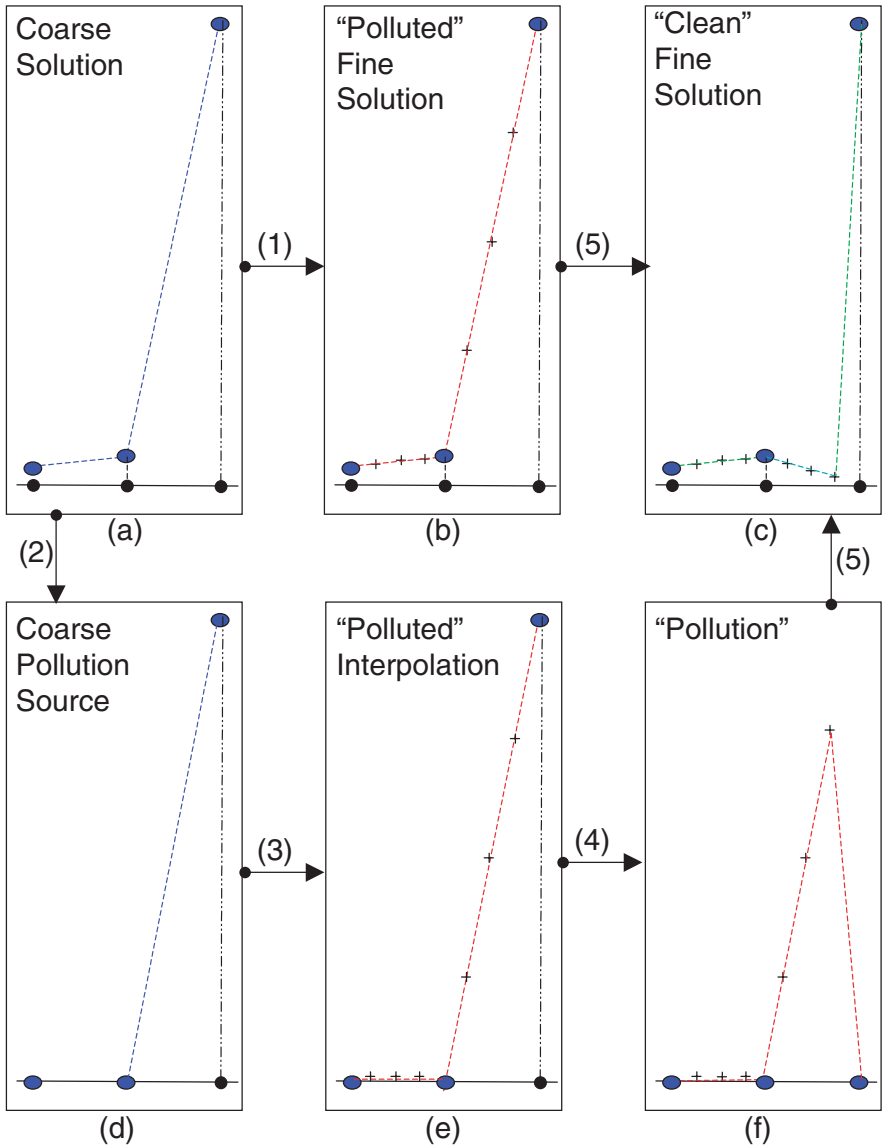


FIG. 3.2. *Representation of the modified coarse-to-fine interpolation* (3.2)*, with* (a) *input* $\boldsymbol{\varphi}_{i-1}$ *and* (c) *output* $\boldsymbol{\varphi}_i$*. The five steps are* (1) *interpolation* $\mathcal{I}_{i-1}^{i}\boldsymbol{\varphi}_{i-1}$*,* (2) *coarse jump values* $\tilde{\boldsymbol{\varphi}}_{i-1} = (\mathbf{I}_{i-1} - \mathcal{Z}_{i-1})\boldsymbol{\varphi}_{i-1}$*,* (3) *polluted* $\tilde{\boldsymbol{\varphi}}_i = \mathcal{I}_{i-1}^{i}\tilde{\boldsymbol{\varphi}}_{i-1}$*,* (4) *pollution isolation* $\mathcal{Z}_i\tilde{\boldsymbol{\varphi}}_i$*, and* (5) *pollution removal* $\boldsymbol{\varphi}_i = \mathcal{I}_{i-1}^{i}\boldsymbol{\varphi}_{i-1} - \mathcal{Z}_i\tilde{\boldsymbol{\varphi}}_i$*.*

This procedure removes the smoothed contributions due to the coarse discontinuities, maintaining, at the fine level, the sharp jumps originally present at the coarse level. It should be noted that the operator $\mathcal{I}_{i-1}^{i}$ is usually available for the coding of a multilevel procedure, and, once the locations where the operator $\boldsymbol{\mathcal{Z}}_i$ needs to act over are known, the implementation of (3.2) is extremely easy and can be performed on ' ' mesh in ' ' dimension, with ' ' number of components.

The Lagrange multipliers reflect an eventual "discontinuity" of the type of equations (or their physical dimensions) between equations in different regions of $\overline{\Omega}$. From

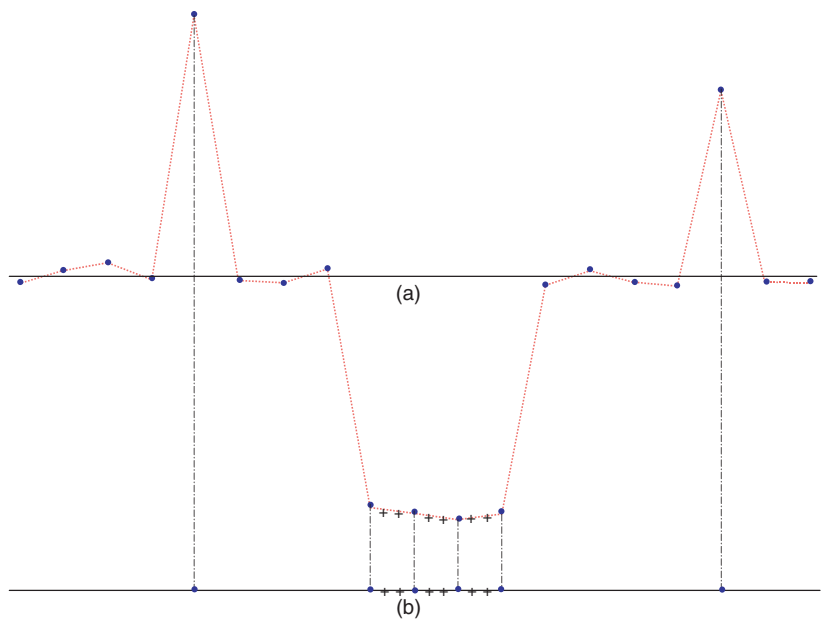


FIG. 3.3. (a) *A schematic example of a possible distribution of Lagrange multiplier components throughout a coarse mesh at level* $i-1$. (b) *The corresponding coarse (and fine) mesh points where the operator* $\mathcal{Z}_{i-1}$ *(and* $\mathcal{Z}_i$*) would act over. The plus sign refers to the fine mesh points where* $\mathcal{Z}_i$ *would act over as well.*

this point of view, it seems "natural" to apply different interpolations to the Lagrange multipliers depending on their location on $\overline{\Omega}$. In the case of the problems in section 4, for instance, a clear difference exists between those equations in Ω and those on $\partial\Omega$, that is, for those problems the operator $\boldsymbol{\mathcal{Z}}_i$ zeros the Lagrange multiplier components located at the boundary. A schematic representation of (3.2) for such a situation is then given in Figure 3.2 for the simpler case of a piecewise linear interpolation. However, the modified interpolation (3.1)–(3.2) is [illegible] at all restricted to situations where the discontinuities are located at the boundary. Figure 3.3(a) gives a schematic example where Lagrange multiplier discontinuities are spread throughout different regions of the domain. With the specification of the region where the operator $\boldsymbol{\mathcal{Z}}_i$ acts over, given by Figure 3.3(b), the pollution (smoothed jumps) created by the usual interpolation operators $\mathcal{I}_{i-1}^{i}$ will be removed in a way similar to the way exemplified in the simpler situation of Figure 3.2.

Whatever the shape of the Lagrange multipliers in the final solution of a PDE-constrained optimization problem, the eventual sharp jumps will tend to appear more on the first Newton steps and, at each Newton iteration, on the coarse corrections related to the first Krylov iterations. As both Newton and Krylov iterations progress towards the respective solutions, the steps and corrections will naturally shrink in magnitude. Nonetheless, even when sharp jumps are not involved anymore, the modified interpolation can still be applied with no harm: The interpolation of an eventual smooth coarse correction with small values on the Lagrange multiplier components will still present small values at the fine level. This fact facilitates the programming of (3.1)–(3.2), since the application of the modified interpolation is not conditional.

In our tests we apply the modified interpolation only for the Lagrange multiplier components of coarse corrections, while the optimization variables continue to be

interpolated with the unmodified process $\mathcal{I}_{i-1}^{i}$. Also, the restriction process remains $\mathcal{I}_{i}^{i-1}$ for all variables; i.e., (2.2) becomes

$$\mathbf{r}^{(\ell+1)} = (\mathbf{I}_1 - \mathbf{K}_1\mathbf{B}_{1,\text{post}}^{-1})^{\nu_1}(\mathbf{I}_1 - \mathbf{K}_1\mathcal{I}_{0,\text{modif}}^{1}\mathbf{B}_0^{-1}\mathcal{I}_1^0)(\mathbf{I}_1 - \mathbf{K}_1\mathbf{B}_{1,\text{pre}}^{-1})^{\mu_1}\mathbf{r}^{(\ell)}.$$

Three important aspects of our algorithm should be highlighted before we proceed. First, operator (3.2) is nonlinear, even for linear operators $\mathcal{I}_{i-1}^{i}$. The multilevel preconditioner is then nonlinear as well. The linearity of preconditioners is theoretically desired, because one can then guarantee the same solution when going from the original system (2.1) to the preconditioned one. In our tests, however, the same solution was obtained with both linear and nonlinear preconditioners.

Second, the application of the modification process to an already modified interpolation operator gives the same modified operator: Denoting

$$\mathcal{I}_{i-1,\text{modif,modif}}^{i} = \mathcal{I}_{i-1,\text{modif}}^{i} - \boldsymbol{\mathcal{Z}}_i\mathcal{I}_{i-1,\text{modif}}^{i}(\mathbf{I}_{i-1} - \boldsymbol{\mathcal{Z}}_{i-1}),$$

we have

$$\begin{aligned}
\boldsymbol{\mathcal{Z}}_i\mathcal{I}_{i-1,\text{modif}}^{i}(\mathbf{I}_{i-1} - \boldsymbol{\mathcal{Z}}_{i-1}) &= \boldsymbol{\mathcal{Z}}_i\left[\mathcal{I}_{i-1}^{i} - \boldsymbol{\mathcal{Z}}_i\mathcal{I}_{i-1}^{i}(\mathbf{I}_{i-1} - \boldsymbol{\mathcal{Z}}_{i-1})\right](\mathbf{I}_{i-1} - \boldsymbol{\mathcal{Z}}_{i-1}) \\
&= \boldsymbol{\mathcal{Z}}_i\left[\mathcal{I}_{i-1}^{i}(\mathbf{I}_{i-1} - \boldsymbol{\mathcal{Z}}_{i-1}) - \boldsymbol{\mathcal{Z}}_i\mathcal{I}_{i-1}^{i}(\mathbf{I}_{i-1} - \boldsymbol{\mathcal{Z}}_{i-1})^2\right] \\
&= \boldsymbol{\mathcal{Z}}_i\left[\mathcal{I}_{i-1}^{i}(\mathbf{I}_{i-1} - \boldsymbol{\mathcal{Z}}_{i-1}) - \boldsymbol{\mathcal{Z}}_i\mathcal{I}_{i-1}^{i}(\mathbf{I}_{i-1} - \boldsymbol{\mathcal{Z}}_{i-1})\right] \\
&= \boldsymbol{\mathcal{Z}}_i(\mathbf{I}_i - \boldsymbol{\mathcal{Z}}_i)\mathcal{I}_{i-1}^{i}(\mathbf{I}_{i-1} - \boldsymbol{\mathcal{Z}}_{i-1}) \\
&= (\boldsymbol{\mathcal{Z}}_i - \boldsymbol{\mathcal{Z}}_i^2)\mathcal{I}_{i-1}^{i}(\mathbf{I}_{i-1} - \boldsymbol{\mathcal{Z}}_{i-1}), \\
&= (\boldsymbol{\mathcal{Z}}_i - \boldsymbol{\mathcal{Z}}_i)\mathcal{I}_{i-1}^{i}(\mathbf{I}_{i-1} - \boldsymbol{\mathcal{Z}}_{i-1}),
\end{aligned}$$

the null operator. This result might be important for eventual theoretical demonstrations of preconditioning properties.

Third, procedure (3.2) assumes that

(3.3) the neighborhood of the mesh points where operator $\boldsymbol{\mathcal{Z}}_i$ acts upon has Lagrange multipliers with values close to zero,

so that the term $(\mathbf{I}_{i-1} - \boldsymbol{\mathcal{Z}}_{i-1})$ reflects the true relative discontinuity between the Lagrange multiplier values at the points where operator $\boldsymbol{\mathcal{Z}}_i$ acts upon (acting points) and the Lagrange multiplier values at the neighboring points. If (3.3) is not valid, then operator $\boldsymbol{\mathcal{Z}}_{i-1}$ in (3.2) should be replaced by an operator $\boldsymbol{\mathcal{T}}_{i-1}$ that sets the Lagrangian multiplier values at the acting points equal to the neighboring values instead. In the case of OTD approaches, where explicit PDEs and boundary conditions are obtained for the Lagrange multipliers before discretization, the correct extension of values from the neighboring to the acting points might be known ahead of time; otherwise, the implementation of $\boldsymbol{\mathcal{T}}_{i-1}$ would be more difficult than that of $\boldsymbol{\mathcal{Z}}_{i-1}$. However, we expect (3.3) to be valid for many optimization problems of physical systems, e.g., (4.1)–(4.2) in section 4.

We remark that pollution removing is an efficient way of dealing with the potential discontinuity between Lagrange multipliers related to PDEs and Lagrange multipliers related to boundary constraints, and it does not assume that the latter constitute the traces of the former. In the case of DTO, the Lagrange multipliers are already finite dimensional vectors in their first appearance in the solution process, and so one does not need to deal with traces. Since the collection of mesh points associated to

a PDE and the collection of mesh points associated to the corresponding boundary conditions are disjoint and cover the entire mesh, it is convenient, for implementation purposes, to join Lagrange multipliers into a unique programming variable defined over the entire mesh and suitable for the application of pollution removing.

4. Numerical experiments. Our numerical experiments in this paper focus on optimal control problems [19], where the optimization space in (1.1) is generally given by $\mathbf{W}=\mathbf{S}\times\mathbf{U}$, with $\mathbf{S}$ being the state space and $\mathbf{U}$ the control space. More specifically, we treat the boundary control of two-dimensional steady-state incompressible Navier–Stokes equations in the velocity-vorticity formulation [20]: $\mathbf{v}=(v_1,v_2)$ is the velocity and ω is the vorticity. Let $\Omega\subset\mathbb{R}^2$ be an open and bounded smooth domain, $\Gamma=\partial\Omega$ its boundary, $\boldsymbol{\nu}$ the unit outward normal vector along Γ, and $\mathbf{f}$ a given external force defined in Ω. Let $L^2(\Omega)$ and $L^2(\Gamma_u)$ be the spaces of square Lebesgue integrable real functions in Ω and $\Gamma_u\subset\Gamma$, respectively. The control problems consist on finding $(\mathbf{s},\mathbf{u})=(v_1,v_2,\omega,u_1,u_2)\in\mathbf{S}\times\mathbf{U}=L^2(\Omega)^3\times L^2(\Gamma_u)^2$ such that the minimization

$$\min_{(\mathbf{s},\mathbf{u})\in\mathbf{S}\times\mathbf{U}}\mathcal{F}(\mathbf{s},\mathbf{u})=\frac{1}{2}\int_\Omega\omega^2\,d\Omega+\frac{c}{2}\int_{\Gamma_u}\|\mathbf{u}\|_2^2\,d\Gamma_u \tag{4.1}$$

is achieved subject to the constraints

$$\left\{\begin{array}{lcll}
-\Delta v_1-\dfrac{\partial\omega}{\partial x_2} & = & 0 & \text{in } \Omega,\\
-\Delta v_2+\dfrac{\partial\omega}{\partial x_1} & = & 0 & \text{in } \Omega,\\
-\Delta\omega+\text{Re } v_1\dfrac{\partial\omega}{\partial x_1}+\text{Re } v_2\dfrac{\partial\omega}{\partial x_2}-\text{Re curl } \mathbf{f} & = & 0 & \text{in } \Omega,\\
v_i-v_{D,i} & = & 0 & \text{on } \Gamma_{D,i},\quad i=1,2,\\
\dfrac{\partial v_i}{\partial\boldsymbol{\nu}}-v_{N,i} & = & 0 & \text{on } \Gamma_{N,i},\quad i=1,2,\\
\mathbf{v}-\mathbf{u} & = & \mathbf{0} & \text{on } \Gamma_u,\\
\omega+\dfrac{\partial v_1}{\partial x_2}-\dfrac{\partial v_2}{\partial x_1} & = & 0 & \text{on } \Gamma,\\
\int_\Gamma\mathbf{v}\cdot\boldsymbol{\nu}\,d\Gamma & = & 0, &
\end{array}\right. \tag{4.2}$$

where curl $\mathbf{f}=-\partial f_1/\partial x_2+\partial f_2/\partial x_1$ and, for $i=1,2$, $\Gamma=\Gamma_{D,i}\cup\Gamma_{N,i}\cup\Gamma_u$ and $\Gamma_{D,i}$ ($\Gamma_{N,i}$) is the part of the boundary where the velocity component v_i is specified through a Dirichlet (Neumann) condition with a prescribed velocity $v_{D,i}$ ($v_{N,i}$). The parameter $c>0$ is used to adjust the relative importance of the control norms in achieving the minimization, so indirectly constraining their sizes. The physical objective in (4.1)–(4.2) is the minimization of turbulence [18]. The last constraint is necessary for consistency with the physical law of mass conservation, making $m\neq n_s$ and causing the complexity of the parallel finite difference approximation of Jacobians to increase, since nonadjacent mesh points become coupled by the integral. An alternative formulation, that compromises between the physical law of mass conservation and the complex computation of Jacobians, eliminates the integral constraint but adds to the objective function the term $\tilde{c}\left[\int_\Gamma\mathbf{v}\cdot\boldsymbol{\nu}\,d\Gamma\right]^2/2$, with $\tilde{c}\gg 1$ [4]. We restrict our numerical experiments to [illegible] boundary control problems, i.e., $\mathbf{u}\cdot\boldsymbol{\nu}=0$ on Γ_u, and the velocity $\mathbf{v}$ is assumed to guarantee mass conservation along $\Gamma\setminus\Gamma_u$, so that $m=n_s$.

We consider rectangular domains $\Omega=(0,L_1)\times(0,L_2)$, $L_2\leqslant L_1$. We define $E_1=\{(x_1,0)\in\mathbb{R}^2:0<x_1<L_1\}$, $E_{1,a}=\{(x_1,x_2)\in E_1:0<x_1\leqslant L_2\}$, $E_{1,b}=$

$E_1 \setminus E_{1,a}$, $E_2 = \{(L_1, x_2) \in \mathbb{R}^2 : 0 < x_2 < L_2\}$, $E_3 = \{(x_1, L_2) \in \mathbb{R}^2 : 0 < x_1 < L_1\}$, $E_4 = \{(0, x_2) \in \mathbb{R}^2 : 0 < x_2 < L_2\}$, $E_{4,a} = \{(x_1, x_2) \in E_4 : \frac{L_2}{2} \leqslant x_2 < L_2\}$, and $E_{4,b} = E_4 \setminus E_{4,a}$, $C_1 = (0, 0)$. Two flow problems are considered: cavity and backward-facing step. In each case we consider both simulation problems and tangential boundary control problems. In the descriptions below we define only Γ_u and the velocity boundary conditions on $\Gamma \setminus \Gamma_u$, since all control problems seek the minimization of the objective function (4.1) with $c = 10^{-2}$ and have in common the three PDEs in Ω, the ω boundary condition on Γ, and the tangential boundary control on Γ_u. In the case of [illegible] flows, $L_1 = L_2 = 1$, $\Gamma_u = \Gamma$, and $\mathbf{f} = (f_1, f_2) = \big(-\sin^2(\pi x_1)\cos(\pi x_2)\sin^2(\pi x_2), \sin^2(\pi x_2)\cos(\pi x_1)\sin^2(\pi x_1)\big)$. In the case of [illegible] flows, $L_1 = 6$, $L_2 = 1$, $\Gamma_u = E_{4,b} \cup \{C_1\} \cup E_{1,a}$, and $\mathbf{f} = \mathbf{0}$. The velocity boundary conditions on $\Gamma \setminus \Gamma_u$ are $v_1 = 0$ on $E_{1,b} \cup \overline{E_3}$, $v_1 = 8(1 - x_2)(x_2 - 1/2)$ on $E_{4,a}$, $v_1 = x_2(1 - x_2)$ on $\overline{E_2}$, and $v_2 = 0$. All corresponding simulation problems, used for comparison with control problems, impose $\mathbf{v} \cdot \boldsymbol{\nu} = 0$ and $\partial \mathbf{v}/\partial \boldsymbol{\nu} = 0$ on Γ_u; i.e., Γ_u becomes a (set of) slip rigid wall(s).

4.1. Details of numerical approaches. We use a five-point second order finite difference method on a uniform nonstaggered mesh. We divide the horizontal edges on $N_{1,\mathrm{div}}$, equally spacing subintervals, and the vertical edges on $N_{2,\mathrm{div}}$, also equally spacing subintervals, generating a rectangular grid with a total of $N_g = (1+N_{1,\mathrm{div}})(1+N_{2,\mathrm{div}})$ points. We denote $h_1 = L_1/N_{1,\mathrm{div}}$ and $h_2 = L_2/N_{2,\mathrm{div}}$. Each grid point is assigned an integer index k and denoted $\mathbf{x}_k = (x_{1,k}, x_{2,k})$, $k \in K = \{k \in \mathbb{N},\ 0 \leqslant k < N_g\}$. K_b is the set of indexes related to the $N_b = 2(N_{1,\mathrm{div}} + N_{2,\mathrm{div}})$ grid points located at the boundary, l_k is the elementary boundary length surrounding a point $\mathbf{x}_k$, $k \in K_b$, and a_k is the elementary area surrounding a point $\mathbf{x}_k$, $k \in K$; that is,

$$l_k = h_1(\text{or } h_2),\ a_k = \frac{h_1 h_2}{2} \text{ for points on horizontal (or vertical) edges,}$$

$$l_k = \frac{h_1 + h_2}{2},\ a_k = \frac{h_1 h_2}{4} \text{ for points on corners,}$$

and $a_k = h_1 h_2$ for $k \in K \setminus K_b$. The total number of discrete state variables is $n_s = 3N_g$. For code implementation convenience, both discrete control components are defined everywhere in the domain; i.e., the total number of discrete control variables is $n_u = 2N_g$. There is no problem with such an approach since all control components not used as Dirichlet controls (i.e., not appearing in the constraints) are automatically forced to zero in the optimality system. The discrete state space is $\mathbf{S}_h = \mathbb{R}^{n_s}$, and the discrete control space is $\mathbf{U}_h = \mathbb{R}^{n_u}$. The discretized version of (4.1) then reads

$$\min_{(\mathbf{s},\mathbf{u}) \in \mathbf{S}_h \times \mathbf{U}_h} \mathcal{F}_h(\mathbf{s}, \mathbf{u}) = \frac{1}{2} \sum_{k \in K} \omega_k^2 a_k + \frac{c}{2} \sum_{k \in K_b} \|\mathbf{u}_k\|_2^2 l_k + \frac{c}{2} \sum_{k \in K \setminus K_b} \|\mathbf{u}_k\|_2^2 a_k.$$

All derivative terms in (4.2) are discretized with a second order scheme, including the ω boundary condition. For parallel performance reasons, only mesh points adjacent to the boundary are used by applying the definition $\omega = -\partial v_1/\partial x_2 + \partial v_2/\partial x_1$ at these points as well [28].

In order to form the algebraic system of nonlinear discretized equations, we need to order the unknowns and the corresponding functions. The unknowns are ordered mesh point by mesh point, in contrast to physical variable by physical variable as usually required by other methods. At each mesh point the unknowns are ordered as $v_1, v_2, \omega, u_1, u_2, \lambda_1, \lambda_2$, and λ_3. The mesh points are ordered subdomain by subdomain, for the purpose of parallel processing. The ordering of the subdomains is

not important, since we use, at each level other than the coarsest, additive methods whose performance has nothing to do with the subdomain ordering. In order to avoid pivoting during the sparse LU method (used in our experiments), the corresponding functions are ordered as $\nabla_{\lambda_1}\mathcal{L}$,$\nabla_{\lambda_2}\mathcal{L}$,$\nabla_{\lambda_3}\mathcal{L}$, $\nabla_{u_1}\mathcal{L}$,$\nabla_{u_2}\mathcal{L}$, $\nabla_{v_1}\mathcal{L}$,$\nabla_{v_2}\mathcal{L}$, and $\nabla_{\omega}\mathcal{L}$. Because the orderings for the unknowns and for the function components are different, the Jacobian matrix becomes nonsymmetric and so we use GMRES [30].

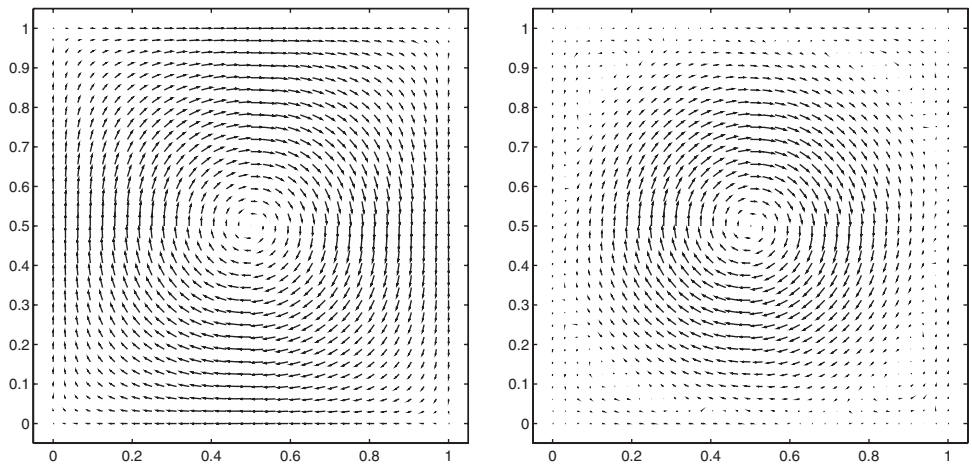
The Jacobian matrix is constructed approximately using a multicolored central finite difference method [11] with step size 10^{-5}. In problem (4.1)–(4.2), since there is no variable with power greater than two, central finite difference approximations of the Jacobian are exact up to roundoff errors. To solve the Jacobian systems we use restarted GMRES with an absolute (relative) tolerance equal to 10^{-8} (10^{-4}), a restart parameter equal to 90, and a maximum number of iterations equal to 5,000. When using left preconditioning, the GMRES tolerances are checked over preconditioned residuals. Regarding the one-level additive Schwarz preconditioner, the number of subdomains is equal to the number of processors, and the extended subdomain problems have zero Dirichlet interior boundary conditions and are solved with sparse LU. All subdomains Ω_j and $\Omega_j^{'}$ are rectangular and made up of an integral number of mesh cells. The computation of coarse matrices at each Newton iteration proceeds recursively from the finest coarse level until the coarsest level by simply first injecting the finer iterate and then computing the Jacobian. We use nested meshes, the redundant LU solver at the coarsest level (all processors construct and solve a full coarsest linear system by sending their local portions of the Jacobian and of the right-hand side to all other processors), and the interpolation is piecewise linear. Richardson smoothers are all used with a damping factor equal to 1. A line search is performed with an augmented Lagrangian merit function and cubic backtracking. For Newton iterations, the maximum allowed number is 100 and the absolute (relative) stopping tolerance is 10^{-6} (10^{-10}). Simulation problems are solved with Newton–Krylov–Schwarz [7]. We do not use Reynolds continuation in any of the algorithms.

If not stated otherwise, the modified interpolation process explained in section 3 is always used in multilevel tests.

4.2. Results of numerical experiments. We have performed tests on a cluster of Linux PCs and developed our parallel object-oriented software using the C++ programming language and the Portable, Extensible Toolkit for Scientific Computing (PETSc) from Argonne National Laboratory [1]. Since the cluster network is relatively slow and is shared with other processes, and since the redundant LU solver used at the coarsest level is not scalable in time w.r.t. the number of processors, our analysis focuses on the number of Newton and Krylov iterations, not on computing times.

4.2.1. Cavity flows. Figure 4.1(b) shows the controlled velocity field for the cavity flow with Re = 200: Although the fluid in the interior follows the clockwise direction imposed by the external force, the movement near the boundary is much less intense than in Figure 4.1(a). In fact, a zoom on the boundary velocity of Figure 4.1(b) shows the control acting in the counterclockwise way, making the integral of ω^2 on Ω decrease from the value of 55.4 for the simulation with slip walls to the controlled value of 32.5. Figure 3.1 shows the sharp jump of the Lagrangian multiplier function λ_1 at the boundary. The other two Lagrangian multipliers behave similarly, no matter how fine the mesh is.

Tables 4.1–4.4 show results for one-level preconditioners. In Tables 4.1–4.2 we fix the mesh size in 280×280 and the preconditioning side to , . Then, for Re = 200 and 300, respectively, we perform tests with 25 and 49 processors, varying the combi-



(a) (b)

FIG. 4.1. *Computed velocity fields for the cavity flow problems with* $\mathrm{Re} = 200$*:* (a) *simulation problem and* (b) *tangential boundary control problem. The Lagrange multiplier* λ_1 *corresponding to* v_1 *is shown in Figure* 3.1.

TABLE 4.1

One-level left *Schwarz preconditioner results for the cavity flow control problem with* $\mathrm{Re} = \mathbf{200}$ *and a* 280×280 *mesh (*631,688 *variables), for different combinations of the number of processors* N_p*, the relative overlap* δ/H*, and the Schwarz preconditioner* $\mathbf{B}^{-1}$*.* k *is the total number of Newton iterations, and* $\overline{\ell}$ *is the average number of Krylov iterations per Newton iteration.*

$\mathbf{B}^{-1}$	N_p	Relative overlap δ/H		
		1/8	1/4	1/2
ASM	25	$k = 6$; $\overline{\ell} \approx \mathbf{110}$	$k = 7$; $\overline{\ell} \approx \mathbf{64}$	$k = 7$; $\overline{\ell} \approx \mathbf{60}$
	49	$k = 6$; $\overline{\ell} \approx \mathbf{196}$	$k = 6$; $\overline{\ell} \approx \mathbf{148}$	$k = 7$; $\overline{\ell} \approx \mathbf{142}$
l-RAS	25	$k = 7$; $\overline{\ell} \approx \mathbf{114}$	$k = 7$; $\overline{\ell} \approx \mathbf{67}$	$k = 8$; $\overline{\ell} \approx \mathbf{52}$
	49	$k = 6$; $\overline{\ell} \approx \mathbf{205}$	$k = 7$; $\overline{\ell} \approx \mathbf{158}$	$k = 7$; $\overline{\ell} \approx \mathbf{83}$
r-RAS	25	$k = 7$; $\overline{\ell} \approx \mathbf{113}$	$k = 7$; $\overline{\ell} \approx \mathbf{67}$	$k = 7$; $\overline{\ell} \approx \mathbf{50}$
	49	$k = 6$; $\overline{\ell} \approx \mathbf{238}$	$k = 7$; $\overline{\ell} \approx \mathbf{356}$	$k = 7$; $\overline{\ell} \approx \mathbf{540}$

nations of Schwarz preconditioner type (ASM, l-RAS, or r-RAS) and relative overlap (1/8, 1/4, or 1/2). In Table 4.3, we do the same for $\mathrm{Re} = 300$, only fixing the preconditioning side to [illegible]. Most of the results in these first three tables are consistent with the behavior of one-level Schwarz preconditioners for positive-definite systems: The average number $\overline{\ell}$ of Krylov iterations per Newton iteration grows with the number N_p of processors and decreases as the overlap gets larger. Whenever some Newton step demanded 5,000 Krylov iterations to be computed, the overall test was considered to fail. This was the case in the six tests not reported in Tables 4.2 and 4.3.

The results in these first three tables suggest that the more robust combination is the [illegible] δ/H 1/2, which is then fixed for the next set of tests, shown in Table 4.4, with varying Re, mesh size, and number of processors, with the number of processors going up to 100 now. However, even with the pretty large relative overlap $\delta/H = 1/2$, the method fails for Re17 = 250 with 100 processors.

Clearly, then, some stabilization is needed. This stabilization is achieved with the use of a coarse mesh and the modified interpolation process explained in section 3. The

TABLE 4.2

One-level left *Schwarz preconditioner results for the cavity flow control problem with* Re = **300** *and a* 280×280 *mesh (*631,688 *variables), for different combinations of the number of processors* N_p*, the relative overlap* δ/H*, and the Schwarz preconditioner* $\mathbf{B}^{-1}$*. k is the total number of Newton iterations, and* $\overline{\ell}$ *is the average number of Krylov iterations per Newton iteration.*

$\mathbf{B}^{-1}$	N_p	Relative overlap δ/H		
		1/8	1/4	1/2
ASM	25	$k = 11$; $\overline{\ell} \approx$ **123**	$k = 10$; $\overline{\ell} \approx$ **84**	$k = 11$; $\overline{\ell} \approx$ **111**
	49	$k = 10$; $\overline{\ell} \approx$ **324**	$k = -$; $\overline{\ell} \approx -$	$k = 11$; $\overline{\ell} \approx$ **333**
l-RAS	25	$k = 11$; $\overline{\ell} \approx$ **165**	$k = 10$; $\overline{\ell} \approx$ **96**	$k = 11$; $\overline{\ell} \approx$ **74**
	49	$k = 10$; $\overline{\ell} \approx$ **454**	$k = -$; $\overline{\ell} \approx -$	$k = 11$; $\overline{\ell} \approx$ **216**
r-RAS	25	$k = 11$; $\overline{\ell} \approx$ **172**	$k = 10$; $\overline{\ell} \approx$ **86**	$k = 11$; $\overline{\ell} \approx$ **72**
	49	$k = 10$; $\overline{\ell} \approx$ **498**	$k = -$; $\overline{\ell} \approx -$	$k = 10$; $\overline{\ell} \approx$ **180**

TABLE 4.3

One-level right *Schwarz preconditioner results for the cavity flow control problem with* Re = **300** *and a* 280×280 *mesh (*631,688 *variables), for different combinations of the number of processors* N_p*, the relative overlap* δ/H*, and the Schwarz preconditioner* $\mathbf{B}^{-1}$*. k is the total number of Newton iterations, and* $\overline{\ell}$ *is the average number of Krylov iterations per Newton iteration.*

$\mathbf{B}^{-1}$	N_p	Relative overlap δ/H		
		1/8	1/4	1/2
ASM	25	$k = 10$; $\overline{\ell} \approx$ **184**	$k = 10$; $\overline{\ell} \approx$ **109**	$k = 10$; $\overline{\ell} \approx$ **108**
	49	$k = 10$; $\overline{\ell} \approx$ **413**	$k = -$; $\overline{\ell} \approx -$	$k = 10$; $\overline{\ell} \approx$ **313**
l-RAS	25	$k = 10$; $\overline{\ell} \approx$ **231**	$k = 10$; $\overline{\ell} \approx$ **136**	$k = 10$; $\overline{\ell} \approx$ **82**
	49	$k = 10$; $\overline{\ell} \approx$ **494**	$k = -$; $\overline{\ell} \approx -$	$k = 10$; $\overline{\ell} \approx$ **266**
r-RAS	25	$k = 10$; $\overline{\ell} \approx$ **201**	$k = 10$; $\overline{\ell} \approx$ **122**	$k = 10$; $\overline{\ell} \approx$ **67**
	49	$k = 10$; $\overline{\ell} \approx$ **646**	$k = -$; $\overline{\ell} \approx -$	$k = 10$; $\overline{\ell} \approx$ **202**

TABLE 4.4

One-level left *l-RAS preconditioner results for the cavity flow control problem with relative overlap* $\delta/H = 1/2$*, for different combinations of* Re*, the number of processors* N_p*, and the mesh size. k is the total number of Newton iterations, and* $\overline{\ell}$ *is the average number of Krylov iterations per Newton iteration. The number of variables is* 2,517,768 *on the case of finest mesh.*

Re	N_p	Mesh		
		140×140	280×280	560×560
200	25	$k = 8$; $\overline{\ell} \approx$ **46**	$k = 8$; $\overline{\ell} \approx$ **52**	Out of memory
	49	$k = 8$; $\overline{\ell} \approx$ **73**	$k = 7$; $\overline{\ell} \approx$ **83**	Out of memory
	100	$k = 8$; $\overline{\ell} \approx$ **247**	$k = 7$; $\overline{\ell} \approx$ **277**	$k = 6$; $\overline{\ell} \approx$ **205**
250	25	$k = 12$; $\overline{\ell} \approx$ **53**	$k = 9$; $\overline{\ell} \approx$ **57**	Not tested
	49	$k = 13$; $\overline{\ell} \approx$ **112**	$k = 9$; $\overline{\ell} \approx$ **117**	Not tested
	100	$k = -$; $\ell^{(10)}$=5,000	$k = -$; $\ell^{(5)}$=5,000	Not tested

results so far suggest that one can focus on RAS preconditioners on the Richardson smoothers and not use ASM. We, however, made some experiments with an ASM preconditioned smoother, and in all cases the preconditioned GMRES diverged, with either left or right preconditioning. We also tried to avoid tests with $\delta = H/2$ as well, in order to check if the two-level method can be used with smaller and larger overlaps.

In Table 4.5 we fix the mesh size in 280×280 the preconditioning side to , , , and, for Re = 200 and 250, we test two-level Schwarz with 25 and 49 processors, varying the

TABLE 4.5

Two-level left *Schwarz preconditioner results for the cavity flow control problem with a* 280×280 *mesh (*$631,688$ *variables), for different combinations of* Re*, the number of processors* N_p*, the relative overlap* δ/H*, and the Schwarz preconditioner* $\mathbf{B}^{-1}$ *for the Richardson smoother.* k *is the total number of Newton iterations, and* $\overline{\ell}$ *is the average number of Krylov iterations per Newton iteration.*

Re	$\mathbf{B}^{-1}$	N_p	Relative overlap δ/H	
			1/8	1/4
200	l-RAS	25	$k = 8$; $\overline{\ell} \approx \mathbf{16}$	$k = -$; $\overline{\ell} \approx -$
		49	$k = 7$; $\overline{\ell} \approx \mathbf{16}$	$k = 8$; $\overline{\ell} \approx \mathbf{13}$
	r-RAS	25	$k = -$; $\overline{\ell} \approx -$	$k = 8$; $\overline{\ell} \approx \mathbf{19}$
		49	$k = 10$; $\overline{\ell} \approx \mathbf{32}$	$k = 8$; $\overline{\ell} \approx \mathbf{22}$
250	l-RAS	25	$k = -$; $\overline{\ell} \approx -$	$k = -$; $\overline{\ell} \approx -$
		49	$k = -$; $\overline{\ell} \approx -$	$k = 10$; $\overline{\ell} \approx \mathbf{18}$
	r-RAS	25	$k = 12$; $\overline{\ell} \approx \mathbf{24}$	$k = 12$; $\overline{\ell} \approx \mathbf{17}$
		49	$k = 20$; $\overline{\ell} \approx \mathbf{36}$	$k = 11$; $\overline{\ell} \approx \mathbf{24}$

TABLE 4.6

Two-level right *Schwarz preconditioner results for the cavity flow control problem with* Re $=$ **200** *and a* 280×280 *mesh (*$631,688$ *variables), for different combinations of the number of processors* N_p*, the relative overlap* δ/H*, and the Schwarz preconditioner* $\mathbf{B}^{-1}$ *for the Richardson smoother.* k *is the total number of Newton iterations, and* $\overline{\ell}$ *is the average number of Krylov iterations per Newton iteration.*

$\mathbf{B}^{-1}$	N_p	Relative overlap δ/H	
		1/8	1/4
l-RAS	25	$k = 6$; $\overline{\ell} \approx \mathbf{20}$	$k = 6$; $\overline{\ell} \approx \mathbf{17}$
	49	$k = 6$; $\overline{\ell} \approx \mathbf{23}$	$k = 6$; $\overline{\ell} \approx \mathbf{17}$
	100	$k = 6$; $\overline{\ell} \approx \mathbf{28}$	$k = 6$; $\overline{\ell} \approx \mathbf{18}$
r-RAS	25	$k = 6$; $\overline{\ell} \approx \mathbf{102}$	$k = 6$; $\overline{\ell} \approx \mathbf{21}$
	49	$k = 6$; $\overline{\ell} \approx \mathbf{59}$	$k = 6$; $\overline{\ell} \approx \mathbf{25}$
	100	$k = 6$; $\overline{\ell} \approx \mathbf{32}$	$k = 6$; $\overline{\ell} \approx \mathbf{26}$

combinations of Schwarz preconditioner type (l-RAS or r-RAS) and relative overlap (1/8 or 1/4). The method with left preconditioning diverges in many cases. The divergences were caused by the fact that the true linear residuals at the Newton steps did not decrease as much as the preconditioned linear residuals used for the Krylov stopping criteria. The overall algorithm then failed after some Newton iterations, either by computing steps that did not provide descent directions or by computing step lengths too small.

In Tables 4.6–4.8 we do basically the same as in Table 4.5, but now we fix the preconditioning side to $\mathcal{M}_{l,r}$ and test up to Re $= 300$ and 100 processors. In the case of Re $= 300$ (Table 4.8) the method demanded too many iterations for convergence with $\delta = H/8$, and so we report experiments with $\delta = H/2$.

It is interesting to observe in Tables 4.7 and 4.8 that, fixing the overlap, one of the RAS smoothers (l-RAS or r-RAS) causes the preconditioner to converge with a smaller average number of Krylov iterations for 25 subdomains, while the other RAS smoother contributes for a better average Krylov convergence in the case of 100 subdomains. That is, there is no concept such as a unique RAS version that is always the best smoother for all possible number of processors.

Based on the results of Tables 4.6–4.8, we perform the tests shown in Tables 4.9–4.11, where we fix the preconditioning side to $\mathcal{M}_{l,r}$ and vary Re, mesh size, and the

TABLE 4.7

Two-level right *Schwarz preconditioner results for the cavity flow control problem with* Re $= 250$ *and a* 280×280 *mesh (*631,688 *variables), for different combinations of the number of processors* N_p*, the relative overlap* δ/H*, and the Schwarz preconditioner* $\mathbf{B}^{-1}$ *for the Richardson smoother.* k *is the total number of Newton iterations, and* $\overline{\ell}$ *is the average number of Krylov iterations per Newton iteration.*

$\mathbf{B}^{-1}$	N_p	Relative overlap δ/H	
		1/8	1/4
l-RAS	25	$k = -$; $\ell^{(6)} > 2,518$	$k = 8$; $\overline{\ell} \approx$ **87**
	49	$k = -$; $\ell^{(7)} = 3,150$	$k = 8$; $\overline{\ell} \approx$ **21**
	100	$k = 8$; $\overline{\ell} \approx$ **33**	$k = 8$; $\overline{\ell} \approx$ **23**
r-RAS	25	$k = 8$; $\overline{\ell} \approx$ **29**	$k = 8$; $\overline{\ell} \approx$ **22**
	49	$k = 8$; $\overline{\ell} \approx$ **106**	$k = 8$; $\overline{\ell} \approx$ **28**
	100	$k = 8$; $\overline{\ell} \approx$ **251**	$k = 8$; $\overline{\ell} \approx$ **86**

TABLE 4.8

Two-level right *Schwarz preconditioner results for the cavity flow control problem with* Re $=$ **300** *and a* 280×280 *mesh (*631,688 *variables), for different combinations of the number of processors* N_p*, the relative overlap* δ/H*, and the Schwarz preconditioner* $\mathbf{B}^{-1}$ *for the Richardson smoother.* k *is the total number of Newton iterations, and* $\overline{\ell}$ *is the average number of Krylov iterations per Newton iteration.*

$\mathbf{B}^{-1}$	N_p	Relative overlap δ/H	
		1/4	1/2
l-RAS	25	$k = -$; $\ell^{(6)} >$ **4,290**	$k = 10$; $\overline{\ell} \approx$ **43**
	49	$k = 10$; $\overline{\ell} \approx$ **73**	$k = -$; $\ell^{(6)} =$ **1,794**
	100	$k = 10$; $\overline{\ell} \approx$ **26**	$k = 10$; $\overline{\ell} \approx$ **25**
r-RAS	25	$k = 10$; $\overline{\ell} \approx$ **23**	$k = 10$; $\overline{\ell} \approx$ **21**
	49	$k = 10$; $\overline{\ell} \approx$ **48**	$k = 10$; $\overline{\ell} \approx$ **21**
	100	$k = 10$; $\overline{\ell} \approx$ **52**	$k = 10$; $\overline{\ell} \approx$ **29**

number of processors. Clearly, the two-level method performed in a very robust way, solving problems for which the one-level version has previously failed. In all three Tables 4.9–4.11, the average number $\overline{\ell}$ of Krylov iterations per Newton iteration is kept stable in the tested range of mesh size and number of processors. The results are consistent with predictions (2.3)–(2.4): In each column of the tables, $\overline{\ell}$ is not very sensitive to the increase in the number N_p of processors (prediction (2.3)), and, in each line of the tables, $\overline{\ell}$ is not too sensitive to mesh refinement (prediction (2.4)).

As intuitively expected, the Jacobian ill-conditioning gets worse as we increase Re and refine the mesh, demanding Schwarz preconditioning with bigger overlap and/or more smoothing iterations.

Many of the results reported so far, for both one-level and two-level Schwarz preconditioners, might be related to comment (2.5). Namely, the fact that the average number of Krylov iterations per Newton step gets eventually worse with an increase in the size H of subdomains or in the overlap δ. Such behavior is "strange" if we consider the literature results on the application of Schwarz preconditioners to the solution of positive definite systems. In the case of one-level tests, examples of strange results exist in Table 4.2, with 49 processors, and in Table 4.3, also with 49 processors. In the case of two-level tests, examples of strange results exist in Table 4.5, with Re $= 200$ and with Re $= 250$, a l-RAS preconditioner, and relative overlap $\delta/H = 1/4$, in Table 4.7, with a l-RAS preconditioner and relative overlap $\delta/H = 1/8$, and in Table 4.8, with a l-RAS preconditioner.

TABLE 4.9

Two-level right *Schwarz preconditioner results for the cavity flow control problem with* Re = 200*, Schwarz preconditioner* $\mathbf{B}^{-1}$ *for the Richardson smoother, and relative overlap* δ/H*, for different combinations of the number of processors* N_p*, and the mesh size.* k *is the total number of Newton iterations, and* $\overline{\ell}$ *is the average number of Krylov iterations per Newton iteration. The number of variables is* 2,517,768 *on the case of finest mesh.*

N_p $[\mathbf{B}^{-1}]$ $\left(\frac{\delta}{H}\right)$	Mesh		
	140×140	280×280	560×560
25 [l-RAS] $\left(\frac{1}{4}\right)$	$k = 7$; $\overline{\ell} \approx \mathbf{10}$	$k = 6$; $\overline{\ell} \approx \mathbf{17}$	Not tested
49 [l-RAS] $\left(\frac{1}{4}\right)$	$k = 7$; $\overline{\ell} \approx \mathbf{10}$	$k = 6$; $\overline{\ell} \approx \mathbf{17}$	Not tested
100 [l-RAS] $\left(\frac{1}{4}\right)$	$k = 7$; $\overline{\ell} \approx \mathbf{11}$	$k = 6$; $\overline{\ell} \approx \mathbf{18}$	$k = 6$; $\overline{\ell} \approx \mathbf{22}$

TABLE 4.10

Two-level right *Schwarz preconditioner results for the cavity flow control problem with* Re = 250*, Schwarz preconditioner* $\mathbf{B}^{-1}$ *for the Richardson smoother, and relative overlap* δ/H*, for different combinations of the number of processors* N_p*, and the mesh size.* k *is the total number of Newton iterations, and* $\overline{\ell}$ *is the average number of Krylov iterations per Newton iteration. The number of variables is* 2,517,768 *on the case of finest mesh.*

N_p $[\mathbf{B}^{-1}]$ $\left(\frac{\delta}{H}\right)$	Mesh		
	140×140	280×280	560×560
25 [r-RAS] $\left(\frac{1}{4}\right)$	$k = 12$; $\overline{\ell} \approx \mathbf{17}$	$k = 8$; $\overline{\ell} \approx \mathbf{22}$	Not tested
49 [l-RAS] $\left(\frac{1}{4}\right)$	$k = 12$; $\overline{\ell} \approx \mathbf{14}$	$k = 8$; $\overline{\ell} \approx \mathbf{21}$	Not tested
100 [l-RAS] $\left(\frac{1}{4}\right)$	$k = 12$; $\overline{\ell} \approx \mathbf{14}$	$k = 8$; $\overline{\ell} \approx \mathbf{23}$	$k = 7$; $\overline{\ell} \approx \mathbf{31}$

TABLE 4.11

Two-level right *Schwarz preconditioner results, for the cavity flow control problem with* Re = 300 *and a* 70×70 *coarse mesh, for different situations of the number of processors* N_p*, and the mesh size.* k *is the total number of Newton iterations, and* $\overline{\ell}$ *is the average number of Krylov iterations per Newton iteration. To each situation corresponds a combination of the number* σ *of Richardson iterations, the RAS preconditioner, and the relative overlap* δ/H *used in the pre- and postsmoothers. The number of variables is* 2,517,768 *in the case of finest mesh.*

N_p $\left(\frac{\delta}{H}\right)$	Mesh		
	140×140	280×280	560×560
25 $\left(\frac{1}{4}\right)$	$\sigma = 1$; r-RAS $k = 13$; $\overline{\boldsymbol{\ell}} = \mathbf{20}$	$\sigma = 1$; r-RAS $k = 10$; $\overline{\boldsymbol{\ell}} = \mathbf{23}$	– –
49 $\left(\frac{1}{2}\right)$	$\sigma = 1$; r-RAS $k = 13$; $\overline{\boldsymbol{\ell}} = \mathbf{18}$	$\sigma = 1$; r-RAS $k = 10$; $\overline{\boldsymbol{\ell}} = \mathbf{21}$	– –
100 $\left(\frac{1}{2}\right)$	$\sigma = 1$; l-RAS $k = 13$; $\overline{\boldsymbol{\ell}} = \mathbf{18}$	$\sigma = 1$; l-RAS $k = 10$; $\overline{\boldsymbol{\ell}} = \mathbf{25}$	$\sigma = 2$; r-RAS $k = 8$; $\overline{\boldsymbol{\ell}} = \mathbf{27}$

Figures 4.2(a) and 4.2(b) clearly show the stabilization on the average number of Krylov iterations provided by the two-level preconditioner with modified interpolation. The one-level preconditioner fails with 100 processors for Re = 250 and Re = 300.

Table 4.12 shows the efficacy of the modified interpolation process, which performs much better than the unmodified one, causing the two-level preconditioner to outperform the one-level preconditioner.

4.2.2. Backward-facing step flows. The numerical behavior observed in these problems is very different than the one observed in the cavity flows of the previous section. The backward-facing step problems demand more Newton iterations, but the

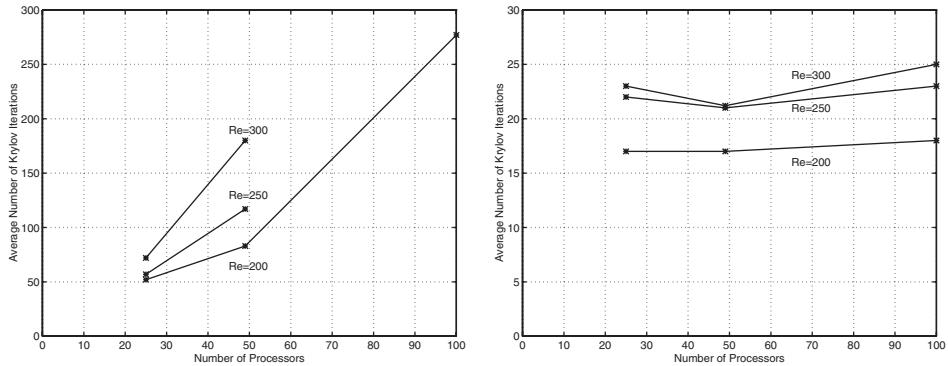


FIG. 4.2. *Cavity tangential boundary control problem results for* (a) *one-level and* (b) *two-level right Schwarz preconditioner with a* 280×280 *mesh (*631,688 *variables) and a* 70×70 *coarse mesh.*

TABLE 4.12

Right Schwarz preconditioner results for the cavity flow control problem with Re $= 250$*, a* 280×280 *mesh (*631,688 *variables),* 49 *processors, relative overlap* $\delta/H = 1/4$*, and a* 70×70 *coarse mesh, for different combinations of the number of levels* L*, piecewise linear interpolation type, the number of pre- and postsmoother iterations* σ*, and a RAS preconditioner.* k *is the total number of Newton iteration, and* $\overline{\ell}$ *is the average number of Krylov iterations per Newton iteration.*

L	Linear interpolation Type	σ	RAS preconditioner	
			l-RAS	r-RAS
1	–	–	$k = 8$; $\overline{\ell} = 336$	$k = 9$; $\overline{\ell} = 973$
2	Unmodified	1	$k = 8$; $\overline{\ell} = 1,110$	$k = 8$; $\overline{\ell} = 1,150$
2	Unmodified	2	$k = 8$; $\overline{\ell} = 356$	$k = 8$; $\overline{\ell} = 222$
2	Modified	1	$k = 8$; $\overline{\ell} = \mathbf{21}$	$k = 8$; $\overline{\ell} = \mathbf{28}$

TABLE 4.13

Two-level left *Schwarz preconditioner results for the backward-facing step flow control problem with* Re $= 300$*, a r-RAS preconditioned Richardson smoother, and relative overlap* $\delta/H = 1/4$*, for different combinations of the number of processors* N_p*, and the mesh size.* k *is the total number of Newton iterations, and* $\overline{\ell}$ *is the average number of Krylov iterations per Newton iteration. The number of variables is* 1,780,232 *on the case of finest mesh.*

N_p	Mesh		
	288×48	576×96	$1,152 \times 192$
24	$k = 46$; $\overline{\ell} \approx \mathbf{4}$	$k = 36$; $\overline{\ell} \approx \mathbf{4}$	Not tested
54	$k = 46$; $\overline{\ell} \approx \mathbf{4}$	$k = 36$; $\overline{\ell} \approx \mathbf{5}$	Not tested
96	$k = 46$; $\overline{\ell} \approx \mathbf{5}$	$k = 36$; $\overline{\ell} \approx \mathbf{6}$	$k = 25$; $\overline{\ell} \approx \mathbf{7}$

computation of each Newton step is easier. In fact, the one-level Schwarz preconditioner now works for problems with 100 processors for all three Re = 200, 250, and 300, the same happening for the two-level method with the polluted (unmodified) interpolation procedure. Also, similarly to the case of positive-definite systems, a left preconditioner now works better than a right preconditioner, as shown by Tables 4.13 and 4.14.

Nonetheless, the pollution-free two-level RAS preconditioner was the best choice, as exemplified by Table 4.15, where the use of the modified interpolation decreases the

TABLE 4.14

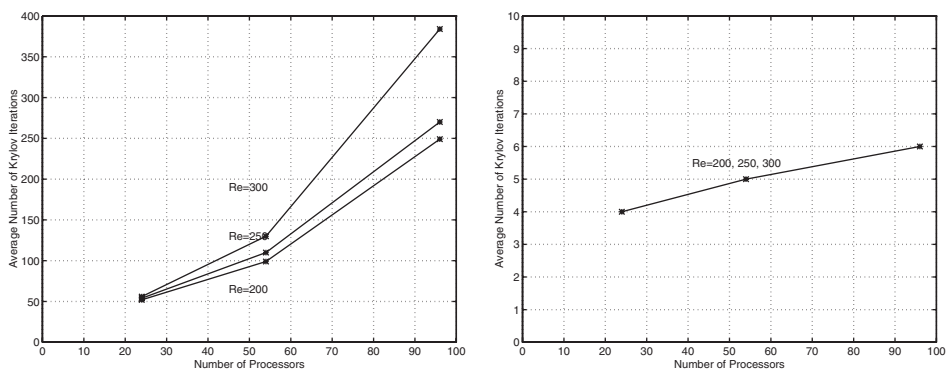
Two-level right *Schwarz preconditioner results for the backward-facing step flow control problem with* Re = **300**, *a r-RAS preconditioned Richardson smoother, and relative overlap* $\delta/H = 1/4$, *for different combinations of the number of processors* N_p, *and the mesh size.* k *is the total number of Newton iterations, and* $\overline{\ell}$ *is the average number of Krylov iterations per Newton iteration. The number of variables is* 1,780,232 *on the case of finest mesh.*

N_p	Mesh		
	288×48	576×96	$1,152 \times 192$
24	$k = 46$; $\overline{\ell} \approx \mathbf{6}$	$k = 36$; $\overline{\ell} \approx \mathbf{7}$	Not tested
54	$k = 46$; $\overline{\ell} \approx \mathbf{7}$	$k = 36$; $\overline{\ell} \approx \mathbf{8}$	Not tested
96	$k = 46$; $\overline{\ell} \approx \mathbf{8}$	$k = 36$; $\overline{\ell} \approx \mathbf{10}$	$k = 26$; $\overline{\ell} \approx \mathbf{11}$

TABLE 4.15

Left Schwarz preconditioner results for the backward-facing step flow control problem with Re = 300, *a* 1152×192 *mesh (*1,780,232 *variables),* 96 *processors, a r-RAS with relative overlap* $\delta/H = 1/4$, *and a* 144×24 *coarse mesh, for different combinations of the number of levels* L, *piecewise linear interpolation type, and the number of pre- and postsmoother iterations* σ. k *is the total number of Newton iterations, and* $\overline{\ell}$ *is the average number of Krylov iterations per Newton iteration.*

L	Linear interpolation type	σ	r-RAS preconditioner
1	–	–	$k = 26$; $\overline{\ell} = 404$
2	Unmodified	1	$k = 27$; $\overline{\ell} = 14$
2	Modified	1	$k = 25$; $\overline{\boldsymbol{\ell}} = \mathbf{7}$



(a) (b)

FIG. 4.3. *Backward facing-step tangential boundary control problems results for* (a) *one-level and* (b) *two-level left Schwarz preconditioner with a* 576×96 *mesh (*447,752 *variables) and a* 144×24 *coarse mesh.*

average number of Krylov iterations by a further factor of two w.r.t. the unmodified interpolation. Figures 4.3(a) and 4.3(b) also show the stabilization on the average number of Krylov iterations provided by the preconditioner. Finally, as in the case of cavity flow control problems, Figure 4.3(b) is consistent with prediction (2.3).

5. Conclusions. We have developed a parallel multilevel Schwarz preconditioner that has shown a robust performance when tested on some tangential boundary flow control problems. With such a preconditioner we were able to extend the one-level Lagrange–Newton–Krylov–Schwarz presented in [28, 29] to a multilevel method.

Our algorithm works well for the discretize-then-optimize approach and has yet to be tested with an optimize-then-discretize approach.

In general, the success of Lagrange–Newton methods with a line search rests on four tasks: the computation of the Jacobian, the computation of a Newton step with descent direction, the choice of a penalty parameter in the merit function, and the computation of the step length. Although we addressed all of these issues in our work, our main contribution in this paper consists in the combination of a general multigrid V-cycle preconditioner with (1) RAS preconditioned Richardson smoothers and (2) a modified interpolation procedure that removes the pollution often generated by the application of common interpolation techniques to the Lagrange multipliers. Such a combination is the key for the successful application of the two-level Schwarz preconditioner in our experiments and the consequent improvement over the one-level method, handling flow control problems with higher Reynolds number, finer meshes, and more processors. Surprisingly, RAS preconditioners performed much better than the classical ones.

We expect multilevel pollution removing preconditioners to have a wide application in many scientific and engineering applications.

Acknowledgments. We thank the PETSc team for helping us using the PETSc library, the Hemisphere cluster team at the University of Colorado at Boulder for their support and Professor Richard Byrd, the editor, and two anonymous referees for their useful comments.

REFERENCES

[1] S. Balay, K. Buschelman, W. D. Gropp, D. Kaushik, M. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang, *Portable, Extensible Toolkit for Scientific Computation (PETSc)*. Homepage http://www.mcs.anl.gov/petsc, 2004.

[2] M. Benzi, G. H. Golub, and J. Liesen, *Numerical solution of saddle point problems*, Acta Numer., 14 (2005), pp. 1–137.

[3] G. Biros and O. Ghattas, *Parallel Lagrange-Newton-Krylov-Schur methods for PDE-constrained optimization. Part* I: *The Krylov-Schur solver*, SIAM J. Sci. Comput., 27 (2005), pp. 687–713.

[4] G. Biros and O. Ghattas, *Parallel Lagrange-Newton-Krylov-Schur methods for PDE-constrained optimization. Part* II: *The Lagrange-Newton solver and its application to optimal control of steady viscous flows*, SIAM J. Sci. Comput., 27 (2005), pp. 714–739.

[5] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, 2nd ed., SIAM, 2000.

[6] X.-C. Cai, M. Dryja, and M. Sarkis, *Restricted additive Schwarz preconditioners with harmonic overlap for symmetric positive definite linear systems*, SIAM J. Numer. Anal., 41 (2003), pp. 1209–1231.

[7] X.-C. Cai, W. D. Gropp, D. E. Keyes, R. G. Melvin, and D. P. Young, *Parallel Newton-Krylov-Schwarz algorithms for the transonic full potential equation*, SIAM J. Sci. Comput., 19 (1998), pp. 246–265.

[8] X.-C. Cai and D. E. Keyes, *Nonlinearly preconditioned inexact Newton algorithms*, SIAM J. Sci. Comput., 24 (2002), pp. 183–200.

[9] X.-C. Cai and M. Sarkis, *A restricted additive Schwarz preconditioner for general sparse linear systems*, SIAM J. Sci. Comput., 21 (1999), pp. 792–797.

[10] X.-C. Cai and O. B. Widlund, *Domain decomposition algorithms for indefinite elliptic problems*, SIAM J. Sci. Comput., 13 (1992), pp. 243–258.

[11] T. F. Coleman and J. J. Moré, *Estimation of sparse Jacobian matrices and graph coloring problems*, SIAM J. Numer. Anal., 20 (1983), pp. 187–209.

[12] J. J. Dongarra, I. S. Duff, D. C. Sorensen, and H. A. van der Vorst, *Numerical Linear Algebra for High-Performance Computers*, SIAM, 1998.

[13] M. Dryja and W. Hackbusch, *On the nonlinear domain decomposition method*, BIT, 37 (1997), pp. 296–311.

[14] M. Dryja and O. B. Widlund, *Domain decomposition algorithms with small overlap*, SIAM J. Sci. Comput., 15 (1994), pp. 604–620.
[15] S. C. Eisenstat and H. F. Walker, *Globally convergent inexact Newton methods*, SIAM J. Optim., 4 (1994), pp. 393–422.
[16] A. V. Fiacco, *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming*, Math. Sci. Eng. 68, Academic, New York, 1983.
[17] A. Frommer and D. B. Szyld, *An algebraic convergence theory for restricted additive Schwarz methods using weighted max norms*, SIAM J. Numer. Anal., 39 (2001), pp. 463–479.
[18] M. D. Gunzburger, *Perspectives in Flow Control and Optimization*, 1st ed., SIAM, 2003.
[19] M. D. Gunzburger and L. S. Hou, *Finite-dimensional approximation of a class of constrained nonlinear optimal control problems*, SIAM J. Control Optim., 34 (1996), pp. 1001–1043.
[20] M. D. Gunzburger, M. Mundt, and J. S. Peterson, *Experiences with finite element methods for the velocity vorticity formulation of three dimensional, viscous, incompressible flows*, in Computational Methods in Viscous Aerodynamics, T. K. S. Murthy and C. A. Brebbia, eds., Elsevier, New York, 1990, pp. 231–272.
[21] M. Heinkenschloss and H. Nguyen, *Neumann-Neumann domain decomposition preconditioners for linear-quadratic elliptic optimal control problems*, SIAM J. Sci. Comput., 28 (2006), pp. 1001–1028.
[22] F.-N. Hwang and X.-C. Cai, *A parallel nonlinear additive Schwarz preconditioned inexact Newton algorithm for incompressible Navier-Stokes equations*, J. Comput. Phys., 204 (2005), pp. 666–691.
[23] D. E. Keyes, *How scalable is domain decomposition in practice?*, in Proceedings of the 11th International Conference on Domain Decomposition Methods, Domain Decomposition Press, 1998, Bergen, Norway, pp. 286–297.
[24] Z. Li and Y. Saad, *SchurRAS: A restricted version of the overlapping Schur complement preconditioner*, SIAM J. Sci. Comput., 27 (2006), pp. 1787–1801.
[25] R. Nabben and D. B. Szyld, *Convergence theory of restricted multiplicative Schwarz methods*, SIAM J. Numer. Anal., 40 (2002), pp. 2318–2336.
[26] J. Nocedal and S. J. Wright, *Numerical Optimization*, 1st ed., Springer-Verlag, New York, 2000.
[27] N. M. Nachtigal, S. C. Reddy, and L. N. Trefethen, *How fast are nonsymmetric matrix iterations?*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 778–795.
[28] E. E. Prudencio, R. Byrd, and X.-C. Cai, *Parallel full space SQP Lagrange-Newton-Krylov-Schwarz algorithms for PDE-constrained optimization problems*, SIAM J. Sci. Comput., 27 (2006), pp. 1305–1328.
[29] E. E. Prudencio, *Parallel Fully Coupled Lagrange-Newton-Krylov-Schwarz Algorithms and Software for Optimization Problems Constrained by Partial Differential Equations*, Ph.D. thesis, Department of Computer Science, University of Colorado at Boulder, 2005.
[30] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, 2003.
[31] J. Schöberl and W. Zulehner, *On Schwarz-type smoothers for saddle point problems*, Numer. Math., 95 (2003), pp. 377–399.
[32] B. Smith, P. Bjørstad, and W. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, 1st ed., Cambridge University Press, London, 1996.
[33] A. Toselli and O. Widlund, *Domain Decomposition Methods - Algorithms and Theory*, Springer-Verlag, New York, 2005.
[34] X. Zhang, *Multilevel Schwarz methods*, Numer. Math., 63 (1992), pp. 521–539.

SIAM J. SCI. COMPUT.
Vol. 29, No. 3, pp. 986–1005
© 2007 Society for Industrial and Applied Mathematics

MIXED HIGHER ORDER SPECTRAL FINITE ELEMENTS FOR REISSNER–MINDLIN EQUATIONS*

G. COHEN† AND P. GROB‡

Abstract. We construct a class of high order numerical approximations for the Reissner–Mindlin plate model in the time domain, based on mixed spectral finite elements with mass lumping. In this way we obtain explicit time-stepping schemes. We first compare the Reissner–Mindlin model to three-dimensional (3D) solutions to validate our method. Then, we show the advantages of the schemes in terms of accuracy and computational time.

Key words. Reissner–Mindlin plate model, higher order spectral elements, mass lumping

AMS subject classifications. 74S05, 74S25, 74B05, 74K20

DOI. 10.1137/050642332

Introduction. The study of vibrations of thin structures is important for many industrial applications. Each of these applications is concerned with numerical simulation on large frequency bandwidths. To avoid having to solve a large linear system for each frequency (which is very costly), it can be efficient to compute the solution in the time domain and to reconstitute the solution in the harmonic domain by an FFT method, for example. In this paper, we consider the Reissner–Mindlin–Naghdi (cf. [15], [17], [19], [20]) plate model in the time domain. This model is more general than that of Kirchhoff–Love (KL), which takes into account shear strain and makes it possible to treat the case of thick plates. Furthermore, we can use H^1-conforming finite elements, while the KL model a priori requires H^2-conforming finite elements. However, we have to deal with vectorial unknowns, and the shear strain terms induce numerical locking which is essentially bad convergence of the finite element method when the thickness tends to zero [1], [4], [6], [7]. To the best of our knowledge, the numerical analysis of the Reissner–Mindlin equations in the time domain is not well understood. We show in this paper the utility of using mixed spectral finite elements with mass lumping (cf. [9], [14]). This method is known to be robust and efficient for the linear elasticity system [10], [13]. We use a mixed formulation combined with the Piola transform [9] to avoid assembling the stiffness matrix and thus to have a very substantial gain in memory storage. Usually, the use of mixed finite elements (MFE) eliminates the problem of numerical locking. Our motivation for using MFEs is thus different. The outline of the article is as follows.

In section 1, we describe the equations that constitute the Reissner–Mindlin plate model. Then, we recall the energy identity, which ensures the well-posedness of the problem.

In section 2, we propose a mixed formulation for the problem and its discretization by mixed spectral finite elements with mass lumping. Mass lumping is obtained thanks to the use of the appropriate quadrature formula (Gauss–Lobatto). To our knowledge, the error analysis of such methods is not complete, particularly concerning the effect

*Received by the editors October 10, 2005; accepted for publication (in revised form) September 6, 2006; published electronically May 10, 2007.
http://www.siam.org/journals/sisc/29-3/64233.html
†INRIA-Rocquencourt, B.P. 105, 78153 Le Chesnay Cedex, France (gary.cohen@inria.fr).
‡INRIA-Rocquencourt, B.P. 105, 78153 Le Chesnay Cedex, France and Université Laval Dept. Math. et Stat., Québec G1K 7P4, Canada (pgrob@mat.ulaval.ca).

of numerical integration. However, we give a short review of some known results in this direction.

In section 3, we propose and analyze a leapfrog scheme for the time discretization.

In section 4, we give several numerical results. First, we validate our method and our results using the fact that the plate model is asymptotic to the linear elasticity system: We compare our results with results given by a three-dimensional (3D) linear elasticity code. Then we show the advantages of the scheme in terms of accuracy and computational time. Finally, we address the question of numerical locking through a numerical experiment which shows that a low order (2 or 3) method is subject to a locking phenomena, while the higher order methods are little affected by this problem. To conclude, the aim of this paper is to propose a finite element method for the Reissner–Mindlin plate model in the time domain, that we claim to be robust, accurate, and high-performance in terms of computation time and memory storage. Some important aspects like boundary layer effects are not treated in this article.

1. The Reissner–Mindlin plate model.

1.1. Construction of the model by an asymptotic approach. In this paper, we define a plate as a parallelepipedic elastic material Ω with a dimension of thickness δ small compared to the other dimensions. The midsurface of the plate is Σ and the boundary is $\Gamma = \Gamma^+ \cup \Gamma^- \cup \Gamma^0$ (Figure 1.1), where

$$\Gamma^{\pm} = \Omega \cap \left\{ z = \frac{\pm\delta}{2} \right\}, \quad \Gamma^0 = \partial\Omega \setminus \Gamma^{\pm}.$$

We suppose that the plate is totally clamped along Γ_0 and is subjected to a loading force $\underline{g}$ on Γ^+ and that there are no internal forces in the plate. So, a displacement field

$$\underline{U}(x_1, x_2, x_3; t) = U_i(x_1, x_2, x_3; t)_{i=1,2,3}, \quad (x_1, x_2, x_3; t) \in \Sigma \times \left[-\frac{\delta}{2}, \frac{\delta}{2} \right] \times [0, T]$$

is characterized by the linear tensor of deformations

$$\underline{\underline{\varepsilon}}(\underline{U}) \stackrel{def}{:=} \underline{\underline{\varepsilon}}(\underline{U})_{i,j} = \frac{1}{2} \left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} \right) \quad \forall\, (i,j) = \{1,2,3\}$$

and satisfies the linear elasticity system

$$\begin{cases} \rho \dfrac{\partial^2 \underline{U}}{\partial t^2} - \underline{\mathrm{Div}} \left(\underline{\underline{\tilde{C}}}\, \underline{\underline{\varepsilon}}(\underline{U}) \right) = 0 & \text{on} \quad \Omega, \\ \underline{\underline{\tilde{C}}}\, \underline{\underline{\varepsilon}}(\underline{U}) \underline{n} = \underline{g} & \text{on} \quad \Gamma^+, \\ \underline{\underline{\tilde{C}}}\, \underline{\underline{\varepsilon}}(\underline{U}) \underline{n} = 0 & \text{on} \quad \Gamma^-, \\ \underline{U} = 0 & \text{on} \quad \Gamma_0 \end{cases}$$

completed by zero initial data. In this system ρ is the density of the plate, $\underline{\underline{\tilde{C}}}$ is the stiffness tensor that arises in Hooke's law, and $\underline{n} = (0, 0, 1)$.

Then we can write the variational formulation of the elastic system

$$\frac{d}{dt^2} \int_\Omega \rho \underline{U} \cdot \underline{U}^*\, dx + \int_\Omega \underline{\underline{\tilde{C}}}\, \underline{\underline{\varepsilon}}(\underline{U}) : \underline{\underline{\varepsilon}}(\underline{U}^*)\, dx = \int_{\Gamma^+} \underline{g} \cdot \underline{U}^* d\gamma \quad \forall\, \underline{U}^* \in V_0, \tag{1.1}$$

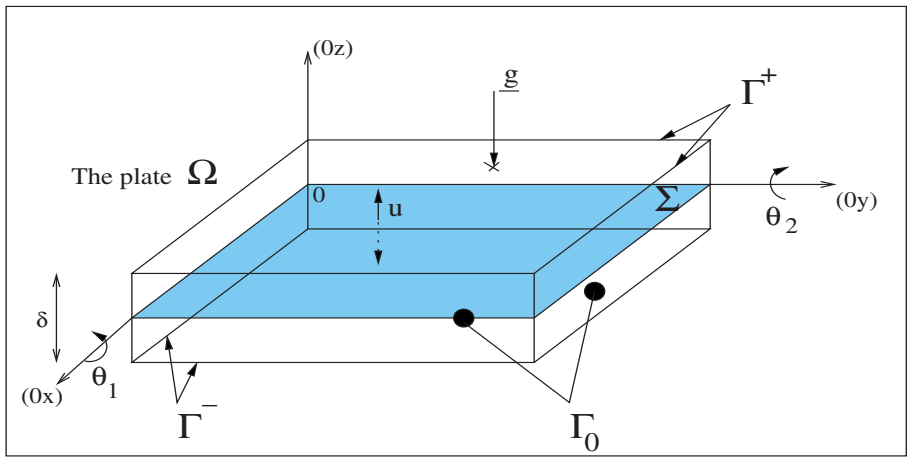


FIG. 1.1. *The plate model.*

where $V_0 = \{\underline{v} \in [H^1(\Omega)]^3,\ \underline{v} = 0 \quad \text{on } \Gamma^+\}$. The use of (1.1) and kinematical assumptions due to Reissner, Mindlin, and Nagdhi [17], [19], [20] on the admissible displacement profile permit us to describe the 3D displacement field $\underline{U}$ from two-dimensional (2D) displacement fields defined on Σ. We recall below the Reissner–Mindlin kinematical assumptions as given in [6].

ASSUMPTION 1.

$$U_\alpha(x_1, x_2, x_3; t) = u_\alpha(x_1, x_2; t) + x_3\theta_\alpha(x_1, x_2; t),\ \alpha = \{1, 2\}, \tag{1.2}$$

$$U_3(x_1, x_2, x_3; t) = u(x_1, x_2; t). \tag{1.3}$$

u_α u Σ $\underline{n}$ $x_3\theta_\alpha$ (Ox) (Oy)

1. In the case of a plate, the membrane displacement and the bending displacement are uncoupled. When there are no internal forces we can show that $u_\alpha = 0,\ \alpha = \{1, 2\}$.

We replace $\underline{U}$ by (1.2) and (1.3) in (1.1) and use test functions $\underline{U}^*$ of the same form. After making several hypotheses given in [6], [19] on the tensor $\underline{\underline{C}}$, and taking into account Remark 1, we can show that u and $\underline{\theta} = (\theta_1, \theta_2)$ satisfy the Reissner–Mindlin plate model,

$$\rho\frac{\delta^3}{12}\frac{\partial^2\underline{\theta}}{\partial t^2} - \frac{\delta^3}{12}\underline{\mathrm{Div}}\left(\underline{\underline{C}}\,\underline{\underline{\varepsilon}}(\underline{\theta})\right) + \delta\mu(\nabla u + \underline{\theta}) = 0 \quad \text{on } \mathcal{Q}_T, \tag{1.4}$$

$$\rho\delta\frac{\partial^2 u}{\partial t^2} - \delta\ \mathrm{div}[\mu(\nabla u + \underline{\theta})] = f \quad \text{on } \mathcal{Q}_T, \tag{1.5}$$

$$u = 0\ ,\ \underline{\theta} = 0 \quad \text{on } \mathcal{Q}_T, \tag{1.6}$$

completed by zero initial data. We have set $\mathcal{Q}_T = \Sigma \times [0, T]$. We denote by E the Young modulus, ν the Poisson coefficient, $\mu = E/[2(1+\nu)]$, and $\lambda = (E\nu)/(1-\nu^2)$.

Then, the new fourth-order stiffness tensor of the plate $\underline{\underline{C}}$ is given by

$$\begin{cases} c^{1111} = c^{2222} = \lambda + 2\mu, & c^{1122} = c^{2211} = \lambda, \\ c^{1221} = c^{2112} = c^{2121} = c^{1212} = \mu, & c^{\alpha\beta\lambda\mu} = 0 \text{ else.} \end{cases}$$

The loading force f is now scalar and depends on $\underline{g}$ and δ. The connection between the 3D model and the 2D model will be used in the validation of the method.

1.2. The well-posedness of the problem. Throughout the rest of the article, if $\xi(t,x)$ is defined on $\mathcal{Q}_T$, we use the notation $\xi(t) = \xi(t,.)$.

PROPOSITION 1.

$$\begin{aligned} E_m[\underline{\theta}(t), u(t)] = \rho\frac{\delta^3}{12}\left\|\frac{\partial\underline{\theta}(t)}{\partial t}\right\|^2_{L^2} + \rho\delta\left\|\frac{\partial u(t)}{\partial t}\right\|^2_{L^2} \\ + \frac{\delta^3}{12}\|\theta(t)\|^2_{\varepsilon,\Sigma} + \delta\,\mu\|(\nabla u + \underline{\theta})(t)\|^2_{L^2} \end{aligned} \tag{1.7}$$

$$\frac{1}{2}\frac{d}{dt}E_m[\theta(t), u(t)] = \int_\Sigma f(t)\frac{\partial u(t)}{\partial t}, \tag{1.8}$$

$\|\theta(t)\|^2_{\varepsilon,\Sigma} = \int_\Sigma \underline{\underline{C}}\,\underline{\underline{\varepsilon}}(\underline{\theta}(t)) : \underline{\underline{\varepsilon}}(\underline{\eta})$ $H^1_0(\Sigma)$. We multiply (1.4) by $\underline{\eta} \in [H^1_0(\Sigma)]^2$ and (1.5) by $v \in H^1_0(\Sigma)$, and we get

$$\begin{aligned} \rho\frac{\delta^3}{12}\frac{d^2}{dt^2}\int_\Sigma \underline{\theta}(t)\cdot\underline{\eta}\,dx - \frac{\delta^3}{12}\int_\Sigma \underline{\mathrm{Div}}\left[\underline{\underline{C}}\,\underline{\underline{\varepsilon}}(\underline{\theta}(t))\right] \\ \cdot\underline{\eta}\,dx + \delta\int_\Sigma \mu(\nabla u + \underline{\theta})(t)\cdot\underline{\eta}\,dx = 0, \end{aligned} \tag{1.9}$$

$$\rho\delta\frac{d^2}{dt^2}\int_\Sigma u(t)v\,dx - \delta\int_\Sigma \mathrm{div}\,[\mu(\nabla u + \underline{\theta})(t)]\,v\,dx = \int_\Sigma f(t)v\,dx. \tag{1.10}$$

By applying Green's formula to the stiffness terms, we get

$$-\int_\Sigma \underline{\mathrm{Div}}\left[\underline{\underline{C}}\,\underline{\underline{\varepsilon}}(\underline{\theta}(t))\right]\cdot\underline{\eta}\,dx = \int_\Sigma \underline{\underline{C}}\,\underline{\underline{\varepsilon}}(\underline{\theta}(t)) : \underline{\underline{\varepsilon}}(\underline{\eta})\,dx, \tag{1.11}$$

$$-\int_\Sigma \mathrm{div}[\mu(\nabla u + \underline{\theta})(t)]v\,dx = \int_\Sigma \mu(\nabla u + \underline{\theta})(t)\cdot\nabla v\,dx. \tag{1.12}$$

Using (1.6), we see that integrals over $\partial\Sigma$ are equal to 0. Then, we substitute (1.11) and (1.12) into (1.9) and (1.10), and by summing (1.9) and (1.10) we obtain

$$\begin{aligned} \rho\frac{\delta^3}{12}\frac{d^2}{dt^2}\int_\Sigma \underline{\theta}(t)\cdot\underline{\eta}\,dx \;+\; \rho\delta\frac{d^2}{dt^2}\int_\Sigma u(t)v\,dx + \frac{\delta^3}{12}\int_\Sigma \underline{\underline{C}}\,\underline{\underline{\varepsilon}}(\underline{\theta}(t)) : \underline{\underline{\varepsilon}}(\underline{\eta})\,dx \\ +\; \delta\int_\Sigma \mu(\nabla u + \underline{\theta})(t)\cdot(\nabla v + \underline{\eta})\,dx = \int_\Sigma f(t)v\,dx. \end{aligned} \tag{1.13}$$

Finally we choose $\underline{\eta} = \frac{\partial\underline{\theta}}{\partial t}(t)$ and $v = \frac{\partial u}{\partial t}(t)$ and we obtain (1.8). □

Then, this type of energy identity ensures the well-posedness of the problem via the Hille–Yosida theorem.

2. Approximation by mixed spectral finite elements.

2.1. The mixed formulation. In order to obtain a stiffness matrix independent of the mesh geometry and the physical coefficients of the problem, we propose in this section a mixed formulation for (1.4)–(1.6). This enables us to avoid assembling the stiffness matrix and thus to save on memory storage (the gains can be quite substantial). MFEs are more commonly used to prevent numerical locking (cf. [4], [6], [7]). As stated above, our motivation for using MFE is different. In the following, $(i,j) = \{1,2\}^2$.

First, we decompose the second-order operators in the stiffness part of (1.4) and (1.6) to get gradient- or divergence-type operators. Using the approach of Cohen–Fauqueux [10] we have the following lemma.

LEMMA 1.

$$\underline{\mathrm{Div}}(\underline{\underline{C}}\,\underline{\underline{\varepsilon}}(\theta))_i = \sum_{j=1}^{2} \mathrm{div}(A_{ij}\nabla\theta_j), \quad A_{ij} = \begin{pmatrix} c^{i1j1} & c^{i1j1} \\ c^{i2j1} & c^{i2j2} \end{pmatrix}.$$

The matrices introduced above are invertible, so by setting

$$\underline{\gamma}_{ij} = A_{ij}\nabla\theta_j \text{ and } \underline{Q} = \mu\,(\nabla u + \underline{\theta}) \quad \forall\, t \in [0,T]$$

and using Lemma 1, (1.4) and (1.6) lead to the following mixed formulation (with appropriated boundary and initial conditions):

$$\rho\frac{\delta^3}{12}\frac{\partial^2\theta_i}{\partial t^2} - \frac{\delta^3}{12}\sum_{j=1}^{2}\mathrm{div}(\underline{\gamma}_{ij}) + \delta\, Q_i = 0 \quad \text{on } \mathcal{Q}_T, \tag{2.1}$$

$$A_{ij}^{-1}\underline{\gamma}_{ij} = \nabla\theta_j \quad \text{on } \mathcal{Q}_T, \tag{2.2}$$

$$\rho\delta\frac{\partial^2 u}{\partial t^2} - \delta\,\mathrm{div}(\underline{\mathrm{Q}}) = \delta \mathrm{f} \quad \text{on } \mathcal{Q}_T, \tag{2.3}$$

$$\underline{Q} = \mu\,(\nabla u + \underline{\theta}) \quad \text{on } \mathcal{Q}_T. \tag{2.4}$$

In order to use a finite element method, we write a variational formulation for this system. We choose to integrate by parts the divergence-type operators. This induces the following space regularity for the four unknowns:

$$u(t) \in H_0^1(\Sigma),\ \underline{\theta}(t) \in [H_0^1(\Sigma)]^2,\ \underline{Q}(t) \in [L^2(\Sigma)]^2,\ \underline{\gamma}_{ij}(t) \in [L^2(\Sigma)]^2 \qquad \forall\, t \in [0,T].$$

Thanks to the Hille–Yosida theorem we can show that if $f \in C^1(0,T;L^2(\Sigma)$, then

$$(u(t),\underline{\theta}(t),\underline{\gamma}_{ij}(t),\underline{Q}(t)) \in C^1(0,T;H_0^1(\Sigma)) \times C^1(0,T;[H_0^1(\Sigma)]^2) \times C^1(0,T;[L^2(\Sigma)]^2) \times C^1(0,T;[L^2(\Sigma)]^2).$$

Then, the variational formulation is

$$\text{Find } (u(t),\underline{\theta}(t),\underline{\gamma}_{ij}(t),\underline{Q}(t)) \in H_0^1(\Sigma) \times [H_0^1(\Sigma)]^2 \times [L^2(\Sigma)]^2 \times [L^2(\Sigma)]^2 \quad \text{such that}$$

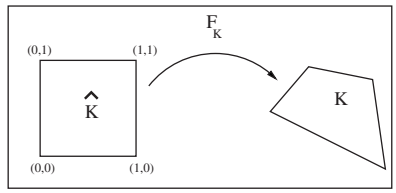


FIG. 2.1. F_K.

$$\rho\frac{\delta^3}{12}\frac{d^2}{dt^2}\int_\Sigma \theta_i(t)\eta_i\,dx + \frac{\delta^3}{12}\sum_{j=1}^{2}\int_\Sigma \underline{\gamma}_{ij}(t)\cdot\nabla\eta_i\,dx + \delta\int_\Sigma Q_i(t)\eta_i\,dx = 0, \tag{2.5}$$

$$\int_\Sigma A_{ij}^{-1}\underline{\gamma}_{ij}(t)\cdot\underline{\xi}_{ij}\,dx = \int_\Sigma \underline{\xi}_{ij}\cdot\nabla\theta_j(t)\,dx, \tag{2.6}$$

$$\rho\delta\frac{d^2}{dt^2}\int_\Sigma u(t)v\,dx + \delta\int_\Sigma \underline{Q}(t)\cdot\nabla v\,dx = \delta\int_\Sigma f(t)v\,dx, \tag{2.7}$$

$$\int_\Sigma \underline{Q}(t)\cdot\underline{q}\,dx = \int_\Sigma \mu\,(\nabla u(t) + \underline{\theta}(t))\cdot\underline{q}\,dx \tag{2.8}$$

for any test function $(v,\underline{\eta},\underline{\xi}_{ij},\underline{q}) \in H_0^1(\Sigma)\times[H_0^1(\Sigma)]^2\times[L^2(\Sigma)]^2\times[L^2(\Sigma)]^2$.

2.2. The choice of a finite element.

2.2.1. Objectives and main tools. Our objective here is to construct a class of higher-order numerical approximations in order to have very good precision, decrease numerical locking, and diminish computation time. Toward this end, we impose three conditions:

(i) To use an explicit time scheme. This is fundamental to decreasing computation time in the dynamic case.
(ii) To use a quadrature formula which keeps the order of the polynomial approximation.
(iii) To construct an approximation space for $[L^2(\Sigma)]^2$ with a "special" transformation in order to avoid the storage of the stiffness matrix.

To satisfy points (i) and (ii) we use the mass lumping principle and Gauss–Lobatto points. Concerning (iii), the use of the Piola transform will be the key point.

2.2.2. The finite element. We denote by $\widehat{K}$ the unit square, by K a quadrilateral, and by $\underline{F}_K$ the bilinear isomorphism between $\widehat{K}$ and K (Figure 2.1), and we consider $\mathcal{T}_h = \cup_{i=1}^{N_e} K_i$ a mesh of Σ consisting of N_e quadrilaterals. As stated above, we want to use a higher order numerical scheme, and so we choose to approximate the solution of the problem with polynomials in $\mathbb{Q}^r$, the set of polynomials of degree r in each variable. This leads us to use the finite element $\mathcal{Q}^r(\widehat{K}) = (\widehat{K},\mathbb{Q}^r,\Xi^2)$, where Ξ is the set of degrees of freedom given by $\Xi = \{\widehat{\xi}_1,\widehat{\xi}_2,\ldots,\widehat{\xi}_r,\widehat{\xi}_{r+1}\}$, where $\{\widehat{\xi}_i\}_{i=\{2,\ldots,r\}}$ are the Gauss–Lobatto points [5], [9], [21]. We have set $\widehat{\xi}_1 = 0$, $\widehat{\xi}_{r+1} = 1$.

REMARK 2. The finite element $\mathcal{Q}^r(\widehat{K})$ is the "tensor product" of the finite element $([0,1],\mathbb{P}^r,\Xi)$.

We will use the notation

$$\underline{\widehat{\xi}}_p = (\widehat{\xi}_m,\widehat{\xi}_n)\ \forall\, p\in\{1\ldots(r+1)^2\},\ \forall\, m\in\{1\ldots(r+1)\},\ \forall\, n\in\{1\ldots(r+1)\}.$$

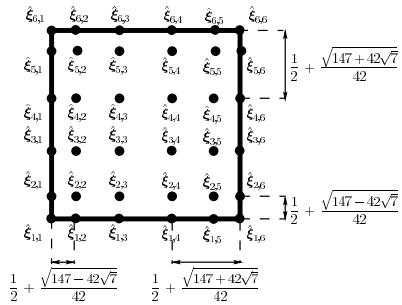


FIG. 2.2. *Example: Gauss–Lobatto points for* $r = 5$.

Even though it is well known, we briefly recall in the next lines what motivates the choice of the Gauss–Lobatto points (Figure 2.2). We have the following proposition [9], [21].

PROPOSITION 2. $\{\widehat{\omega}_i\}_{i\in\{1,\dots,r+1\}}$, $\widehat{\omega}_i > 0$

$$\forall p \in \mathbb{P}^{2r-1},\ \int_0^1 p(\widehat{x})\, d\widehat{x} = \sum_{m=1}^{r+1} \widehat{\omega}_m\, p(\widehat{\xi}_m).$$

Remark 2 and Proposition 2 yield the following corollary.

COROLLARY 1.

$$\forall p \in \mathbb{Q}^{2r-1},\ \int_{\widehat{K}} p(\widehat{x},\widehat{y})\, d\widehat{x}d\widehat{y} = \sum_{n=1}^{r+1}\sum_{m=1}^{r+1} \widehat{\omega}_n\, \widehat{\omega}_m\, p(\widehat{\underline{\xi}}_n, \widehat{\underline{\xi}}_m).$$

THEOREM 1. 1 [8] [5] (ii)

In the following we will use the notation for quadrature formula

$$\oint_K f(x)\, dx = \oint_{\widehat{K}} f(\widehat{x})|J_K(\widehat{x})|\, d\widehat{x} = \sum_{m=1}^{(r+1)^2} \widehat{\omega}_m\, f\circ F_K(\widehat{\underline{\xi}}_m)\, |J_K(\widehat{\underline{\xi}}_m)|.$$

3. This type of construction for our finite element is obviously inappropriate if we choose a triangular geometry. In this case it is difficult to construct a family of points that satisfies such a criterion of accuracy for arbitrary orders [8, 16].

Then, by the definition of $\mathcal{Q}^r(\widehat{K})$ we use the degrees of freedom as quadrature nodes, so we respect the mass lumping principle and satisfy (i).

2.3. The semidiscrete scheme.

2.3.1. Approximated spaces and variational formulation. We now introduce the following approximation spaces of $H_0^1(\Sigma)$ and $[L^2(\Sigma)]^2$, respectively:

$$U_{h,0}^r = \{\varphi \in C^0(\mathcal{T}_h) \text{ such that } \varphi_{|_K}\circ \underline{F}_K \in \mathbb{Q}_r(\widehat{K}) \text{ and } \varphi = 0 \text{ on } \partial\Omega\},$$

$$V_h^r = \{\underline{v} \in [L^2(\mathcal{T}_h)]^2 \text{ such that } |J_K| DF_K^{-1}\, \underline{v}_{|_K}\circ \underline{F}_K \in [\mathbb{Q}_r(\widehat{K})]^2\}.$$

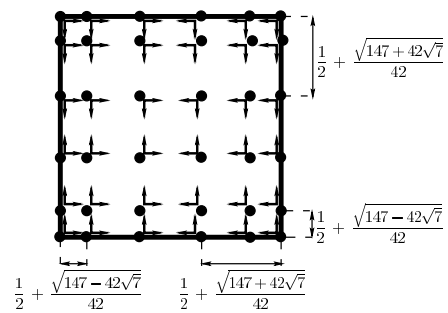


FIG. 2.3. *Example: Degrees of freedom for $r = 5$.*

In the next subsections we need a basis for these spaces. For now, we formally define

$$U_{h,0}^r = \mathrm{Vect}\{\varphi_n\}, \quad \dim(U_{h,0}^r) = n_h, \tag{2.9}$$

$$V_h^r = \mathrm{Vect}\{\underline{\psi}_m\}, \quad \dim(U_{h,0}^r) = m_h. \tag{2.10}$$

The space $U_{h,0}^r$ is a classical approximation of $H_0^1(\Sigma)$: $\varphi_K \in U_{h,0}^r$, defined on a quadrilateral K of the mesh; it is transformed into a function $\widehat{\varphi}$ defined on $\widehat{K}$ by the classical transformation

$$\varphi_K \circ F_K = \widehat{\varphi}. \tag{2.11}$$

The space V_h^r introduces a nonstandard approach: $\underline{\psi}_K \in V_h^r$, defined on a quadrangle K of the mesh, is transformed into a function $\underline{\widehat{\psi}}$ defined on $\widehat{K}$ by the [illegible]

$$\underline{\psi}_K \circ F_K = \frac{1}{|J_K|} DF_K \, \underline{\widehat{\psi}}. \tag{2.12}$$

[illegible] 4. Even if functions of V_h^r are discontinuous on the mesh, the Piola transform ensures that their normal components are continuous. It is in fact the condition to have an H(div)-conforming approximation.

Figure 2.3 shows that we have nodal continuous degrees of freedom at Gauss–Lobatto points for $U_{h,0}^r$ and vectorial discontinuous degrees of freedom at Gauss–Lobatto points for V_h^r. Using the previous spaces we consider the following semidiscrete variational problem.

Find $(u^h(t), \underline{\theta}^h(t), \underline{\gamma}_{ij}^h(t), \underline{Q}^h(t)) \in U_{h,0}^r \times [U_{h,0}^r]^2 \times V_h^r \times V_h^r$ such that

$$\rho \frac{\delta^3}{12} \frac{d^2}{dt^2} \oint_\Sigma \theta_i^h(t) \varphi_n \, dx + \frac{\delta^3}{12} \sum_{j=1}^{2} \oint_\Sigma \underline{\gamma}_{ij}^h(t) \cdot \nabla \varphi_n \, dx + \delta \oint_\Sigma Q_i^h(t) \, \varphi_n \, dx = 0, \tag{2.13}$$

$$\oint_\Sigma A_{ij}^{-1} \underline{\gamma}_{ij}^h(t) \cdot \underline{\psi}_m \, dx = \oint_\Sigma \underline{\psi}_m \cdot \nabla \theta_j^h(t) \, dx, \tag{2.14}$$

$$\rho\delta \frac{d^2}{dt^2} \oint_\Sigma u^h(t) \varphi_q^r \, dx + \delta \oint_\Sigma \underline{Q}^h(t) \cdot \nabla \varphi_n \, dx = \delta \oint_\Sigma f(t) \varphi_n \, dx, \tag{2.15}$$

$$\oint_\Sigma \underline{Q^h} \cdot \underline{\psi_m} \, dx = \mu \oint_\Sigma (\nabla u^h + \underline{\theta}) \cdot \underline{\psi_m} \, dx \tag{2.16}$$

for all indices $(n, m) \in \{1, \ldots, n_h\} \times \{1, \ldots, m_h\}$.

PROPOSITION 3. [illegible] (2.13)–(2.16) [illegible]

$$\rho\frac{\delta^3}{12}\frac{d^2}{dt^2}\oint_\Sigma \underline{\theta}_h(t)\cdot\underline{\eta}_h\,dx+\rho\delta\frac{d^2}{dt^2}\oint_\Sigma u_h(t)v_h\,dx+\frac{\delta^3}{12}\oint_\Sigma \underline{\underline{C}}\,\underline{\underline{\varepsilon}}(\underline{\theta}(t)):\underline{\underline{\varepsilon}}(\underline{\eta}_h)\,dx$$
$$+\delta\oint_\Sigma \mu(\nabla u_h+\underline{\theta}_h)(t)\cdot(\nabla v_h+\underline{\eta}_h)\,dx=\oint_\Sigma f(t)v_h\,dx \tag{2.17}$$

[illegible] $(v_h,\underline{\eta}_h)\in U_h^r\times[U_h^r]^2$

The proof is given in [18] for more general hyperbolic systems. Key points are the use of the same quadrature formulas and polynomial approximation for functions of U_h^r and V_h^r.

2.4. Matricial formulation. Now we give matrices of the bilinear forms which constitute the variational system (2.13)–(2.16). First, we deal with the mass matrices. Let

$$D:\quad (u_h,v_h)\longrightarrow \oint_\Sigma \rho u_h v_h \quad \text{in}\quad U_{h,0}^r\times U_{h,0}^r,$$
$$B:\quad (\underline{w}_h,\underline{w}_h)\longrightarrow \oint_\Sigma \underline{w}_h\cdot\underline{w}_h \quad \text{in}\quad V_h^r\times V_h^r,$$
$$M_{ij}:\quad (\underline{w}_h,\underline{w}_h)\longrightarrow \oint_\Sigma (A_{ij}\underline{w}_h)\cdot\underline{w}_h \quad \text{in}\quad V_h^r\times V_h^r.$$

These matrices have already been described in [13], [10] and we just recall their elementary terms which are useful for computation. Toward this end, we introduce some notation:

- q and p will be indices for local degrees of freedom: $(q,p)\in\{1,\dots,(r+1)\}^2$.
- Let A^K be the elementary matrix for the element K associated to a finite element matrix A.

PROPOSITION 4. [illegible] D, B [illegible] M_{ij} [illegible]

$$(D^K)_{q,q}=\widehat{\omega}_q\,|J_K(\widehat{\underline{\xi}}_q)|,$$
$$(B^K)_{q,q}=\frac{\omega_q}{|J_K(\widehat{\underline{\xi}}_q)|}\,DF_K^*(\widehat{\underline{\xi}}_q)\,DF_K(\widehat{\underline{\xi}}_q),$$
$$(M_{ij}^K)_{q,q}=\frac{\widehat{\omega}_q}{|J_K(\widehat{\underline{\xi}}_q)|}\,DF_K^*(\widehat{\underline{\xi}}_q)\,A_{ij}\,DF_K(\widehat{\underline{\xi}}_q).$$

[illegible] 5. We use the mass-lumping principle so D is diagonal. Furthermore, B and the M_{ij} are 2×2-block diagonal and do not need assembling because functions of V_h^r are discontinuous. Finally, the term DF_K appears in $\widehat{B}^K$ and $\widehat{M}_{ij}^K$ because of (2.12).

Now we deal with the stiffness matrix.

PROPOSITION 5. [illegible] R [illegible]

$$(u_h,\underline{w}_h)\in U_{h,0}^r\times V_h^r\longrightarrow \int_{\mathcal{T}_h}\underline{v}_h\cdot\nabla u_h\,dx.$$

[illegible] R [illegible] $\widehat{K}$

$$\forall\, K\in\mathcal{T}_h,\quad \int_K v_h\cdot\nabla u_h\,dx=\int_{\widehat{K}}\widehat{\underline{v}}_h\cdot\widehat{\nabla}\widehat{u}_h\,d\widehat{x}. \tag{2.18}$$

To give a proof we recall the algebraic identity

$$(\nabla\varphi)\circ F_K = DF_K^{*-1}\widehat{\nabla}\widehat{\varphi}. \tag{2.19}$$

For each element $K \in \mathcal{T}_h$ we have

$$\begin{aligned}\int_K v_h \cdot \nabla u_h \, d &= \int_{\widehat{K}} \underline{v}_h \circ F_K \cdot (\nabla u_h)\circ F_K |J_K| \, d\widehat{x} \\ &= \int_{\widehat{K}} \frac{1}{|J_K|} \, DF_K \underline{\widehat{v}}_h \cdot DF_K^{*-1}\widehat{\nabla}\widehat{u}_h \, |J_K| \, d\widehat{x} \quad \text{using (2.11), (2.12), (2.19),} \\ &= \int_{\widehat{K}} \underline{\widehat{v}}_h \cdot \widehat{\nabla}\widehat{u}_h \, d\widehat{x}. \qquad \square\end{aligned}$$

COROLLARY 2.

$$\forall\, K \in \mathcal{T}_h,\ R^K = R^{\widehat{K}} \qquad (R^K)_{q,p} = \widehat{\omega}_q \frac{\partial \widehat{\varphi}_p}{\partial x_l}(\widehat{\xi}_q)\delta_{2q+l,p} \qquad \forall\, l \in \{1,2\}.$$

Proposition 6 permits us to deduce an algorithm in which we avoid storing a stiffness matrix. More precisely, when we want to compute the product matrix vector $R\,U$, we do not assemble R but we compute directly the product from the storage of $R^{\widehat{K}}$. In fact we "assemble" the product $R\,U$. This method of computation is also possible in the harmonic or stationary case when we use an iterative algorithm to solve the associated linear system. It is obvious that the gain in terms of memory storage is quite substantial. Indeed, we just have to store an elementary matrix instead of the whole matrix. However, it is possible that this procedure penalizes the computation time. In fact, this algorithm also induces a gain in terms of computation time when we use elements with high enough order. When the order increases, the multiplicity of interactions of degrees of freedom also increases. Then a local computation of the product $R\,U$ with our algorithm is more efficient than a computation after assembling R. In Figure 2.4, we represent the computation time as a function of the order to compute $R\,U$ for a given number of degrees of freedom (at each order we have a mesh such that the number of degrees of freedom is constant). The standard formulation corresponds to the computation of $R\,U$ with the assembling of R.

To complete the definition of our matrices, we take into account in our model the shear strain which induces a coupling between u and $\underline{\theta}$ in the term $\underline{Q} = \mu(\nabla u + \underline{\theta})$. So it introduces a new coupling matrix.

PROPOSITION 6.

$$(\underline{u}_h, \underline{w}_h) \longrightarrow \oint_\Sigma \underline{u}_h \cdot \underline{w}_h \quad in\ [U_{h,0}^r]^2 \times V_h^r.$$

$(C^K)_{q,p} = \widehat{\omega}_q \, DF_K(\widehat{\xi}_q)\, \delta_{q,p}$.

We set $\overrightarrow{e}_1 = \binom{1}{0}$, $\overrightarrow{e}_2 = \binom{0}{1}$. From (2.9) we define a basis for $[U_{h,0}^r]^2$:

$$[U_{h,0}^r]^2 = \mathrm{Vect}\{\varphi_q \overrightarrow{e}_i\} \quad \forall i = \{1,2\},\ \forall n = \{1 \dots n_h\}.$$

Because functions of V_h^r are vectorial and discontinuous, $n_h = 2N_e(r+1)^2$ (we do not take into account Dirichlet conditions). So, instead of the formal basis given in (2.10) we use the enumeration

$$V_h^r = \mathrm{Vect}\{\underline{\psi}_{p,e,l}\} \quad \forall\,(p,e,l) \in \{1 \dots (r+1)^2\} \times \{1 \dots N_e\} \times \{1,2\},$$

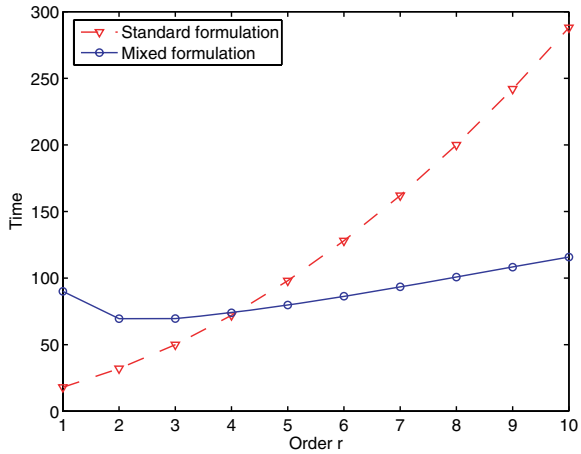


FIG. 2.4. *Compution time versus order for standard and mixed formulations.*

where p is still an index for local degrees of freedom, e is an index that describes elements, and l is an index that describes the vectorial components. Let K_e be the eth-element of the mesh. Then, the basis functions of V_h^r are constructed via the Piola transform and are given by the algebraic expression

$$\underline{\psi}_{p,e,l} \circ \underline{F}_{K_e} = \frac{\widehat{\varphi}_p}{|J_{K_e}|} DF_{K_e} \overrightarrow{e}_l. \tag{2.20}$$

With this notation we have

$$\begin{aligned} \oint_{K_e} \underline{\psi}_{p,e,l} \cdot \varphi_q \overrightarrow{e}_i \, dx &= \oint_{\widehat{K}} (\underline{\psi}_{p,e,l} \circ \underline{F}_{K_e}) \cdot (\varphi_q \circ \underline{F}_{K_e}) \overrightarrow{e}_i \, |J_{K_e}| \, d\widehat{x} \\ &= \oint_{\widehat{K}} \widehat{\varphi}_p \, \widehat{\varphi}_q \, (DF_{K_e} \overrightarrow{e}_l) \cdot \overrightarrow{e}_i \, d\widehat{x}; \end{aligned} \tag{2.21}$$

see (2.20). By applying a Gauss–Lobatto quadrature formula we get

$$\begin{aligned} \oint_{K_e} \underline{\psi}_{p,e,l} \cdot \varphi_q \overrightarrow{e}_i \, dx &= \sum_{m=1}^{(r+1)^2} \widehat{\omega}_m \, [DF_{K_e}(\widehat{\xi}_m) \overrightarrow{e}_l] \cdot \overrightarrow{e}_i \, \delta_{p,m} \, \delta_{q,m} \\ &= \widehat{\omega}_q \, [DF_{K_e}(\widehat{\xi}_q) \overrightarrow{e}_l] \cdot \overrightarrow{e}_q \, \delta_{p,q}. \qquad \square \end{aligned}$$

To write the semidiscrete system, we denote by, respectively, $U_h(t), \Theta_h(t), \Gamma_h^{ij}(t)$, and $\mathcal{Q}_h(t)$ the components of $u_h(t), \theta^h(t), \gamma_{ij}^h(t)$, and $\mathcal{Q}^h(t)$ in the appropriate basis. Using the definitions of the mass, stiffness, and coupling matrices, (2.13)–(2.16) lead to the matricial formulation

$$\frac{\delta^3}{12} D \frac{d^2 \Theta_h^i(t)}{dt^2} + \frac{\delta^3}{12} \sum_{j=1}^{2} R \, \Gamma_h^{ij}(t) + \delta \, C^i \mathcal{Q}_h^i(t) = 0, \tag{2.22}$$

$$M_{ij} \Gamma_h^{ij}(t) = R \, \Theta_h^j(t), \tag{2.23}$$

$$\rho\delta \, D \frac{d^2 U_h(t)}{dt^2} + \delta \, R^* \mathcal{Q}_h(t) = \delta \, F_h(t), \tag{2.24}$$

$$B \mathcal{Q}_h(t) = \mu \, [R U_h(t) + C^* \Theta_h(t)]. \tag{2.25}$$

In (2.22), the matrices C_i are related to C by the identity $C = [C_1\ C_2]$. The matrices B and M_{ij} being invertible, we can substitute $M_{ij}^{-1} R\ \Theta_h^j(t)$ and $\mu\ B^{-1}[RU_h(t) + C^*\Theta_h(t)]$ for the auxiliary unknowns $\Gamma_h^{ij}(t)$ and $\mathcal{Q}_h(t)$ (needed for the mixed formulation). To have a block expression we set

$$\mathcal{D} = \begin{pmatrix} D & 0 \\ 0 & D \end{pmatrix} \quad \text{and} \quad \mathcal{K} = \begin{pmatrix} R^*(M^{11})^{-1}R & R^*(M^{12})^{-1}R \\ R^*(M^{21})^{-1}R & R^*(M^{22})^{-1}R \end{pmatrix}.$$

So, the semidiscrete scheme that we propose the for Reissner–Mindlin plate equation is

$$\frac{\delta^3}{12}\,\mathcal{D}\frac{d^2\Theta_h(t)}{dt^2} + \frac{\delta^3}{12}\,\mathcal{K}\,\Theta_h(t) + \delta\ C\ B^{-1}\Big[R\,U_h(t) + C^*\Theta_h(t)\Big] = 0, \tag{2.26}$$

$$\delta D\frac{d^2U_h(t)}{dt^2} + \delta\ R^*B^{-1}\Big[R\,U_h(t) + C^*\Theta_h(t)\Big] = \delta\ F_h(t). \tag{2.27}$$

2.5. About error estimates and numerical integration. The error analysis of our method, in particular, the justification of the use of quadrature formula, is still an open question to our knowledge and will be the object of future research. Proposition 4 shows that we have to consider the analysis of a classical hyperbolic system of order 2 in space. We recall here some known results in this direction. There are two sources of error—the error due to the approximation of $H_0^1(\Sigma)$ that can be reduced to the study of interpolation error and the error related to numerical integration. One difficulty of the analysis is that our finite element subspaces are constructed from a space of reference shape functions which are then transformed to quadrilateral elements via the bilinear (and not affine) isomorphism F_K. The general theory of such finite elements is quite recent; however, in [2], [3], and [5] the authors give criteria on the mesh geometry to have the following estimates:

$$\inf_{v\in U_h^r} \|(u-v)(t)\|_{L^p(\Sigma)} \le Ch^{r+1}|u(t)|_{W^{p,r+1}}, \tag{2.28}$$

$$\inf_{v\in U_h^r} \|\nabla(u-v)(t)\|_{L^p(\Sigma)} \le Ch^{r}|u(t)|_{W^{p,r+1}}. \tag{2.29}$$

Then, using technical arguments given in [8] and [11] we prove the optimal error estimate for our finite elements method when the mass and stiffness matrices are computed exactly. Concerning quadratures formulas, using the approach in [8] we can show that the Gauss–Lobatto quadrature rule keeps the order of the polynomial approximation if we compute the stiffness matrix exactly. As far as we know, for general quadrilaterals the question of the influence of numerical integration is not solved because this case requires numerical integration on rational fractions. However, numerical experiments performed on the Helmholtz equation showed an optimal convergence with general quadrilateral meshes using the Gauss–Lobatto quadrature rule [12].

3. Time discretization and stability analysis.

3.1. The leapfrog scheme. Common software that solves the plate or shell equations uses, in the time domain, an implicit scheme for time discretization like the Newmark scheme. For our mass-lumped formulation, it is more suitable to use an explicit leapfrog scheme.

For $[0,T] \backsimeq \{t_n\} = n\Delta t\ ,\ \Delta t = \frac{T}{N}\ ,\ n \in [0,N]$, we set $U_h(t_n) = U_h^n,\ \Theta_h(t_n) = \Theta_h^n,\ \mathcal{Q}_h^n = \mathcal{Q}_h(t_n)$, and we consider the following scheme:

$$\frac{\delta^3}{12}\,\mathcal{D}\frac{\Theta_h^{n+1} - 2\Theta_h^n + \Theta_h^{n-1}}{\Delta t^2} + \frac{\delta^3}{12}\,\mathcal{K}\,\Theta_h^n + \delta\,C\mathcal{Q}_h^n = 0, \tag{3.1}$$

$$\rho\delta\,D\frac{U_h^{n+1} - 2U_h^n + U_h^{n-1}}{\Delta t^2} + \delta\,R^*\mathcal{Q}_h^n = \delta\,F_h^n, \tag{3.2}$$

$$B\,\mathcal{Q}_h^n = \mu\,(R\,U_h^n + C^*\Theta_h^n). \tag{3.3}$$

Remark 6. In practice (for the implementation) we use the leapfrog scheme for (2.26) and (2.27). For the stability analysis, we prefer to keep the unknown $\mathcal{Q}$ which justifies (3.3).

3.2. Stability results. Because of the mass-lumping principle, our time scheme is explicit. We prove in this section (by a classical technique using a discrete energy identity) that it is stable under a CFL condition. First, we introduce the notation

$$\left|\begin{array}{lcll} b_h : (X,Y) & \longrightarrow & (B\,X)\cdot Y, & \\ k_h : (X,Y) & \longrightarrow & (\mathcal{K}\,X)\cdot Y, & \\ (X,Y) & = & D\,X\cdot Y, & \text{and} \quad |X|^2 = (X,X) \end{array}\right.$$

for any vector $(X,Y) \in \mathbb{R}^{n_h}$. Then we define the discrete quantity at time $t_{n+\frac{1}{2}}$:

$$E_m^{n+\frac{1}{2}} = \frac{\delta^3}{12}\left|\frac{\Theta_h^{n+1} - \Theta_h^n}{\Delta t}\right|^2 + \delta\left|\frac{U_h^{n+1} - U_h^n}{\Delta t}\right|^2 + \frac{\delta^3}{12}k_h(\Theta_h^{n+1},\Theta_h^n) + \delta\,b_h(\mathcal{Q}_h^{n+1},\mathcal{Q}_h^n).$$

Proposition 7. [illegible] (3.1)–(3.3) [illegible]

$$\frac{E_m^{n+\frac{1}{2}} - E_m^{n-1/2}}{2\Delta t} = \delta\,F_h^n \cdot \frac{U_h^{n+1} - U_h^{n-1}}{2\Delta t}. \tag{3.4}$$

[illegible]. Multiplying (3.1) by $\frac{\Theta_h^{n+1} - \Theta_h^{n-1}}{2\Delta t}$ and (3.2) by $\frac{U_h^{n+1} - U_h^{n-1}}{2\Delta t}$ we get

$$\frac{\delta^3}{12}\left(\frac{\Theta_h^{n+1} - 2\Theta_h^n + \Theta_h^{n-1}}{\Delta t^2}, \frac{\Theta_h^{n+1} - \Theta_h^{n-1}}{2\Delta t}\right) + \frac{1}{2\Delta t}\frac{\delta^3}{12}\,[k_h(\Theta_h^n,\Theta_h^{n+1}) - k_h(\Theta_h^n,\Theta_h^{n-1})]$$
$$+ \frac{\delta}{2\Delta t}\,[C\mathcal{Q}_h^n\cdot\Theta_h^{n+1} - C\mathcal{Q}_h^n\cdot\Theta_h^{n-1}] = 0,$$

$$\delta\left(\frac{U_h^{n+1} - 2U_h^n + U_h^{n-1}}{\Delta t^2}, \frac{U_h^{n+1} - U_h^{n-1}}{2\Delta t}\right) + \frac{\delta}{2\Delta t}\,[R^*\mathcal{Q}_h^n\cdot U_h^{n+1} - R^*\mathcal{Q}_h^n\cdot U_h^{n-1}]$$
$$= \delta\,F_h^n\cdot\frac{U_h^{n+1} - U_h^{n-1}}{2\Delta t}.$$

Then we use the equality

$$(V^{n+1} - 2V^n + V^{n-1}, V^{n+1} - V^{n-1}) = |V^{n+1} - V^n|^2 - |V^n - V^{n-1}|^2 \qquad \forall V \in \mathbb{R}^{n_h}\ ,$$

and we obtain

(3.5)

$$\frac{\delta^3}{12}\left[\frac{1}{2\Delta t}\left|\frac{\Theta_h^{n+1}-\Theta_h^n}{\Delta t}\right|^2-\frac{1}{2\Delta t}\left|\frac{\Theta_h^{n}-\Theta_h^{n-1}}{\Delta t}\right|^2\right]+\frac{1}{2\Delta t}\frac{\delta^3}{12}[k_h(\Theta_h^n,\Theta_h^{n+1})-k_h(\Theta_h^n,\Theta_h^{n-1})]$$
$$+\frac{\delta}{2\Delta t}[\mathcal{Q}_h^n\cdot C^*\Theta_h^{n+1}-\mathcal{Q}_h^n\cdot C^*\Theta_h^{n-1}]=0,$$
$$\delta\left[\frac{1}{2\Delta t}\left|\frac{U_h^{n+1}-U_h^n}{\Delta t}\right|^2-\frac{1}{2\Delta t}\left|\frac{U_h^{n}-U_h^{n-1}}{\Delta t}\right|^2\right]+c\frac{\delta}{2\Delta t}[\mathcal{Q}^n\cdot RU_h^{n+1}-\mathcal{Q}^n\cdot RU_h^{n-1}]$$
$$=F_h^n\cdot\frac{U_h^{n+1}-U_h^{n-1}}{2\Delta t}.$$

We notice that

$$\mathcal{Q}^n\cdot C^*\Theta_h^{n+1}+\mathcal{Q}^n\cdot R\,U_h^{n+1}=\mathcal{Q}^n\cdot(R\,U_h^{n+1}+C^*\Theta_h^{n+1})=\mathcal{Q}^n\cdot B\mathcal{Q}^{n+1}=b_h(\mathcal{Q}^{n+1},\mathcal{Q}^n).$$

Since B is symmetric we show in the same way that

$$\mathcal{Q}^n\cdot C^*\Theta_h^{n-1}+\mathcal{Q}^n\cdot R\,U_h^{n-1}=b_h(\mathcal{Q}^n,\mathcal{Q}^{n-1}).$$

Finally, we use the symmetry of k_h : $k_h(\Theta_h^n,\Theta_h^{n+1})=k_h(\Theta_h^{n+1},\Theta_h^n)$, and by adding (3.5) and (3.5) we get (3.4). □

We now use this energy identity to write a condition of stability for the scheme (3.1)–(3.3).

1. Let

$$\mathcal{M}_h=\begin{pmatrix}\frac{\delta^3}{12}\mathcal{D} & 0\\ 0 & \delta D\end{pmatrix}\text{ and }\mathcal{S}_h=\begin{pmatrix}\frac{\delta^3}{12}\mathcal{K}+\delta\,C\,B^{-1}C^* & \delta\,C\,B^{-1}R\\ \delta\,R^*B^{-1}C^* & \delta\,R^*B^{-1}R^*,\end{pmatrix}.$$

Using the same substitution as that needed to obtain (2.26)–(2.27) the scheme (3.1)–(3.3) can be rewritten as

$$\mathcal{M}_h\frac{V_h^{n+1}-2V_h^n+V_h^{n-1}}{\Delta t^2}+\mathcal{S}_hV_h^n=\mathcal{F}_h^n\ ,\ V_h^n=\begin{pmatrix}\Theta_h^n\\ U_h^n\end{pmatrix},\ \mathcal{F}_h^n=\begin{pmatrix}\delta F_h^n\\ 0\end{pmatrix}.$$

With this notation, we can also write

$$E_m^{n+\frac{1}{2}}=\mathcal{M}_h\frac{V_h^{n+1}-V_h^n}{\Delta t}\cdot\frac{V_h^{n+1}-V_h^n}{\Delta t}+\mathcal{S}_hV_h^{n+1}\cdot V_h^n.$$

Using a discrete Gronwall lemma, (3.4) enables us to write the following proposition.

PROPOSITION 8.

$$\frac{\Delta t^2}{4}\sup_{V_h\in\mathbb{R}^{3n_h}}\frac{\mathcal{S}_hV_h\cdot V_h}{\mathcal{M}_hV_h\cdot V_h}<1.$$

We can show that a necessary condition to satisfy the condition in Proposition 8 is

$$\frac{\Delta t}{h}<\alpha,$$

where α depends on the highest eigenvalue of the matrix $\mathcal{M}_h^{-1}\mathcal{S}_h$. In practice we compute the limit of the sequence

$$\frac{\|X_{n+1}\|_{l^2}}{\|X_n\|_{l^2}},\ \text{where } X_n \text{ is given by } X_{n+1}=\mathcal{M}_h^{-1}\mathcal{S}_hX_n$$

to have an approximation of this eigenvalue.

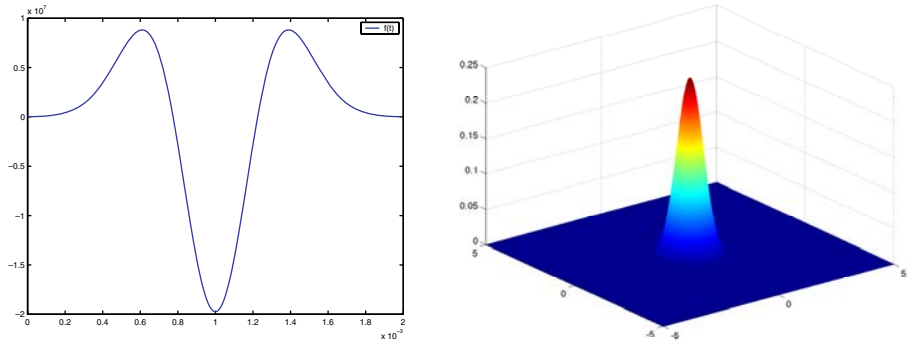


FIG. 4.1. *The loading force: on the left f_1, and on the right f_2.*

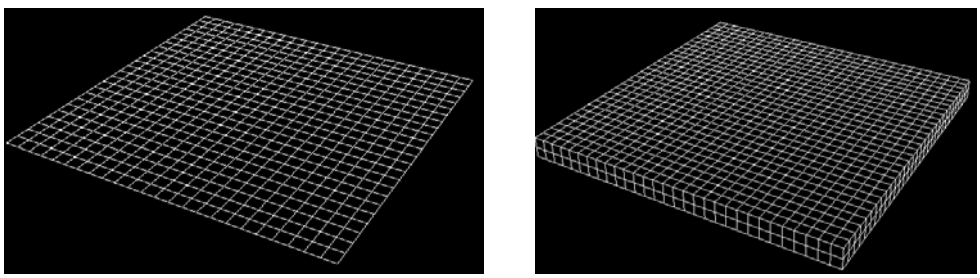


FIG. 4.2. 2D *and* 3D *meshes for the plate and the elastic media.*

4. Numerical results. In the following, the loading force $f(t,x) = f_1(t)f_2(x)$, where f_2 is a Gaussian-type function centered at x_0 with radius $r = 0.5$ and

$$f_1(t) = 2\pi^2 f_0^2(2\pi^2(f_0 t - 1)e^{-\pi^2(f_0 t-1)^2}$$

is a Ricker-type function with central frequency $f_0 = 1000$ Hz (Figure 4.1). We consider an aluminium plate, the physical coefficients of which are

$$\rho = 2700 kg.m^{-3}, \quad E = 7.4\ 10^{10} P_a, \quad \nu = 0.34.$$

The dimensions of the plate are $\Sigma = [-0.5; 0.5] \times [-0.5; 0.5]$, and we denote $x_0 = (0, 0)$ as its center. For all simulations, we consider the time interval $[0; T] = [0; 2e-2]$ which corresponds to many reflections of propagating waves in the plate.

4.1. A validation of the method. To the best of our knowledge, there is no analytical solution for the Reissner–Mindlin plate model. So, in order to validate our method and our results, we use the fact that the plate model is asymptotic to the linear elasticity system. We compute a 3D displacement field $U(t,x)$, solution of (1.1), with a code implemented from Cohen and Fauqueux's approach [10] and we compute $u(t,x)$ by our finite element method. Toward this end, we use a 3D mesh with two space steps in the thickness and a 2D mesh of the midsurface of the 3D mesh (Figure 4.2). We choose $\mathcal{Q}_5$ finite elements for the two computations. Then, taking into account relation (1.3), we compare the two quantities $U_3(t,x_0)$ and $u(t,x_0)$ as a function of time for different cases when the thickness goes to zero.

We can see in Figures 4.3, 4.4, and 4.5 that when the thickness goes to zero, the two solutions tend to be identical.

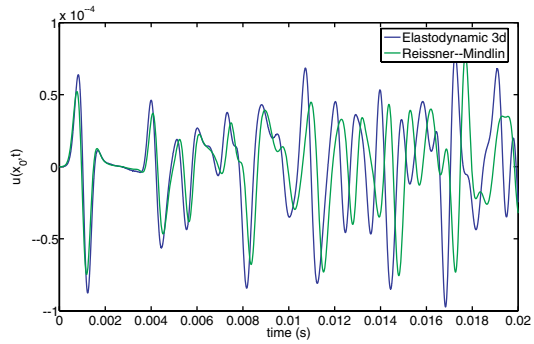


FIG. 4.3. *Comparaison with* $\delta = 1.4cm$.

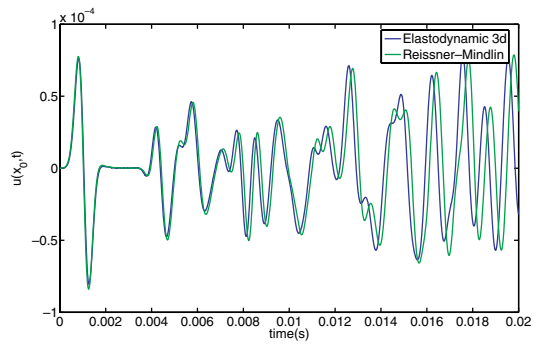


FIG. 4.4. *Comparaison with* $\delta = 0.5cm$.

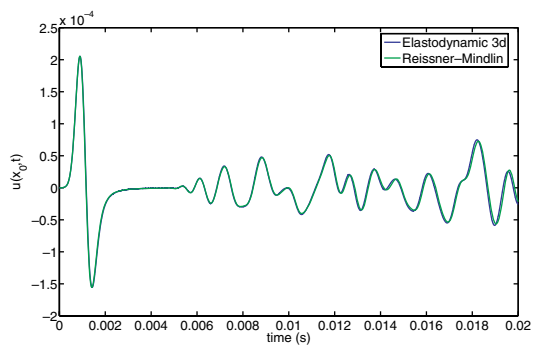


FIG. 4.5. *Comparaison with* $\delta = 0.01cm$.

4.2. Influence of the order of the method on CPU time and accuracy. In this section we want to show the advantages of our scheme in terms of accuracy and computational time. We fix $\delta = 1mm$ and we compute a reference solution with $\mathcal{Q}_7$-finite elements with an extremely fine mesh (5000 elements). Then, we compute the solution for orders $r = 2$ to $r = 7$ with a regular mesh with a space step h_r, such that the relative error (defined by (4.1)) of this solution compared to the reference solution is around 5%. For each case of computation, we take the optimal time step in terms of computation time, i.e., the maximum Δt authorized by the stability condition. In practice we compute the relative error e_h at the point x_0, so that our criterion of

accuracy is

$$e_h = \frac{\|u_h(x_0,t) - u_{ref}(x_0,t)\|_{l_2[0,T]}}{\|u_{ref}(x_0,t)\|_{l_2[0,T]}} \simeq 5\%. \tag{4.1}$$

The results we obtain are given in the following table:

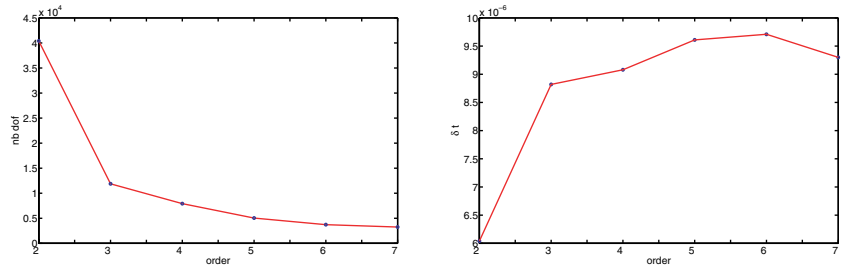
Order	Elements	Storage	Δt	CPU time(s)	$e_h(\%)$
2	10000	40401 dof	0.604e-5	556	5.2
3	1296	11881 dof	0.882e-5	84	5.4
4	484	7921 dof	0.908e-5	49	4.8
5	196	5041 dof	0.961e-5	26	4.9
6	100	3721 dof	0.971e-5	18	5.1
7	64	3249 dof	0.93e-5	16	5

The results of this table clearly illustrate the advantage of using higher order elements. The use of $\mathcal{Q}_7$-finite elements divides by 3 the computation time compared to $\mathcal{Q}_4$-finite elements, and divides by 35 the computation time compared to $\mathcal{Q}_2$-finite elements. We can also see the gain in terms of memory storage: Indeed, both the number of elements and the number of degrees of freedom needed to satisfy (4.1) decrease when we use higher orders.

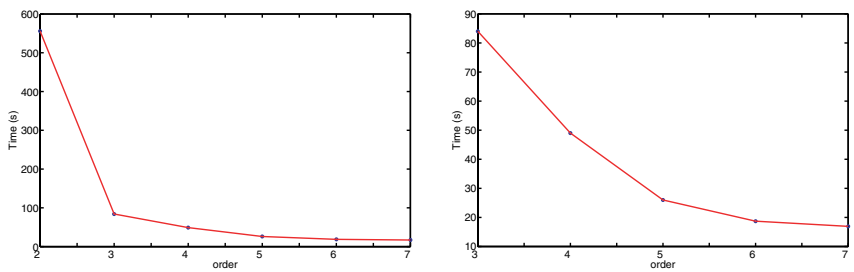
One can, however, notice a slower convergence for higher orders. This is due to the increase of the size of the discrete operators with the order, which finally balance and even overcome the gain realized by the use of larger elements. On the other hand, the step Δt remains approximately the same for orders from 4 to 7. Indeed, even if a high order enforces a smaller Δt than for low orders, we can use a less fine mesh to satisfy (4.1) and thus the CFL condition remains almost constant. All these comments are illustrated by Figure 4.6. Figure 4.6(a) represents the number of degrees of freedom as a function of the order to satisfy (4.1), Figure 4.6(b) represents the optimal step Δt as a function of the order, and Figure 4.6(c) represents the CPU time as a function of the order with a zoom at large orders in Figure 4.6(d).

4.3. Influence of the thickness in the finite element approximation. The previous table shows that for low orders (orders 2 and 3) the convergence is abnormally slow, and we need very many degrees of freedom (a very small space step) to be accurate. This is a locking phenomenon. Many papers [1], [4], [7] deal with this problem, which is not the subject of this paper. However, we were interested in this aspect. Figures 4.6 and 4.7 represent the number of degrees of freedom (and in some sense the size of the space step) needed to satisfy the criterion of accuracy (4.1) for different thicknesses. We can see that as the thickness goes to zero, the number of degrees of freedom needed to satisfy (4.1) becomes higher and higher for orders 2 and 3, while this is not the case from the order 4. For orders higher than 4, the influence of the thickness on the mesh is weak. Our conclusion is that a way to weaken locking phenomena is to use a finite element with a high enough order. Nevertheless we do not claim that this method eliminates numerical locking.

4.4. The case of a heterogeneous plate. To conclude we present some snapshots from a numerical simulation for a heterogeneous plate. The plate consists of three media. Medium 1 is plexiglass where the smallest velocity of propagating waves in the plate is 758 $m.s^{-1}$. Medium 2 is aluminium where the smallest velocity of propagating waves in the plate is 3097 $m.s^{-1}$, and medium 3 is cast iron where the smallest velocity of propagating waves in the plate is 3215 $m.s^{-1}$. So we have to construct a structured mesh that will be finer in the first media than in the two



(a) Degree of freedom/order (b) Δt/order



(c) CPU time/order (d) CPU time/order

FIG. 4.6. *Influence of the order on the performance of the method.*

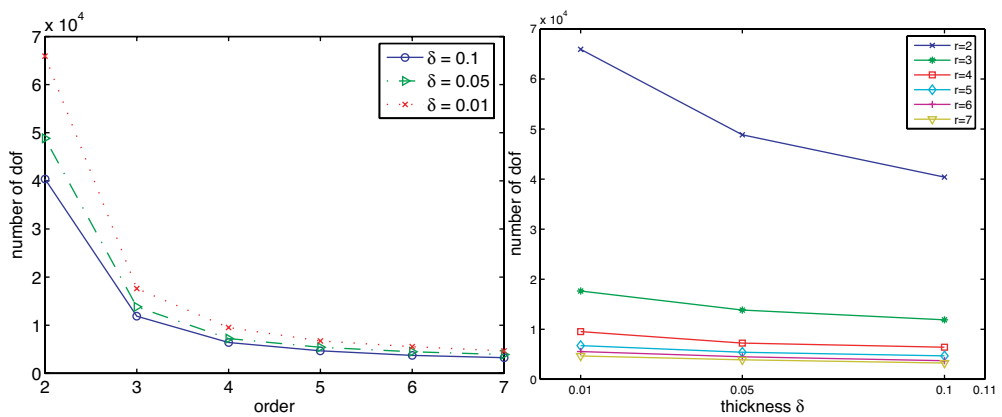


FIG. 4.7. *Numerical locking for low orders.*

others. Most of the software available gives appropriate triangular meshes. Then we obtain our quadrilateral mesh from a triangular mesh by cutting each triangle in three quadrilaterals. The mesh and the way to get the mesh are illustrated in Figure 4.8. We center the Gaussian function at the point $(-1, 0)$ in the medium 2.

Figure 4.9 brings to the fore that the plate equation is dispersive. The picture on the left shows the signal at the beginning of the simulation and the picture and the right shows the solution after a short time of propagation. We see the modification of the wave front during this time. We can see in Figure 4.10 that propagated waves are diffracted by the plexiglass because the velocity is slower than in the other media. We

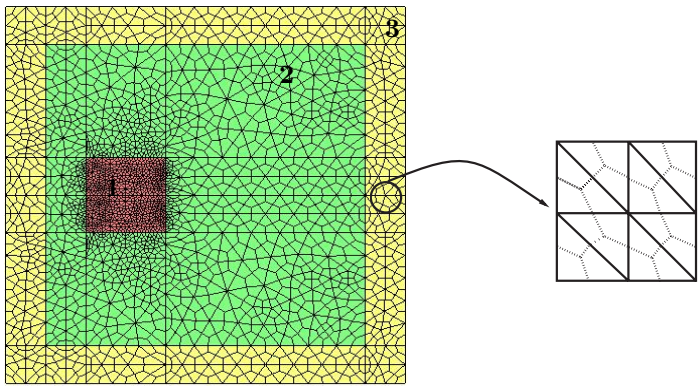


FIG. 4.8. *The mesh of the heterogeneous plate.*

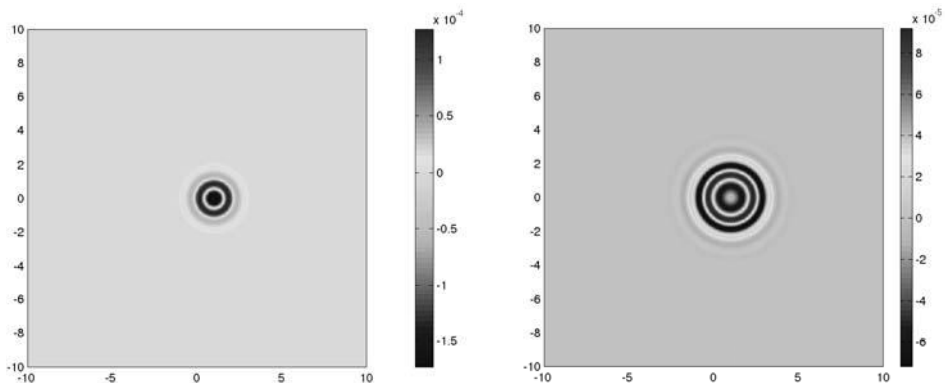


FIG. 4.9. *Snapshots of the bending displacement.*

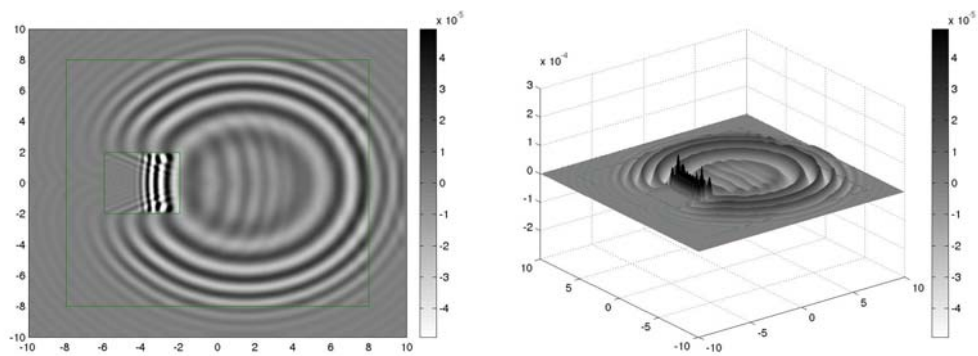


FIG. 4.10. *Snapshots of the bending displacement at the time* $t = 1.7 \times 10^{-4} s$.

do not see a similar phenomenon at the interface of aluminium and cast iron because the velocities are almost equal. To conclude, the amplitude of the displacement is higher in plexiglass than in aluminium and is higher in aluminium than in cast iron. This is consistent with the stiffness properties of each medium.

Conclusion. In this paper we have proposed a robust and accurate finite element method in the time domain for the Reissner–Mindlin plate model. This method is based on mixed spectral finite elements with mass lumping. A mixed formulation of the model has enabled us to construct an approximation space with the Piola transform to avoid the storage of the stiffness matrix which induces a very substantial gain in memory storage. Then we have proposed an explicit leapfrog scheme thanks to the mass-lumping principle, which induces a gain of CPU time. After a validation of our method with results of a 3D code, numerical experiments demonstrate the usefulness of high orders in terms of computation time, memory storage, and accuracy. Finally, we have brought to the fore that the use of high orders is a way to weaken locking phenomena.

REFERENCES

[1] D.N. Arnold and F. Brezzi, *Locking-free finite element methods for shells*, Math. Comp., 66 (1997), pp. 1–14.

[2] D.N. Arnold, D. Boffi, and R. Falk, *Approximation by quadrilateral finite elements*, Math. Comp., 71 (2002), pp. 909–922.

[3] D.N. Arnold, D. Boffi, and R. Falk, *Quadrilateral H(*div*) finite elements*, SIAM J. Numer. Anal., 42 (2005), pp. 2429–2451.

[4] F. Brezzi, M. Fortin, and R. Stenberg, *Error analysis of mixed-interpolated elements for Reissner-Mindlin plates*, Math. Models Methods Appl. Sci., 1 (1991), pp. 125–151.

[5] P.G. Ciarlet, *Handbook of Numerical Analysis, Vol.* 2, North–Holland, Amsterdam, 1991.

[6] D. Chapelle and K. Bathe, *The Finite Element Analysis of Shells—Fundamentals*, Springer-Verlag, Berlin, 2003.

[7] D. Chapelle and R. Stenberg, *An optimal low-order locking-free finite element method for Reissner-Mindlin plates*, Math. Models Methods Appl. Sci., 8 (1998), pp. 407–430.

[8] G. Cohen, P. Joly, J. E. Roberts, and N. Tordjman, *Higher order triangular finite elements with mass lumping for the wave equation*, SIAM J. Numer. Anal., 38 (2001), pp. 2047–2078.

[9] G. Cohen, *Higher Order Numerical Methods for Transient Wave Equations*, Springer-Verlag, New York, 2001.

[10] G. Cohen and S. Fauqueux, *Mixed spectral finite elements for the linear elasticity system in unbounded domains*, SIAM J. Sci. Comput., 26 (2005), pp. 864–884.

[11] T. Dupont, *L^2-estimates for Galerkin methods for second order hyperbolic equations*, SIAM J. Numer. Anal., 10 (1973), pp. 880–889.

[12] M. Duruflé, *Intégration numérique et méthodes d'éléments finis d'ordre élevé pour les équations de Maxwell en régime harmonique*, Ph.D. thesis, University of Paris-Dauphine, 2006.

[13] S. Fauqueux, *Elements finis mixtes spectraux et couches absorbantes parfaitement adaptées pour la propagation d' ondes élastiques en régime transitoire*, Ph.D. thesis, University of Paris-Dauphine, 2003.

[14] Y. Maday and E. Rønquist, *Optimal error analysis of spectral methods with emphasis on non-constant coefficients and deformed geometries*, Comput. Methods Appl. Mech. Engrg., 80 (1990), pp. 91–115.

[15] R.D. Mindlin, *Influence of rotary inertia and shear on flexural motion of isotropic elastic plates*, J. Appl. Mech., 18 (1951), pp. 31–38.

[16] W.A. Mulder, *Higher-order mass-lumped finite elements for the wave equation*, J. Comput. Acoust., 9 (2001), pp. 671–680.

[17] P.M. Naghdi, *Foundations of elastic shell theory*, in Progress in Solid Mechanics, Vol. 4, North–Holland, Amsterdam, 1963, pp. 1–90.

[18] S. Pernet, *Etude de méthodes d'ordre élevé pour résoudre les équations de Maxwell dans le domaine temporel. Application à la détection et à la compatibilité électromagnétique*, Ph.D. thesis, University of Paris-Dauphine, 2004.

[19] E. Reissner, *The effect of tranverse shear deformation on the bending of elastic plates*, J. Appl. Mech., 12 (1945), pp. 69–77.

[20] E. Reissner, *Stress strain relations in the theory of thin elastic shells*, J. Math. Phys., 31 (1952), pp. 109–119.

[21] O. Zienkiewicz and R. Taylor, *The Finite Element Method, Vol.* 2, Butterworth–Heinemann, Oxford, UK, 2000.

SIAM J. SCI. COMPUT.
Vol. 29, No. 3, pp. 1006–1041
© 2007 Society for Industrial and Applied Mathematics

ON THE VARIATIONAL APPROXIMATION OF COMBINED SECOND AND FOURTH ORDER GEOMETRIC EVOLUTION EQUATIONS*

JOHN W. BARRETT†, HARALD GARCKE‡, AND ROBERT NÜRNBERG†

Abstract. We present a variational formulation of combined motion by minus the Laplacian of curvature and mean curvature flow, as well as related flows. The proposed scheme covers both the closed curve case and the case of curves that are connected via triple or quadruple junction points or intersect the external boundary. On introducing a parametric finite element approximation, we prove stability bounds and compare our scheme with existing approaches. The presented scheme has very good properties with respect to the equidistribution of mesh points and, if applicable, area conservation.

Key words. surface diffusion, (inverse) mean curvature flow, surface attachment limited kinetics, nonlinear curve evolution, triple junctions, parametric finite elements, Schur complement, tangential movement

AMS subject classifications. 65M60, 65M12, 35K55, 53C44

DOI. 10.1137/060653974

1. Introduction.

The motion of curves or surfaces driven by second or fourth order geometric evolution equations arises in many applications in materials science and, of course, in differential geometry. Well known is the mean curvature flow, where a hypersurface moves in the direction of its mean curvature vector. Also frequently arising are evolution laws where the normal speed of an evolving hypersurface is given as a function of mean curvature, in applications, such as image processing or mathematical physics. In mathematical physics the inverse mean curvature flow plays a role in the context of the positive mass conjecture. In materials science the motion of networks of curves or surfaces is also often important. The evolution of grain networks—arising in polycrystalline materials—is also given by mean curvature flow, where at junctions angle conditions have to hold.

Fourth order geometric evolution equations also frequently are found in geometry and materials science. The situation that the normal velocity of a hypersurface is given by minus the Laplacian of mean curvature is called surface diffusion. This evolution law arises in situations where the diffusion of material is restricted to interfacial regions; see, e.g., [36]. When many phases appear, networks also have to be taken into account. As the evolution law is of fourth order, additional conditions, which act as boundary conditions for the evolution law on the curves or surfaces, have to hold.

The goal of this paper is to propose and analyze a new approach to numerically solving geometric evolution laws of second and fourth order. The numerical method is variational and very flexible. In particular, it is possible to couple fourth and second order laws on the surfaces or at triple junctions in a straightforward way.

*Received by the editors March 9, 2006; accepted for publication (in revised form) October 25, 2006; published electronically May 10, 2007.
http://www.siam.org/journals/sisc/29-3/65397.html
†Department of Mathematics, Imperial College, London, SW7 2AZ, UK (j.barrett@imperial.ac.uk, robert.nurnberg@imperial.ac.uk).
‡Naturwissenschaftliche Fakultät I – Mathematik, Universität Regensburg, 93040 Regensburg, Germany (harald.garcke@mathematik.uni-regensburg.de).

There exist numerous methods for numerically solving geometric evolution equations. Approaches are based on, e.g., the parametric formulation, the graph formulation, the level set method, or the phase field approach. We do not intend to give an outline of all these methods, but rather refer to [14] for a comprehensive recent review. Our approach makes use of a fundamental idea in [16], which used the identity

$$\Delta_s \, \vec{x} = \vec{\varkappa}, \tag{1.1}$$

where Δ_s is the surface Laplacian, $\vec{x}$ is the position vector, and $\vec{\varkappa}$ is the mean curvature vector, for the first time in order to design a finite element method for geometric partial differential equations and mean curvature flow; see also [17]. A second major idea stems from the paper [3], which came up with a splitting method and employed a Schur complement approach, in order to compute solutions of the surface diffusion law

$$\mathcal{V} = -\Delta_s \, \varkappa, \tag{1.2}$$

where $\varkappa$ is the sum of the principal curvatures of Γ and $\mathcal{V}$ is the normal velocity of the surface.

The approaches of Dziuk and of Bänsch, Morin, and Nochetto both require that the parameterization evolve only in the normal direction. In the recent paper [6], the present authors introduced a novel parametric finite element approximation for surface diffusion that also allowed for a tangential movement of mesh points, which of course does not change the geometry. This new approach has the advantage that the discretization has very good properties with respect to the equidistribution of mesh points. In the case of surface diffusion of curves, a semidiscrete version of this scheme, i.e., a scheme where one discretizes only with respect to space, leads to a precise equidistribution of mesh points. With the new ansatz it is also possible to formulate the rather complicated conditions that need to hold at triple junction points in a natural variational way. In this paper we want to demonstrate that the ideas developed in [6] can also be used to compute other geometric evolution laws of both second and fourth order. For instance, the inverse mean curvature flow and an evolution flow which couples mean curvature flow to surface diffusion at triple junctions (see, e.g., [12]) can be approximated in the spirit of the approach proposed in [6].

In this paper, we will consider evolution laws that couple fourth and second order geometric evolution laws. The coupling can be either through triple junctions or through a coupling that combines mean curvature flow and the fourth order surface diffusion flow via an appropriate interpolation; see [42] for applications of this flow in materials science. Let us introduce first the case of pure second order geometric evolution equations; that means in our context situations where the normal velocity is given by a function of mean curvature. For a closed hypersurface Γ in $\mathbb{R}^d$, mean curvature flow is given by

$$\mathcal{V} = \varkappa \tag{1.3}$$

or, more generally, we are also going to consider flows of the form

$$\mathcal{V} = f(\varkappa), \tag{1.4}$$

where $f : (a, b) \to \mathbb{R}$ with $-\infty \leq a < b \leq \infty$ is a strictly monotonically increasing continuous function, e.g.,

$$f(r) := |r|^{\beta-1} r, \quad \beta \in \mathbb{R}_{>0}\,; \tag{1.5}$$

see [35] and the references therein. For example, the evolution law (1.4), with (1.5) for $\beta = \frac{1}{3}$, has been studied in [1, 40, 2].

We will also consider computations for (1.4) with

$$f(r) := -r^{-1}, \tag{1.6}$$

i.e., the inverse mean curvature flow; see, e.g., [29, 32] for the origins of this flow in mathematical physics, and [31], and the references therein, for a consideration of this flow in differential geometry. To our knowledge, the only numerical results in the literature for (1.4) with (1.6) can be found in [38, 22], where a finite volume and a finite element approximation, respectively, of a regularized level set formulation of (1.4) with (1.6) are employed. We know of no direct approach for the approximation of the inverse mean curvature flow in the literature.

For a parameterization $\vec{x}(\rho, t) \in \mathbb{R}^d$ of Γ, (1.4) can be written as a second order equation:

$$\mathcal{V} := \vec{x}_t \,.\, \vec{\nu} = f(\varkappa), \qquad \varkappa\,\vec{\nu} = \Delta_s\,\vec{x}, \tag{1.7}$$

where $\vec{\nu}$ is a unit normal to Γ. Note that because the tangential component $\vec{x}_t - (\vec{x}_t \,.\, \vec{\nu})\,\vec{\nu}$ of the velocity $\vec{x}_t$ is not prescribed in (1.7), there exists a whole family of solutions $\vec{x}$, even though the evolution of Γ is uniquely determined by (1.3). Our numerical scheme will directly discretize (1.7), in contrast to the scheme considered by Dziuk (see [16, 17]), which discretizes $\vec{x}_t = \Delta_s\,\vec{x}$ and hence, on noting (1.1) and that $\vec{\varkappa} = \varkappa\,\vec{\nu}$, enforces a movement of the parameterization $\vec{x}$ in the normal direction only.

A version of (1.4) that preserves the enclosed volume is given by

$$\mathcal{V} = f(\varkappa) - \frac{\int_\Gamma f(\varkappa)\,\mathrm{d}s}{\int_\Gamma 1\,\mathrm{d}s}, \tag{1.8}$$

the so-called conserved mean curvature flow, also called surface attachment limited kinetics (SALK), if $f(r) := r$. An intermediate law between (1.4), with $f(r) := r$, and (1.2) is the following evolution law:

$$\mathcal{V} = -\Delta_s \left(\frac{1}{\alpha} - \frac{1}{\xi}\,\Delta_s\right)^{-1} \varkappa, \tag{1.9}$$

where $\alpha, \xi \in \mathbb{R}_{>0}$. The flow (1.9) interpolates between surfaces diffusion (1.2) and SALK, (1.8) with $f(r) := r$, and was first discussed in [42]; see also [20]. It is similar to (1.2) and (1.8) in that the enclosed volume is conserved while the area of the hypersurface decreases. We observe that for $\alpha \to \infty$ and $\xi = 1$ the solutions to (1.9) converge to solutions of (1.8) with $f(r) := r$, as has been shown in [21]. In the limit $\xi \to \infty$ and $\alpha = 1$ the solutions to (1.9) should converge to solutions of the law (1.2), although to our knowledge no rigorous proof exists. Given a parameterization $\vec{x}(\rho, t) \in \mathbb{R}^d$ of Γ, (1.9) can be written as a system of second order equations:

$$\vec{x}_t \,.\, \vec{\nu} = -\Delta_s\, y\,, \qquad \left(\frac{1}{\alpha} - \frac{1}{\xi}\,\Delta_s\right) y = \varkappa\,, \qquad \varkappa\,\vec{\nu} = \Delta_s\,\vec{x}. \tag{1.10}$$

Analogously surface diffusion can be rewritten as

$$\vec{x}_t \,.\, \vec{\nu} = -\Delta_s\,\varkappa, \qquad \varkappa\,\vec{\nu} = \Delta_s\,\vec{x}. \tag{1.11}$$

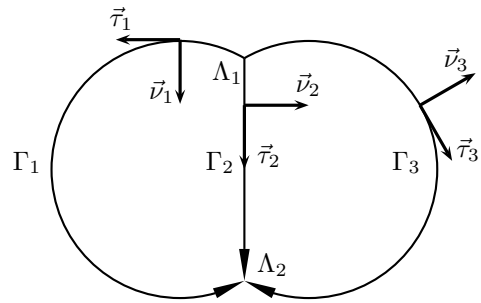


FIG. 1. *The setup of* $\Gamma = (\Gamma_1, \Gamma_2, \Gamma_3)$.

In section 2, we will consider a finite element approximation of a variational formulation of (1.11), as well as (1.7), (1.8), and (1.10), where throughout we will restrict our attention to the case $d = 2$, i.e., curves in the plane. In addition, we will compare our approximation of (1.3) with two other parametric approximations in the literature, namely, [17, 13].

A network of curves under motion by mean curvature with triple junctions can also be considered. In the example network in Figure 1, let Γ_1, Γ_2, Γ_3 be the given curves in $\mathbb{R}^d$, $d = 2$, that intersect at two triple junction points Λ_1 and Λ_2. Then the normal velocity for each curve is given by

$$\mathcal{V}_i = \sigma_i \, \varkappa_i, \qquad i = 1 \to 3, \tag{1.12}$$

where $\varkappa_i$ is the curvature of Γ_i and σ_i is the surface energy density of Γ_i. Let $\vec{\tau}_i \in \mathbb{R}^d$ be the unit tangent to Γ_i pointing away from the triple junction point Λ_1 and towards point Λ_2. The curvature is said to be positive if Γ_i is curved in the direction of the normal $\vec{\nu}_i \in \mathbb{R}^d$, which is the unique unit vector such that $(\vec{\tau}_i, \vec{\nu}_i)$ forms a positively orientated orthonormal system. For parameterizations $\vec{x}_i : [0,1] \times [0,T] \to \mathbb{R}^d$ of Γ_i, $i = 1 \to 3$, (1.12) can be written as a system of second order equations:

$$(\vec{x}_i)_t \, . \, \vec{\nu}_i = \sigma_i \, \varkappa_i, \qquad \varkappa_i \, \vec{\nu}_i = \Delta_s \, \vec{x}_i. \tag{1.13}$$

Then, in addition to (1.12), the following conditions have to hold at the triple junction points Λ_1 and Λ_2:

$$\text{the triple junction does not pull apart,} \tag{1.14a}$$

$$\sigma_1 \, \vec{\tau}_1 + \sigma_2 \, \vec{\tau}_2 + \sigma_3 \, \vec{\tau}_3 = \vec{0}. \tag{1.14b}$$

The condition (1.14a) is an attachment condition, and (1.14b) is Young's law, which is a balance-of-force equation at the triple junction. Young's law is equivalent to the angle condition $\frac{\sin\theta_1}{\sigma_1} = \frac{\sin\theta_2}{\sigma_2} = \frac{\sin\theta_3}{\sigma_3}$, where

$$\theta_1 = \sphericalangle(\vec{\tau}_2, \vec{\tau}_3), \quad \theta_2 = \sphericalangle(\vec{\tau}_3, \vec{\tau}_1), \quad \text{and} \quad \theta_3 = \sphericalangle(\vec{\tau}_1, \vec{\tau}_2)\,. \tag{1.15}$$

A variational formulation of (1.13) with (1.14a,b) will form the basis for the scheme that we present in section 2.

In most physical applications only triple junctions are of interest. However, in certain situations quadruple junctions are also possible. In the case that four interfaces meet at a quadruple junction the balance-of-force condition of the quadruple junction is

$$\sigma_1 \, \vec{\tau}_1 + \sigma_2 \, \vec{\tau}_2 + \sigma_3 \, \vec{\tau}_3 + \sigma_4 \, \vec{\tau}_4 = \vec{0}\,, \tag{1.16}$$

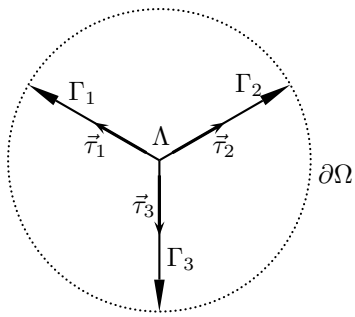


FIG. 2. *The second possible setup of* $\Gamma = (\Gamma_1, \Gamma_2, \Gamma_3)$.

where we use notation analogous to that in (1.14b); e.g., the four curves Γ_i, $i = 1 \to 4$, meeting at the quadruple junction are parameterized such that the tangent vectors point away from the junction. In this case the angles at the quadruple junction are not specified, and in fact there is one degree of freedom. For details we refer to [10, 11, 24].

In [6] the possibility of curve intersections with a fixed external boundary $\partial\Omega$, where Ω is a domain in $\mathbb{R}^d$, was not considered (see Figure 2 for an example). To state the necessary conditions that have to hold at an intersection with the external boundary we assume that $\partial\Omega$ is given by a function $F \in C^1(\mathbb{R}^d)$ such that

$$\partial\Omega = \{\vec{z} \in \mathbb{R}^d : F(\vec{z}) = 0\} \qquad \text{and} \qquad |\nabla F(\vec{z})| = 1 \quad \forall\ \vec{z} \in \partial\Omega\,.$$

Then, for three given curves Γ_i, $i = 1 \to 3$, evolving according to (1.12), the conditions (1.14a,b) need to hold only at the triple junction $\Lambda = \vec{x}_i(0)$, $i = 1 \to 3$, while at the boundary intersection points $\vec{x}_i(1)$, $i = 1 \to 3$, the following conditions have to hold:

(1.17a) the curve endpoint remains attached to $\partial\Omega$,

(1.17b) $$\vec{\tau}_i\,.\,[\nabla F(\vec{x}_i(1))]^\perp = 0,$$

where $\cdot^\perp$ denotes the clockwise rotation by $\frac{\pi}{2}$. The second condition, (1.17b), requires that forces at the outer boundary act only normal to the boundary, which is equivalent to a 90° angle condition. This is the case when the two phases that meet the external boundary have the same contact energy with the boundary. If this is not the case, then condition (1.17b) has to be replaced by the angle condition

(1.18) $$\vec{\tau}_i \cdot \vec{t} = \vec{\tau}_i \cdot [\nabla F(\vec{x}_i(1))]^\perp = \cos\alpha_i,$$

where $\vec{t}$ is a unit tangent to $\partial\Omega$. Condition (1.18) states that the curve Γ_i intersects the outer boundary with a given angle α_i; see, e.g., [23] for the physical background.

Before coupling fourth and second order geometric evolution equations at triple junctions, we now turn our attention to a network evolving according to surface diffusion. Using the same notation as above, we rewrite the evolution laws

(1.19) $$\mathcal{V}_i = -\sigma_i\,\Delta_s\,\varkappa_i, \qquad i = 1 \to 3,$$

as

(1.20) $$(\vec{x}_i)_t\,.\,\vec{\nu}_i = -\sigma_i\,\Delta_s\,\varkappa_i, \qquad \varkappa_i\,\vec{\nu}_i = \Delta_s\,\vec{x}_i.$$

These equations are coupled at triple junctions and have to fulfill boundary conditions at points that intersect the external boundary. At the triple junction we require in addition to (1.14a,b) the conditions

$$\sigma_1 \varkappa_1 + \sigma_2 \varkappa_2 + \sigma_3 \varkappa_3 = 0, \tag{1.21a}$$

$$\sigma_1 \vec{\tau}_1 \,.\, \nabla_s \varkappa_1 = \sigma_2 \vec{\tau}_2 \,.\, \nabla_s \varkappa_2 = \sigma_3 \vec{\tau}_3 \,.\, \nabla_s \varkappa_3, \tag{1.21b}$$

where $\nabla_s |_{\Gamma_i} \equiv \vec{\tau}_i \frac{\partial}{\partial s}$ with s being the arclength. Equation (1.21a) follows from the continuity of chemical potentials, and (1.21b) is a flux balance condition; see [28] for details. At an intersection with an external boundary we require (1.17a,b) and the following no-flux condition at the external boundary:

$$\nabla_s \varkappa_i = \vec{0}\,. \tag{1.22}$$

Finally, another possible setup is to require motion by mean curvature only on a subset of the given curves, while the remaining curves move by surface diffusion. This is of relevance, e.g., in thermal grooving [36], in interface motion in polycrystalline two-phase materials [10], and in the evolution of boundaries in the electromigration of intergranular voids [7]. For parameterizations $\vec{x}_i : [0,1] \times [0,T] \to \mathbb{R}^d$ of Γ_i, $i = 1 \to K_C$, this gives rise to the following system of equations:

$$\begin{aligned} &(\vec{x}_i)_t \,.\, \vec{\nu}_i = \sigma_i \varkappa_i, \quad i = 1 \to i_0, \qquad (\vec{x}_i)_t \,.\, \vec{\nu}_i = -\sigma_i \Delta_s \varkappa_i, \quad i = i_0 + 1 \to K_C, \\ &\varkappa_i \vec{\nu}_i = \Delta_s \vec{x}_i, \quad i = 1 \to K_C, \end{aligned} \tag{1.23}$$

subject to the triple junction conditions (1.14a,b). In addition, we require that at triple junctions

$$\sigma_i \vec{\tau}_i \,.\, \nabla_s \varkappa_i = \sigma_j \vec{\tau}_j \,.\, \nabla_s \varkappa_j \tag{1.24a}$$

for all interfaces $i, j \in \{i_0 + 1, \dots, K_C\}$ which are present at the junction. Further, we require that

$$\sum_{i \in T_{SD}} \sigma_i \varkappa_i = 0, \tag{1.24b}$$

where $T_{SD} \subset \{i_0 + 1, \dots, K_C\}$ are the interfaces present at the triple junction which move according to surface diffusion. We note that for practical applications of (1.23) only triple junctions, where one curve that moves by mean curvature flow and two curves moving by surface diffusion meet, are of interest. A variational formulation of (1.23) with the appropriate conditions at triple junctions and along the fixed external boundary will also be considered in section 2.

As for previous work on the approximation of curve networks, we refer to [9, 43, 37]. Also relevant is [8], where (1.12) and (1.14a,b) are derived from an Allen–Cahn system. A parametric finite element approximation of the mean curvature flow of a curve that intersects an external boundary was considered in [15]. A level set approach for mean curvature flow of curve networks has been considered in [34, 44, 41]. A phase field model for the combined motion of mean curvature flow and surface diffusion, as well as SALK, was considered in [7]. A phase field model for a mean curvature flow system is considered in [27], and a surface diffusion flow system in [4].

This paper is organized as follows. In section 2 we formulate a finite element approximation of problem (1.13) and derive stability bounds. Here we first introduce our

approximation for the simpler case of a closed curve, (1.7), and then generalize that scheme to cover (1.8), as well as (1.20) and (1.14a,b) in the case of a triple junction configuration, as in Figure 1. We indicate how to generalize the approach to a configuration as in Figure 2, as well as to an arbitrary setup of curves, triple junctions, and external boundary intersections. We will also indicate how to combine this approach with the algorithm presented in [6] to yield a scheme for combined mean curvature flow and surface diffusion. Finally, we consider the adaptation of our scheme to approximate (1.10). In section 3 we present a large number of numerical computations and compare our results, where possible, with those from other parametric algorithms in the literature.

2. Finite element approximation.

2.1. Closed curves. We introduce the following finite element approximation. Let $J := \mathbb{R}/\mathbb{Z} = \bigcup_{j=1}^{N} J_j$, $N \geq 3$, be a decomposition of J into intervals given by the nodes q_j, $J_j = [q_{j-1}, q_j]$. Let $h_j = |J_j|$ and $h = \max_{j=1\to N} h_j$ be the maximal length of a grid element. Then the necessary finite element spaces are defined as follows:

$$\underline{V}_0^h := \{\vec{\chi} \in C(J,\mathbb{R}^d) : \vec{\chi}|_{J_j} \text{ is linear } \forall\ j = 1 \to N\} =: [W_0^h]^d \subset H^1(J,\mathbb{R}^d), \tag{2.1}$$

where $W_0^h \subset H^1(J,\mathbb{R})$ is the space of scalar continuous (periodic) piecewise linear functions, with $\{\phi_l\}_{l=1}^N$ denoting the standard basis of W_0^h. Throughout this paper, we make use of the periodicity of J, i.e., $q_N \equiv q_0$, $q_{N+1} \equiv q_1$, and so on.

In addition, let $0 = t_0 < t_1 < \cdots < t_{M-1} < t_M = T$ be a partitioning of $[0,T]$ into possibly variable time steps $\tau_m := t_{m+1} - t_m$, $m = 0 \to M-1$. We set $\tau := \max_{m=0\to M-1} \tau_m$. Let $\vec{X}^m \in \underline{V}_0^h$ be an approximation to $\vec{x}(\cdot,t_m)$, and similarly $\kappa^m \in W_0^h$ for $\varkappa(\cdot,t_m)$.

For scalar and vector functions $u, v \in L^2(J,\mathbb{R}^{(d)})$ we introduce the L^2 inner product $\langle\cdot,\cdot\rangle_m$ over the current polygonal curve Γ^m, which is described by the vector function $\vec{X}^m \in \underline{V}_0^h$, as follows:

$$\langle u, v\rangle_m := \int_{\Gamma^m} u \,.\, v \,\mathrm{d}s = \int_J u \,.\, v\, |\vec{X}_\rho^m| \,\mathrm{d}\rho\,,$$

where, here and throughout, $\vec{X}_\rho^m$ denotes the derivative of $\vec{X}^m$ with respect to ρ, where $\rho \in [0,1]$ is the parameterization variable, and $\cdot^{(*)}$ denotes an expression with or without the superscript $*$, and similarly for subscripts. In addition, if u, v are piecewise continuous, with possible jumps at the nodes $\{q_j\}_{j=1}^N$, we define the mass lumped inner product $\langle\cdot,\cdot\rangle_m^h$ as

$$\langle u, v\rangle_m^h := \tfrac{1}{2} \sum_{j=1}^{N} |\vec{X}^m(q_j) - \vec{X}^m(q_{j-1})| \left[(u \,.\, v)(q_j^-) + (u \,.\, v)(q_{j-1}^+)\right], \tag{2.2}$$

where we define $u(q_j^\pm) := \lim_{\varepsilon \searrow 0} u(q_j \pm \varepsilon)$. Furthermore, we note that $\nabla_s u \,.\, \nabla_s v = \frac{u_\rho \,.\, v_\rho}{|\vec{X}_\rho^m|^2}$ and $\vec{\nu}^m = -\frac{(\vec{X}_\rho^m)^\perp}{|\vec{X}_\rho^m|}$.

We propose the following approximation to (1.7): Find $\{\vec{X}^{m+1}, \kappa^{m+1}\} \in \underline{V}_0^h \times W_0^h$ such that

$$\left\langle \frac{\vec{X}^{m+1} - \vec{X}^m}{\tau_m}, \chi\, \vec{\nu}^m \right\rangle_m^h - \langle f(\kappa^{m+1}), \chi\rangle_m^h = 0 \qquad \forall\ \chi \in W_0^h, \tag{2.3a}$$

$$\langle \kappa^{m+1}\, \vec{\nu}^m, \vec{\eta}\rangle_m^h + \langle \nabla_s \vec{X}^{m+1}, \nabla_s \vec{\eta}\rangle_m = 0 \qquad \forall\ \vec{\eta} \in \underline{V}_0^h, \tag{2.3b}$$

where, as noted above, the inner products $\langle\cdot,\cdot\rangle_m^{(h)}$ as well as ∇_s depend on m.

In order to approximate (1.8), we adapt (2.3a) to

(2.4)
$$\left\langle \frac{\vec{X}^{m+1}-\vec{X}^m}{\tau_m}, \chi\,\vec{\nu}^m \right\rangle_m^h - \langle f(\kappa^{m+1}),\chi\rangle_m^h = -\frac{\langle f(\kappa^m),1\rangle_m^h}{\langle 1,1\rangle_m}\,\langle 1,\chi\rangle_m \qquad \forall\ \chi\in W_0^h,$$

where $\kappa^0\in W_0^h$ is suitably chosen; see section 3.

Remark 2.1. We note that in (2.3a,b), and throughout this paper, we use piecewise linear finite elements, e.g., as defined in (2.1). This is a sensible choice for the type of evolution equations in consideration, both from a theoretical and practical points of view. It is also in line with other approaches in the literature; see, e.g., [16, 17, 13, 15, 19, 3]. However, from a computational point of view, one could also choose to use higher order finite elements.

Before we can proceed to prove existence and uniqueness of (2.3a,b), we have to make the following very mild assumption.

($\mathcal{A}_0$) Let $|\vec{X}_\rho^m|>0$ for almost all $\rho\in J$. For $j=1\to N$, let $\vec{\nu}^m_{j-\frac12} := -\frac{(\vec{X}_\rho^m)^\perp}{|\vec{X}_\rho^m|}\,|_{J_j}$, and set

$$\vec{\omega}_j^m := \frac{|\vec{X}^m(q_j)-\vec{X}^m(q_{j-1})|\,\vec{\nu}^m_{j-\frac12}+|\vec{X}^m(q_{j+1})-\vec{X}^m(q_j)|\,\vec{\nu}^m_{j+\frac12}}{|\vec{X}^m(q_j)-\vec{X}^m(q_{j-1})|+|\vec{X}^m(q_{j+1})-\vec{X}^m(q_j)|}$$

$$= \frac{-[\vec{X}^m(q_{j+1})-\vec{X}^m(q_{j-1})]^\perp}{|\vec{X}^m(q_j)-\vec{X}^m(q_{j-1})|+|\vec{X}^m(q_{j+1})-\vec{X}^m(q_j)|}\,. \tag{2.5}$$

Then we further assume that $\dim \operatorname{span}\{\vec{\omega}_j^m\}_{j=1}^N = d = 2$.

Remark 2.2. We note that one can interpret $\vec{\omega}_j^m$ as a weighted normal defined at the node $\vec{X}^m(q_j)$ of the curve Γ^m, where in general $|\vec{\omega}_j^m|<1$. In addition, we note that ($\mathcal{A}_0$) is violated only in very rare occasions. For example, it always holds for curves without self intersections; see [6, Remark 2.2] for details.

THEOREM 2.3. [illegible] ($\mathcal{A}_0$) [illegible] $f:(a,b)\to\mathbb{R}$, [illegible] $-\infty\le a<0<b\le\infty$, [illegible] $f((a,b))=\mathbb{R}$ [illegible] $\{\vec{X}^{m+1},\kappa^{m+1}\}\in \underline{V}_0^h\times W_0^h$ [illegible] (2.3a,b) [illegible]. For later developments involving networks of curves, we first discuss the linear case when $f(r):=r$, and so existence follows from uniqueness. To investigate the latter, we consider the following system: Find $\{\vec{X},\kappa\}\in\underline{V}_0^h\times W_0^h$ such that

$$\langle\vec{X},\chi\,\vec{\nu}^m\rangle_m^h - \tau_m\,\langle\kappa,\chi\rangle_m^h = 0 \qquad \forall\ \chi\in W_0^h, \tag{2.6a}$$

$$\langle\kappa\,\vec{\nu}^m,\vec{\eta}\rangle_m^h + \langle\nabla_s\,\vec{X},\nabla_s\,\vec{\eta}\rangle_m = 0 \qquad \forall\ \vec{\eta}\in\underline{V}_0^h. \tag{2.6b}$$

Choosing $\chi=\kappa\in W_0^h$ in (2.6a) and $\vec{\eta}=\vec{X}\in\underline{V}_0^h$ in (2.6b) yields that

$$\langle\nabla_s\,\vec{X},\nabla_s\,\vec{X}\rangle_m + \tau_m\,\langle\kappa,\kappa\rangle_m^h = 0. \tag{2.7}$$

It follows from (2.7) that $\kappa\equiv 0$ and $\vec{X}\equiv\vec{X}^c\in\mathbb{R}^d$, and hence that

$$\langle\vec{X}^c,\chi\,\vec{\nu}^m\rangle_m^h = 0 \quad \forall\ \chi\in W_0^h. \tag{2.8}$$

Choosing $\chi=\phi_j$ in (2.8) yields that $\vec{X}^c\,.\,\vec{\omega}_j^m=0$ for all $j=1\to N$. It follows from assumption ($\mathcal{A}_0$) that $\vec{X}^c=\vec{0}$. Hence we have shown that (2.3a,b) with $f(r):=r$ has a unique solution $\{\vec{X}^{m+1},\kappa^{m+1}\}\in\underline{V}_0^h\times W_0^h$.

For a general function $f : (a,b) \to \mathbb{R}$ fulfilling the assumptions of the theorem, we can rewrite (2.3a,b), on noting (2.2) and (2.5), as: Find $\vec{X}^{m+1} \in \underline{V}_0^h$ such that

$$\langle \nabla_s \vec{X}^{m+1}, \nabla_s \vec{\eta} \rangle_m + \left\langle f^{-1}\left(\frac{\vec{X}^{m+1} - \vec{X}^m}{\tau_m} \,.\, \vec{\omega}^m \right), \vec{\eta} \,.\, \vec{\omega}^m \right\rangle_m^h = 0 \qquad \forall\ \vec{\eta} \in \underline{V}_0^h, \tag{2.9}$$

where $\vec{\omega}^m := \sum_{j=1}^N \vec{\omega}_j^m \phi_j$. Then $\kappa^{m+1} \in W_0^h$ is uniquely determined from

$$\kappa^{m+1}(q_j) = f^{-1}\left(\frac{\vec{X}^{m+1}(q_j) - \vec{X}^m(q_j)}{\tau_m} \,.\, \vec{\omega}_j^m \right), \qquad j = 1 \to N. \tag{2.10}$$

On noting our assumption $(\mathcal{A}_0)$, similarly as in the linear case above, it follows that (2.9) is the Euler–Lagrange variation of the strictly convex minimization problem

$$\min_{\vec{\eta} \in \underline{V}_0^h} \left[\tfrac{1}{2} \langle \nabla_s \vec{\eta}, \nabla_s \vec{\eta} \rangle_m + \tau_m \left\langle \Phi\left(\frac{\vec{\eta} - \vec{X}^m}{\tau_m} \,.\, \vec{\omega}^m \right), 1 \right\rangle_m^h \right],$$

where Φ is an antiderivative of f^{-1}. We note that $\Phi : \mathbb{R} \to \mathbb{R}$ is strictly convex with $\Phi'(f(0)) = f^{-1}(f(0)) = 0$, and hence we obtain that Φ is bounded from below and coercive. Therefore there exists a unique solution $\vec{X}^{m+1} \in \underline{V}_0^h$ to (2.9), and hence a unique solution $\{\vec{X}^{m+1}, \kappa^{m+1}\} \in \underline{V}_0^h \times W_0^h$ to (2.3a,b). ∎

The above proof immediately applies to the case when (2.3a) is replaced by (2.4). We remark also that we still obtain uniqueness for strictly increasing continuous functions $f : (a,b) \to \mathbb{R}$ with $-\infty \le a < b \le \infty$. This follows, since Φ defined as above is still strictly convex. Existence cannot be established as easily as above, because Φ is not coercive any longer. This discussion is relevant, e.g., for $f(r) = -r^{-1}$ with $r \in (0,\infty)$, which is the case of the inverse mean curvature flow if $\varkappa(\cdot,0) > 0$. In this case we obtain that $\Phi : (-\infty, 0) \to \mathbb{R}$ is defined as $\Phi(r) = \ln|r|$.

In addition, stability results for (2.3a,b) and the variants involving (2.4) can be established in certain cases; see Theorem 2.10, and the ensuing comment, below. For example, the term $\langle \kappa^{m+1}, \kappa^{m+1} \rangle_m^h$ in the analogue of (2.38) is replaced by $\langle f(\kappa^{m+1}), \kappa^{m+1} \rangle_m^h$, which once again is nonnegative if f is monotonically increasing with $f(0) = 0$. Of course, it may be computationally more convenient to consider a linearized version of (2.3a). For example, for (1.5) with $\beta \ge 1$ one could replace (2.3a) by

$$\left\langle \frac{\vec{X}^{m+1} - \vec{X}^m}{\tau_m}, \chi\, \vec{\nu}^m \right\rangle_m^h - \left\langle \frac{f(\kappa^m)}{\kappa^m}\, \kappa^{m+1}, \chi \right\rangle_m^h = 0 \qquad \forall\ \chi \in W_0^h,$$

with $\kappa^0 \in W_0^h$ suitably chosen. Once again, it is then straightforward to prove existence and uniqueness and to derive a stability result for this scheme.

In order to solve the (nonlinear) algebraic systems arising from (2.3a,b) and its generalizations, we apply a Schur complement approach. For later developments involving networks of curves, we describe it here for the linear case when $f(r) := r$. However, it easily caries over to nonlinear f; see (2.14a,b) below. Let $\vec{Id}_n \in (\mathbb{R}^{d\times d})^{n\times n}$ be the identity matrix, and similarly for $Id_n \in \mathbb{R}^{n\times n}$. We introduce also the matrices $\vec{N}_0 \in (\mathbb{R}^d)^{N\times N}$, $M_0 \in \mathbb{R}^{N\times N}$, and $\vec{A}_0 \in (\mathbb{R}^{d\times d})^{N\times N}$ with entries

$$[M_0]_{kl} := \langle \phi_k, \phi_l \rangle_m^h, \quad [\vec{N}_0]_{kl} := \int_{\Gamma^m} \pi^h[\phi_k\, \phi_l]\, \vec{\nu}^m \,\mathrm{d}s, \quad [\vec{A}_0]_{kl} := \langle \nabla_s \phi_k, \nabla_s \phi_l \rangle_m\, \vec{Id}_1\,, \tag{2.11}$$

where $\pi^h : C(J,\mathbb{R}) \to W_0^h$ is the standard interpolation operator at the nodes $\{q_j\}_{j=1}^N$. We can then formulate (2.3a,b) with $f(r) := r$ as: Find $\{\delta\vec{X}^{m+1}, \kappa^{m+1}\} \in (\mathbb{R}^d)^N \times \mathbb{R}^N$ such that

$$\begin{pmatrix} \tau_m M_0 & -\vec{N}_0^T \\ \vec{N}_0 & \vec{A}_0 \end{pmatrix} \begin{pmatrix} \kappa^{m+1} \\ \delta\vec{X}^{m+1} \end{pmatrix} = \begin{pmatrix} 0 \\ -\vec{A}_0\,\vec{X}^m \end{pmatrix}, \tag{2.12}$$

where, with an obvious abuse of notation, $\delta\vec{X}^{m+1} = (\delta\vec{X}_1^{m+1}, \dots, \delta\vec{X}_N^{m+1})^T$ and $\kappa^{m+1} = (\kappa_1^{m+1}, \dots, \kappa_N^{m+1})^T$ are the vectors of coefficients with respect to the standard basis for $\vec{X}^{m+1} - \vec{X}^m$ and κ^{m+1}, respectively. We can transform (2.12) into

$$\kappa^{m+1} = \frac{1}{\tau_m}\, M_0^{-1}\, \vec{N}_0^T\, \delta\vec{X}^{m+1}, \tag{2.13a}$$

$$\left(\vec{A}_0 + \frac{1}{\tau_m}\, \vec{N}_0\, M_0^{-1}\, \vec{N}_0^T\right) \delta\vec{X}^{m+1} = -\vec{A}_0\, \vec{X}^m. \tag{2.13b}$$

As (2.13b) is clearly symmetric and positive definite under our assumption $(\mathcal{A}_0)$, there exists a unique solution to (2.13b). Moreover, the solution to (2.13a,b) uniquely solves (2.3a,b) with $f(r) := r$.

For later purposes, we note that for the approximation (2.3a,b) with nonlinear f, the linear Schur system (2.13a,b) has to be replaced by the corresponding nonlinear system

$$\kappa^{m+1} = f^{-1}\left[\frac{1}{\tau_m}\, M_0^{-1}\, \vec{N}_0^T\, \delta\vec{X}^{m+1}\right], \tag{2.14a}$$

$$\vec{A}_0\, \delta\vec{X}^{m+1} + \vec{N}_0\, f^{-1}\left[\frac{1}{\tau_m}\, M_0^{-1}\, \vec{N}_0^T\, \delta\vec{X}^{m+1}\right] = -\vec{A}_0\, \vec{X}^m, \tag{2.14b}$$

where $f^{-1}(z) \in \mathbb{R}^N$ is defined by $[f^{-1}(z)]_i := f^{-1}(z_i)$, $i = 1 \to N$, for any $z \in \mathbb{R}^N$.

REMARK 2.4. In section 3, we will report on computations for our scheme (2.3a,b) and compare our results in the case of $f(r) := r$ with two other schemes in the literature. The first scheme is from [17] and can be formulated as: Find $\vec{X}^{m+1} \in \underline{V}_0^h$ such that

$$\left\langle \frac{\vec{X}^{m+1} - \vec{X}^m}{\tau_m}, \vec{\eta} \right\rangle_m^h + \langle \nabla_s\, \vec{X}^{m+1}, \nabla_s\, \vec{\eta} \rangle_m = 0 \qquad \forall\ \vec{\eta} \in \underline{V}_0^h. \tag{2.15}$$

The system (2.15) is a discretization of the variational formulation of

$$\vec{x}_t = \vec{\varkappa}, \qquad \vec{\varkappa} := \varkappa\, \vec{\nu} = \Delta_s\, \vec{x}, \tag{2.16}$$

as opposed to (1.7) with $f(r) := r$. From (2.10), we see that our scheme (2.3a,b) with $f(r) := r$ can be rewritten as: Find $\vec{X}^{m+1} \in \underline{V}_0^h$ such that

$$\left\langle \frac{\vec{X}^{m+1} - \vec{X}^m}{\tau_m} \cdot \vec{\omega}^m, \vec{\eta} \cdot \vec{\omega}^m \right\rangle_m^h + \langle \nabla_s\, \vec{X}^{m+1}, \nabla_s\, \vec{\eta} \rangle_m = 0 \qquad \forall\ \vec{\eta} \in \underline{V}_0^h\,, \tag{2.17}$$

which clearly highlights the key difference between the two schemes. The second scheme is from [13] and can be formulated as follows: Find $\vec{X}^{m+1} \in \underline{V}_0^h$ such that

$$\int_J |\vec{X}_\rho^m|^2\, \pi^h\left[\frac{\vec{X}^{m+1} - \vec{X}^m}{\tau_m} \cdot \vec{\eta}\right] \mathrm{d}\rho + \int_J \vec{X}_\rho^{m+1} \cdot \vec{\eta}_\rho\, \mathrm{d}\rho = 0 \qquad \forall\ \vec{\eta} \in \underline{V}_0^h\,, \tag{2.18}$$

or equivalently

$$\left\langle |\vec{X}^m_\rho| \, \frac{\vec{X}^{m+1}-\vec{X}^m}{\tau_m}, \vec{\eta} \right\rangle^h_m + \langle\, |\vec{X}^m_\rho| \, \nabla_s \vec{X}^{m+1}, \nabla_s \vec{\eta}\rangle_m = 0 \qquad \forall\ \vec{\eta} \in \underline{V}^h_0\,.$$

We note that the scheme (2.15) changes the approximation of $\vec{x}$ predominantly in the normal direction (recall (2.16)), whereas the scheme (2.3a,b) proposed in this paper, as well as (2.18), also induce tangential changes. This is a crucial difference. Without movement in the tangential direction, clustering and coalescence of nodes can occur, which leads to a breakdown of the algorithm. In addition, we note that the schemes (2.15) and (2.18) do not easily generalize to the case of nonlinear f, in contrast to our scheme (2.3a,b). However, it should be noted that the numerical analysis of the schemes (2.15) and (2.18) is well developed, in that error bounds in the semidiscrete case have been derived; see [17, 13]. This is something that we hope to achieve for our scheme in the future.

REMARK 2.5. Similarly to [6, Remark 2.3], one can consider a continuous-in-time semidiscrete version of our scheme (2.3a,b). In particular, we let

$$\langle \vec{X}_t, \chi\, \vec{\nu}^h\rangle^h - \langle f(\kappa), \chi\rangle^h = 0 \qquad \forall\, \chi \in W^h_0, \tag{2.19a}$$

$$\langle \kappa\, \vec{\nu}^h, \vec{\eta}\rangle^h + \langle \nabla_s \vec{X}, \nabla_s \vec{\eta}\rangle = 0 \qquad \forall\, \vec{\eta} \in \underline{V}^h_0, \tag{2.19b}$$

where we always integrate over the current curve Γ^h, described by $\vec{X}$, and so $\vec{\nu}^h = -\frac{(\vec{X}_\rho)^\perp}{|\vec{X}_\rho|}$ and $\langle\cdot,\cdot\rangle^{(h)}$ is the same as $\langle\cdot,\cdot\rangle^{(h)}_m$ with Γ^m and $\vec{X}^m$ replaced by Γ^h and $\vec{X}$, respectively. It is then possible to show that the scheme (2.19a,b) will always equidistribute the nodes along Γ^h if the corresponding intervals are not locally parallel; see [6, Remark 2.4] for details. Although it does not appear possible to prove an analogue for the fully discrete scheme (2.3a,b), in practice we see that the nodes are moved tangentially so that they will eventually be equidistributed; see section 3 for details.

While the scheme (2.18) from [13] also induces a tangential movement of vertices, it does not appear possible to show an analogous result for that scheme.

2.1.1. Intermediate evolution laws. In this subsection we consider the intermediate motion (1.10). We introduce the following approximation to (1.10): Find $\{\vec{X}^{m+1}, Y^{m+1}, \kappa^{m+1}\} \in \underline{V}^h_0 \times [W^h_0]^2$ such that

$$\left\langle \frac{\vec{X}^{m+1}-\vec{X}^m}{\tau_m}, \chi\, \vec{\nu}^m \right\rangle^h_m - \langle \nabla_s Y^{m+1}, \nabla_s \chi\rangle_m = 0 \qquad \forall\ \chi \in W^h_0, \tag{2.20a}$$

$$\frac{1}{\xi}\,\langle \nabla_s Y^{m+1}, \nabla_s \chi\rangle_m + \frac{1}{\alpha}\,\langle Y^{m+1}, \chi\rangle^h_m - \langle \kappa^{m+1}, \chi\rangle^h_m = 0 \qquad \forall\ \chi \in W^h_0, \tag{2.20b}$$

$$\langle \kappa^{m+1}\, \vec{\nu}^m, \vec{\eta}\rangle^h_m + \langle \nabla_s \vec{X}^{m+1}, \nabla_s \vec{\eta}\rangle_m = 0 \qquad \forall\ \vec{\eta} \in \underline{V}^h_0. \tag{2.20c}$$

REMARK 2.6. The scheme (2.20a–c) is close in concept to the approximation for (1.11) in [6]: Find $\{\vec{X}^{m+1}, \kappa^{m+1}\} \in \underline{V}^h_0 \times W^h_0$ such that

$$\left\langle \frac{\vec{X}^{m+1}-\vec{X}^m}{\tau_m}, \chi\, \vec{\nu}^m \right\rangle^h_m - \langle \nabla_s \kappa^{m+1}, \nabla_s \chi\rangle_m = 0 \qquad \forall\ \chi \in W^h_0, \tag{2.21a}$$

$$\langle \kappa^{m+1}\, \vec{\nu}^m, \vec{\eta}\rangle^h_m + \langle \nabla_s \vec{X}^{m+1}, \nabla_s \vec{\eta}\rangle_m = 0 \qquad \forall\ \vec{\eta} \in \underline{V}^h_0. \tag{2.21b}$$

A semidiscrete version of (2.21a,b) enjoys exact area conservation and an equidistribution of vertices; see [6, Remarks 2.3 and 2.4] for the relevant details. We note that the same holds true for the semidiscrete analogue of (2.20a–c).

THEOREM 2.7. [illegible] $(\mathcal{A}_0)$ [illegible] $\{\vec{X}^{m+1}, Y^{m+1}, \kappa^{m+1}\} \in \underline{V}_0^h \times [W_0^h]^2$ [illegible] (2.20a–c) [illegible]

$$|\Gamma^k| + \frac{1}{\alpha} \sum_{m=0}^{k-1} \tau_m \, |\nabla_s \, Y^{m+1}|_m^2 + \xi \sum_{m=0}^{k-1} \tau_m \, |\kappa^{m+1} - \tfrac{1}{\alpha} \, Y^{m+1}|_{m,h}^2 \leq |\Gamma^0| \tag{2.22}$$

[illegible] $k = 1 \to M$ [illegible] $|\cdot|_{m(,h)}^2 := \langle \cdot, \cdot \rangle_m^{(h)}$ [illegible]. The uniqueness proof is a straightforward adaption of the proof of Theorem 2.3. As (2.20a–c) is linear, existence follows from uniqueness, and the latter is easily established for the relevant equations

$$\begin{aligned} \langle \vec{X}, \chi \, \vec{\nu}^m \rangle_m^h - \tau_m \, \langle \nabla_s \, Y, \nabla_s \, \chi \rangle_m &= 0 \qquad \forall \; \chi \in W_0^h, \\ \frac{1}{\xi} \, \langle \nabla_s \, Y, \nabla_s \, \chi \rangle_m + \frac{1}{\alpha} \, \langle Y, \chi \rangle_m^h - \langle \kappa, \chi \rangle_m^h &= 0 \qquad \forall \; \chi \in W_0^h, \\ \langle \kappa \, \vec{\nu}^m, \vec{\eta} \rangle_m^h + \langle \nabla_s \, \vec{X}, \nabla_s \, \vec{\eta} \rangle_m &= 0 \qquad \forall \; \vec{\eta} \in \underline{V}_0^h, \end{aligned}$$

on choosing $\chi = \frac{\alpha}{\xi} \, \kappa$, $\chi = \tau_m \, (\alpha \, \kappa - Y)$, and $\vec{\eta} = \frac{\alpha}{\xi} \, \vec{X}$, respectively. Combining yields that

$$\frac{\alpha}{\xi} \, \langle \nabla_s \, \vec{X}, \nabla_s \, \vec{X} \rangle_m + \frac{\tau_m}{\xi} \, \langle \nabla_s \, Y, \nabla_s \, Y \rangle_m + \frac{\tau_m}{\alpha} \, \langle \alpha \, \kappa - Y, \alpha \, \kappa - Y \rangle_m^h = 0 \, . \tag{2.23}$$

It follows from (2.23) that $\vec{X} = \vec{X}^c \in \mathbb{R}^d$, $Y = Y^c \in \mathbb{R}$, and $\kappa = \frac{Y^c}{\alpha} \in \mathbb{R}$, and hence, similarly to (2.8) on recalling assumption $(\mathcal{A}_0)$, that $\vec{X}^c = \vec{0}$ and $Y^c = 0$. Hence we have existence of a unique solution $\{\vec{X}^{m+1}, Y^{m+1}, \kappa^{m+1}\} \in \underline{V}_0^h \times [W_0^h]^2$ to the system (2.20a–c). Finally, choosing $\chi = \frac{\alpha}{\xi} \, \kappa^{m+1}$, $\chi = \tau_m \, (\alpha \, \kappa^{m+1} - Y^{m+1})$, and $\vec{\eta} = \frac{\alpha}{\xi} \, (\vec{X}^{m+1} - \vec{X}^m)$ in (2.20a–c) gives, similarly to (2.23), that

$$\langle \nabla_s \, \vec{X}^{m+1}, \nabla_s \, (\vec{X}^{m+1} - \vec{X}^m) \rangle_m + \frac{\tau_m}{\alpha} \, |\nabla_s \, Y^{m+1}|_m^2 + \xi \, \tau_m \left| \kappa^{m+1} - \frac{1}{\alpha} \, Y^{m+1} \right|_{m,h}^2 = 0 \, . \tag{2.24}$$

Combining (2.24) with the closed curve analogue of (2.40), below, yields (2.22). ∎

On recalling the definitions (2.11) and on similarly introducing the matrix $A_0 \in \mathbb{R}^{N \times N}$, we can reformulate (2.20a–c) as: Find $\{\delta \vec{X}^{m+1}, Y^{m+1} \, \kappa^{m+1}\} \in (\mathbb{R}^d)^N \times [\mathbb{R}^N]^2$ such that

$$\begin{pmatrix} 0 & \tau_m \, A_0 & -\vec{N}_0^T \\ -M_0 & \frac{1}{\xi} \, A_0 + \frac{1}{\alpha} \, M_0 & 0 \\ \vec{N}_0 & 0 & \vec{A}_0 \end{pmatrix} \begin{pmatrix} \kappa^{m+1} \\ Y^{m+1} \\ \delta \vec{X}^{m+1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -\vec{A}_0 \, \vec{X}^m \end{pmatrix}, \tag{2.25}$$

where, with obvious abuse of notation, $\delta \vec{X}^{m+1}$, Y^{m+1}, and κ^{m+1} are the vectors of coefficients with respect to the standard basis of $\vec{X}^{m+1} - \vec{X}^m$, Y^{m+1}, and κ^{m+1}, respectively. Introducing the inverse S_0 of A_0 restricted on the set $(\ker A_0)^\perp \equiv (\mathrm{span}\{1\})^\perp$, where $1 := (1, \ldots, 1)^T \in \mathbb{R}^N$, and noting that the first equation in (2.25)

implies that $1^T \vec{N}_0^T \, \delta\vec{X}^{m+1} = 0$, one can transform (2.25) to

$$Y^{m+1} = \frac{1}{\tau_m} \, S_0 \, \vec{N}_0^T \, \delta\vec{X}^{m+1} + \mu \, 1, \tag{2.26a}$$

$$\kappa^{m+1} = \frac{1}{\tau_m} \left(\frac{1}{\alpha} \, S_0 + \frac{1}{\xi} \, M_0^{-1} \right) \vec{N}_0^T \, \delta\vec{X}^{m+1} + \frac{\mu}{\alpha} \, 1, \tag{2.26b}$$

$$\begin{aligned} &\left(\vec{A}_0 + \frac{1}{\tau_m} \, \vec{N}_0 \left[\frac{1}{\alpha} \, S_0 + \frac{1}{\xi} \, M_0^{-1} \right] \vec{N}_0^T \right) \delta\vec{X}^{m+1} = -\vec{A}_0 \, \vec{X}^m - \frac{\mu}{\alpha} \, \vec{N}_0 1, \\ &\qquad (\delta\vec{X}^{m+1})^T \, \vec{N}_0 1 = 0, \end{aligned} \tag{2.26c}$$

where $\mu = \frac{1^T Y^{m+1}}{1^T 1} \in \mathbb{R}$ is unknown. We introduce also the orthogonal projection $\vec{\Pi}_0$ onto $\mathcal{R}_0^\perp := \{\vec{X} \in (\mathbb{R}^d)^N : \vec{X}^T \vec{N}_0 1 = 0\}$ by $\vec{\Pi}_0 := \vec{Id}_N - \frac{\vec{w}\,\vec{w}^T}{\vec{w}^T \vec{w}}$, where $\vec{w} := \vec{N}_0 1$. Then (2.26c), on noting that $\vec{\Pi}_0 \, \delta\vec{X}^{m+1} = \delta\vec{X}^{m+1}$, is replaced by

$$\vec{\Pi}_0 \left(\vec{A}_0 + \frac{1}{\tau_m} \vec{N}_0 \left[\frac{1}{\alpha} \, S_0 + \frac{1}{\xi} \, M_0^{-1} \right] \vec{N}_0^T \right) \vec{\Pi}_0 \, \delta\vec{X}^{m+1} = -\vec{\Pi}_0 \, \vec{A}_0 \, \vec{X}^m. \tag{2.27}$$

As (2.20a–c) has a unique solution, it is easily established that there exists a unique solution to (2.27). Moreover, the system (2.27) is symmetric and positive definite on $\mathcal{R}_0^\perp$. For details of a similar situation involving surface diffusion and triple junctions, we refer to [6, Theorem 2.4].

2.2. Triple junctions. In this section, we consider the case where a network of curves meeting at triple junction points moves under motion by mean curvature. Here the curves can meet at triple junction points or can intersect the external boundary $\partial\Omega$. For ease of exposition, from now on we consider the two cases of three curves $(\Gamma_1, \Gamma_2, \Gamma_3)$ with surface energies $\sigma := (\sigma_1, \sigma_2, \sigma_3)$ meeting either at two triple junction points Λ_1 and Λ_2, as in Figure 1, or meeting at a single triple junction point Λ and each intersecting the external boundary $\partial\Omega$, as in Figure 2. In particular, we note the stated choices of the direction of the unit tangents. We will outline also how the ideas presented for these cases can be carried over to an arbitrary setup of curves; see Remark 2.13 below.

We begin with the first setup. The main idea for the necessary trial ($\equiv$ test) spaces is to make sure that the conditions (1.14a,b) hold either essentially or weakly at the triple junctions. Here we will enforce condition (1.14a) explicitly through the trial space, whereas condition (1.14b) will be enforced weakly, similarly to a Neumann boundary condition for a standard second order elliptic PDE.

Let $I := [0,1]$ be the unit interval, and let $I = \bigcup_{j=1}^{N_i} I_j^i$, $i = 1 \to 3$, be decompositions of I into intervals $I_j^i = [q_{j-1}^i, q_j^i]$ based on the nodes $\{q_j^i\}_{j=0}^{N_i}$, $N_i \geq 2$. Let $h_j^i = |I_j^i|$ and $h = \max_{i=1\to 3} \max_{j=1\to N_i} h_j^i$ be the maximal length of a grid element. Let $\underline{V} := \{(\vec{\chi}_1, \vec{\chi}_2, \vec{\chi}_3) \in [C(I, \mathbb{R}^d)]^3 : \vec{\chi}_1 = \vec{\chi}_2 = \vec{\chi}_3 \text{ on } \partial I\}$ and $W_{\mathcal{M}} := \{(\chi_1, \chi_2, \chi_3) \in [C(I, \mathbb{R})]^3\}$. The appropriate finite element spaces are then defined by

$$\underline{V}^h := \{(\vec{\chi}_1, \vec{\chi}_2, \vec{\chi}_3) \in \underline{V} : \vec{\chi}_i\,|_{I_j^i} \text{ is linear } \forall\ j = 1 \to N_i,\ i = 1 \to 3\} \subset \underline{V} \tag{2.28}$$

and similarly for the space of scalar functions $W_{\mathcal{M}}^h \subset W_{\mathcal{M}}$.

Recall the time partitioning $\{\tau_m\}_{m=0}^{M-1}$, and let $\vec{X}^m \in \underline{V}^h$ be an approximation to $\vec{x}(\cdot, t_m) \equiv (\vec{x}_1, \vec{x}_2, \vec{x}_3)(\cdot, t_m)$, and similarly $\kappa^m \in W_{\mathcal{M}}^h$ for $\varkappa(\cdot, t_m)$. We introduce the σ weighted L^2 inner product $\langle \cdot, \cdot \rangle_m$ and the mass lumped inner product $\langle \cdot, \cdot \rangle_m^h$ over

the current surface $\Gamma^m := (\Gamma_1^m, \Gamma_2^m, \Gamma_3^m)$, which is described by the vector function $\vec{X}^m \in \underline{V}^h$, for scalar and vector functions $u, v \in [L^2(I, \mathbb{R}^{(d)})]^3$ as follows:

$$\langle u, v \rangle_m := \int_{\Gamma^m} u \,.\, v \,\mathrm{d}s := \sum_{i=1}^{3} \sigma_i \int_I u_i \,.\, v_i \, |(\vec{X}_i^m)_\rho| \,\mathrm{d}\rho,$$

(2.29)

$$\langle u, v \rangle_m^h := \sum_{i=1}^{3} \frac{\sigma_i}{2} \sum_{j=1}^{N_i} |\vec{X}_i^m(q_j^i) - \vec{X}_i^m(q_{j-1}^i)| \left[(u_i \,.\, v_i)([q_j^i]^-) + (u_i \,.\, v_i)([q_{j-1}^i]^+) \right].$$

In addition, we note that

$$(\nabla_s u \,.\, \nabla_s v)\,|_{\Gamma_i^m} = \frac{(u_i)_\rho \,.\, (v_i)_\rho}{|(\vec{X}_i^m)_\rho|^2}, \qquad \vec{\nu}^m\,|_{\Gamma_i^m} = -\frac{(\vec{X}_i^m)_\rho^\perp}{|(\vec{X}_i^m)_\rho|}, \qquad i = 1 \to 3.$$

We then propose the following approximation to (1.13) with (1.14a,b): Find $\{\vec{X}^{m+1}, \kappa^{m+1}\} \in \underline{V}^h \times W_{\mathcal{M}}^h$ such that

$$\left\langle \frac{\vec{X}^{m+1} - \vec{X}^m}{\tau_m}, \chi\, \vec{\nu}^m \right\rangle_m^h - \langle \sigma\, \kappa^{m+1}, \chi \rangle_m^h = 0 \qquad \forall\ \chi \in W_{\mathcal{M}}^h, \tag{2.30a}$$

$$\langle \kappa^{m+1}\, \vec{\nu}^m, \vec{\eta} \rangle_m^h + \langle \nabla_s \vec{X}^{m+1}, \nabla_s \vec{\eta} \rangle_m = 0 \qquad \forall\ \vec{\eta} \in \underline{V}^h. \tag{2.30b}$$

Observe that (2.30a,b) was derived from (1.13) using integration by parts and the definition of the space $\underline{V}^h$. On noting that $(\nabla_s \vec{X}^{m+1})\,|_{\Gamma_i^m}$ approximates $\vec{\tau}_i\, \vec{\tau}_i^T$, $i = 1 \to 3$, we see that (2.30b) weakly approximates Young's law (1.14b) at the triple junction points Λ_1 and Λ_2.

An extension of the scheme (2.18) that incorporates intersections with an external boundary was given in [15]. The authors considered a setup where a single curve intersects a fixed external boundary, and, for their fully discrete scheme, introduced the following trial and test space: For a given $\vec{X}^m \in C(I, \mathbb{R}^d)$,

$$\widetilde{\underline{V}}(\vec{X}^m) := \{\vec{\chi} \in C(I, \mathbb{R}^d) : \vec{\chi} \,.\, \nabla F(\vec{X}^m) = 0 \ \text{ on } \partial I\}, \tag{2.31}$$

and in addition $\widetilde{\underline{V}}^h(\vec{X}^m) := \{\vec{\chi} \in \widetilde{\underline{V}}(\vec{X}^m) : \vec{\chi}\,|_{I_j^1} \text{ is linear } \forall\ j = 1 \to N_1,\} \subset \widetilde{\underline{V}}(\vec{X}^m)$. Then their fully discrete approximation can be formulated as: Find $\delta\vec{X}^{m+1} \in \widetilde{\underline{V}}^h(\vec{X}^m)$, where $\vec{X}^{m+1} := \vec{X}^m + \delta\vec{X}^{m+1}$ such that

$$\int_I |\vec{X}_\rho^m|^2\, \pi^h \left[\frac{\delta\vec{X}^{m+1}}{\tau_m} \,.\, \vec{\eta} \right] \mathrm{d}\rho + \int_I \vec{X}_\rho^{m+1} \,.\, \vec{\eta}_\rho \,\mathrm{d}\rho = 0 \qquad \forall\ \vec{\eta} \in \widetilde{\underline{V}}^h(\vec{X}^m). \tag{2.32}$$

We now adapt the definition (2.31) to the setup as depicted in Figure 2. For a given $\vec{X}^m \in [C(I, \mathbb{R}^d)]^3$, let

$$\underline{V}_\partial(\vec{X}^m) := \{(\vec{\chi}_1, \vec{\chi}_2, \vec{\chi}_3) \in [C(I, \mathbb{R}^d)]^3 : \vec{\chi}_1(0) = \vec{\chi}_2(0) = \vec{\chi}_3(0) \quad \text{and}$$
$$\vec{\chi}_i(1) \,.\, \nabla F(\vec{X}_i^m(1)) = 0 \quad \forall\ i = 1 \to 3\}.$$

The finite element space $\underline{V}_\partial^h(\vec{X}^m)$ is then defined accordingly, similarly to (2.28). Moreover, the system (2.30a,b) is then adapted to: Find $\{\delta\vec{X}^{m+1}, \kappa^{m+1}\} \in \underline{V}_\partial^h(\vec{X}^m) \times$

$W^h_{\mathcal{M}}$, where $\vec{X}^{m+1} := \vec{X}^m + \delta\vec{X}^{m+1}$, such that

$$\left\langle \frac{\delta\vec{X}^{m+1}}{\tau_m}, \chi\,\vec{\nu}^m \right\rangle^h_m - \langle \sigma\,\kappa^{m+1}, \chi\rangle^h_m = 0 \qquad \forall\ \chi \in W^h_{\mathcal{M}}, \tag{2.33a}$$

$$\langle \kappa^{m+1}\,\vec{\nu}^m, \vec{\eta}\rangle^h_m + \langle \nabla_s\,\vec{X}^{m+1}, \nabla_s\,\vec{\eta}\rangle_m = 0 \qquad \forall\ \vec{\eta} \in \underline{V}^h_\partial(\vec{X}^m). \tag{2.33b}$$

Once again (2.33a,b) was derived from (1.13) using integration by parts and the definition of the space $\underline{V}^h_\partial(\vec{X}^m)$. On recalling that $(\nabla_s\,\vec{X}^{m+1})|_{\Gamma^m_i}$ approximates $\vec{\tau}_i\,\vec{\tau}_i^{\,T}$, $i = 1 \to 3$, we see that (2.33b) weakly approximates Young's law (1.14b) at the triple junction point Λ and (1.17b) at the boundary intersections. In order to approximate the general contact angle condition (1.18), we need to replace (2.33b) by

$$\begin{aligned} &\langle \kappa^{m+1}\,\vec{\nu}^m, \vec{\eta}\rangle^h_m + \langle \nabla_s\,\vec{X}^{m+1}, \nabla_s\,\vec{\eta}\rangle_m \\ &\quad = \sum_{i=1}^3 \sigma_i\,\frac{[\nabla F(\vec{X}^m_i(1))]^\perp}{|\nabla F(\vec{X}^m_i(1))|}\,.\,\vec{\eta}_i(1)\,\cos\alpha_i \qquad \forall\ \vec{\eta} \in \underline{V}^h_\partial(\vec{X}^m). \end{aligned} \tag{2.34}$$

Furthermore, the constraint $\delta\vec{X}^{m+1} \in \underline{V}^h_\partial(\vec{X}^m)$ weakly enforces (1.17a), as it is a linearized approximation of these constraints. In particular, for curved boundaries the equations $F(\vec{X}^{m+1}_i(1)) = 0$, $i = 1 \to 3$, are only approximately satisfied. Similarly to [15, p. 651] it formally follows, on assuming that $F(\vec{X}^0_i(1)) = 0$ for $i = 1 \to 3$, that

$$|F(\vec{X}^{m+1}_i(1))| \leq \sum_{k=0}^m |F(\vec{X}^{k+1}_i(1)) - F(\vec{X}^k_i(1))| = \mathcal{O}(\tau), \qquad i = 1 \to 3.$$

Hence, for small time steps the endpoints of the curve segments should stay close to $\partial\Omega$, and this is what one observes in practice; see section 3 for details. However, one could also employ a projection step that orthogonally projects $\vec{X}^{m+1}$ onto $\partial\Omega$ at every time step, which would have the advantage of satisfying (1.17a) exactly throughout the evolution. But this complicates the stability proof below; hence our preference for the approximation (2.33a,b). However, we do include numerical results using both approaches in section 3.

We remark that as the parameterization $\vec{X}^{m+1}$ does not "see" the boundary for $\rho < 1$, it is possible that the evolving curves touch the external boundary at some interior point, in which case the evolution is no longer described by the approximation. We note that for a convex domain this cannot happen for mean curvature flow; see [39]. However, in the surface diffusion case this can happen even for convex domains.

2.8. We note that, as stated in [15, p. 640], it is not clear how to naturally generalize the scheme (2.32) to triple junctions, as there are severe difficulties with how to approximate the condition (1.14b) within that scheme.

Before we can proceed to proving existence and uniqueness for (2.30a,b), we have to make the following very mild assumption, the analogue of $(\mathcal{A}_0)$ for a network of curves.

$(\mathcal{A})$ Let $|(\vec{X}^m_i)_\rho| > 0$ for almost all $\rho \in I$, $i = 1 \to 3$. Let

$$\vec{\nu}^m_{i,j-\frac12} := -\frac{(\vec{X}^m_i)^\perp_\rho}{|(\vec{X}^m_i)_\rho|}\,|_{I^i_j}, j = 1 \to N_i,$$

and set

$$\vec{\omega}^m_{i,j} := \frac{|\vec{X}^m_i(q^i_j) - \vec{X}^m_i(q^i_{j-1})|\,\vec{\nu}^m_{i,j-\frac{1}{2}} + |\vec{X}^m_i(q^i_{j+1}) - \vec{X}^m_i(q^i_j)|\,\vec{\nu}^m_{i,j+\frac{1}{2}}}{|\vec{X}^m_i(q^i_j) - \vec{X}^m_i(q^i_{j-1})| + |\vec{X}^m_i(q^i_{j+1}) - \vec{X}^m_i(q^i_j)|},$$
$$j = 1 \to N_i - 1, i = 1 \to 3.$$

Then we assume further that $\dim \operatorname{span}\{\{\vec{\omega}^m_{i,j}\}_{j=1}^{N_i-1}\}_{i=1}^3 = d = 2$.

The assumption $(\mathcal{A})$ basically assures that none of the curves Γ^m_i, $i = 1 \to 3$, is a "zig zagging" connection between the two triple junctions points Λ_1 and Λ_2. A sufficient condition for $(\mathcal{A})$ to hold is that at least one of the three curves is not a "saw tooth"-like curve, similar to the one in [6, Figure 3], where all the vertex normals $\vec{\omega}^m_{i,j}$, $j = 1 \to N_i - 1$, are linearly dependent.

THEOREM 2.9. $(\mathcal{A})$ $\{\vec{X}^{m+1}, \kappa^{m+1}\} \in \underline{V}^h \times W^h_{\mathcal{M}}$ (2.30a,b)

. As (2.30a,b) is linear, existence follows from uniqueness. To investigate the latter, we consider the system: Find $\{\vec{X}, \kappa\} \in \underline{V}^h \times W^h_{\mathcal{M}}$ such that

$$\langle \vec{X}, \chi\,\vec{\nu}^m\rangle^h_m - \tau_m\,\langle \sigma\,\kappa, \chi\rangle^h_m = 0 \qquad \forall\ \chi \in W^h_{\mathcal{M}}, \tag{2.35a}$$

$$\langle \kappa\,\vec{\nu}^m, \vec{\eta}\rangle^h_m + \langle \nabla_s\,\vec{X}, \nabla_s\,\vec{\eta}\rangle_m = 0 \qquad \forall\ \vec{\eta} \in \underline{V}^h. \tag{2.35b}$$

Similarly to (2.6a,b), choosing $\chi = \kappa \in W^h_{\mathcal{M}}$ in (2.35a) and $\vec{\eta} = \vec{X} \in \underline{V}^h$ in (2.35b) yields that

$$\langle \nabla_s\,\vec{X}, \nabla_s\,\vec{X}\rangle_m + \tau_m\,\langle \sigma\,\kappa, \kappa\rangle^h_m = 0\,. \tag{2.36}$$

It follows from (2.36) that $\kappa \equiv 0$ and $\vec{X} \equiv \vec{X}^c = (\vec{X}^c_1, \vec{X}^c_2, \vec{X}^c_3)^T \in (\mathbb{R}^d)^3$ with $\vec{X}^c_1 = \vec{X}^c_2 = \vec{X}^c_3$, and hence

$$\langle \vec{X}^c, \chi\,\vec{\nu}^m\rangle^h_m = 0 \quad \forall\ \chi \in W^h_{\mathcal{M}}. \tag{2.37}$$

Similarly to (2.8), choosing $\chi = \varphi^i_j$ in (2.37) and noting that $\vec{X}^c_1 = \vec{X}^c_2 = \vec{X}^c_3$ yields that $\vec{X}^c_1 \cdot \vec{\omega}^m_{i,j} = 0$ for all $j = 1 \to N_i - 1$, $i = 1 \to 3$. Assumption $(\mathcal{A})$ then yields that $\vec{X}^c_1 = \vec{0}$, and hence $\vec{X}^c = \vec{0}$. Hence we have shown that (2.30a,b) has a unique solution $\{\vec{X}^{m+1}, \kappa^{m+1}\} \in \underline{V}^h \times W^h_{\mathcal{M}}$. ∎

The proof immediately carries over to the system (2.33a,b).

Furthermore, we can establish that our scheme is unconditionally stable.

THEOREM 2.10. $\{\vec{X}^m, \kappa^m\}_{m=1}^M$ (2.30a,b) $k = 1 \to M$

$$|\Gamma^k| + \sum_{m=0}^{k-1} \tau_m\,\langle \sigma\,\kappa^{m+1}, \kappa^{m+1}\rangle^h_m \le |\Gamma^0|\,, \tag{2.38}$$

$|\Gamma^k| := \int_{\Gamma^k} 1\,\mathrm{d}s \equiv \sum_{i=1}^3 \sigma_i\,|\Gamma^k_i|$ (2.29)

. Choosing $\chi = \kappa^{m+1} \in W^h_{\mathcal{M}}$ in (2.30a) and $\vec{\eta} = \frac{\vec{X}^{m+1} - \vec{X}^m}{\tau_m} \in \underline{V}^h$ in (2.30b) yields that

$$\langle \nabla_s\,\vec{X}^{m+1}, \nabla_s\,(\vec{X}^{m+1} - \vec{X}^m)\rangle_m + \tau_m\,\langle \sigma\,\kappa^{m+1}, \kappa^{m+1}\rangle^h_m = 0\,. \tag{2.39}$$

We now analyze the first term in (2.39), using the ideas in [18]. Let $\vec{h}_j^{\,i,m} := \vec{X}_i^m(q_{j+1}^i) - \vec{X}_i^m(q_j^i)$. Then it holds that

$$\begin{aligned}\langle \nabla_s \vec{X}^{m+1}, \nabla_s (\vec{X}^{m+1} - \vec{X}^m)\rangle_m &= \sum_{i=1}^{3} \sigma_i \int_{\Gamma_i^m} [\nabla_s \vec{X}^{m+1}] \,.\, [\nabla_s (\vec{X}^{m+1} - \vec{X}^m)] \,\mathrm{d}s \\ &= \sum_{i=1}^{3} \sigma_i \sum_{j=1}^{N_i-1} \left[\frac{|\vec{h}_j^{\,i,m+1}|^2 - \vec{h}_j^{\,i,m+1} \,.\, \vec{h}_j^{\,i,m}}{|\vec{h}_j^{\,i,m}|} \right] \\ &\geq \sum_{i=1}^{3} \sigma_i \sum_{j=1}^{N_i-1} \left[|\vec{h}_j^{\,i,m+1}| - |\vec{h}_j^{\,i,m}| \right] = |\Gamma^{m+1}| - |\Gamma^m| \,. \end{aligned} \tag{2.40}$$

Combining (2.39) and (2.40) yields that

$$|\Gamma^{m+1}| - |\Gamma^m| + \tau_m \int_{\Gamma^m} \sigma \, |\kappa^{m+1}|^2 \,\mathrm{d}s \leq 0. \tag{2.41}$$

Summing (2.41) for $m = 0 \to k-1$ yields the desired result. ∎

The proof above is written explicitly for (2.30a,b), but as it depends solely on a specific choice of test functions, it immediately carries over to (2.33a,b), as well as (2.3a,b) with the changes, in the case of nonlinear f, stated after Theorem 2.3.

Let $N := \sum_{i=1}^{3}(N_i + 1)$. We define the orthogonal projection $\vec{P} : (\mathbb{R}^d)^N \to \underline{\mathbb{X}} := \{(\vec{z}_1, \vec{z}_2, \vec{z}_3) \in (\mathbb{R}^d)^N : [\vec{z}_1]_0 = [\vec{z}_2]_0 = [\vec{z}_3]_0, [\vec{z}_1]_{N_1} = [\vec{z}_2]_{N_2} = [\vec{z}_3]_{N_3}\}$ onto the Euclidean space associated with $\underline{V}^h$.

In order to give a matrix formulation for (2.30a,b) we introduce the matrices $M^i \in \mathbb{R}^{(N_i+1)\times(N_i+1)}$, $\vec{N}^i \in (\mathbb{R}^d)^{(N_i+1)\times(N_i+1)}$, $A^i \in \mathbb{R}^{(N_i+1)\times(N_i+1)}$, and $\vec{A}^i \in (\mathbb{R}^{d\times d})^{(N_i+1)\times(N_i+1)}$, $i = 1 \to 3$, defined by

$$\begin{aligned} M_{kl}^i &:= \sigma_i \int_{\Gamma_i^m} \pi_i^h[\phi_k^i \, \phi_l^i] \,\mathrm{d}s, \qquad \vec{N}_{kl}^i := \sigma_i \int_{\Gamma_i^m} \pi_i^h[\phi_k^i \, \phi_l^i] \, \vec{\nu}^m \,\mathrm{d}s, \\ A_{kl}^i &:= \sigma_i \int_{\Gamma_i^m} \nabla_s \phi_k^i \,.\, \nabla_s \phi_l^i \,\mathrm{d}s, \qquad \vec{A}_{kl}^i := A_{kl}^i \, \vec{Id}_1 \,, \end{aligned} \tag{2.42}$$

where $\{\phi_l^i\}_{l=0}^{N_i}$ is the standard basis of $S_i^h := \{\chi \in C(I,\mathbb{R}) : \chi\,|_{I_j^i} \text{ is linear } \forall\ j = 1 \to N_i\}$ and $\pi_i^h : C(I,\mathbb{R}) \to S_i^h$ is the standard interpolation operator at the nodes $\{q_j^i\}_{j=0}^{N_i}$. Then on introducing the matrices

$$M := \begin{pmatrix} \sigma_1 M^1 & 0 & 0 \\ 0 & \sigma_2 M^2 & 0 \\ 0 & 0 & \sigma_3 M^3 \end{pmatrix}, \ \vec{A} := \begin{pmatrix} \vec{A}^1 & 0 & 0 \\ 0 & \vec{A}^2 & 0 \\ 0 & 0 & \vec{A}^3 \end{pmatrix}, \ \vec{N} := \begin{pmatrix} \vec{N}^1 & 0 & 0 \\ 0 & \vec{N}^2 & 0 \\ 0 & 0 & \vec{N}^3 \end{pmatrix}, \tag{2.43}$$

where $M : \mathbb{R}^N \to \mathbb{R}^N$, $\vec{A} : (\mathbb{R}^d)^N \to (\mathbb{R}^d)^N$, and $\vec{N} : \mathbb{R}^N \to (\mathbb{R}^d)^N$, the system of equations (2.30a,b) can be written as: Find $\{\delta\vec{X}^{m+1}, \kappa^{m+1}\} \in \underline{\mathbb{X}} \times \mathbb{R}^N$ such that

$$\begin{pmatrix} \tau_m M & -\vec{N}^T \vec{P} \\ \vec{P}\vec{N} & \vec{P}\vec{A}\vec{P} \end{pmatrix} \begin{pmatrix} \kappa^{m+1} \\ \delta\vec{X}^{m+1} \end{pmatrix} = \begin{pmatrix} 0 \\ -\vec{P}\vec{A}\vec{P}\,\vec{X}^m \end{pmatrix}. \tag{2.44}$$

Here, with an obvious abuse of notation as in (2.12), $\kappa^{m+1} = (\kappa_1^{m+1}, \kappa_2^{m+1}, \kappa_3^{m+1})^T$ with $\kappa_i^{m+1} = ([\kappa_i^{m+1}]_0, \ldots, [\kappa_i^{m+1}]_{N_i})$, $i = 1 \to 3$, and $\delta\vec{X}^{m+1} = (\delta\vec{X}_1^{m+1}, \delta\vec{X}_2^{m+1},$

$\delta\vec{X}_3^{m+1})^T$ with $\delta\vec{X}_i^{m+1} = ([\delta\vec{X}_i^{m+1}]_0, \ldots, [\delta\vec{X}_i^{m+1}]_{N_i})$, $i = 1 \to 3$, are the vectors of coefficients with respect to the standard basis $\{\{\phi_l^i\}_{l=0}^{N_i}\}_{i=1}^3$ of κ^{m+1} and $\vec{X}^{m+1} - \vec{X}^m$ in (2.30a,b), respectively.

As M is nonsingular, we can reformulate (2.44), similarly to (2.13a,b), as

$$\kappa^{m+1} = \frac{1}{\tau_m}\, M^{-1}\, \vec{N}^T \vec{P}\, \delta\vec{X}^{m+1}, \tag{2.45a}$$

$$\left(\vec{P}\vec{A}\vec{P} + \frac{1}{\tau_m}\, \vec{P}\vec{N}\, M^{-1}\, \vec{N}^T \vec{P}\right) \delta\vec{X}^{m+1} = -\vec{P}\vec{A}\vec{P}\, \vec{X}^m. \tag{2.45b}$$

THEOREM 2.11. [illegible] $\{\delta\vec{X}^{m+1}, \kappa^{m+1}\} \in \underline{\mathbb{X}} \times \mathbb{R}^N$ [illegible] (2.44) [illegible] $\delta\vec{X}^{m+1}$ [illegible] (2.45b) [illegible] (2.45b) [illegible]. The proof is straightforward. □

The system (2.45a,b) is easily adapted to cover the approximation (2.33a,b) of the setup displayed in Figure 2. In particular, we need to introduce the orthogonal projection $\vec{P}_\partial : (\mathbb{R}^d)^N \to \underline{\mathbb{X}}_\partial := \{(\vec{z}_1, \vec{z}_2, \vec{z}_3) \in (\mathbb{R}^d)^N : [\vec{z}_1]_0 = [\vec{z}_2]_0 = [\vec{z}_3]_0$ and $[\vec{z}_i]_{N_i} \,.\, \nabla F([\vec{X}_i^m]_{N_i}) = 0 \;\forall i = 1 \to 3\}$ onto the Euclidean spaces associated with $\underline{V}_\partial^h(\vec{X}^m)$. Then (2.45b) is readily replaced by

$$\left(\vec{P}_\partial\vec{A}\vec{P}_\partial + \frac{1}{\tau_m}\, \vec{P}_\partial\vec{N}\, M^{-1}\, \vec{N}^T\vec{P}_\partial\right) \delta\vec{X}^{m+1} = -\vec{P}_\partial\vec{A}\, \vec{X}^m. \tag{2.46}$$

We note that (2.13b) and (2.46) can easily be generalized to curve networks with an arbitrary number of triple junctions and external boundary intersections; see Remark 2.13 below for the relevant details.

2.2.1. Surface diffusion of curve networks. We recall the following approximation to (1.19), with (1.14a,b) and (1.21a,b), as proposed in [6]: Find $\{\vec{X}^{m+1}, \kappa^{m+1}\} \in \underline{V}^h \times W^h$ such that

$$\left\langle \frac{\vec{X}^{m+1} - \vec{X}^m}{\tau_m}, \chi\, \vec{\nu}^m \right\rangle_m^h - \langle \sigma\, \nabla_s\, \kappa^{m+1}, \nabla_s\, \chi\rangle_m = 0 \qquad \forall\ \chi \in W^h, \tag{2.47a}$$

$$\langle \kappa^{m+1}\, \vec{\nu}^m, \vec{\eta}\rangle_m^h + \langle \nabla_s\, \vec{X}^{m+1}, \nabla_s\, \vec{\eta}\rangle_m = 0 \qquad \forall\ \vec{\eta} \in \underline{V}^h, \tag{2.47b}$$

where

$$W^h := \{(\chi_1, \chi_2, \chi_3) \in W : \chi_i|_{I_j^i} \text{ is linear } \forall\ j = 1 \to N_i,\ i = 1 \to 3\} \subset W \tag{2.48}$$

and $W := \{(\chi_1, \chi_2, \chi_3) \in [C(I, \mathbb{R})]^3 : \sigma_1\, \chi_1 + \sigma_2\, \chi_2 + \sigma_3\, \chi_3 = 0 \text{ on } \partial I\}$. We recall further that the system (2.47a,b) can be solved by applying a Schur complement approach and then solving for $\delta\vec{X}^{m+1} \in \underline{\mathbb{X}}$:

$$\vec{\Pi}\, \vec{P}\, \left(\vec{A} + \frac{1}{\tau_m}\, \vec{N} K\, S\, K \vec{N}^T\right)\, \vec{P}\, \vec{\Pi}\, \delta\vec{X}^{m+1} = -\vec{\Pi}\, \vec{P}\vec{A}\vec{P}\, \vec{X}^m. \tag{2.49}$$

Here $K : \mathbb{R}^N \to \mathbb{X} := \{(z_1, z_2, z_3) \in \mathbb{R}^N : \sum_{i=1}^3 \sigma_i\, [z_i]_0 = \sum_{i=1}^3 \sigma_i\, [z_i]_{N_i} = 0\}$ is the orthogonal projection onto $\mathbb{X}$, and S is the inverse of KAK, where A is defined similarly to (2.43) using A^i in (2.42), on the space $(\ker KAK)^\perp$. Also $\vec{\Pi} : (\mathbb{R}^d)^N \to \mathcal{R}^\perp$ is the orthogonal projection onto $\mathcal{R}^\perp$, where $\mathcal{R} := \text{span}\,\{\vec{P}\vec{N}K\, e_i : i = 1 \to 2\} \equiv$

$\{\vec{P}\vec{N}K\,v : v \in \ker KAK\} \subset \underline{\mathbb{X}}$ with $\{e_i\}_{i=1}^2$, $e_1 := (\frac{1}{\sigma_1}1^1, -\frac{1}{\sigma_2}1^2, 0) \in \mathbb{X}$ and $e_2 := (0, \frac{1}{\sigma_2}1^2, -\frac{1}{\sigma_3}1^3) \in \mathbb{X}$, where $1^i := (1,\dots,1)^T \in \mathbb{R}^{N_i+1}$, $i = 1 \to 3$, being a basis of the space $E = \ker A \cap \mathbb{X}$.

We now want to adapt the above scheme to include possible boundary intersections. Naturally in the case of (1.17b), (2.47a,b) is changed to: Find $\{\delta\vec{X}^{m+1}, \kappa^{m+1}\} \in \underline{V}_\partial^h(\vec{X}^m) \times W_\partial^h$ such that

$$\left\langle \frac{\delta\vec{X}^{m+1}}{\tau_m}, \chi\,\vec{\nu}^m \right\rangle_m^h - \langle \sigma\,\nabla_s\,\kappa^{m+1}, \nabla_s\,\chi\rangle_m = 0 \qquad \forall\ \chi \in W_\partial^h, \tag{2.50a}$$

$$\langle \kappa^{m+1}\,\vec{\nu}^m, \vec{\eta}\rangle_m^h + \langle \nabla_s\,\vec{X}^{m+1}, \nabla_s\,\vec{\eta}\rangle_m = 0 \qquad \forall\ \vec{\eta} \in \underline{V}_\partial^h(\vec{X}^m), \tag{2.50b}$$

where $W_\partial := \{(\chi_1,\chi_2,\chi_3) \in [C(I,\mathbb{R})]^3 : \sigma_1\,\chi_1(0)+\sigma_2\,\chi_2(0)+\sigma_3\,\chi_3(0) = 0\}$ and W_∂^h is defined similarly to (2.48). On defining the orthogonal projection $K_\partial : \mathbb{R}^N \to \mathbb{X}_\partial := \{(z_1,z_2,z_3) \in \mathbb{R}^N : \sum_{i=1}^3 \sigma_i\,[z_i]_0 = 0\}$ and using the projection $\vec{P}_\partial$ defined earlier, we can apply a Schur complement approach to yield

$$\vec{\Pi}_\partial\,\vec{P}_\partial\left(\vec{A} + \frac{1}{\tau_m}\,\vec{N}K_\partial\,S_\partial\,K_\partial\vec{N}^T\right)\vec{P}_\partial\,\vec{\Pi}_\partial\,\delta\vec{X}^{m+1} = -\vec{\Pi}_\partial\,\vec{P}_\partial\vec{A}\,\vec{X}^m, \tag{2.51}$$

where S_∂ is the inverse of $K_\partial AK_\partial$ on the space $(\ker K_\partial AK_\partial)^\perp$ and $\vec{\Pi}_\partial$ is the orthogonal projection from $\mathbb{R}^N$ onto $\mathcal{R}_\partial^\perp$ with $\mathcal{R}_\partial := \vec{P}_\partial\vec{N}K_\partial(\ker K_\partial AK_\partial)$. In fact, for the setup in Figure 2, $\vec{\Pi}_\partial \equiv \vec{\Pi}$. However, in general this is not always the case; see Remark 2.14 below. Of course, the above is extended to (1.18) by replacing (2.50b) by (2.34).

2.12. A possible definition for the projection $\vec{\Pi}$ is $\vec{\Pi} := \vec{Id}_N - \vec{Q}\,\vec{Q}^T$, where $\operatorname{im}\vec{Q} = \mathcal{R}$ and $\vec{Q}^T\vec{Q} = Id_2$. In other words, the columns of $\vec{Q} \in (\mathbb{R}^d)^{N\times 2}$ are an orthonormal basis of the subspace $\mathcal{R} \subset (\mathbb{R}^d)^N$ spanned by $\vec{P}\vec{N}Ke_i \equiv \vec{P}\vec{N}\,e_i$, where $e_i \in \mathbb{X}$, $i = 1 \to 2$, are the above mentioned null vectors of KAK. We note that the definitions of $\vec{P}$ and $\vec{N}$ yield that $\dim\mathcal{R} = 2$. Hence $\vec{\Pi}$ is the orthogonal projection from $(\mathbb{R}^d)^N$ onto $(\operatorname{im}\vec{Q})^\perp \equiv \mathcal{R}^\perp$.

2.13. The definitions of the spaces $\underline{V}$ and W can easily be generalized to a situation with K_B bubbles (enclosed areas), K_C curves, and K_T triple junction points. Note that Euler's formula yields that $6\,(K_B-1) = 2\,K_C = 3\,K_T$ in the absence of external boundary intersections. For example, $K_B = 2$, $K_C = 3$, and $K_T = 2$ in Figure 1. In particular, we would have that $\underline{V} := \{(\chi_1,\dots,\chi_{K_C}) \in [C(I,\mathbb{R}^d)]^{K_C} : \chi_{i_j}(p_{j,i_j}) = \chi_{1_j}(p_{j,1_j}),\ i = 2 \to 3,\ \forall j = 1 \to K_T\}$ and

$$W := \left\{(\chi_1,\dots,\chi_{K_C}) \in [C(I,\mathbb{R})]^{K_C} : \sum_{i=1}^3 (-1)^{p_{j,i_j}}\,\sigma_{i_j}\,\chi_{i_j}(p_{j,i_j}) = 0\ \ \forall j = 1 \to K_T\right\}. \tag{2.52}$$

Here $i_j \in \{1,\dots,K_C\}$, $i = 1 \to 3$, denotes the three curves meeting at triple junction j, while $p_{j,i_j} \in \{0,1\}$ denotes whether these curves start ($p_{j,i_j} = 0$) or end ($p_{j,i_j} = 1$) at the triple junction point j. That is, $|\{i_j : i = 1 \to 3\}| = 3$ for all $j = 1 \to K_T$, $|\{j : i_j = c\}| = 2$ for all $c = 1 \to K_C$, and $\sum_{j=1}^{K_T} p_{j,c} = 1$ for all $c = 1 \to K_C$.

The above definitions are easily generalized to the possible presence of external boundary intersections. Let K_B denote the number of bubbles (enclosed areas), K_T the number of triple junctions, K_C the number of curves, and K_I the number of

intersections with the external boundary. Then Euler's formula yields that $6\,(K_B - 1) = 2\,(K_C + K_I) = 3\,(K_T + K_I)$. (Note that the special case $K_I = 0$ corresponds to the formula given above.) For example, $K_B = 3$, $K_C = 3$, $K_T = 1$, and $K_I = 3$ in Figure 2. The corresponding precise definitions of the spaces $\underline{V}_\partial(\vec{X}^m)$ and W_∂ are straightforward, and we omit the details here.

Remark 2.14. The definition of $\vec{\Pi}$ can easily be adapted to a situation with K_B bubbles. In the case of no external boundary intersections, the subspace $E := \mathbb{X} \cap \ker A$ of the kernel of KAK has dimension K_B, and a possible basis consists of vectors that each "describe an admissible orientation of the boundary of a bubble" in terms of the given K_C curves. For example, if $K_B = 3$, one area is enclosed by curves $1, 2, 4$, and curve 2 is parameterized in the opposite direction to curves 1 and 4, then the corresponding eigenvector would be $(\frac{1}{\sigma_1}1^1, -\frac{1}{\sigma_2}1^2, 0, \frac{1}{\sigma_4}1^4, 0, 0)$. See the appendix in [6] for a more rigorous definition.

When boundary intersections are present, the subspace E has dimension $K_B - 1$, and a possible basis consists of vectors that each "describe an admissible orientation of the boundary of a bubble" (ignoring the boundary part of any bubble that is made up by the external boundary $\partial\Omega$) in terms of the given K_C curves. Similarly to [6, Appendix] one can show that it indeed holds that $\dim E = K_C - K_T = (K_C + K_I) - (K_T + K_I) = 3\,(K_B - 1) - 2\,(K_B - 1) = K_B - 1$.

One can interpret each set of basis vectors of E as a linearly independent collection of regions that preserve their areas. Then it is intuitive that the respective dimensions of E in the two described cases are K_B and $K_B - 1$.

2.2.2. Combined surface diffusion and mean curvature flow. We now adapt (2.47a,b) so that it approximates the evolution law (1.23), with (1.14a,b) and (1.24a,b). Choosing $\underline{V}^h$ based on $\underline{V}$, as defined in Remark 2.13, and $W_\star^h$ based on (recall (2.52)) $W_\star := \{(\chi_1, \ldots, \chi_{K_C}) \in [C(I, \mathbb{R})]^{K_C} : \sum_{i=1}^{3} \mathcal{H}_{\{i_0+1 \to K_C\}}(i_j)\,[(-1)^{p_{j,i_j}}\,\sigma_{i_j}\,\chi_{i_j}(p_{j,i_j})] = 0 \;\; \forall\; j = 1 \to K_T\}$, where $\mathcal{H}_{\mathcal{F}}$ is the indicator function for the set $\mathcal{F}$, yields the following approximation: Find $\{\vec{X}^{m+1},\, \kappa^{m+1}\} \in \underline{V}^h \times W_\star^h$ such that

$$\left\langle \frac{\vec{X}^{m+1} - \vec{X}^m}{\tau_m}, \chi\,\vec{\nu}^m \right\rangle_m^h - \langle \sigma\,\kappa^{m+1}, \chi \rangle_m^\star = 0 \qquad \forall\; \chi \in W_\star^h, \tag{2.53a}$$

$$\langle \kappa^{m+1}\,\vec{\nu}^m, \vec{\eta} \rangle_m^h + \langle \nabla_s\,\vec{X}^{m+1}, \nabla_s\,\vec{\eta} \rangle_m = 0 \qquad \forall\; \vec{\eta} \in \underline{V}^h, \tag{2.53b}$$

where $\langle \sigma\,\eta, \chi \rangle_m^\star := \langle \sigma\,\eta\,|_{\Gamma^{m,MC}}, \chi\,|_{\Gamma^{m,MC}} \rangle_m^h + \langle \sigma\,\nabla_s\,\eta\,|_{\Gamma^{m,SD}}, \nabla_s\,\chi\,|_{\Gamma^{m,SD}} \rangle_m$ for all $\eta,\, \chi \in W_\star^h$. Here $\Gamma^{m,MC} := \bigcup_{i=1}^{i_0} \Gamma_i^m$ and $\Gamma^{m,SD} := \bigcup_{i=i_0+1}^{K_C} \Gamma_i^m$. The above (weakly) approximates the correct conditions at the triple junctions, (1.14a,b) and (1.24a,b). Once again, it is a straightforward matter to derive an existence and uniqueness result and a stability result for (2.53a,b) under an appropriate mild assumption.

Using the natural extensions from the setup in Figure 1 to K_C curves of the matrices defined earlier, recalling (2.42) and (2.43), introducing $A_\star := \mathrm{diag}(\sigma_1\,M^1, \ldots, \sigma_{i_0}\,M^{i_0}, \sigma_{i_0+1}\,A^{i_0+1}, \ldots, \sigma_{K_C}\,A^{K_C})$, extending the associated operators, and introducing the orthogonal projection $K_\star$ onto the Euclidean space $\mathbb{X}_\star$ associated with $W_\star^h$, we can derive the following equations for the extended coefficient vectors:

$$\begin{pmatrix} \tau_m\,K_\star A_\star K_\star & -K_\star \vec{N}^T \vec{P} \\ \vec{P}\vec{N}K_\star & \vec{P}\vec{A}\vec{P} \end{pmatrix} \begin{pmatrix} \kappa^{m+1} \\ \delta\vec{X}^{m+1} \end{pmatrix} = \begin{pmatrix} 0 \\ -\vec{P}\vec{A}\vec{P}\,\vec{X}^m \end{pmatrix}. \tag{2.54}$$

Introducing the inverse $S_\star$ of $K_\star A_\star K_\star$ restricted on the set $(\ker K_\star A_\star K_\star)^\perp = \mathbb{X}_\star \cap \ker A_\star$, i.e., $S_\star\,K_\star A_\star K_\star\,v = K_\star A_\star K_\star\,S_\star\,v = v$ for all $v \in \mathbb{X}_\star \cap \ker A_\star$, we can employ

a Schur complement approach in order to transform (2.54) to

$$\kappa^{m+1} = \frac{1}{\tau_m}\, S_\star\, K_\star \vec{N}^T \vec{P}\, \delta\vec{X}^{m+1} + z, \qquad z \in \mathbb{X}_\star \cap \ker A_\star, \tag{2.55a}$$

$$\vec{\Pi}_\star \left(\vec{P}\vec{A}\vec{P} + \frac{1}{\tau_m}\, \vec{P}\vec{N} K_\star\, S_\star\, K_\star \vec{N}^T \vec{P} \right) \vec{\Pi}_\star\, \delta\vec{X}^{m+1} = -\vec{\Pi}_\star \vec{P}\vec{A}\vec{P}\, \vec{X}^m, \tag{2.55b}$$

where $\vec{\Pi}_\star : (\mathbb{R}^d)^N \to \mathcal{R}_\star^\perp$ is the orthogonal projection onto $\mathcal{R}_\star^\perp$ with $\mathcal{R}_\star := \vec{P}\vec{N} K_\star\, E_\star$ and $E_\star := \mathbb{X}_\star \cap \ker A_\star$.

Remark 2.15. A basis of $\mathcal{R}_\star$ can be found similarly to the pure surface diffusion case; see Remark 2.14. The corresponding basis of $E_\star$ is now made up of vectors that assign orientations for bubble boundaries to linear independent collections of curves Γ_i^m with $i > i_0$. Of course, one can extend the approximation (2.53a,b) to include external boundaries in the natural way. Although we omit the details here, some numerical results are given in section 3.

2.3. Quadruple junctions. A further generalization of the schemes (2.33a,b), (2.50a,b), and (2.53a,b) in this paper is the extension from triple junction points to quadruple junction points. This needs only a minor change to the introduced finite element spaces, and we omit the exact details here. However, numerical results for this practically interesting situation can be found in the next section.

3. Results. The Schur complement systems (2.13b), (2.26c), (2.45b), (2.46), (2.49), (2.51), and (2.55b) can be easily solved with a conjugate gradient solver. Where necessary, the solution of $KAK\, y = x$ and its variants in order to compute $S\, x$, $S_\partial\, x$, $S_\star\, x$, and $S_0\, x$, respectively, can be obtained with an (inner loop) CG solver without a projection, as the right-hand-side vector x always satisfies the necessary compatibility condition, e.g., $x \in (\ker KAK)^\perp$. See [30] for a justification for using a CG solver for a positive semidefinite system.

The system (2.14b), on the other hand, can be solved with an inexact Newton method. When f is given by (1.6), because of the singularity of $f^{-1} = f$ at the origin, the discrete system (2.14b) needs to be solved with a damped inexact Newton method, where as initial guess for the Newton iteration we choose $\delta\vec{X}^{m+1,0} := \vec{N}_0\, 1$. Moreover, we perform computations only for (1.7) with (1.6), where the evolution is well defined for all times, e.g., where the initial data $\vec{x}(\cdot,0)$ is such that that $\varkappa(\cdot,0) > 0$. In practice, the damped Newton method always converged in these cases and we always observed that $\kappa^m > 0$, $m = 0 \to M$. In addition, for the scheme (2.4), (2.3b) we will use the initial data $\kappa^0 := -(\vec{N}_0^{\,T} \vec{N}_0)^{-1} \vec{N}_0^{\,T} \vec{A}_0\, \vec{X}^0$, on noting that $\vec{N}_0^{\,T} \vec{N}_0$ is a diagonal matrix with strictly positive diagonal entries.

Throughout this section we use uniform time steps $\tau_m = \tau$, $m = 0 \to M-1$. For later purposes, we define $\vec{X}(t) := \frac{t-t_{m-1}}{\tau}\, \vec{X}^m + \frac{t_m - t}{\tau}\, \vec{X}^{m-1}$ for $t \in [t_{m-1}, t_m]$, $m \geq 1$.

3.1. Closed curves. Here we compare our scheme (2.3a,b) with $f(r) := r$, i.e., (2.17), with two other algorithms in the literature, namely the schemes (2.15) from [17] and (2.18) from [13]. As a first test, we repeat the computations for a true solution as given in [17, p. 604]. An exact solution to (1.7) with $f(r) := r$, so that the resulting $\Gamma(\cdot)$ solves (1.3), is given by

$$\vec{x}(\rho,t) = (1-2\,t)^{\frac{1}{2}}\, (\cos g(\rho), \sin g(\rho))^T, \quad \varkappa(\rho,t) = (1-2\,t)^{-\frac{1}{2}}, \quad t \in [0, \overline{T}),\ \overline{T} = 0.5, \tag{3.1}$$

TABLE 1
Absolute errors $\|\vec{X} - \vec{x}\|_{L^\infty}$ *for* $T = \frac{1}{2}\,\overline{T} = 0.25$ *and* $T = \overline{T} - \tau$, *respectively.*

	(2.15)		(2.18)		(2.17)	
N	$T = \frac{1}{2}\,\overline{T}$	$T = \overline{T} - \tau$	$T = \frac{1}{2}\,\overline{T}$	$T = \overline{T} - \tau$	$T = \frac{1}{2}\,\overline{T}$	$T = \overline{T} - \tau$
16	3.9879e-02	1.3476e-01	4.2132e-02	1.3978e-01	3.1574e-02	1.1731e-01
32	1.2994e-02	1.2155e-01	1.3973e-02	1.2496e-01	1.0287e-02	1.1149e-01
64	3.4556e-03	8.4151e-02	3.7408e-03	8.5576e-02	2.7043e-03	7.8982e-02
128	8.7924e-04	5.3324e-02	9.5359e-04	5.3825e-02	6.8469e-04	5.0839e-02
256	2.2112e-04	3.1728e-02	2.3963e-04	3.1887e-02	1.7184e-04	3.0545e-02
512	5.5339e-05	1.8217e-02	5.6193e-05	1.8270e-02	4.2953e-05	1.7655e-02
1024	1.3846e-05	1.0227e-02	1.4049e-05	1.0245e-02	1.0735e-05	9.9610e-03

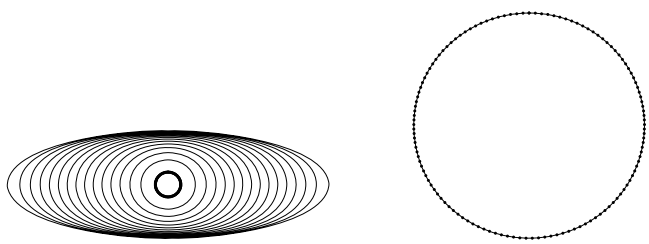


FIG. 3. *A plot of* $\vec{X}(t)$ *at times* $t = 0,\ 0.1, \ldots, 1.5$ *and at time* $t = T$ *(scaled).*

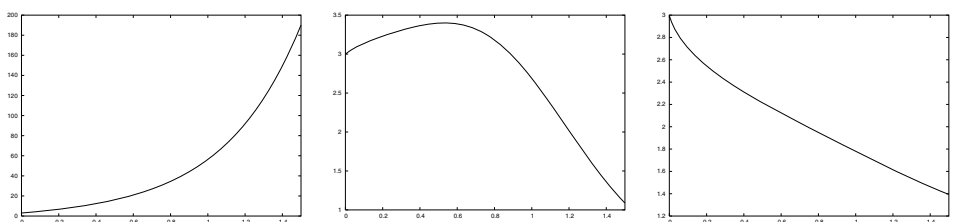


FIG. 4. *A plot of the ratio* $r = h_{\vec{X}^m}/\ell_{\vec{X}^m}$ *for the three schemes* (2.15), (2.18), *and* (2.17).

where $g(\rho) = 2\pi\rho + 0.1\,\sin(2\pi\rho)$ in order to make the initial distribution of nodes nonuniform. We note that $\vec{x}_t \,.\, \vec{\tau} = 0$ for the solution (3.1). We compare our results from (2.17) to the schemes (2.15) and (2.18); see Table 1. We use $\tau = 0.5\,h^2$ and either $T = \frac{1}{2}\,\overline{T}$ or $T = \overline{T} - \tau$. Here and in what follows we always compute the error $\|\vec{X} - \vec{x}\|_{L^\infty} := \max_{m=1\to M} \|\vec{X}(t_m) - \vec{x}(\cdot, t_m)\|_{L^\infty}$, where $\|\vec{X}(t_m) - \vec{x}(\cdot, t_m)\|_{L^\infty} := \max_{j=1\to N} \min_{\rho\in J} |\vec{X}^m(q_j) - \vec{x}(\rho, t_m)|$ between $\vec{X}$ and the true solution on the interval $[0, T]$. We note that the experiments indicate that the convergence rate for the error away from the singularity is $O(h^2)$, and up to the singularity at time $\overline{T}$ is of order less than $O(h)$, for all three schemes, as is to be expected.

The next experiment is for a mild ellipse. The parameters were chosen as follows. $N = 128$, $\tau = 10^{-2}$, $T = 1.5$, and the initial curve is a $3:1$ ellipse with a unit semiminor axis. In order to highlight one difference between the three schemes in consideration, we plot for each of them the ratio $r := h_{\vec{X}^m}/\ell_{\vec{X}^m}$, where $h_{\vec{X}^m} := \max_{j=1\to N} |\vec{X}^m(q_j) - \vec{X}^m(q_{j-1})|$ and $\ell_{\vec{X}^m} := \min_{j=1\to N} |\vec{X}^m(q_j) - \vec{X}^m(q_{j-1})|$, over time. The evolution of our scheme (2.17) can be seen in Figure 3. Plots of the ratio r for the three schemes can be seen in Figure 4. One can clearly see that the ratio increases for scheme (2.15), while the tangential movement of vertices induced by the

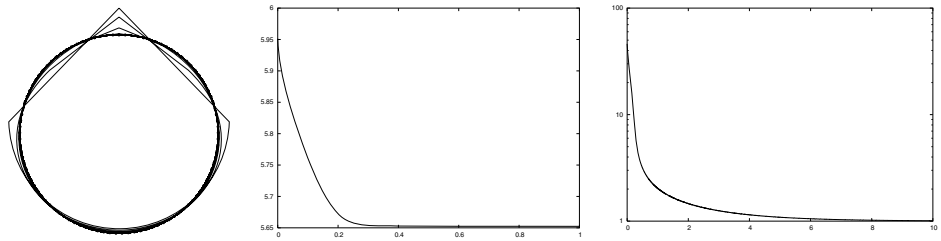


FIG. 5. *Plots of* $\vec{X}(t)$ *at* $t = 0, 0.1, \ldots, T = 10$, $|\Gamma(t)|$ *for* $t \in [0,1]$, *and* $\log r(t)$ *for* $t \in [0,T]$.

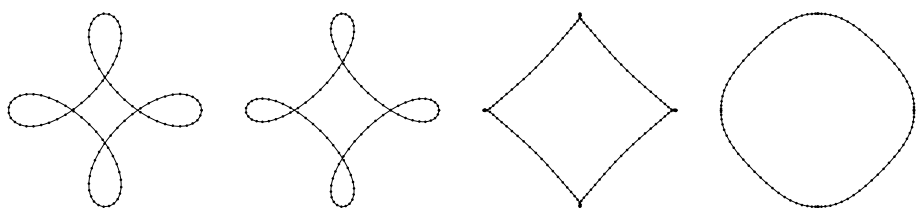


FIG. 6. *Plot of* $\vec{X}(t)$ *at times* $t = 0.08, 0.085, 0.0885, 0.9$ *(scaled).*

other two schemes, as discussed in Remark 2.5, results in a decrease of the ratio r, which approaches the optimal value 1. In order to underline this point further, we conducted an experiment for area preserving mean curvature flow, (1.8). The initial curve for our approximation (2.4), with $f(r) := r$, and (2.3b) consists of a semicircle and a single additional node on the periphery of the circle with unit radius. We used the parameters $N = 100$, $T = 10$, and $\tau = 10^{-3}$. In Figure 5 we show $\vec{X}(t)$ at different times, as well as a plot of $|\Gamma(t)|$ and $\log r(t)$ over time. One can clearly see that although an approximation to the true steady state, a circle, is reached very quickly (at around time $t = 0.6$), in the remaining time the vertices are continually moved tangentially, which results in a further decrease in the ratio r that eventually approaches the optimal value 1.

In addition, we computed (2.3a,b) with $f(r) := r$ for a self-intersecting curve that exhibits cusps during the evolution. The initial curve is taken from [13, Figure 1] and can be parameterized by $\vec{x}(\rho,0) = \cos(4\pi\,\rho)\,(\cos(2\pi\,\rho), \sin(2\pi\,\rho))^T$. We used the parameters $N = 100$, $\tau = 10^{-3}$, and $T = 0.09$. We see in Figure 6 that for this example the mesh points are clustered in regions where the curvature is high, due to the very fast evolution there. Of course this clustering decreases as the time step size τ is reduced in line with an equidistribution result for the semidiscrete scheme; recall Remark 2.5. Similarly to other parametric algorithms, our approximation jumps across the singularity in the evolution; see, e.g., [13, Figure 1] for (2.4) and [14, Figure 4.2] for (2.15).

For the first experiment for the nonlinear approximation (2.3a,b) we used the exact solution to (1.4) with (1.5):

$$\vec{x}(\rho,t) = (1-(\beta+1)\,t)^{\frac{1}{\beta+1}}\,(\cos g(\rho), \sin g(\rho))^T, \quad \varkappa(\rho,t) = (1-(\beta+1)\,t)^{-\frac{1}{\beta+1}}, \quad t \in [0,\overline{T}),$$

where $\overline{T} = \frac{1}{\beta+1}$ and g is given as in (3.1). We note once again that here $\vec{x}_t \,.\, \vec{\tau} = 0$. We report on a corresponding error table for $\beta = \frac{1}{2}$ and $\beta = \frac{1}{3}$ in Table 2, where we used $\tau = 0.5\,h^2$.

TABLE 2
Absolute errors $\|\vec{X} - \vec{x}\|_{L^\infty}$ *for* $T = \frac{1}{2}\,\overline{T} = \frac{1}{2(\beta+1)}$, $T = \overline{T} - \tau$, *and* $T = 1$, *respectively.*

	$\beta = \frac{1}{2}$		$\beta = \frac{1}{3}$		$f(r) = -r^{-1}$
N	$T = \frac{1}{2}\,\overline{T}$	$T = \overline{T} - \tau$	$T = \frac{1}{2}\,\overline{T}$	$T = \overline{T} - \tau$	$T = 1$
16	1.8812e-02	8.4620e-02	1.2804e-02	6.0554e-02	2.6054e-01
32	5.4838e-03	6.0185e-02	3.3827e-03	3.8905e-02	7.5559e-02
64	1.4210e-03	3.4475e-02	8.6680e-04	1.9613e-02	1.9709e-02
128	3.5637e-04	1.7774e-02	2.1613e-04	9.0762e-03	4.9869e-03
256	8.9173e-05	8.6226e-03	5.3925e-05	3.9824e-03	1.2502e-03
512	2.2226e-05	4.0244e-03	1.3431e-05	1.6607e-03	3.1255e-04

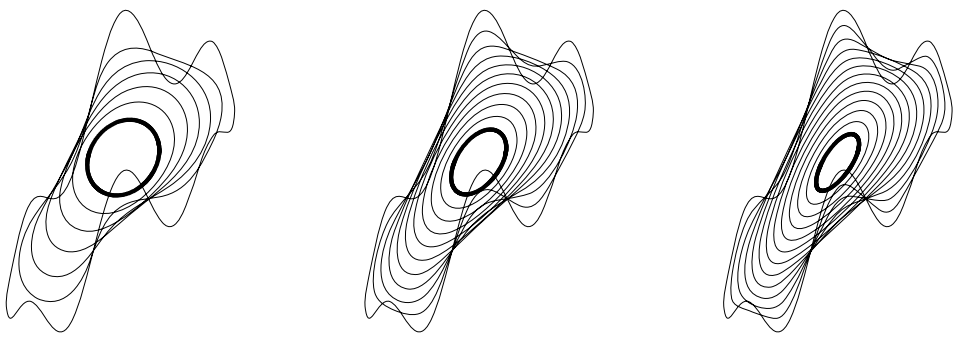


FIG. 7. $\mathcal{V} = \varkappa^\beta$, *with* $\beta = 1, \frac{1}{2}, \frac{1}{3}$. *We plot* $\vec{X}(t)$ *at times* $t = 0, 0.05, \ldots, T$, *with* $T = 0.3, 0.5$, *and* 0.6, *respectively.*

Next, we considered (2.3a,b) with (1.5) for the initial curve as given in [35, p. 1494] for the three different choices $\beta = 1, \frac{1}{2}, \frac{1}{3}$; i.e., we chose

$$\vec{x}(\rho, 0) = \Bigg(\cos g_0(\rho), \\ \frac{1}{2}\,\sin g_0(\rho) + \sin\left(\cos g_0(\rho)\right) + \sin g_0(\rho)\,\left[\frac{1}{5} + \sin g_0(\rho)\,\sin^2 g_0(3\,\rho)\right]\Bigg)^T,$$

where $g_0(\rho) = 2\pi\,\rho$, and set $\vec{X}^0 = \pi^h\,\vec{x}(\cdot, 0)$ for a uniform partitioning of J. The numerical results for the approximation (2.3a,b) can be seen in Figure 7, where we used $N = 256$ and $\tau = 10^{-3}$. It should be noted that in [35] a system of second order PDEs is derived to model (1.4), which introduces a tangential movement that locally equidistributes nodes under discretization by a finite difference approximation. However, the system and subsequent approximation is far more complicated than our simple approximation (2.3a,b).

3.1.1. Inverse mean curvature flow. Here we consider the flow (1.4) with (1.6). First, we performed a convergence test for the approximation (2.3a,b) with f given by (1.6). A true solution to (1.7) and (1.6) with $\vec{x}_t \,.\, \vec{\tau} = 0$ is given by

$$\vec{x}(\rho, t) = \exp(t)\,(\cos g(\rho), \sin g(\rho))^T, \quad \varkappa(\rho, t) = \exp(-t), \quad t \in [0, \infty),$$

where g is defined as in (3.1). We report on the corresponding errors for $\tau = 0.5\,h^2$ in the last column of Table 2. The evolution for $N = 256$ can be seen on the left

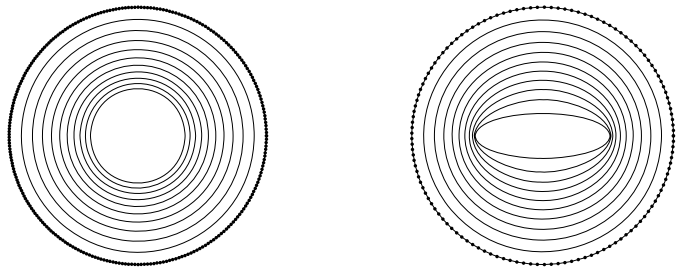


FIG. 8. $\vec{X}(t)$ *for* $t = 0, 0.1, \ldots, T = 1$ *for the inverse mean curvature flow of a circle (left) and an ellipse (right).*

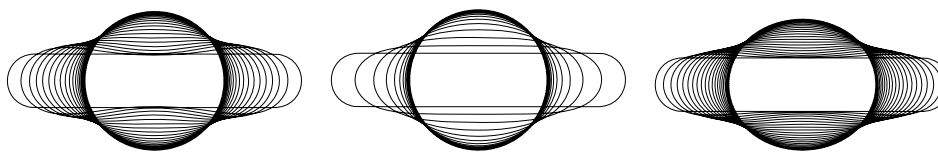


FIG. 9. *Different evolutions for surface diffusion, area preserving mean curvature flow, and the intermediate flow* (1.9)*, with* $\alpha = \xi = 1$*. In each case, we plot* $\vec{X}(t)$ *for* $t = 0, 1, \ldots, T = 20$*.*

of Figure 8. The next experiment is for a $3:1$ ellipse that expands to a circle. The discretization parameters are $N = 128$, $\tau = 10^{-3}$, and $T = 1$; see the right-hand side of Figure 8.

3.1.2. Intermediate evolution laws. In this subsection, we report on numerical results for our approximation (2.20a–c) of the intermediate evolution law (1.10). First, we compare the different evolutions of (1.2), (1.8) with $f(r) := r$, and (1.9) with $\alpha = \xi = 1$, for an initial curve that is given by an elongated tube of dimensions 10×1. As discretization parameters for the schemes (2.21a,b), (2.4) with (2.3b) and $f(r) := r$, and (2.20a–c) we used $N = 256$, $\tau = 10^{-3}$, and $T = 20$, and the corresponding results are shown in Figure 9. One can clearly see that while the curve that moves under area preserving mean curvature flow remains convex throughout the evolution, this is not the case for the other two evolution laws. The area losses for the respective schemes were 0.004%, 0.016%, and 0.002%. As noted before for $\alpha \to \infty$ and $\xi = 1$, the solutions to (1.9) converge to solutions of (1.8), while $\xi \to \infty$ and $\alpha = 1$ corresponds to the law (1.2). We now investigate this property numerically. To this end, we repeat the above experiments for (2.20a–c) with $\xi = 1$, $\alpha = 1000$ and $\xi = 1000$, $\alpha = 1$, respectively. The results, for which the respective area losses were 0.008% and 0.004%, can be seen in Figure 10. One can clearly see the similarity between these evolutions and their corresponding limits in Figure 9. We note also that the curve in the first evolution remains convex, while the curve for the second evolution does not.

3.2. Triple junctions. In the first experiment for triple junctions (see Figure 11) we simulate how two initially elliptic bubbles move under motion by mean curvature. Throughout, we assume equal surface energies, $\sigma_i = 1$, $i = 1 \to K_C$, unless stated otherwise. The plot on the left-hand side of Figure 11 shows the evolution

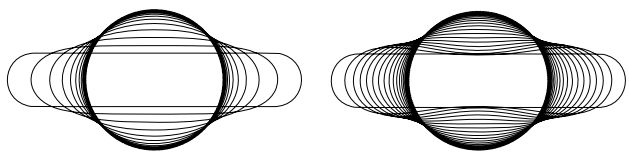


FIG. 10. *The flow* (1.9) *with* $\xi = 1$, $\alpha = 1000$ *(left) and* $\xi = 1000$, $\alpha = 1$ *(right). In each case, we plot* $\vec{X}(t)$ *for* $t = 0, 1, \ldots, T = 20$.

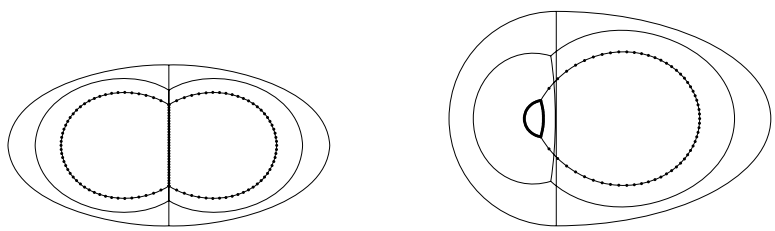


FIG. 11. $\vec{X}(t)$ *for* $t = 0, 0.2, T = 0.4$. *An equal area double bubble under mean curvature flow is shown on the left, and a nonequal area case on the right.*

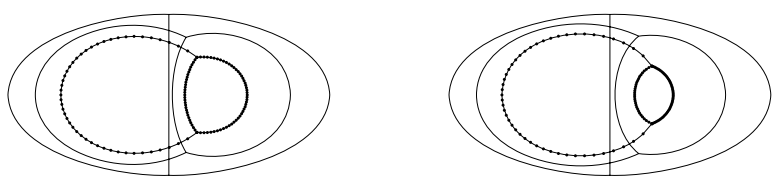


FIG. 12. $\vec{X}(t)$ *for* $t = 0, 0.2, T = 0.4$. *An equal area double bubble under mean curvature flow is shown for surface energies* $(1, 1, \frac{3}{2})$ *and* $(1, 1, \frac{7}{4})$, *respectively.*

for two equal area bubbles, while the nonequal area case is displayed on the right-hand side. The chosen parameters were $N = 128$, $\tau = 0.01$, and $T = 0.4$. The initial shapes are given by two segments of a $2:1$ ellipse with unit semiminor axis and a straight line, or a semicircle together with an elliptic segment from the above ellipse and a straight line, respectively. We repeated the experiment on the left-hand side of Figure 11 for different surface energies. The surface energies were chosen to be $(\sigma_1, \sigma_2, \sigma_3) = (1, 1, \frac{3}{2})$ and $(1, 1, \frac{7}{4})$. That means that the length of the curve Γ_3 is weighted more in the overall energy $|\Gamma|$, so that it will shorten faster during the evolution; see Figure 12.

The next experiment is for motion by surface diffusion. The initial setup, as depicted in Figure 13, consists of a segmentation of the $[-1, 1]^2$ square, with the middle segments having width 0.3. Our discretization parameters were $N = 256$, $\tau = 10^{-3}$, and $T = 0.04$. The evolution leads to one curve segment shrinking to a single point. Of course, our approximation cannot compute beyond that singularity. However, on the right-hand side of Figure 13 we show how, after a topological change, the evolution continues to a numerically steady state. This numerical result was

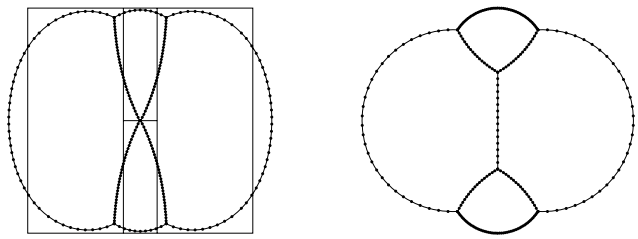


FIG. 13. *A curve shrinks to a point: $\vec{X}(t)$ for $t = 0, T$ (left). The eventual numerically steady state after a change of topology (right).*

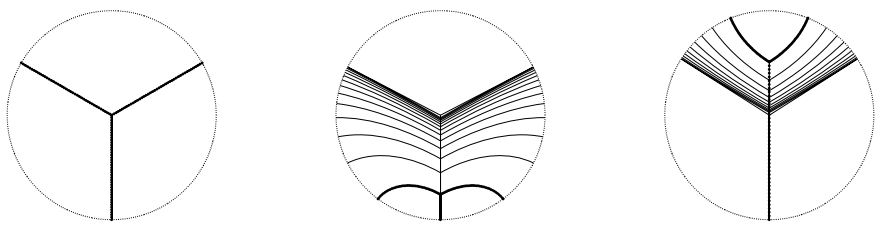


FIG. 14. *A triple junction inside the unit circle moving under mean curvature flow.*

obtained by restarting the computation from a curve network that was topologically equivalent to the steady state, and with enclosed regions having the same area as before, i.e., 0.7, 0.3, 0.7, and 0.3.

3.3. Boundary intersections. In the first experiment for curve networks with triple junctions and intersections with the external boundary (see Figure 14), we simulate how an initial "letter Y" inside the unit circle moves under motion by mean curvature. We varied the initial angle θ_3 between the two upper curves, which meet at the origin. Here and throughout, $\theta := (\theta_1, \theta_2, \theta_3)$ is defined as the angles formed by the three curve segments meeting at a triple junction, and θ_i, $i = 1 \to 3$, denotes the angle opposite the curve Γ_i at a triple junction; recall (1.15).

The values used for θ_3 are 120°, 125°, and 115° with $\theta_1 = \theta_2$. The first setup is a steady state for this law of motion, while the other two experiments show that the symmetric boundary intersections will either move up or down the unit circle. The chosen discretization parameters for (2.33a,b) were $N = 128$, $\tau = 0.01$, and $T = 1, 2.6$, and 2, respectively. The times shown in Figure 14 are $t = 0, 0.2, \ldots, T$. The maximum distance $d_{\partial\Omega} := \max_{i=1\to3} \operatorname{dist}(\vec{X}_i^M(1), \partial\Omega)$ of the curve endpoints at time T was $d_{\partial\Omega} = 0$, 8.6×10^{-3}, and 1.8×10^{-3}, respectively. Moreover, note that eventually the solution for the latter two experiments becomes either a single point (on the boundary $\partial\Omega$) or a single straight line.

We repeated the last experiment inside the square $[-1, 1]^2$. Note that in this case $F \notin C^1(\mathbb{R}^2)$, but as the points of the derivative discontinuities are never reached by the evolving curves, this has no effect on the evolution; this will always be true in all examples below for such F. Here no steady state solution exists for this law of motion. For the results see Figure 15, where we chose as angles $\theta_3 = 180°,\ 95°,\ 85°$ and integrated until time $T = 1.4,\ 2$, and 0.4, respectively. We observe that in the first two cases the solution exhibits a traveling wave character. In fact, an exact solution can be found for this setup; see [27, p. 313]. Let $\gamma(\rho, t) := \frac{6}{\pi} \ln(\cos(\frac{\pi}{6}(1-\rho))) - \frac{\pi}{6} t + 1$

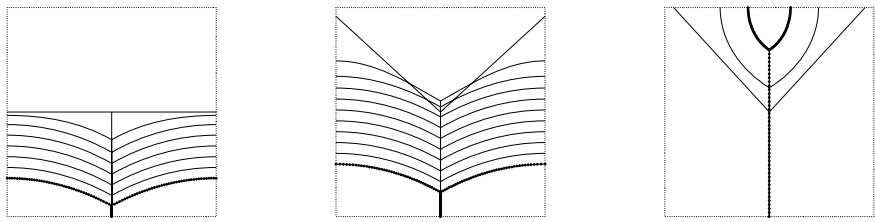
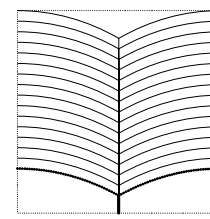


FIG. 15. *A triple junction inside the square* $[-1,1]^2$ *moving under mean curvature flow.*

TABLE 3

Absolute errors $\|\vec{X}-\vec{x}\|_{L^\infty}$ *and triple junction angles for the test problem. On the right is shown an approximation of the exact solution* (3.2) *for* $t = 0, 0.2, \ldots, 3$

N	$\|\vec{X}-\vec{x}\|_{L^\infty}$	Angles $\theta = (\theta_1, \theta_2, \theta_3)$
16	3.1156e-02	(115.9, 115.9, 128.1)
32	8.7298e-03	(118.2, 118.2, 123.7)
64	2.2632e-03	(119.2, 119.2, 121.6)
128	5.7263e-04	(119.6, 119.6, 120.7)
256	1.4362e-04	(119.8, 119.8, 120.4)
512	3.5954e-05	(119.9, 119.9, 120.2)



and

(3.2)
$$\vec{x}_1(\rho,t) = \begin{pmatrix} -\rho \\ \gamma(\rho,t) \end{pmatrix}, \quad \vec{x}_2(\rho,t) = \begin{pmatrix} \rho \\ \gamma(\rho,t) \end{pmatrix}, \quad \vec{x}_3(\rho,t) = (1-\rho)\begin{pmatrix} 0 \\ \gamma(0,t) \end{pmatrix} + \rho \begin{pmatrix} 0 \\ -1 \end{pmatrix}.$$

Then $\vec{x} := (\vec{x}_1, \vec{x}_2, \vec{x}_3)$ is a solution to (1.12), (1.14a,b), and (1.17a,b). We used this exact solution to perform a convergence test for our approximation (2.33a,b). See Table 3, where we show the error $\|\vec{X}-\vec{x}\|_{L^\infty}$ on the time interval $[0,T]$ for $T = 3$ and $\tau = 0.5\,h^2$. Here $\|\vec{X}-\vec{x}\|_{L^\infty} := \max_{m=1\to M} \|\vec{X}(t_m) - \vec{x}(\cdot,t_m)\|_{L^\infty}$ with $\|\vec{X}(t_m) - \vec{x}(\cdot,t_m)\|_{L^\infty} := \max_{i=1\to 3} \max_{j=0\to N_i} \min_{\rho\in I} |\vec{X}_i^m(q_j^i) - \vec{x}_i(\rho,t_m)|$ is computed by employing a Newton method for the curved segments of the true solution. The evolution for $N = 128$ is shown on the right of Table 3.

3.3.1. Surface diffusion with boundary intersections. We repeated the above type of experiments for motion by surface diffusion, i.e., the approximation (2.50a,b) of the system (1.19)–(1.22) and (1.17a,b). In the first experiment we simulate how an initial "letter Y" inside the unit circle moves under motion by surface diffusion. We varied the initial angle θ_3 between the two upper curves at the origin with values 120°, 180°, and 45°. While the first setup is already a steady state for this law of motion, the other two experiments soon reach a steady state. The chosen parameters were $N = 128$, $\tau = 10^{-4}$, and $T = 0.05$. In Figure 16 we show $\vec{X}(t)$ at times $t = 0,\ T$. Note that the maximum distance of the curve endpoints to the external boundary $\partial\Omega$ at time T was $d_{\partial\Omega} = 0,\ 3\times10^{-5}$, and 8×10^{-5}, respectively. The results for the same experiments inside the domain $\Omega = [-1,1]^2$ can be found in Figure 17. We note that here $d_{\partial\Omega} = 0$ in each case, as the external boundary consists of straight lines.

We repeated the experiment on the left-hand side of Figure 16 for different surface energies. The surface energies were chosen to be $(\sigma_1,\sigma_2,\sigma_3) = (1,1,\frac{3}{2})$ and $(1,1,\frac{7}{4})$, respectively. We observe that as the length of the curve Γ_3 is weighted

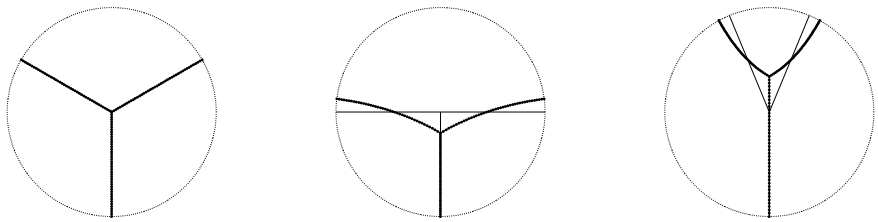


FIG. 16. *A triple junction inside the unit circle moving under motion by surface diffusion.*

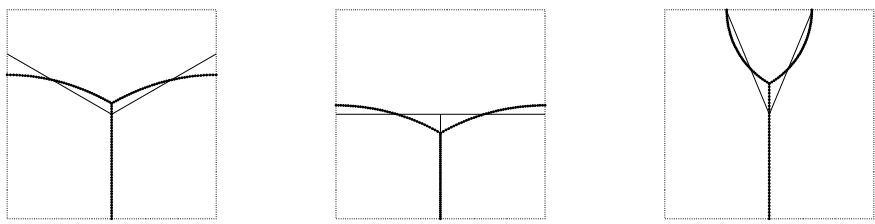


FIG. 17. *A triple junction inside the square* $[-1,1]^2$ *moving under motion by surface diffusion.*

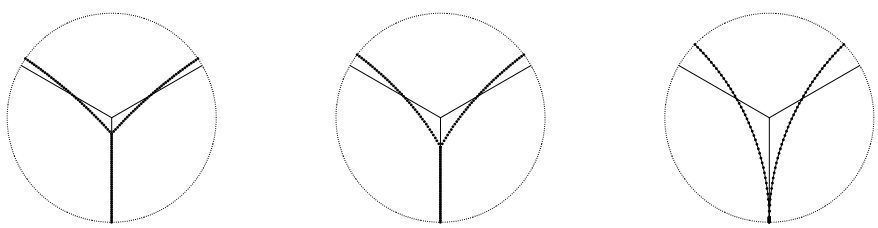


FIG. 18. *A triple junction inside the unit circle moving under motion by surface diffusion for surface energies* $(1,1,\frac{3}{2})$, $(1,1,\frac{7}{4})$, *and* $(1,1,2)$.

more in the overall energy $|\Gamma|$, it shortens in comparison to the other two curves during the evolution; see Figure 18. The observed angles at the triple junction are $\theta = (138.4, 138.4, 83.2)$ and $\theta = (150.6, 150.6, 58.8)$. (Note that Young's law yields $\theta = (138.6, 138.6, 82.8)$ and $\theta = (151, 151, 57.9)$, respectively, for the exact solution.) Finally, we give the same evolution for the surface energies $(1,1,2)$ on the right-hand side of Figure 18. Here in the true steady state the curve Γ_3 has shrunk to a point on the boundary $\partial\Omega$, which, as it represents a change of topology, our approximation cannot compute. We give a plot of $\vec{X}(T)$ for $T = 0.5$, when the observed triple junction angles are $\theta = (178.2, 178.2, 3.7)$.

An example with only two enclosed areas can be seen in Figure 19. Here the initial curve is given by a straight line through the origin that forms an angle of 10° with the x-axis inside a 2:1 elliptic domain Ω with unit semiminor axis. The chosen parameters were $N = 128$, $\tau = 10^{-4}$, $T = 2$. On the left-hand side of Figure 19 we plot $\vec{X}(t)$ at times $t = 0, 0.05, \ldots, T = 2$. The right-hand side shows a similar experiment, where the initial curve goes through the point $(0, -\frac{1}{2})^T$. The parameters were $N = 32$, $\tau = 10^{-3}$, and $T = 10$, and we display $\vec{X}(t)$ at times $t = 0, 0.5, \ldots, T = 10$. In both cases the evolution finds the minimum of a corresponding partitioning problem in which the length of the interface between two phases is minimized subject to an area constraint for the phases.

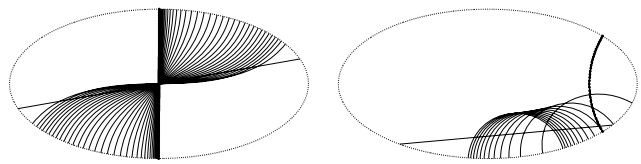


FIG. 19. *A curve inside an elliptic domain moving under motion by surface diffusion.*

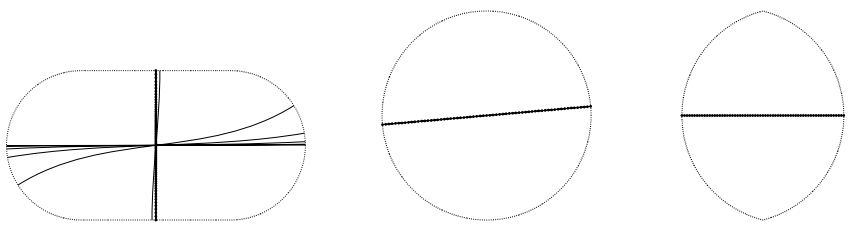


FIG. 20. *Initially straight lines that are unstable (left), neutral (middle), or stable (right). On the left,* $\vec{X}(t)$ *at times* $t = 0, 2, \ldots, 16$. *The remaining plots show* $\vec{X}(t)$ *at times* $t = 0, 6$.

Next, we investigate numerically the stability of some of the setups discussed in [25]. We start by considering a domain Ω that has a boundary with piecewise constant curvature. The analysis in [25, Figure 6] predicts that a straight line inside Ω is only stable if it is sufficiently short. In the situation considered in Figure 20, it must not exceed twice the radius of the two arcs. In the first experiment, the total dimensions of the domain are 4×2. The initial line has a slope of $0.1°$, and as the straight line on the x_1-axis is not stable, it evolves to a straight line on the x_2-axis. If, however, the dimensions of the domain are only $\frac{3}{2} \times 2$, then the straight line is stable, and this is seen in the corresponding experiment. The case where the domain is a circle is neutral, as any straight line through the origin is a steady state solution. As an example, we show an initially straight line with slope $5°$, which does not change under the surface diffusion flow. The final solutions depicted in Figure 20 are all numerically steady states.

The next experiment investigates the behavior for the setup in [25, Figure 7]. In each case, we use a domain where the lower boundary follows a sine shape. In particular, the lower boundary is given by $x_2 = -1 + \frac{1}{8}\sin(2\pi\, x_1)$. The initial profile is always a semicircle with radius 0.5. Depending on the sign of the curvature of the domain boundary, that solution is either stable or unstable. We investigate the stability numerically, by using $\vec{z} = (\mp 0.24, -1 \pm \frac{1}{8})$ as the center of the half-circle. Note that $\vec{z} = (\mp\frac{1}{4}, -1 \pm \frac{1}{8})$ corresponds to a steady state. The results are shown in Figure 21, where we also include the neutral case of a semicircle attached to a straight part of the boundary.

Next, we performed computations with three enclosed areas and two vertices on the external boundary $\partial\Omega$; see Figure 22. Starting with an initially circular area (with radius 0.75) that is connected by two straight lines to the external boundary inside a $2:1$ rectangular domain, we observe different steady state solutions depending on the chosen surface energies $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$. We note that for a physically relevant setup $\sigma_1 = \sigma_4$, where Γ_1 and Γ_4 are the straight line curve segments. We used the

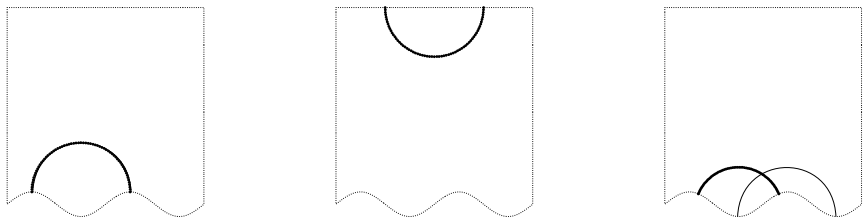


FIG. 21. *Stable (left) and unstable (right) setup. In the middle a neutral situation is shown. Each plot shows* $\vec{X}(t)$ *at times* $t = 0, 1$.

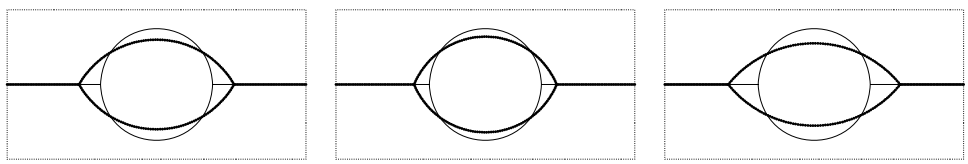


FIG. 22. *Steady state solutions for different angles* θ_1.

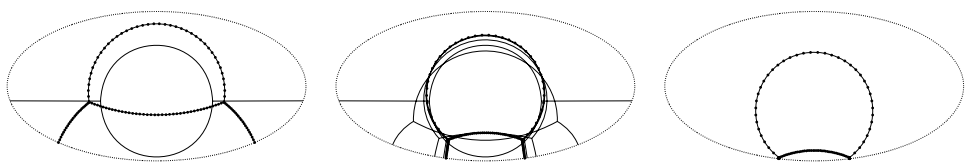


FIG. 23. *Solution for circular setup for pure surface diffusion (left) and combined mean curvature and surface diffusion inside an elliptic domain. In the middle* $\vec{X}^m(t)$ *at times* $t = 0, 0.5, \dots, 2$ *and on the right* $\vec{X}^m(T)$ *at* $T = 40$.

values $\sigma = (1, \lambda, \lambda, 1)$ with $\lambda = 1,\ \frac{\sqrt{7}}{2},\ (\frac{5}{8})^{\frac{1}{2}}$, corresponding to [7, Figures 3 and 8], so that the true triple junction angle is $\theta_1 = 120°,\ 135°,\ 102°$, respectively. The chosen discretization parameters for each experiment were $N = 256$, $\tau = 10^{-4}$, and $T = 0.2$. The observed angles inside the circular area were $\theta_1 = 118.6°$, $133.9°$, and $100.2°$, respectively.

For later purposes, we repeated the first experiment in Figure 22, but now inside a 2:1 elliptic domain and with the initial profile at the height $y = -0.2$. The chosen parameters were $N = 128$, $\tau = 10^{-3}$, and $T = 2$, by which time the numerical solution has reached a steady state. The results are shown on the left-hand side of Figure 23.

Next, we report on some experiments for (2.50a) and (2.34), i.e., for the surface diffusion of a curve network with specified contact angles α_i that the curves form with the external boundary $\partial\Omega$. Here we use as initial data a "letter Y" inside the unit circle, similarly to Figure 14. We varied the angle α_i, $i = 1 \to 3$, that the three curves form with the external boundary. In Figure 24 we show the solution $\vec{X}(t)$ at times $t = 0,\ T$ for the values $\alpha_1 = \alpha_2 = \alpha_3 = \alpha$ with $\alpha = 60°$, $30°$, and $5°$. The chosen discretization parameters were $N = 128$, $\tau = 10^{-3}$, and $T = 0.1$. The same experiments inside the square $[-1, 1]^2$, where the initial profile is now a "letter T,"

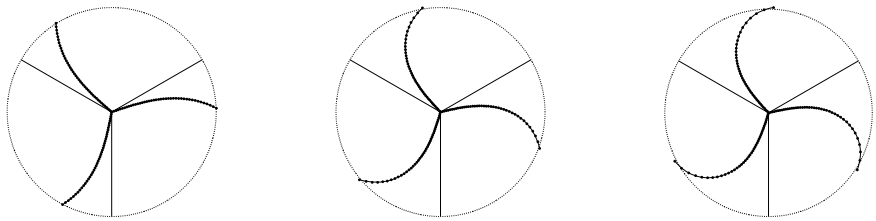


FIG. 24. *A triple junction inside the unit circle with different contact angles at* $\partial\Omega$.

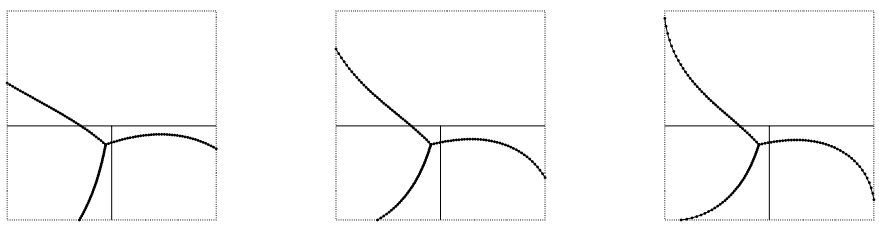


FIG. 25. *A triple junction inside the square* $[-1,1]^2$ *with different contact angles at* $\partial\Omega$.

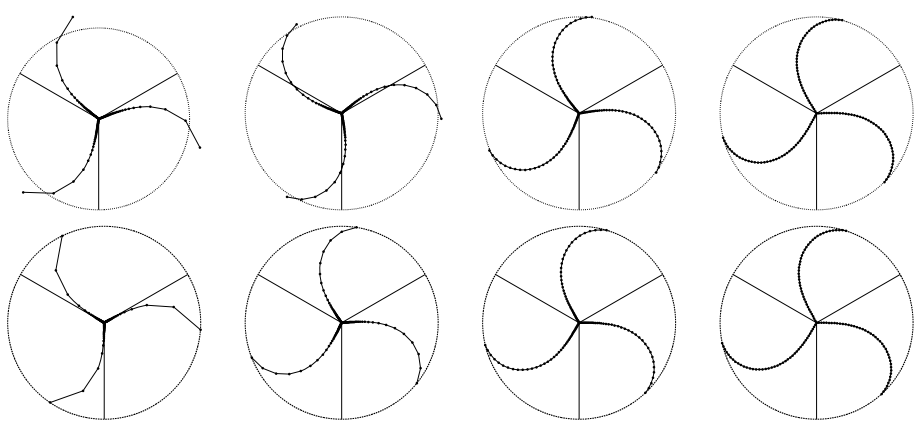


FIG. 26. *Tangential movement and boundary vertex mislocation for different time step sizes,* $\tau = 10^{-k}$, $k = 2 \to 5$. *Results for* (2.50a,b) *(top) and the corresponding scheme with a projection onto* $\partial\Omega$ *at each time step (bottom).*

can be seen in Figure 25.

We note that in practice we observe a tangential movement of mesh points away from the boundary intersection towards the triple junction. On the other hand, we have mentioned in Remark 2.5 that it was shown in [6] that for a semidiscrete in time approximation the vertices equidistribute along each curve Γ_i^h, $i = 1 \to 3$. Hence for τ sufficiently small, we expect the parameterizations $\vec{X}_i^m$ to become more and more uniform. We investigate this behavior with the following set of experiments. We repeat the last experiment in Figure 24 and integrate until time $T = 1$ for different choices of $\tau = 10^{-k}$, $k = 2 \to 5$. In the top row of Figure 26 we see clearly that the distribution of vertices improves as τ decreases. For comparison, we include the results for the same set of experiments for a version of our scheme, where one employs

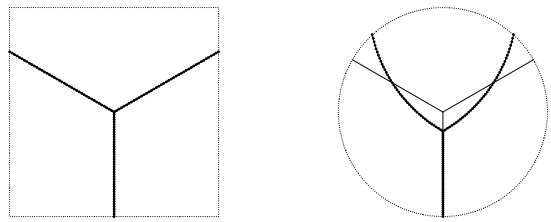


FIG. 27. *A steady state solution for given boundary contact angles inside the square* $[-1,1]^2$*, and the corresponding evolution inside the unit circle.*

a projection step that orthogonally projects the boundary vertices of $\vec{X}^{m+1}$ onto $\partial\Omega$ at every time step. As can be seen from the results in the bottom row of Figure 26, for small time step sizes the results are virtually indistinguishable, while for large time steps it can be beneficial to employ the described projection step.

Finally, we provide an experiment for a steady state inside the domain $\Omega = [-1,1]^2$. The chosen contact angles are $\alpha_1 = 60°$, $\alpha_2 = 120°$, and $\alpha_3 = 90°$, so that the initial "letter Y" is a true steady state solution. We confirm this with a numerical experiment for (2.50a) and (2.34) with $N = 128$, $\tau = 10^{-3}$, and $T = 0.5$; see Figure 27, where we complement this result with the corresponding evolution inside the unit circle.

3.3.2. Combined surface diffusion and mean curvature flow. In this subsection, we report on numerical results for the approximation (2.53a,b) of (1.23). First, we repeated the experiment reported on earlier, shown on the left-hand side of Figure 23, but now for a combined motion of surface diffusion and mean curvature flow. In particular, we prescribe surface diffusion on the two initially circular curves, while the initially straight-lined curves experience motion by mean curvature. See Figure 23 for the results, where we plot $\vec{X}^m(t)$ at different times. The chosen parameters were $N = 128$, $\tau = 10^{-3}$, and $T = 40$. One can clearly see the very different evolution compared to the pure surface diffusion flow, and note in particular that each of the two curves moving under motion by mean curvature now shrink to a single point (on the boundary $\partial\Omega$).

The next experiment is motivated by the considerations on a traveling wave solution to (1.23) that was first mentioned in [36] and which plays a major role in the study of grain boundary motion; see also [33] and [7, Figure 7]. The computations shown in Figure 28 start with three curves meeting at a single triple junction, of which the two horizontal ones experience motion by surface diffusion, while the third curve undergoes motion by mean curvature. As surface energies we chose $\sigma = (1, 1, \sigma_3)$ with $\sigma_3 = 1$ and $\sigma_3 = \frac{3}{2}$, respectively. Moreover, the evolution law for the curve moving under motion by mean curvature is here given by $\mathcal{V}_3 = \mu_3\,\sigma_3\,\varkappa_3$ in place of (1.12), where for the mobility we chose either $\mu_3 = 1$ or $\mu_3 = 2$. Note that our approximation (2.53a,b) can easily be adapted to this situation by replacing σ_3 with $\mu_3\,\sigma_3$ in (2.53a). The domain Ω is an 8×2 rectangle. The chosen discretization parameters were $N = 256$ and $\tau = 10^{-3}$.

3.4. Quadruple junctions. Here we consider a setup of three enclosed areas meeting at two triple junctions and one quadruple junction. A physical interpretation of this configuration is an alloy material with two grains and two phases, similar to the situation considered in [10, 26]. For the initial curves depicted in Figure 29 this

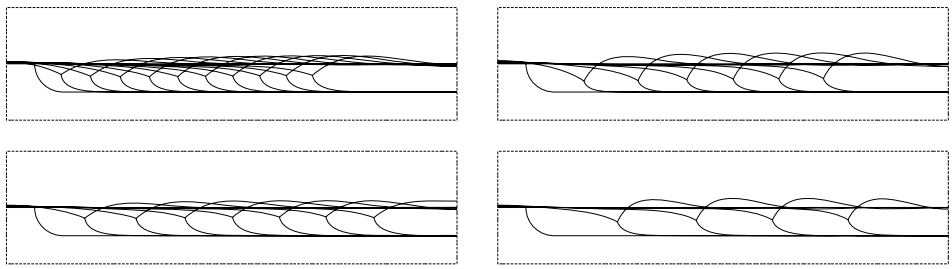


FIG. 28. *Traveling wave solutions* $\vec{X}^m(t)$ *at times* $t = 0, 0.1, \ldots, T$, *with* $T = 1,\ 0.6,\ 0.7$, *and* 0.4, *respectively. Surface energies are* $\sigma_3 = 1$ *(left) and* $\sigma_3 = \frac{3}{2}$ *(right), with mobilities* $\mu_3 = 1$ *(top) and* $\mu_3 = 2$ *(bottom).*

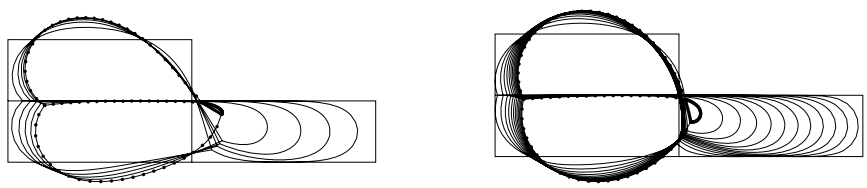


FIG. 29. $\vec{X}(t)$ *for* $t = 0, 0.1, \ldots, T$ *for a quadruple junction with* $\lambda = \frac{3}{2}$ *(left) and* $\lambda = \frac{2}{3}$ *(right).*

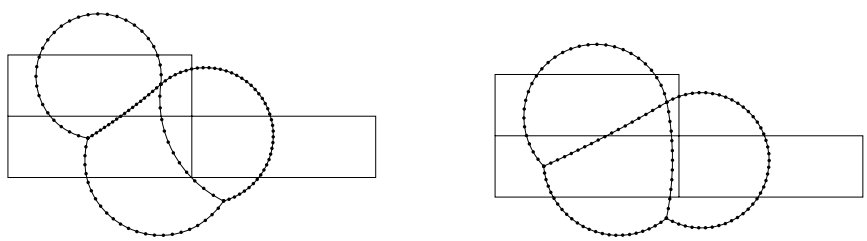


FIG. 30. $\vec{X}(t)$ *for* $t = 0, T$ *for a quadruple junction with* $\lambda = \frac{3}{2}$ *(left) and* $\lambda = \frac{2}{3}$ *(right).*

means that the two left enclosed areas represent phase B, while the rest of the plane is made up of phase A. Moreover, the lower two enclosed areas represent grain I, with the rest of the domain being grain II. Then we have motion by surface diffusion for the interfaces Γ_1 and Γ_3 separating the two phases, and mean curvature flow for the interfaces Γ_2 and Γ_4 between the two grains. The surface energies were chosen to be $\sigma_{SD} = 1$ for the curves moving by surface diffusion and $\sigma_{MC} = \lambda$ for the remaining curves. Two evolutions for $\lambda = \frac{3}{2}$ and $\lambda = \frac{2}{3}$, where the initial curves create a collection of $3:1$ rectangles, can be seen in Figure 29. The discretization parameters were $N = 128$, $\tau = 10^{-3}$, and $T = 0.49$ and $T = 1.2$, respectively. In both evolutions we can see that the two curves enclosing grain I in phase A eventually shrink to a point. The change of topology needed to continue the evolution is beyond the scope of our direct parametric approximation. We also give the corresponding numerical steady states for the pure surface diffusion flow version of this experiment in Figure 30. The discretization parameters were $N = 128$, $\tau = 10^{-3}$, and $T = 8$.

Finally, we remark that, on recalling $\sigma_{SD} = 1$ and $\sigma_{MC} = \lambda$, the balance of forces equation (1.16) becomes $(\vec{\tau}_1 + \vec{\tau}_3) + \lambda\,(\vec{\tau}_2 + \vec{\tau}_4) = \vec{0}$. It follows from this that $\theta_1 = \theta_3$ and $\theta_2 = \theta_4$, where $\theta_i := \sphericalangle(\vec{\tau}_i, \vec{\tau}_{i+1})$, $i = 1 \to 3$, and $\theta_4 := \sphericalangle(\vec{\tau}_4, \vec{\tau}_1)$, as adding the four vectors $\vec{\tau}_1, \lambda\,\vec{\tau}_2, \vec{\tau}_3, \lambda\,\vec{\tau}_4$ in the plane leads to a 4-polygon with two opposite sides having length 1 and two opposite sides having length λ. This implies that the 4-polygon is a parallelogram, and hence opposite angles in the parallelogram have to be equal. This fact can be seen clearly in the numerical computations. For example, on labeling the curves anticlockwise starting from the top most one, the angles observed in Figure 30 at the quadruple junction are $\theta = (133.1, 44.6, 133.7, 48.6)$ and $\theta = (107.5, 69.6, 105.8, 77.1)$, respectively; i.e., the stated angle condition is approximately satisfied.

4. Conclusions. We have presented a fully practical finite element approximation for the (combined) motion by mean curvature and motion by surface diffusion of curves in $\mathbb{R}^2$, as well as for other related second and fourth order geometric evolution equations. Our scheme can handle both triple and quadruple junction points between different curves, and intersections of curves with a fixed external boundary $\partial\Omega$. To our knowledge, this is the first such scheme in the literature. Moreover, the presented scheme intrinsically moves the vertices tangentially along the curves, so that no artificial redistribution of vertices is necessary in practice. Finally, we note that a generalization of the presented method to geometric flows of two-dimensional hypersurfaces in $\mathbb{R}^3$ will be considered in the forthcoming manuscript [5]. Moreover, extending our schemes for curves and hypersurfaces to include the case of fully anisotropic surface energies is the subject of our ongoing research in this area.

REFERENCES

[1] L. Alvarez, F. Guichard, P.-L. Lions, and J.-M. Morel, *Axioms and fundamental equations of image processing*, Arch. Ration. Mech. Anal., 123 (1993), pp. 199–257.

[2] S. Angenent, G. Sapiro, and A. Tannenbaum, *On the affine heat equation for nonconvex curves*, J. Amer. Math. Soc., 11 (1998), pp. 601–634.

[3] E. Bänsch, P. Morin, and R. H. Nochetto, *A finite element method for surface diffusion: The parametric case*, J. Comput. Phys., 203 (2005), pp. 321–343.

[4] J. W. Barrett, H. Garcke, and R. Nürnberg, *Finite element approximation of a phase field model for multicomponent surface diffusion*, 2007, in preparation.

[5] J. W. Barrett, H. Garcke, and R. Nürnberg, *On the Parametric Finite Element Approximation of Evolving Hypersurfaces in* $\mathbb{R}^3$, Preprint No. 18/2006, University Regensburg, Regensburg, Germany, 2006.

[6] J. W. Barrett, H. Garcke, and R. Nürnberg, *A parametric finite element method for fourth order geometric evolution equations*, J. Comput. Phys., 222 (2007), pp. 441–467.

[7] J. W. Barrett, H. Garcke, and R. Nürnberg, *A phase field model for electromigration of intergranular voids*, Interfaces Free Bound., to appear.

[8] L. Bronsard and F. Reitich, *On three-phase boundary motion and the singular limit of a vector-valued Ginzburg–Landau equation*, Arch. Ration. Mech. Anal., 124 (1993), pp. 355–379.

[9] L. Bronsard and B. T. R. Wetton, *A numerical method for tracking curve networks moving with curvature motion*, J. Comput. Phys., 120 (1995), pp. 66–87.

[10] J. W. Cahn, *Stability, microstructural evolution, grain growth, and coarsening in a two-dimensional two-phase microstructure*, Acta Metall., 39 (1991), pp. 2189–2199.

[11] J. W. Cahn, E. A. Holm, and D. J. Srolovitz, *Modeling microstructural evolution in two-dimensional two-phase microstructures*, Mater. Sci. Forum, 94–96 (1992), pp. 141–158.

[12] J. W. Cahn and A. Novick-Cohen, *Evolution equations for phase separation and ordering in binary alloys*, J. Statist. Phys., 76 (1994), pp. 877–909.

[13] K. Deckelnick and G. Dziuk, *On the approximation of the curve shortening flow*, in Calculus of Variations, Applications, and Computations (Pont-à-Mousson, 1994), Pitman Res. Notes Math. 326, Longman Scientific and Technical, Harlow, UK, 1995, pp. 100–108.

[14] K. Deckelnick, G. Dziuk, and C. M. Elliott, *Computation of geometric partial differential equations and mean curvature flow*, Acta Numer., 14 (2005), pp. 139–232.
[15] K. Deckelnick and C. M. Elliott, *Finite element error bounds for a curve shrinking with prescribed normal contact to a fixed boundary*, IMA J. Numer. Anal., 18 (1998), pp. 635–654.
[16] G. Dziuk, *An algorithm for evolutionary surfaces*, Numer. Math., 58 (1991), pp. 603–611.
[17] G. Dziuk, *Convergence of a semidiscrete scheme for the curve shortening flow*, Math. Models Methods Appl. Sci., 4 (1994), pp. 589–606.
[18] G. Dziuk, *Numerical schemes for the mean curvature flow of graphs*, in Variations of Domain and Free-Boundary Problems in Solid Mechanics (Paris, 1997), Solid Mech. Appl. 66, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999, pp. 63–70.
[19] G. Dziuk, E. Kuwert, and R. Schätzle, *Evolution of elastic curves in $\mathbb{R}^n$: Existence and computation*, SIAM J. Math. Anal., 33 (2002), pp. 1228–1245.
[20] C. M. Elliott and H. Garcke, *Existence results for diffusive surface motion laws*, Adv. Math. Sci. Appl., 7 (1997), pp. 465–488.
[21] J. Escher, Y. Giga, and K. Ito, *On a limiting motion and self-intersections of curves moved by the intermediate surface diffusion flow*, Nonlinear Anal., 47 (2001), pp. 3717–3728.
[22] X. Feng and A. Prohl, *Finite Element Approximations of the Inverse Mean Curvature Flow Arising from the General Relativity*, FIM Preprint, ETH Zurich, Zurich, Switzerland, 2005.
[23] R. Finn, *Equilibrium Capillary Surfaces*, Grundlehren Math. Wiss. 284, Springer, New York, 1986.
[24] P. Freudenberger, *Tripel- und Quadrupelpunkte im Krümmungsfluß*, Master's thesis, Mathematics Department, University of Bonn, Bonn, Germany, 1997.
[25] H. Garcke, K. Ito, and Y. Kohsaka, *Linearized stability analysis of stationary solutions for surface diffusion with boundary conditions*, SIAM J. Math. Anal., 36 (2005), pp. 1031–1056.
[26] H. Garcke, B. Nestler, and B. Stinner, *A diffuse interface model for alloys with multiple components and phases*, SIAM J. Appl. Math., 64 (2004), pp. 775–799.
[27] H. Garcke, B. Nestler, and B. Stoth, *A multiphase field concept: Numerical simulations of moving phase boundaries and multiple junctions*, SIAM J. Appl. Math., 60 (1999), pp. 295–315.
[28] H. Garcke and A. Novick-Cohen, *A singular limit for a system of degenerate Cahn–Hilliard equations*, Adv. Differential Equations, 5 (2000), pp. 401–434.
[29] R. Geroch, *Energy extraction*, Ann. New York Acad. Sci., 224 (1973), pp. 108–117.
[30] M. R. Hestenes, *Pseudoinverses and conjugate gradients*, Comm. ACM, 18 (1975), pp. 40–43.
[31] G. Huisken and T. Ilmanen, *The inverse mean curvature flow and the Riemannian Penrose inequality*, J. Differential Geom., 59 (2001), pp. 353–437.
[32] P. S. Jang, *On the positive energy conjecture*, J. Math. Phys., 17 (1976), pp. 141–145.
[33] J. Kanel, A. Novick-Cohen, and A. Vilenkin, *Coupled surface and grain boundary motion: Nonclassical traveling wave solutions*, Adv. Differential Equations, 9 (2004), pp. 299–327.
[34] B. Merriman, J. K. Bence, and S. J. Osher, *Motion of multiple functions: A level set approach*, J. Comput. Phys., 112 (1994), pp. 334–363.
[35] K. Mikula and D. Ševčovič, *Evolution of plane curves driven by a nonlinear function of curvature and anisotropy*, SIAM J. Appl. Math., 61 (2001), pp. 1473–1501.
[36] W. W. Mullins, *The effect of thermal grooving on grain boundary motion*, Acta Metall., 6 (1958), pp. 414–427.
[37] R. Neubauer, *Ein Finiteelementeansatz für Krümmungsfluß von unter Tripelpunktbedingungen verbundenen Kurven*, Master's thesis, Mathematics Department, University of Bonn, Bonn, Germany, 2002.
[38] E. Pasch, *Numerische Verfahren zur Berrechnung von Krümmungsflüssen*, Ph.D. thesis, Mathematics Department, University of Tübingen, Tübingen, Germany, 1998.
[39] J. Rubinstein, P. Sternberg, and J. B. Keller, *Fast reaction, slow diffusion, and curve shortening*, SIAM J. Appl. Math., 49 (1989), pp. 116–133.
[40] G. Sapiro and A. Tannenbaum, *On affine plane curve evolution*, J. Funct. Anal., 119 (1994), pp. 79–120.
[41] K. A. Smith, F. J. Solis, and D. L. Chopp, *A projection method for motion of triple junctions by levels sets*, Interfaces Free Bound., 4 (2002), pp. 263–276.
[42] J. E. Taylor and J. W. Cahn, *Linking anisotropic sharp and diffuse surface motion laws via gradient flows*, J. Statist. Phys., 77 (1994), pp. 183–197.
[43] B. Thaddey, *Numerik für die Evolution von Kurven mit Tripelpunkt*, Master's thesis, Mathematics Department, University of Freiburg, Freiburg, Germany, 1999.
[44] H.-K. Zhao, B. Merriman, S. Osher, and L. Wang, *Capturing the behavior of bubbles and drops using the variational level set approach*, J. Comput. Phys., 143 (1998), pp. 495–518.

SIAM J. SCI. COMPUT.
Vol. 29, No. 3, pp. 1042–1072
© 2007 Society for Industrial and Applied Mathematics

NEW ACYCLIC AND STAR COLORING ALGORITHMS WITH APPLICATION TO COMPUTING HESSIANS*

ASSEFAW H. GEBREMEDHIN†, ARIJIT TARAFDAR†, FREDRIK MANNE‡, AND ALEX POTHEN†

Abstract. Acyclic and star coloring problems are specialized vertex coloring problems that arise in the efficient computation of Hessians using automatic differentiation or finite differencing, when both sparsity and symmetry are exploited. We present an algorithmic paradigm for finding heuristic solutions for these two NP-hard problems. The underlying common technique is the exploitation of the structure of two-colored induced subgraphs. For a graph G on n vertices and m edges, the time complexity of our star coloring algorithm is $O(n\bar{d}_2)$, where $\bar{d}_k$, a generalization of vertex degree, denotes the average number of distinct paths of length at most k edges starting at a vertex in G. The time complexity of our acyclic coloring algorithm is larger by a multiplicative factor involving the inverse of Ackermann's function. The space complexity of both algorithms is $O(m)$. To the best of our knowledge, our work is the first practical algorithm for the acyclic coloring problem. For the star coloring problem, our algorithm uses fewer colors and is considerably faster than a previously known $O(n\bar{d}_3)$-time algorithm. Computational results from experiments on various large-size test graphs demonstrate that the algorithms are fast and produce highly effective solutions. The use of these algorithms in Hessian computation is expected to reduce overall runtime drastically.

Key words. acyclic coloring, star coloring, computation of Hessians, automatic differentiation, finite differences

AMS subject classifications. 05C15, 05C85, 68R10

DOI. 10.1137/050639879

1. Introduction.

The cost involved in computing a Jacobian or a Hessian matrix using automatic differentiation or finite differencing is proportional to the number of columns in the matrix. When the target derivative matrix A is sparse, huge savings in runtime can be attained by first computing a matrix B, where each column of B is a sum of several structurally "independent" columns of A, and then recovering the nonzero elements in A from B. Thus, an important component of the efficient computation of a sparse derivative matrix, whose sparsity pattern is known a priori, is the problem of partitioning the columns of the matrix into the fewest groups, each consisting of structurally independent columns.

The appropriate notion of structural independence—and the corresponding partitioning problem—depends on whether the derivative matrix to be computed is or , and on whether the nonzero entries are to be recovered from the compressed matrix or via . Conceptually, direct recovery corresponds to solving a diagonal system of equations, and recovery via substitution corresponds to solving a set of simple triangular systems of equations. Several previous studies have shown that these matrix partitioning problems can be modeled and effectively solved using specialized graph coloring problems and their algorithms [9, 12, 13, 14, 22, 27]. See our earlier work [17] for a comprehensive review.

*Received by the editors September 8, 2005; accepted for publication (in revised form) November 7, 2006; published electronically May 10, 2007. This work was supported by the U.S. National Science Foundation through grants ACI 0203722 and CCF 0515218, by DOE grant DE-FC02-01ER25476 and by subcontract B 533520 from the Lawrence Livermore National Laboratory.

http://www.siam.org/journals/sisc/29-3/63987.html

†Department of Computer Science and Center for Computational Science, Old Dominion University, Norfolk, VA 23529-0162 (assefaw@cs.odu.edu, tarafdar@cs.odu.edu, pothen@cs.odu.edu).

‡Department of Informatics, University of Bergen, N-5020 Bergen, Norway (fredrikm@ii.uib.no).

This paper is concerned with the design and implementation of effective algorithms for [illegible] and [illegible] coloring, models for efficient computation of Hessians. In a star coloring, adjacent vertices receive distinct colors (distance-1 coloring), and every [illegible] (P_4) uses at least three colors. In an acyclic coloring, adjacent vertices receive distinct colors, and every [illegible] uses at least three colors. Coleman and Moré [13] showed that star coloring models the partitioning problem that occurs in the computation of a Hessian via a direct method; Coleman and Cai [9] showed that the corresponding model in a substitution-based computation is acyclic coloring. Both problems, whose objectives are to minimize the number of colors used, are known to be NP-hard [9, 13]. As we show in section 2, these problems are also hard to approximate.

Star coloring as a model for direct Hessian computation was preceded by two less accurate models: the [illegible] 2 coloring model due to McCormick [27] and the [illegible] coloring model due to Powell and Toint [28]. Similarly, acyclic coloring as a model for Hessian computation via substitution was preceded by a less accurate model, [illegible] coloring, due to Coleman and Moré [13]. These models and the interrelationships amongst them are discussed in detail in section 5.

In a previous work [17], we suggested a simple heuristic algorithm for the star coloring problem. In each step of that algorithm, the color of a vertex is determined by examining every P_4 starting at the vertex. The time complexity of the algorithm for a graph on n vertices is $O(n\bar{d}_3)$, where $\bar{d}_k$ denotes the average number of paths of length at most k edges starting at a vertex. A similar $O(n\bar{d}_2)$-time algorithm for restricted star coloring, originally due to Powell and Toint [28], exists. However, when used for solving the star coloring problem, the latter algorithm uses significantly more colors than the $O(n\bar{d}_3)$-time algorithm mentioned earlier.

Although Coleman and Cai [9] identified acyclic coloring as the model for computing a Hessian via a substitution method, the algorithm they suggested is an algorithm for triangular coloring. In their approach, first a graph G' is constructed by adding edges to the input graph G, and then a distance-1 coloring algorithm is applied on the graph G'. A similar approach for triangular coloring was taken earlier by Coleman and Moré [13]. An optimal solution for the triangular coloring problem is not necessarily an optimal solution for the acyclic coloring problem. Moreover, working with a denser graph G' is undesirable for large-scale problems, since the approach may require excessive storage space and lead to memory overflow. As for the acyclic coloring problem, we are not aware of any previous practically relevant algorithm.

In this paper, we present new algorithms for both the acyclic and the star coloring problems. The common paradigm underlying the two algorithms is the exploitation of the structure of [illegible]. In the acyclic coloring case, every subgraph induced by any two color classes is a collection of [illegible]; the corresponding structure in the star coloring case is a collection of [illegible]. In both cases, the two-colored induced subgraphs [illegible] the edge set of the input graph. Our algorithms are based on these observations.

The algorithms presented in this paper are [illegible]—they progressively extend a partial coloring by processing one vertex at a time. In each step, a vertex is assigned the [illegible] (assuming colors are positive integers) allowable color with respect to the current partial coloring, and the color assigned to the vertex remains unchanged during the course of the algorithm. Greedy coloring heuristics stand in contrast to [illegible] heuristics, where the color assigned to a vertex may change several times during the course of an algorithm in an attempt to ultimately

reduce the number of colors used. Greedy coloring heuristics are in general fast, which makes them attractive approaches for solving subproblems in scientific computing applications, such as the case in this paper. An important issue in greedy coloring algorithms is the order in which vertices are processed. We consider several effective ordering techniques in this paper.

A key component of our new algorithms is the efficient management of two-colored induced subgraphs. For the acyclic coloring case, the disjoint-set data structure is used for this purpose; for the star coloring case, a simpler data structure is shown to be sufficient. In addition, a few intricate techniques which yet use simple data structures are employed to avoid two-colored cycles in the acyclic coloring case, and two-colored P_4's in the star coloring case. The time complexity of the star coloring algorithm is $O(n\bar{d}_2)$, and that of the acyclic coloring algorithm is larger by a multiplicative factor involving the inverse of Ackermann's function. The space complexity of both algorithms is $O(m)$, where m denotes the number of edges in a graph. The acyclic coloring algorithm will be presented (in section 3) before the star coloring algorithm (in section 4) to reflect that the former in an algorithmic sense corresponds to the more general case.

We have implemented the new acyclic and star coloring algorithms presented in this paper as well as several previously known algorithms for related coloring problems; the latter algorithms will be discussed in section 6. Results from experiments carried out on various large-size test graphs show that the new algorithms are highly effective, both in terms of the number of colors used and in practical runtime. On the test graphs used, the number of colors used by the acyclic coloring algorithm is observed to be very close to (within a factor of two) the [illegible] of a graph—the number of colors used by a greedy distance-1 coloring algorithm that employs a [illegible] vertex ordering. The coloring number is an important graph parameter with close relationship to graph notions such as degeneracy, maximal core, and arboricity. The runtime of the acyclic coloring algorithm is observed to be nearly the same as that of a greedy distance-2 coloring algorithm. In comparison with our implementation of an algorithm for triangular coloring—the previously known approach for Hessian computation via a substitution method—the acyclic coloring algorithm runs faster, uses fewer colors, and requires less memory and storage space. In agreement with time complexity analyses, our new star coloring algorithm is observed to be significantly faster than our previous $O(n\bar{d}_3)$-time algorithm—it offers a relative speedup as high as ten on the test graphs used. Moreover, the new algorithm uses slightly fewer colors. In comparison with the restricted star coloring algorithm of Powell and Toint [28], which is of the same time complexity as the new star coloring algorithm proposed here, our algorithm significantly reduces the number of colors used. In terms of observed runtime, the new star coloring algorithm is slower than the restricted star coloring algorithm by only a small factor, typically two.

2. Background.

2.1. Preliminary concepts and notations. Throughout this paper, we consider simple, undirected graphs without loops or parallel edges. Two classes of (sub)graphs that we have need for are trees and stars. A [illegible] is a connected graph without any cycle, and a [illegible] is a collection of trees. A [illegible] is a complete bipartite graph in which one of the vertex sets consists of a single vertex. In a star with at least two edges, we call the vertex at which the edges meet the [illegible] of the star.

We denote an edge whose endpoints are the vertices v and w by vw. Two distinct vertices v and w are [illegible] k [illegible] if a shortest path connecting them consists

TABLE 2.1
Characterization of subgraphs induced by a pair of color classes. The coloring variants are listed in an increasing order of restrictiveness; correspondingly, the two-colored induced subgraphs get increasingly less connected as one goes from top to bottom in the table.

Coloring variant	A two-colored induced subgraph
Distance-1 coloring	A bipartite graph
Acyclic coloring	A collection of trees
Star coloring	A collection of stars
Distance-2 coloring	A distance-2 matching

of at most k edges. We denote the set of distance-k neighbors of a vertex v, excluding the vertex v itself, by $N_k(v)$. As usual, the [illegible] of a vertex is the number of edges incident on it. We extend this notion and define the [illegible] k of a vertex v, denoted by $d_k(v)$, to be the number of distinct paths of length at most k edges starting at v. (Two paths are [illegible] if they differ in at least one edge.) We denote the average degree-k in a graph by $\overline{d}_k$. Note that degree-1 is synonymous with degree, and that the degree-2 of a vertex v is equal to the number of edges that are incident either on v or on vertices adjacent to v. Note also that in general $d_k(v) \geq |N_k(v)|$.

A [illegible] of a graph $G = (V, E)$ is a mapping $\phi : V \to \{1, 2, \ldots, p\}$ such that whenever vertices v and w are distance-k neighbors, $\phi(v) \neq \phi(w)$. By definition, two distance-k neighboring vertices are distance-k' neighbors for $k' > k$, and a distance-k coloring is a distance-k'' coloring for $k'' < k$. The k[illegible] of a graph $G = (V, E)$ is the graph $G^k = (V, F)$, where $vw \in F$ whenever vertices v and w are distance-k neighbors in G. A mapping ϕ is a distance-k coloring of a graph G if and only if ϕ is a distance-1 coloring of the graph G^k. A [illegible] of a graph is a distance-1 coloring with the additional requirement that [illegible]. An [illegible] of a graph is a distance-1 coloring with the further restriction that [illegible]. In a distance-k, a star, an acyclic, or any other coloring variant on a graph, a [illegible] is a set of vertices that have received the same color.

The names star and acyclic coloring are due to the structures of subgraphs induced by any two color classes. In an acyclically colored graph, a subgraph induced by any two color classes is clearly a collection of trees, and thus acyclic. Similarly, in a star-colored graph, a subgraph induced by any two color classes is a collection of stars. It is easy to see that a two-colored connected induced subgraph here ought to be a star, for otherwise there would exist a two-colored path on four vertices, violating a condition of star coloring. These structures are used in the design of the acyclic and star coloring algorithms to be presented in the next two sections. Although not so useful for algorithm design, the structures of two-colored induced subgraphs in the cases of distance-1 and distance-2 coloring are also interesting. A coloring of a graph is a distance-1 coloring if and only if every two-colored induced subgraph is a bipartite graph. In the case of a distance-2 coloring, every two-colored induced subgraph is a set of edges in which a pair of edges is apart by a path of length at least two edges in the original graph, a structure appropriately called a [illegible] 2[illegible]. These observations on two-colored induced subgraphs are summarized in Table 2.1.

2.2. Complexity results. The objective of an optimization problem associated with each of the coloring variants we have introduced is to minimize the number of colors used. Lin and Skiena [26] proved that the distance-k graph coloring problem is NP-hard for every fixed integer $k \geq 1$. Coleman and Moré [13] showed that the

star coloring problem is NP-hard even if the graph is bipartite. The acyclic coloring problem was proven to be NP-hard by Coleman and Cai [9].

The distance-1 coloring problem is also known to be hard to approximate: for a graph on n vertices, there exists $\delta > 0$ such that approximating distance-1 coloring within $O(n^{\delta})$ is NP-hard (Corollary 10.1 in [5]). We use this to prove similar inapproximability results for acyclic and star coloring. We first introduce some notation. The [illegible] of a graph G, denoted by $\chi_k(G)$, is the least number of colors required to distance-k color G. The acyclic and star chromatic numbers are defined analogously, and are denoted by χ_a and χ_s, respectively.

THEOREM 2.1. [illegible] $\epsilon > 0$ [illegible] $\chi_a(G)$ [illegible] G [illegible] $O(n^{\epsilon})$ [illegible] n [illegible] G [illegible]. The reduction is from distance-1 coloring, and it is a small modification of the reduction for proving the NP-completeness of acyclic coloring in Coleman and Cai [9].

Assume that we are given a graph $G = (V, E)$ which we are required to distance-1 color; from G we create a new graph $H = (W, F)$ for acyclic coloring as follows. Let Δ denote the maximum degree of a vertex in G. Each edge $uv \in E$ is replaced by a subgraph (a gadget) in H consisting of vertices u', v', and Δ new vertices w_1, w_2, $\ldots$, w_{Δ}; and edges join vertices u' and v' to every one of the Δ vertices w_i.

We claim that G has a distance-1 coloring using p colors if and only if H has an acyclic coloring using p colors, where $3 \leq p \leq \Delta$.

Suppose G has a distance-1 coloring with p colors. In any edge $uv \in E$, the vertices u and v receive distinct colors. In the graph H, assign u' the same color as u, and v' the same color as v. Since each vertex w_i is adjacent only to the pair of vertices u' and v', it can be assigned any one of the $p-2$ colors distinct from the two colors that have been used for u' and v'; note that the quantity $p-2$ is positive (there exists a color for the vertex w_i) since p is assumed to be at least three. The coloring of H obtained by "processing" every edge in G in this fashion is acyclic. To see this, notice that every cycle in H contains one u' vertex, one v' vertex, and at least one w_i vertex from the gadget we created for some edge $uv \in E$, and these three vertices receive distinct colors in H. Hence every cycle in H uses at least three colors, and one direction of the claim follows.

Now consider an acyclic coloring of H with p colors, again where $3 \leq p \leq \Delta$.

We claim that the vertices u' and v' corresponding to an edge $uv \in E$ receive distinct colors. Suppose not. Then every pair of vertices w_i and w_j in this gadget would have to be colored with distinct colors, or else there would be a cycle u', w_i, v', w_j, u' which would be colored with two colors. But then this subgraph of H would need $\Delta + 1$ colors, contradicting the choice of p.

Thus in every gadget the vertices u' and v' receive distinct colors in the acyclic coloring, and these same colors can be assigned to the vertices u and v to obtain a distance-1 coloring of G using at most p colors.

There exists $\delta > 0$ such that approximating distance-1 coloring within $O(n^{\delta})$ is NP-hard [5], where $n = |V|$, the number of vertices in G. Let N denote the number of vertices in H. Then $N = n + \Delta|E| = O(n + n^3)$. Thus it follows that there exists an $\epsilon > 0$ such that approximating acyclic coloring within $O(N^{\epsilon})$ is NP-hard. □

Under stronger complexity conjectures ($NP \neq ZPP$, the class of problems with polynomial time zero-error randomized algorithms), we can prove that it is hard to approximate acyclic coloring within $O(n^{1/3-\epsilon})$, using a technique similar to that in

Agnarsson and Halldórsson [3].

We can also prove inapproximability results for star coloring, using the same gadget used in the acyclic coloring proof. We omit the details.

THEOREM 2.2. [illegible] $\epsilon > 0$ [illegible] $\chi_s(G)$ [illegible] G [illegible] $O(n^{\epsilon})$

2.3. Related work. Some of the coloring problems introduced in section 2.1 have been studied by other authors, from an application or a pure graph-theoretic point of view. Krumke, Marathe, and Ravi [24] studied the role of distance-2 coloring in channel assignment problems in radio networks. Kumar et al. [25] have undertaken a related study on scheduling problems in wireless networks. In these two studies, several approximation and distributed algorithms for distance-2 coloring of disk and other special classes of graphs have been suggested. Balakrishnan et al. [6] showed that the problem of maximizing concurrent transmissions in certain ad-hoc wireless networks can be modeled as a maximum cardinality distance-2 matching problem, a problem related to distance-2 coloring as discussed at the end of section 2.1. The distance-2 matching problem was first studied and shown to be NP-hard by Stockmeyer and Vazirani [29].

Agnarsson and Halldórsson [2] have studied distance-k coloring of planar graphs, and a similar study of chordal graphs is available in Agnarsson, Greenlaw, and Halldórsson [1]. Acyclic colorings were first defined and studied by Grünbaum [21], who asked whether the acyclic chromatic number $\chi_a(G)$ of a graph G satisfies the inequality $\chi_a(G) \leq \Delta + 1$. A negative answer to this question was given by Erdös, who proved probabilistically the existence of graphs where $\chi_a(G) \geq \Delta^{4/3-\epsilon}$ [23]. Borodin [7] proved that every planar graph can be acyclically colored using at most five colors. Star coloring has recently been studied by Albertson et al. [4], who showed that every acyclic coloring that uses q colors can be refined to a star coloring with at most $2q^2 - q$ colors. These authors also prove that planar graphs have star colorings with at most 20 colors, and they exhibit an example in which 10 colors are required. For more pointers to theoretical works on distance-k, acyclic, and star coloring, see the monograph by Jensen and Toft [23] and our earlier work [17]. The focus of this paper is on practical, fast, suboptimal algorithms for acyclic and star coloring.

3. The new acyclic coloring algorithm.

3.1. Overview. Suppose a subset $U \subset V$ of the vertices of a graph $G = (V, E)$ has been colored satisfying the two requirements of an acyclic coloring. As discussed in section 2.1, every subgraph of G induced by a set of vertices assigned any two colors is a forest. Furthermore, every edge in G both of whose endpoints are in the set U of colored vertices belongs to a unique tree (in a two-colored forest). Therefore, the union of all possible two-colored forests induced by the vertices in the set U is a collection of edge-disjoint trees. Our algorithm maintains such a collection of trees efficiently and extends the partial coloring by processing one vertex at a time.

We use the illustration in Figure 3.1 to provide an overview of the algorithm. The graph shown at the left is the input graph G which has been partially colored, and v is the only vertex that remains to be colored. The right half of Figure 3.1 shows two-colored forests induced by the rest of the vertices of the graph. The input graph currently uses four colors, red, blue, green, and yellow (to ease black-and-white viewing, each colored vertex in the figure has been labeled with a letter reflecting the color used). Thus there could exist at most six two-colored induced forests. Out of these, the subgraph induced by green and yellow vertices is a graph whose edge set

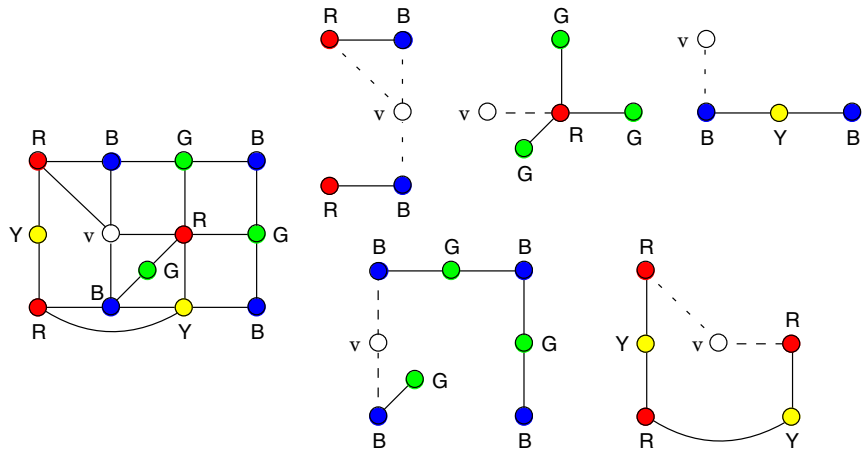


FIG. 3.1. *Left: A partial acyclic coloring of a graph; vertex v is the only vertex that remains to be colored. Every other vertex has been assigned one of the four colors red, blue, green, or yellow; the label on each vertex reflects the color used. Right: All of the nonempty, two-colored induced subgraphs; each subgraph is a forest.*

is empty. The remaining five forests are depicted in the figure. Each of the red-blue and blue-green forests consists of two trees, while each of the other forests consists of a single tree. In the example, all of the two-colored trees are incident on the vertex v.

The task of choosing a valid color for the vertex v can be divided into two natural parts. The first part involves excluding the colors of each of the distance-1 neighbors of v from its set of allowable colors. In our illustration, this results in the exclusion of the colors red and blue from the allowable set of colors for v. The second part involves ensuring that a color chosen for the vertex v does not lead to a two-colored cycle in the graph G. Since the vertex v (and, in general, any vertex in a graph) could be part of many, arbitrarily long cycles, the second part is more difficult than the first, which is confined to the distance-1 neighborhood of v in G.

The key point in our algorithm is the detection of a potential two-colored cycle that involves the vertex v in an efficient way. In particular, the cycle detection is done efficiently by examining two-colored trees incident on v, a task accomplished by visiting the distance-2 (and not any further) neighborhood of v once. Due to the distance-1 coloring requirement in an acyclic coloring, an odd cycle would necessarily use at least three colors. Thus in avoiding two-colored cycles, we will be concerned only with even cycles.

A two-colored cycle that involves the vertex v can arise only if the following two conditions are satisfied:

1. the vertex v is adjacent to , vertices in a two-colored tree, and
2. these adjacent vertices are of the , color.

Note that condition 2 restricts the concern to only even cycles, for if the vertex v is adjacent to two vertices with different colors in a two-colored tree, the corresponding cycle involving v is of odd length and thus uses three colors, posing no concern. For example, in Figure 3.1, though vertex v is adjacent to two vertices in the upper red-blue tree, v will not induce a cycle in that tree since the adjacent vertices have distinct colors.

Now let F_v denote a collection of two-colored trees that are incident on the vertex

v and satisfy the aforementioned two conditions. For each tree $T_v \in F_v$, let c_1 denote the color of the vertices adjacent to the vertex v, and let c_2 be the other color used in the tree T_v. Then, for each tree T_v, we exclude the color c_2 from the allowable set of colors for the vertex v, for otherwise v might be assigned the color c_2, creating a two-colored cycle involving v. In Figure 3.1 this process would lead to the exclusion of the color yellow from the allowable set of colors for the vertex v, for otherwise a two-colored cycle could arise in the red-yellow tree.

When both the first and the second tasks discussed above are completed for the vertex v, we know exactly the colors that cannot be assigned to v, and thus we can choose the smallest allowable color for v.

To complete the overview of the algorithm, we now need to update the collection of two-colored trees to reflect the fact that the vertex v has been colored. The update can be performed in two stages. In the first stage, edges vw that connect the vertex v and an already colored vertex w are grouped into two-colored stars. In the second stage, these stars are merged with existing two-colored trees as needed. A point that should be noted here is that the second stage of the update may cause previously disconnected trees in a two-colored forest to be connected. For example, if the vertex v in Figure 3.1 receives the color green, then the two (currently disconnected) trees in the green-blue forest need to be merged. Just as the task of choosing a color for the vertex v, we will show that the task of updating the collection of two-colored trees after v is assigned a color can be accomplished by visiting the distance-2 neighborhood of v once.

3.2. Detailed description. For an efficient implementation of the algorithm sketched in section 3.1, we use the disjoint-set data structure to maintain the collection of two-colored trees. For a discussion of the disjoint-set data structure, see, e.g., [15]. A set in our context corresponds to a two-colored tree, and an element of a set corresponds to an edge. The disjoint-set data structure supports three operations: MAKESET, FIND, and UNION. Operation MAKESET(e) creates a new set consisting of the element e. Operation FIND(e) returns the representative element of the set to which element e belongs, and operation UNION(e_1, e_2) merges the two sets $S(e_1)$ and $S(e_2)$ to which elements e_1 and e_2 belong.

Further, in our implementation of the sketched algorithm, we use the two arrays color and forbiddenColors for coloring purposes. The vertex-indexed array color is used to store the colors assigned to vertices. In particular, color$[u] = c$ indicates that the vertex u has been assigned the color c, a value that once set remains unchanged during the course of the algorithm. The color-indexed array forbiddenColors is used to mark the colors that are forbidden to a specific vertex. In particular, forbiddenColors$[c] = u$ indicates that the color c is impermissible for the vertex u; otherwise the color c is a candidate color for the vertex u. Unlike the array color, the value stored at a particular index in the array forbiddenColors varies during the course of the algorithm, an issue that will be clear in a moment.

3.2.1. The main routine. The driver routine in our acyclic coloring algorithm is outlined in Algorithm 3.1 and the current section is devoted to its discussion. The subroutines called by Algorithm 3.1 will be presented in section 3.2.2.

Leaving the initialization of various data structures in line 2 aside for a moment, the body of the outermost for-loop of Algorithm 3.1 consists of two parts. The first part (lines 4–10) is concerned with finding a color for the vertex v, the current vertex in the iteration over the vertex set V. The second part (lines 11–16) deals with updating the collection of two-colored trees incident on the vertex v.

ALGORITHM 3.1. Input: a graph G. Effect: finds an acyclic coloring of G.

```
1: procedure AcyclicColoring(G = (V, E))
2:     Initialize data structures
3:     for each v ∈ V do
4:         for each colored w ∈ N1(v) do
5:             forbiddenColors[color[w]] ← v
6:         for each colored w ∈ N1(v) do
7:             for each colored x ∈ N1(w) do
8:                 if forbiddenColors[color[x]] ≠ v then
9:                     preventCycle(v, w, x)
10:        color[v] ← min{i > 0, forbiddenColors[i] ≠ v}
11:        for each colored w ∈ N1(v) do      ▷ grow two-colored stars around the vertex v
12:            growStar(v, w)
13:        for each colored w ∈ N1(v) do
14:            for each colored x ∈ N1(w), x ≠ v do
15:                if color[x] = color[v] then
16:                    mergeTrees(v, w, x)   ▷ merge trees T1 ∋ vw and T2 ∋ wx if T1 ≠ T2
```

Pursuant to the first requirement of acyclic coloring, the for-loop in lines 4–5 marks the colors of the distance-1 neighbors of the vertex v as forbidden colors to v. The subsequent for-loop in lines 6–9 forbids additional colors that can result in two-colored cycles involving the vertex v, to enforce the second requirement of acyclic coloring. Here, a subset of the two-colored trees incident on the vertex v—through edges wx, where w is an already colored vertex adjacent to v and x is an already colored vertex adjacent to w—is examined. The routine preventCycle (to be discussed in section 3.2.2) checks whether the vertex v is connected to the two-colored tree containing the edge wx via two edges. If this turns out to be the case, the routine forbids the color of the vertex x from being a candidate color for the vertex v by updating the global array forbiddenColors. The if-test in line 8 ensures that a two-colored tree incident on the vertex v is investigated only if there is a potential for a two-colored cycle to arise in the same tree; recall that the upper red-blue tree in Figure 3.1 need not be examined while coloring the vertex v since it cannot lead to a two-colored cycle involving v. The if-test also avoids an unnecessary computation: once a color c in a two-colored tree T is forbidden to the vertex v to prevent a potential two-colored cycle C (that involves T and v), then the tree T need not be investigated any further in the determination of a color for the vertex v, for the existence of any other cycle C' (that also involves T and v) would only result in the exclusion of the same color c.

Once all the colors that are not allowed for the vertex v are marked in the array forbiddenColors, the array is scanned sequentially in search of the first index in which a value other than v is stored. This corresponds to the smallest allowable color for the vertex v and is chosen as the color for v (line 10). The array forbiddenColors is initialized at the beginning of the algorithm with some value outside the set V. This array can be used without being reinitialized while coloring another vertex, since in each iteration a unique vertex is colored (i.e., the identity of a vertex serves as a "time-stamp").

The remainder of Algorithm 3.1 is concerned with creating and updating two-colored trees incident on the vertex v. This

task is done in two stages, reflected by the two for-loops in lines 11–12 and lines 13–16. In the first stage (the first for-loop), two-colored trees consisting of edges incident on the vertex v are grown. More precisely, the structures grown are two-colored stars, and the hub of each star is the vertex v. To this end, for every edge vw that connects the vertex v with an already colored vertex w, (1) a new two-colored star consisting only of the edge vw is created, and (2) if there is already a similarly two-colored star S with one or more edges of the form vw' (hence $\mathsf{color}[w] = \mathsf{color}[w']$), then the newly created star (edge vw) is merged with S. Both of these items are achieved by the call to the routine GROWSTAR in line 12; the details of this routine will be discussed in section 3.2.2.

At the end of the for-loop in lines 11–12, every edge vw connecting v and an already colored vertex w has been placed in a star in the right two-colored forest. There may now be a need for another level of merges in a forest, constituting the second stage of the data structure update. Such merges involve stars grown at the current iteration, and trees grown in iterations prior to the coloring of v. The piece of code in lines 13–16 determines whether there is a need for such merges, and if so, updates the disjoint-set data structure appropriately. In particular, for each colored vertex w adjacent to v, and each colored vertex x adjacent to w with $\mathsf{color}[x] = \mathsf{color}[v]$, the routine MERGETREES is called in line 16 with the arguments v, w, and x. The routine checks whether the tree that contains the edge vw is the same as the tree that contains the edge wx; if the two trees turn out to be different, then they are merged.

To illustrate the points discussed in the previous two paragraphs without revealing the details of the routines GROWSTAR and MERGETREES, consider the example in Figure 3.1. Suppose the vertex v has been assigned the color green. At the end of the for-loop in lines 11–12, two stars are grown around the vertex v. These are the star (path) B–v–B and the star R–v–R, where B and R stand for a blue and a red neighbor of the vertex v, respectively. In the subsequent for-loop, the star R–v–R is merged with the only tree in the current red-green forest. In the same loop, the star B–v–B is first merged with one of the two disconnected trees in the current blue-green forest, and then the resulting tree is further merged with the other tree. As a result, the blue-green forest now becomes a single tree (to be more precise, a path on eight vertices).

3.2.2. The subroutines. Let us now take an inside look at the routines PREVENTCYCLE, GROWSTAR, and MERGETREES. In the discussion below, we will refer to the actions performed on the vertex v in lines 4–16 of Algorithm 3.1 as v.

The routine PREVENTCYCLE (see Algorithm 3.2) takes three vertices v, w, and x as input parameters: v is the vertex currently being processed, w is a colored vertex adjacent to v, and x is a colored vertex adjacent to w. The primary task of the routine is to forbid the color of the vertex x from being chosen as a color for the vertex v, if it could lead to a two-colored cycle in the input graph. This would happen if the vertex v is incident on the two-colored tree to which the edge wx belongs via two edges. An auxiliary task of the routine is to update an associated data structure.

To determine whether or not a vertex v is connected to a two-colored tree via two edges in an efficient way, we each two-colored tree containing an edge wx incident on the vertex v. While doing so, if a tree is marked twice, then we know that the vertex v is incident on the tree via two edges. We use an edge-indexed array called $\mathsf{firstVisitToTree}$ for this purpose. An index of the array $\mathsf{firstVisitToTree}$ is the

ALGORITHM 3.2. Input: the vertex v being colored, a colored neighbor w of v, and a colored neighbor x of w. Associated data structure: firstVisitToTree$[e] = (p, q)$ indicates that firstVisitToTree$[e]$ was last set when processing vertex p and at that time the tree represented by the edge e was *first* visited through the edge pq. Effect: (1) updates the array firstVisitToTree, or (2) excludes color$[x]$ from the allowable set of colors for v if it could otherwise lead to a two-colored cycle involving v.

```
1: procedure PREVENTCYCLE(v, w, x)
2:     e ← FIND(wx)        ▷ edge wx belongs to the 2-colored tree T represented by edge e
3:     (p, q) ← firstVisitToTree[e]
4:     if p ≠ v then                ▷ T is being visited from vertex v for the first time
5:         firstVisitToTree[e] ← (v, w)
6:     else if q ≠ w then           ▷ T is connected to vertex v via at least two edges
7:         forbiddenColors[color[x]] ← v
```

ID of the representative edge of a two-colored tree. A vertex pair (p, q) is stored at index e of the array to indicate that firstVisitToTree$[e]$ was last set when processing the vertex p, and at that time the two-colored tree whose representative edge is e was [illegible] visited through the edge pq. In firstVisitToTree and other similar data structures used elsewhere in this paper, the word [illegible] refers to the context in which the neighborhood of the vertex p being processed is visited. The array firstVisitToTree is globally declared and initialized at the beginning of Algorithm 3.1 with a pair of values not corresponding to any vertex. As in the array forbiddenColors, the "marker" vertex p in the array firstVisitToTree serves as a time-stamp. Hence, this array need not be reinitialized while processing another vertex.

The routine PREVENTCYCLE begins by getting the two-colored tree T to which the edge wx belongs. Let e be the representative edge of T returned by the FIND operation. In line 3, the vertex pair (p, q) stored at index e of the array firstVisitToTree is retrieved. If $p \neq v$, then the implication is that the tree T is being visited from the vertex v for the first time. To reflect this, the vertex pair (v, w) is stored at firstVisitToTree$[e]$. If, on the other hand, $p = v$, then there are two possibilities, corresponding to the cases $q = w$ and $q \neq w$. The case $q = w$ implies that the tree T has previously been visited from the vertex v, but the visit was through the same edge vw. This occurs if the vertex w is adjacent to at least two vertices (besides the vertex v) in the tree T, and does not cause the color of the vertex x to be a forbidden color for the vertex v. The second case, $q \neq w$, implies that the tree T has previously been visited from the vertex v through an edge other than vw. In this case, setting color$[v]$ = color$[x]$ would create a two-colored cycle, and therefore color$[x]$ is forbidden from being a color for the vertex v.

[illegible] As discussed earlier, the first stage of updating the collection of two-colored trees incident on the vertex v being processed involves growing two-colored stars around the vertex v. This is achieved by the calls GROWSTAR(v, w) made in line 12 of Algorithm 3.1 for each colored vertex w adjacent to the vertex v.

To achieve the desired merging of edges into two-colored stars in an efficient manner, we maintain a color-indexed array, firstNeighbor, in which we store pairs of vertices. The array is used to determine whether or not a particular color is encountered for the first time while visiting the distance-1 neighborhood of a particular vertex. Specifically, we let firstNeighbor$[c] = (p, q)$ to indicate that while processing the vertex p, the vertex q was the first-to-be-encountered neighbor having the color

ALGORITHM 3.3. Input: the vertex v being processed and a colored neighbor w of v. Associated data structure: $\mathsf{firstNeighbor}[c] = (p, q)$ indicates that the vertex q is the first-to-be-encountered neighbor of the vertex p having the color c. Effect: (1) creates a new two-colored tree T_{vw} consisting only of the edge vw, and (2) either updates the array firstNeighbor or merges the tree T_{vw} with a two-colored star being grown around v.

```
1: procedure GROWSTAR(v, w)
2:     MAKESET(vw)                          ▷ create a new tree T_vw consisting only of edge vw
3:     (p, q) ← firstNeighbor[color[w]]
4:     if p ≠ v then          ▷ a neighbor of v with color color[w] encountered for 1st time
5:         firstNeighbor[color[w]] ← (v, w)
6:     else                              ▷ merge T_vw with a two-colored star being grown around v
7:         e1 ← FIND(vw); e2 ← FIND(pq)
8:         UNION(e1, e2)
```

ALGORITHM 3.4. Input: the vertex v being processed, a colored neighbor w of v, and a neighbor x of w having the same color as v. Effect: merges the tree containing the edge vw with the tree containing the edge wx, if the two trees are distinct.

```
1: procedure MERGETREES(v, w, x)
2:     e1 ← FIND(vw); e2 ← FIND(wx)
3:     if e1 ≠ e2 then
4:         UNION(e1, e2)
```

c. As the array firstVisitToTree discussed earlier, the array firstNeighbor is globally declared and initialized in Algorithm 3.1, and can be used without reinitialization while iterating through the vertex set V.

The routine GROWSTAR (Algorithm 3.3) begins by creating a two-colored tree (star) consisting only of the edge vw, a task accomplished by the call MAKESET(vw). The routine then retrieves the vertex pair (p, q) stored at the index $\mathsf{color}[w]$ of the array firstNeighbor. If $p \neq v$, then the implication is that the color used by the vertex w is encountered for the first time while visiting the vertices adjacent to the vertex v. To signify this, the vertex pair (v, w) is stored at the index $\mathsf{color}[w]$ of the array firstNeighbor. Otherwise, if $p = v$, then the implication is that a vertex q (distinct from w) that has the same color as the vertex w has already been encountered while visiting the neighbors of the vertex v. In this case, the edge vw—the two-colored star just created—is merged with the (growing) two-colored star containing the edge pq.

The second stage of the update involves merging the two-colored stars just grown with existing two-colored trees as needed. This is achieved by the calls to MERGETREES made in line 16 of Algorithm 3.1. Each such call is made with the arguments v, w, and x, where v is the vertex currently being processed, w is a colored vertex adjacent to v, and x is a colored vertex adjacent to w with $\mathsf{color}[x] = \mathsf{color}[v]$. The routine MERGETREES (Algorithm 3.4) first queries for the representative edges e_1 and e_2 of the two-colored trees T_1 and T_2 to which the edges vw and wx, respectively, belong. If $e_1 \neq e_2$, then it means that the two trees T_1 and T_2 are distinct, and therefore they need to be merged. Otherwise no further action needs to be taken.

3.3. Complexity. The outermost for-loop in Algorithm 3.1 consists of $n = |V|$ iterations. In each iteration, the two visits to the distance-2 neighborhood of the vertex v being processed, i.e., the for-loops in lines 6–9 and 13–16, constitute the dominant part of the computation. On each visited neighbor, a constant number

of operations is performed. Thus, disregarding the time spent on updating and querying the disjoint-set data structure, the time complexity of the algorithm is $O(\sum_{v \in V} d_2(v)) = O(n\bar{d}_2)$.

The disjoint-set data structure in our context contains at most m distinct elements, the edges of G. Thus there can be at most $m-1$ UNION calls and m MAKESET calls. Further, there can be at most $m' = 2n \cdot \bar{d}_1 + 3n \cdot \bar{d}_2 \leq 5n \cdot \bar{d}_2$ FIND calls, since the routine FIND is called once in the procedure PREVENTCYCLE and twice in each of the procedures GROWSTAR and MERGETREES. In our implementation of the disjoint-set data structure, we use [illegible] for the FIND operation and [illegible] for the UNION operation to achieve the best runtime performance. When these techniques are used, the average time spent on each such operation is at most $\alpha(m', m)$, where α is the inverse of Ackermann's function [15]. Thus the overall time complexity of our acyclic coloring algorithm is $O(n\bar{d}_2 \cdot \alpha(m', m))$. Clearly, the space complexity of the algorithm is $O(m)$.

4. The new star coloring algorithm.

4.1. Overview. Our star coloring algorithm is based on the same idea as the acyclic coloring algorithm presented in section 3. Instead of managing a collection of edge-disjoint two-colored trees, we now manage a collection of edge-disjoint two-colored stars. Recall that the [illegible] of a star is the vertex at which the edges meet. The key difference between the acyclic and the star coloring algorithms stems from the following fact: An edge may be added to a tree at any vertex of the tree, whereas the only way an edge may be added to a star is at the hub of the star. A consequence of this fact is that the addition of an edge to a star does not lead to the merging of two existing stars. Hence, a simpler data structure than disjoint-set suffices. We use a one-dimensional array that maps edges to stars, and a similar array that maps stars to hubs, to manage the collection of two-colored stars.

Figure 4.1 provides an overview of the algorithm. Part (a) of the figure shows the input graph G where every vertex except v has been colored consistent with the requirements of a star coloring. The illustration uses the six colors red, blue, green, yellow, pink, and black (Bk). Parts (b) through (f) show a subset of the two-colored stars with nonempty edge sets that are incident on v.

Since a star coloring is also a distance-1 coloring, the set of colors used by vertices adjacent to the vertex v is impermissible for v. Enforcing this requirement is straightforward. In addition, any color that leads to a two-colored path involving v and three other vertices is impermissible for v. There are exactly two cases that need to be considered to identify and forbid a color of this type.

- [illegible] 1. If vertex v has two adjacent vertices w and x of the [illegible] color, then the color of every vertex adjacent to w, and the color of every vertex adjacent to x should be excluded from the allowable set of colors for v.
- [illegible] 2. If vertex v has exactly one adjacent vertex w of color c, and w belongs to a star S containing at least two edges but is not the hub of S, then the color used by the hub of S should be disallowed from being a candidate color for v.

Parts (b) and (c) of Figure 4.1 illustrate Case 1; neither the color blue nor the color green can be assigned to the vertex v. Similarly, parts (d) and (e) illustrate Case 2; the colors green and pink used by the hubs of the respective stars are not allowed for the vertex v. As illustrated in part (f), if the star S in a situation similar to Case 2 has only one edge, then it does not impose any additional color restriction on the vertex v. We say that the hub of such a star consisting of a single edge is [illegible].

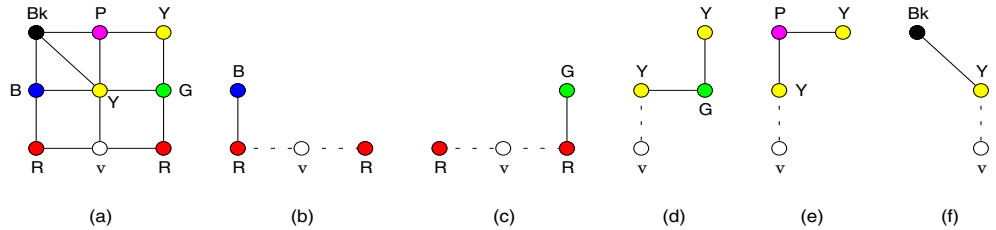


FIG. 4.1. (a): *A partial star coloring of a graph; vertex v is the only vertex that remains to be colored.* (b)–(f): *A subset of the two-colored induced subgraphs incident on the vertex v; each subgraph is a collection of stars. The illustration uses the six colors red, blue, green, yellow, pink, and black (Bk).*

The hub is defined later if and when another edge joins the star, in which case the point of incidence between the two edges becomes the hub.

Like the acyclic coloring algorithm, our star coloring algorithm iterates over the vertex set V in some order, coloring one vertex at a time. In the step where a vertex v is colored, the color used by each adjacent vertex and every additional color identified by checking Cases 1 and 2 is excluded from the allowable set of colors for v. Then the smallest permissible color is chosen and assigned to the vertex v. After this, every edge vw connecting the vertex v and an already colored vertex w is placed in the appropriate two-colored star. In particular, if an edge vw is incident on some (color$[v]$,color$[w]$)-star S, then the edge vw joins the star S; otherwise, the edge vw forms a star on its own with an undefined hub. Unlike the acyclic coloring algorithm in which the addition of an edge to a two-colored tree may result in the merging of two existing trees, the addition of an edge to a two-colored star does not lead to the merging of two existing stars.

4.2. Detailed description. To maintain the collection of two-colored stars effectively, we use an edge-indexed array, star, that maps edges to stars, and a star-indexed array, hub, that maps stars to hubs. In particular, star$[vw] = s$ signifies that the edge vw belongs to the star s, and hub$[s] = u$ indicates that the hub of the star s is the vertex u. We work with the two separate arrays, star and hub, in order to distinguish between stars consisting of at least two edges (having defined hubs) and stars consisting of a single edge (having undefined hubs). In addition, we make use of the two arrays color and forbiddenColors in just the same way as in section 3.

Our star coloring algorithm is outlined in Algorithm 4.1. The body of the outer for-loop has two parts. The first part, the piece of code in lines 4–16, is concerned with finding a color for the vertex v, the current vertex in the iteration over the vertex set. The second part deals with updating the collection of two-colored stars incident on the vertex v, and is achieved by the call to UPDATESTARS in line 17. Hereafter, we shall refer to the actions performed on the vertex v in lines 4–17 as [illegible] v.

[illegible] The for-loop in lines 4–15 of Algorithm 4.1 determines the colors that cannot be assigned to the vertex v. In line 5, the color used by a vertex w adjacent to the vertex v is marked as a forbidden color for v, since a star coloring is also a distance-1 coloring. To determine and forbid additional colors that lead to two-colored paths involving v and three other vertices, Cases 1 and 2 outlined earlier need to be checked. To detect Case 1 in an efficient manner, we use the color-indexed array firstNeighbor which stores pairs of vertices: firstNeighbor$[c] = (p, q)$ indicates

ALGORITHM 4.1. Input: a graph G. Associated data structure: $\mathsf{firstNeighbor}[c] = (p, q)$ indicates that the vertex q is the first-to-be-encountered neighbor of the vertex p having the color c; $\mathsf{treated}[y] = x$ indicates that the color of every vertex adjacent to the vertex y is already forbidden from being a color for the vertex x. Effect: finds a star coloring of G.

```
1:  procedure StarColoring(G = (V, E))
2:      Initialize data structures
3:      for each v ∈ V do
4:          for each colored w ∈ N1(v) do
5:              forbiddenColors[color[w]] ← v
6:              (p, q) ← firstNeighbor[color[w]]
7:              if p = v then                                   ▷ Case 1
8:                  if treated[q] ≠ v then
9:                      TREAT(v, q)                  ▷ forbid colors of neighbors of q
10:                     TREAT(v, w)                  ▷ forbid colors of neighbors of w
11:             else
12:                 firstNeighbor[color[w]] ← (v, w)
13:                 for each colored x ∈ N1(w), x ≠ v do
14:                     if x = hub[star[wx]] then                ▷ potential Case 2
15:                         forbiddenColors[color[x]] ← v
16:         color[v] ← min{i > 0, forbiddenColors[i] ≠ v}
17:         updateStars(v)
```

ALGORITHM 4.2. Input: the vertex v being processed and a colored vertex w adjacent to v. Effect: excludes the colors used by vertices adjacent to w from the allowable set of colors for v, and updates the array treated to reflect this.

```
1:  procedure TREAT(v, w)
2:      for each colored x ∈ N1(w) do
3:          forbiddenColors[color[x]] ← v
4:      treated[w] ← v
```

that when processing the vertex p, the vertex q was the first neighbor with color c to be encountered. (An array with the same name was used for a similar purpose in the acyclic coloring algorithm discussed in the previous section.) In line 6, a lookup in the table firstNeighbor at the index $\mathsf{color}[w]$ is made and the stored value is retrieved in the vertex pair (p, q).

If $p = v$, then the implication is that a vertex which is distinct from, but has the same color as, the vertex w—the vertex q—has already been encountered while visiting the distance-1 neighbors of the vertex v. This clearly corresponds to Case 1. The appropriate action to be taken is to forbid the colors used by the vertices adjacent to q and the colors used by the vertices adjacent to w. The calls to the subroutine TREAT (outlined in Algorithm 4.2) in lines 9 and 10 do precisely that. In connection with this case, we use the vertex-indexed array treated in which we store vertices; $\mathsf{treated}[y] = x$ indicates that the color of every vertex adjacent to the vertex y has already been marked as a forbidden color for the vertex x. The if-test in line 8 is performed to avoid unnecessary computation in the case where $\mathsf{color}[w]$ is encountered for a third time or later.

If, on the other hand, $p \neq v$, then the implication is that the color used by the vertex w is encountered for the first time while visiting the distance-1 neighborhood

ALGORITHM 4.3. Input: the vertex v being processed. Effect: adds edges vw, where w is colored, to appropriate two-colored stars.

```
1: procedure updateStars(v)
2:     for each colored w ∈ N1(v) do
3:         if ∃x ∈ N1(w) where x ≠ v and color[x] = color[v] then      ▷ vw, wx ∈ E
4:             hub[star[wx]] ← w                                      ▷ this may already be true
5:             star[vw] ← star[wx]
6:         else
7:             (p, q) ← firstNeighbor[color[w]]
8:             if (p = v) and (q ≠ w) then                            ▷ vw, vq ∈ E ∧ color[w] = color[q]
9:                 hub[star[vq]] ← v                                  ▷ this may already be true
10:                star[vw] ← star[vq]
11:            else                                                   ▷ vw forms a new star
12:                starID ← starID + 1
13:                star[vw] ← starID
```

of the vertex v. This information is recorded in line 12 for future use. The piece of code in the subsequent for-loop takes a proactive measure in anticipation of Case 2. In particular, the loop iterates over edges wx that belong to existing two-colored stars. For each star to which an edge wx belongs, if the star has a defined hub and the hub is the vertex x, then the color of x is forbidden from being a color for the vertex v (line 14). Potentially, this corresponds to Case 2; if the vertex v is later found to have a neighbor vertex $w' \neq w$ with the same color as w, then it reduces to Case 1. Recall that only stars containing at least two edges have defined hubs; for a star with only one edge the test in line 14 will fail and therefore will not impose an additional color restriction (cf. Figure 4.1 (f)).

At the end of the for-loop in lines 4–15, every color that is not allowed to be a color for the vertex v is marked in the array forbiddenColors. The smallest permissible color is then chosen and assigned to the vertex v in line 16.

As with the arrays used in section 3, in the arrays firstNeighbor and treated the marker vertex p serves as a time-stamp; it is used to determine whether the information stored in the data structures concerns the vertex currently being processed. These data structures are initialized at the beginning of Algorithm 4.1, and no reinitialization is required in the course of iteration over the vertex set.

The call to the routine updateStars (outlined in Algorithm 4.3) is made at line 17 to update the collection of two-colored stars to reflect the fact that the vertex v has been assigned a color. In Algorithm 4.3, each edge vw that joins the vertex v with an already colored vertex w is placed in an existing or a new two-colored star depending on the colors of the vertex v and its distance-2 neighbors. The edge vw joins an existing star if (1) there exists an edge wx and color$[v]$ = color$[x]$, or (2) there exists an edge vq and color$[w]$ = color$[q]$. The first case can be detected by looking at the distance-1 neighborhood of the vertex w; for the second case, the information in the array firstNeighbor, which is set in Algorithm 4.1, is utilized. Clearly, in the first case the vertex w becomes the hub of the resulting star (if it was not already so), and in the second case the vertex v becomes the hub (again, if it was not already so). If neither of cases (1) or (2) turns out to be positive, then the edge vw forms a new two-colored star on its own with an undefined hub. Notice that the definitions of the hubs in lines 4 and 9 are necessary, since the edges wx and

vq could be single-edge-stars with undefined hubs at the time the routine is called.

4.3. Complexity. Given a graph on n vertices and m edges, the computational work involved in each of the n iterations of the outer for-loop in Algorithm 4.1 is proportional to $d_2(v)$. Hence the overall time complexity of our star coloring algorithm is $O(n\bar{d}_2)$. Its space complexity is clearly $O(m)$.

5. Coloring models in Hessian computation. We have experimentally compared the new algorithms for acyclic and star coloring presented in this paper with previously suggested algorithms for these and other related coloring problems that occur in the computation of sparse Hessians. In this section, we briefly review such coloring models and discuss their interrelationships. In section 6, we will discuss corresponding greedy algorithms.

One step in an efficient computation of a sparse derivative matrix A using automatic differentiation (or finite differencing) is to use structural information about A to partition its n columns into p disjoint groups, with p as small as possible. This step can be posed as a problem of seeking an $n \times p$ matrix S whose (j,k) entry is defined as follows:

$$s_{jk} = \begin{cases} 1 & \text{if column } a_j \text{ belongs to group } k, \\ 0 & \text{otherwise.} \end{cases} \tag{5.1}$$

Since the matrix S specified by (5.1) corresponds to a partitioning of the columns of the matrix A, in every row r of the matrix S there exists exactly one column c in which the entry s_{rc} is equal to one. In the automatic differentiation literature, there exist approaches that use a seed matrix where a row-sum is not necessarily equal to one [19]. Such approaches are not partitioning-based as they allow a column of A to be placed in more than one group. In this paper we consider only seed matrices that define a partition.

The specific criteria used to define a seed matrix S depend on whether the entries of the matrix A are to be recovered from the compressed representation AS (by conceptually solving a diagonal system of equations) or via $A'S$ (by conceptually solving a set of triangular systems of equations). The criteria also depend on whether the derivative matrix A is Jacobian (nonsymmetric) or Hessian (symmetric). In general, a seed matrix S suitable for a direct method requires a larger p compared to one suitable for a substitution method. On the other hand, the recovery of entries via substitution incurs an additional, but affordable, computational cost.

5.1. Models for direct methods. Curtis, Powell, and Reid [16] observed that a structurally orthogonal partition of a Jacobian matrix A—a partition of the columns of A in which no two columns in a group share a nonzero at the same row index—gives a seed matrix S where the entries of A can be directly recovered from the compressed representation AS. Obviously, a structurally orthogonal partition could also be used in the context of computing a Hessian if one were to settle for not exploiting the available symmetry. Following this approach, McCormick [27] showed that a structurally orthogonal partition of a Hessian is equivalent to a star 2 of its adjacency graph.

The adjacency graph $G(A)$ of a Hessian A has a vertex for each column, and an edge joins column vertices a_i and a_j whenever the entry a_{ij}, $i \neq j$, is nonzero; the diagonal entries in A are assumed to be nonzero, and they are not explicitly represented by edges in $G(A)$. In the distance-2 coloring model for structurally orthogonal

partition as well as in other models discussed in this section, a set of vertices having the same color corresponds to a set of columns that belong to the same group in the induced partition. In other words, each column of the seed matrix corresponds to a color class.

Powell and Toint [28] were the first to suggest symmetry-exploiting partitions for both direct and substitution-based methods for computing Hessians. When translated to a coloring ϕ of the adjacency graph, the partition Powell and Toint suggested for a direct Hessian computation requires that (1) ϕ be a distance-1 coloring, and (2) in every path on three vertices $P_3 = v, w, x$, the vertices v and x receive different colors whenever $\phi(w) > \min\{\phi(v), \phi(x)\}$. We call a coloring that satisfies these two requirements a [illegible]. Notice that, unlike in a distance-2 coloring, in a restricted star coloring, the terminal vertices v and x in a path v, w, x are allowed to have the same color as long as the color of the middle vertex w is lower in value.

Coleman and Moré [13] generalized the approach of Powell and Toint and showed that a symmetrically orthogonal partition of a Hessian suffices for a direct recovery of its entries from a compressed representation. A partition of the columns of a Hessian matrix A is [illegible] if for every nonzero element a_{ij}, either (1) the group containing column a_j has no other column with a nonzero in row i, or (2) the group containing column a_i has no other column with a nonzero in row j. Coleman and Moré established that a symmetrically orthogonal partition of a Hessian is equivalent to a [illegible] of its adjacency graph.

5.2. Models for substitution methods. Based on the work of Powell and Toint [28], Coleman and Moré [13] found a coloring model for a partitioning of a Hessian matrix A that defines a seed matrix S such that the nonzero entries of A can be recovered from the compressed representation AS via a (restricted) substitution method. The recovery of the entries of the matrix A from the matrix AS requires solving a triangular system of equations, a system that heavily relies on the structure of A. Appropriately, Coleman and Moré called the model triangular coloring. Given a graph $G = (V, E)$, a mapping $\phi : V \to \{1, 2, \ldots, p\}$ is a [illegible] if there exists a vertex ordering π such that (1) ϕ is a distance-1 coloring, and (2) in every $P_3 = v, w, x$, the vertices v and x receive different colors whenever $\pi(w) > \max\{\pi(v), \pi(x)\}$. A vertex ordering π that is well suited for triangular coloring exists and will be discussed in section 5.3.

Triangular coloring exploits symmetry only to a limited extent. This limitation was later addressed by Coleman and Cai [9], who introduced a generalized notion of substitutable partitions for which they provided a simple graph-theoretic characterization. In particular, they proved that an [illegible] of the adjacency graph of a Hessian induces a substitutable partition of its columns. They also showed that, using an acyclic coloring, the nonzeros of the Hessian can be recovered from its compressed representation by considering [illegible] (groups of columns) at a time. The key difference between a partition induced by a triangular coloring and a partition induced by an acyclic coloring is the following. Triangular coloring induces a partition that defines one large, coupled triangular system of equations, whereas acyclic coloring induces a partition that defines several smaller, decoupled triangular systems of equations, each defined by a pair of color classes.

Table 5.1 summarizes the coloring models in Hessian computation discussed thus far. In the table, the coloring variants are listed in an increasing order of restrictiveness, with the least constrained variant at the top and the most constrained variant at the bottom. In the following subsection, we present results that make the interre-

TABLE 5.1

*Overview of coloring models used in Hessian computation. In each case, the adjacency graph $G = (V, E)$ of the underlying Hessian is used, and the mapping $\phi : V \rightarrow \{1, 2, \ldots, p\}$ defines a coloring. In addition to the input graph G, the definition of triangular coloring assumes the existence of a vertex ordering π. *Powell and Toint [28] stated these models using their equivalent matrix partitioning counterparts; the translations into graph colorings are due to Coleman and Moré [13].*

Name	Coloring conditions	Computation modeled
Acyclic	1. $vw \in E \Rightarrow \phi(v) \neq \phi(w)$	substitution
	2. every cycle uses ≥ 3 colors	Coleman and Cai [9]
Triangular	1. $vw \in E \Rightarrow \phi(v) \neq \phi(w)$	restricted substitution
	2. in every $P_3 = v, w, x$:	Powell and Toint [28]*
	$\pi(w) > \max\{\pi(v), \pi(x)\} \Rightarrow \phi(v) \neq \phi(x)$	Coleman and Moré [13]
Star	1. $vw \in E \Rightarrow \phi(v) \neq \phi(w)$	direct
	2. every P_4 uses ≥ 3 colors	Coleman and Moré [13]
Restricted	1. $vw \in E \Rightarrow \phi(v) \neq \phi(w)$	restricted direct
star	2. in every $P_3 = v, w, x$:	Powell and Toint [28]*
	$\phi(w) > \min\{\phi(v), \phi(x)\} \Rightarrow \phi(v) \neq \phi(x)$	Coleman and Moré [13]
Distance-2	1. $vw \in E \Rightarrow \phi(v) \neq \phi(w)$	direct (ignores symm.)
	2. $P_3 = v, w, x \in G \Rightarrow \phi(v) \neq \phi(x)$	McCormick [27]

lationships among these variations more precise.

5.3. Interrelationships. Recall that we denote specialized chromatic numbers by χ with an appropriate subscript. For example, $\chi_a(G)$ denotes the acyclic chromatic number of G. In Lemma 5.1, we relate the acyclic, star, and restricted star chromatic numbers of a graph. Similarly, in Lemma 5.2 we show the relationship among the acyclic, triangular, and restricted star chromatic numbers of a graph.

LEMMA 5.1. [illegible] G $\chi_a(G) \leq \chi_s(G) \leq \chi_{rs}(G)$

[illegible]. Let ϕ be a star coloring of a graph G. Observe that a cycle in G on three vertices uses three colors, since ϕ is also a distance-1 coloring of G. Moreover, a cycle on at least four vertices contains a P_4 and therefore uses at least three colors. Hence ϕ is an acyclic coloring of G, and the first inequality follows.

For the second inequality, we show that if ϕ is a restricted star coloring of G, then ϕ is also a star coloring of G. Let $P = v, w, x$ be a P_3 in G. Since ϕ is a restricted star coloring, $\phi(v) \neq \phi(w)$ and $\phi(w) \neq \phi(x)$. Furthermore, either $\phi(v) \neq \phi(x)$ or $\phi(v) = \phi(x) > \phi(w)$ holds. In the former case, in any P_4 in G that contains the path P, adjacent vertices have different colors, and the P_4 uses at least three colors, satisfying the requirements of a star coloring. To see that the latter case also fulfills the requirements of a star coloring, consider any $P_4 = v, w, x, y$ in G that extends the path P in one of its ends. (A similar argument would hold for a path u, v, w, x that extends P in the other end.) Again, since ϕ is a restricted star coloring, $\phi(y) \neq \phi(x)$. Moreover, the relationship $\phi(y) \neq \phi(w)$ holds, since otherwise the relationship $\phi(x) > \phi(w) = \phi(y)$ in the path w, x, y would hold, contradicting the fact that ϕ is a restricted star coloring. Hence, in the path v, w, x, y, adjacent vertices have different colors, and at least three colors are used. Therefore ϕ is a star coloring of G. □

LEMMA 5.2. [illegible] G $\chi_a(G) \leq \chi_t(G) \leq \chi_{rs}(G)$

[illegible]. Suppose ϕ is a triangular coloring of a graph G for a vertex ordering π. We show that ϕ is then an acyclic coloring of G. Consider any cycle C in G, and let

z be the vertex in C with the [illegible] index in the ordering π. Since C is a cycle, there exists a path $P = u, z, v$ in C for some vertices u and v. Further, since ϕ is a triangular coloring, $\phi(u) \neq \phi(v)$ by choice of z. Since ϕ is also a distance-1 coloring, both of these colors are distinct from $\phi(z)$. Hence, the cycle C uses at least three colors, and therefore ϕ is an acyclic coloring of G.

We now show the second inequality. Suppose ϕ is a restricted star coloring of a graph G on n vertices. Consider a vertex ordering $\pi = v_1, v_2, \ldots, v_n$ such that the sequence $\{\phi(v_i)\}$ is nondecreasing. In other words, the ordering π is such that vertices in the same color class are listed consecutively, and the color classes in turn are listed in increasing order. Consider any path $P = v, w, x$ on three vertices in G. Since ϕ is a restricted star coloring, a pair of adjacent vertices in the path P have different colors. There are two possibilities regarding the terminal vertices v and x in P: either $\phi(v) \neq \phi(x)$ or $\phi(v) = \phi(x) > \phi(w)$. In the former case, the path P trivially satisfies the requirements of a triangular coloring. Given the ordering π as defined in the proof, the path P in the latter case also satisfies the requirements of a triangular coloring, since $\pi(w) < \min\{\pi(v), \pi(x)\}$. Hence, the coloring ϕ and the ordering π together define a triangular coloring of G. □

The inequalities in Lemmas 5.1 and 5.2 can easily be extended in both ends. Clearly, $\chi_1(G) \leq \chi_a(G)$ holds, since an acyclic coloring is also a distance-1 coloring. In a distance-2 colored graph, the vertices in every P_3 use three distinct colors. Hence, a distance-2 coloring is also a restricted star coloring, implying the relationship $\chi_{rs}(G) \leq \chi_2(G)$. We summarize these relationships in Theorem 5.3.

THEOREM 5.3. [illegible] G [illegible]

$$\chi_1(G) \leq \chi_a(G) \leq \chi_t(G) \leq \chi_{rs}(G) \leq \chi_2(G) = \chi_1(G^2),$$

$$\chi_1(G) \leq \chi_a(G) \leq \chi_s(G) \leq \chi_{rs}(G) \leq \chi_2(G) = \chi_1(G^2).$$

The relationship between $\chi_t(G)$ and $\chi_s(G)$ is not clear. There are examples of star colorings where there does not exist a vertex ordering that makes a given star coloring a triangular coloring. An example would be a star coloring of a graph in which every vertex is the hub of a two-colored P_3. Conversely, there are examples of colorings that are triangular but not star. A simple example is a two-colored P_4 in which adjacent vertices have different colors. Thus the class $C_s(G)$ of star colorings of a graph G and the class $C_t(G)$ of triangular colorings of G are such that neither $C_t(G) \subseteq C_s(G)$ nor $C_s(G) \subseteq C_t(G)$ holds. However, the relationship $\chi_t(G) \leq \chi_s(G)$ might still hold. We leave settling the latter issue as an open problem.

The last equality in Theorem 5.3 follows from the fact that a distance-2 coloring of G is equivalent to a distance-1 coloring of the square graph G^2. In fact, most of the coloring variants on a graph $G = (V, E)$ listed in Table 5.1 can also be viewed as a distance-1 coloring of an appropriately defined "filled" graph $G' = (V, E \cup F)$.

When the original graph G is the adjacency graph of a Hessian, the filled graphs in the distance-2 and triangular coloring cases have interesting interpretations. The filled graph in the distance-2 coloring case (the square graph G^2) is the [illegible] of the underlying Hessian A. The column intersection graph of a matrix has a vertex for each column, and an edge between a pair of columns exists whenever the two columns share nonzero entries at some common row index. In the case of a triangular coloring for a given vertex ordering π, the filled graph G', in which each fill edge f is generated as per the specification

$$\forall P_3 = v, w, x \text{ in } G : \pi(w) > \max\{\pi(v), \pi(x)\} \Rightarrow f = vx, \tag{5.2}$$

is the column intersection graph of the [illegible] L_π of the permuted matrix P^TAP, where P is a permutation matrix defined by the ordering π. These relationships were observed and used by Coleman and Moré [13], who showed that a [illegible] (SL) vertex ordering as a choice for π is particularly well suited for triangular coloring.

Given a graph on n vertices, a vertex ordering $v_1, v_2, \ldots, v_n$ is an SL ordering if for all $1 \leq i \leq n$, the vertex v_i is a vertex with the [illegible] degree in the graph induced by vertices $v_1, \ldots, v_i$. Here is why an SL ordering is well suited for triangular coloring: Among the $n!$ possible orderings (permutations) π, an SL ordering minimizes the maximum number of nonzeros in a row in L_π. Thus, an SL ordering provides a lower bound on the triangular chromatic number, a result proven by Coleman and Moré [13].

Orthogonal to its role in triangular coloring, an SL ordering is also known to be one of the effective ordering techniques for reducing the number of colors used by a greedy coloring algorithm. The number of colors used by a greedy distance-1 coloring algorithm that employs an SL ordering to color a graph G is an important graph parameter. The parameter is known as the [illegible] $col(G)$ of G and is closely related to several other graph notions, including degeneracy, maximal core, and arboricity [17]. In our context, since an SL ordering can be computed in linear time in the number of edges, the coloring number is an easy-to-compute, nontrivial upperbound for the distance-1 chromatic number. As mentioned in the previous paragraph, the coloring number is also a lower bound for the triangular chromatic number. Theorem 5.4 summarizes these relationships.

THEOREM 5.4. [illegible] G $\chi_1(G) \leq col(G) \leq \chi_t(G)$

6. Previous algorithms. Coleman and Cai [9] accurately modeled the partitioning problem that occurs in substitution-based Hessian computation as the acyclic coloring problem. However, the algorithm they suggested is an algorithm for triangular coloring. Specifically, they suggested that a filled graph G' be explicitly constructed as described by the expression (5.2), using an SL vertex ordering for π, and then that G' be distance-1 colored. The same approach has been taken earlier by Coleman and Moré [13] and in the related software papers [10, 11]. However, as we will soon show in the current section, it is possible to achieve a triangular coloring of G by using a greedy algorithm that works directly on G. This section also discusses previously known greedy algorithms for star and distance-k coloring. All of the algorithms discussed in this section are included in our experiments.

In a previous work [17], we suggested an algorithm for star coloring that enforces the coloring requirements in a straightforward manner—by checking paths on four vertices. We refer to this algorithm here as NAIVESTARCOLORING (,). Powell and Toint [28] suggested an algorithm of the same spirit, formulated in the language of matrices, for the restricted star coloring problem. We will call this algorithm here RESTRICTEDSTARCOLORING (,).

Algorithms , and , are both greedy. In determining a set of forbidden colors for a vertex v, algorithm , consults the colors used by the distance-3 neighbors of v, whereas algorithm , checks the colors used by only the distance-2 neighbors. Once the set of forbidden colors is obtained, both algorithms choose the smallest color not included in the set as the color for the vertex v. In a similar fashion, one can design greedy algorithms for distance-1 coloring, distance-2 coloring, and triangular coloring for a given ordering π. We refer to such algorithms as DISTANCE1COLORING (1), DISTANCE2COLORING (2), and TRIANGULARCOLORING (π). The suffix in the

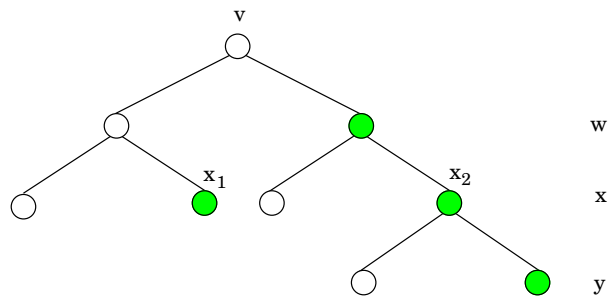


FIG. 6.1. *Visualizing a step in greedy coloring algorithms; v is the vertex currently being colored. Table 6.1 shows how the set of forbidden colors for v is determined in various cases.*

TABLE 6.1
Conditions under which a particular color is forbidden to v (see in conjunction with Figure 6.1). Here ϕ is a coloring, and π is a vertex ordering.

	w	x_1	x_2
$D1$	always		
$D2$	always	always	always
NS	always	always	only if $\phi(w) = \phi(y)$
RS	always	always	only if $\phi(w) > \phi(x)$
T-π	always	only if $\pi(w) > \max\{\pi(v), \pi(x)\}$	only if $\pi(w) > \max\{\pi(v), \pi(x)\}$

notation π signifies that a specific vertex ordering π is used by the algorithm.

We use Figure 6.1 in conjunction with Table 6.1 to illustrate how the set of forbidden colors for a vertex is determined by the various greedy algorithms in coloring a graph G on n vertices and m edges. The tree depicted in Figure 6.1 is not a subgraph of G, but rather a visualization of the classification of the neighborhood $N_k(v)$ of the vertex v being colored at the current step of the particular algorithm. The classification is based on whether a vertex in the set $N_k(v)$ is already colored or not. In the figure, the vertices of the graph G that are one, two, and three edges away from the vertex v are represented by the tree nodes at the levels w, x, and y, respectively. A shaded signifies already colored and an unshaded node signifies vertices not yet colored. The shaded nodes at level x are further labeled x_1 and x_2, to distinguish between the cases where a vertex w in a path v, w, x is already colored or not.

Table 6.1 summarizes the conditions under which a color used by a vertex in the scenario described in Figure 6.1 is forbidden to the vertex v for the various algorithms. Cases x_1 and x_2 correspond to distinct decisions in algorithms and . These two cases are indistinguishable in algorithms 2 and π and are irrelevant to algorithm 1. As column x_1 in Table 6.1 shows, in algorithms and , the color of vertex x in a path v, w, x where vertex w is not yet colored is immediately forbidden to the vertex v. This ensures that any extension v, w, x, y of the path v, w, x would end up using at least three colors, as desired, since in the step in which the vertex w is colored, the distance-1 coloring requirement guarantees that the vertex w gets a color distinct from the colors of vertices v and x. In contrast, in the case where vertex w is already colored (see column x_2 in the table), determining whether the color of vertex x should be forbidden to vertex v or not requires a further check. In algorithm , the check involves vertices w and y (note that y is three edges away from v). In

algorithm , the check involves vertices w and x. In each of the algorithms listed in Table 6.1, once the set of forbidden colors to the vertex v is determined, the smallest color not included in the set is chosen as the color for v.

The illustration should make it clear that the complexity of algorithm 1 is $O(n\bar{d}_1) = O(m)$, the complexity of each of algorithms 2, , , and π is $O(n\bar{d}_2)$, and the complexity of algorithm is $O(n\bar{d}_3)$.

7. Experimental results.

7.1. Setup. To demonstrate the performance of our new algorithms, we ran experiments on 29 graphs obtained from molecular dynamics applications [8, 30]. The test graphs were chosen because of their large size and irregular structure. The left half of Table 7.1 shows the number of vertices $|V|$, the number of edges $|E|$, the maximum degree Δ, and the average degree $\bar{d}_1$ in each test graph. The right half of the table shows certain computed values that we will discuss shortly. All of the algorithms used in our experiments were implemented in C++. The hardware used was a Linux machine equipped with Dual AMD Opteron Processor 244, 1.8 GHz, 2 GB memory, and 1 GB cache. The operating system was RHEL 3 and the compiler was GCC.

7.2. Algorithms compared and orderings used. In addition to the new acyclic and star coloring algorithms presented in sections 3 and 4—abbreviated henceforth as and —in this section, we report experimental results on the algorithms 1, 2, , , , and discussed in section 6. The suffix in the triangular coloring algorithm indicates that we used an SL vertex ordering as the ordering π to define the coloring, since this is optimal as discussed in the paragraph leading to Theorem 5.4. Algorithms 1 and 2 are included in our experiments primarily to serve as references against which quantities could be normalized.

The order in which vertices are colored in a greedy algorithm determines the number of colors used by the algorithm. As mentioned earlier, an SL ordering (more precisely, a distance-1 SL ordering) is one of the effective known ordering techniques for distance-1 coloring. As a natural extension, for algorithms such as 2, where the distance-2 neighborhood of a vertex is explored at each step, a distance-2 SL ordering (D2SL) can be defined and used to attain a similar benefit. Specifically, we define a D2SL ordering using the quantity $|N_2(v)|$ associated with each vertex v in a graph. Using linear-time sorting techniques, a D2SL ordering can be computed within the same order of time as a subsequent distance-2 coloring, i.e., in $O(n\bar{d}_2)$ time.

In our experiments (results to be presented in section 7.3), for each of the algorithms 1, 2, , , , , and , we used one of the orderings among natural (the order in which vertices appear in the input graph), D1SL, and D2SL orderings that led to the fewest colors on a majority of the test cases. For algorithm 1, this ordering was D1SL; for algorithms 2, , , , and , this was the D2SL ordering; and for algorithm , it was the natural ordering. We have also experimented with appropriately extended variants of and orderings [17]. The results we obtained using these are in general similar to or worse than the ones obtained using the SL orderings and hence are not reported here.

In a greedy algorithm for triangular coloring, vertex orderings arise in two different contexts: (1) as part of the definition of the coloring, and (2) as part of the coloring algorithm. In terms of an ordering (π) that defines the coloring, a D1SL ordering in the input graph G is optimal. The same ordering can be used while coloring the vertices sequentially, and that is what we did in algorithm . To (potentially) reduce the number of colors used, one could also use a D1SL coloring order based on

TABLE 7.1

*Test graph statistics (left half) and computed values (right half). Column col(G) lists the coloring number of each graph, and columns D1 and D2 list the runtime in seconds of algorithms D1 and D2. The last column shows the ratio of number of edges in a derived graph G′ to that in an original G; the graph G′ is obtained from the graph G as per the specification in expression (5.2) where π is an SL vertex ordering in G. The symbol *** indicates that a graph G′, being too large to fit in memory, could not be computed.*

ID	Name	$\|V\|$	$\|E\|$	Δ	$\overline{d}_1$	$col(G)$	Runtime (sec)		$\frac{\|E'\|}{\|E\|}$
							$D1$	$D2$	
1	er-gre-2	36,573	53,046	8	3	4	0.01	4.44	1.1
2	apoa1-2	92,224	139,351	8	3	4	0.00	31.74	1.1
3	HIV-3	11,414	64,134	20	11	8	0.00	0.44	1.6
4	er-gre-3	36,573	169,787	19	9	8	0.00	4.34	1.7
5	popc-br-3	24,916	102,058	22	8	9	0.01	2.05	1.6
6	apoa1-3	92,224	456,220	20	10	9	0.02	26.85	1.7
7	mol1-2	131,072	1,179,648	18	18	11	0.08	65.47	2.8
8	HIV-4	11,414	130,332	39	23	14	0.00	0.58	2.5
9	apoa1-4	92,224	1,131,436	43	25	15	0.04	28.39	3
10	er-gre-4	36,573	451,355	42	25	15	0.01	4.91	3
11	popc-br-4	24,916	255,047	43	20	15	0.01	2.35	2.7
12	popc-br-5	24,916	497,269	75	40	24	0.01	3.16	4.2
13	apoa1-5	92,224	2,254,805	73	49	25	0.08	33.02	5.2
14	HIV-6	11,414	412,623	116	72	34	0.01	1.74	6
15	er-gre-6	36,573	1,482,904	116	81	36	0.04	9.76	7.8
16	popc-br-6	24,916	850,043	125	68	38	0.03	4.89	6.4
17	apoa1-6	92,224	3,864,429	123	84	38	0.11	42.14	8.1
18	mol1-3	131,072	5,636,096	86	86	42	0.27	123.59	9.8
19	popc-br-7	24,916	1,316,771	192	106	54	0.04	8.39	9
20	popc-br-8	24,916	1,914,476	275	154	72	0.05	14.67	12.4
21	popc-br-9	24,916	2,668,555	380	214	98	0.06	25.78	16.4
22	er-gre-10	36,573	6,511,122	460	356	124	0.16	92.92	30
23	popc-br-10	24,916	3,587,724	514	288	126	0.08	43.83	21
24	mol1-4	131,072	14,680,064	224	224	104	0.59	354.34	***
25	HIV-10	11,414	1,655,383	454	290	115	0.03	18.29	***
26	apoa1-10	92,224	17,100,850	503	371	135	0.39	305.52	***
27	HIV-12	11,414	2,683,956	760	470	183	0.05	47.78	***
28	er-gre-12	36,573	10,928,521	775	598	198	0.24	275.53	***
29	popc-br-12	24,916	5,924,137	855	476	199	0.14	116.7	***
tot	(1–23)	1,139,699	35,129,231						
tot	(1–29)	1,447,312	88,102,142						

vertex degrees in the filled graph G' derived from G according to the specification given by expression (5.2).

In a method they call , Coleman and Moré [13] construct such a filled graph G' using a D1SL ordering in G as π, and then distance-1 color the graph G' using a D1SL ordering in G' to obtain a triangular coloring of the original graph G. We have implemented this method as well; here we call it to reflect that it is a triangular coloring of the original graph. The last column of Table 7.1 shows the ratio of the number of edges in the derived graph G' to that in the original graph G. For 6 out of the 29 test graphs we used (the last six in Table 7.1), the graph G' could not be computed as it was too big to fit in the size of the available memory. The asterisks used in the corresponding entries in the last column of Table 7.1 indicate this.

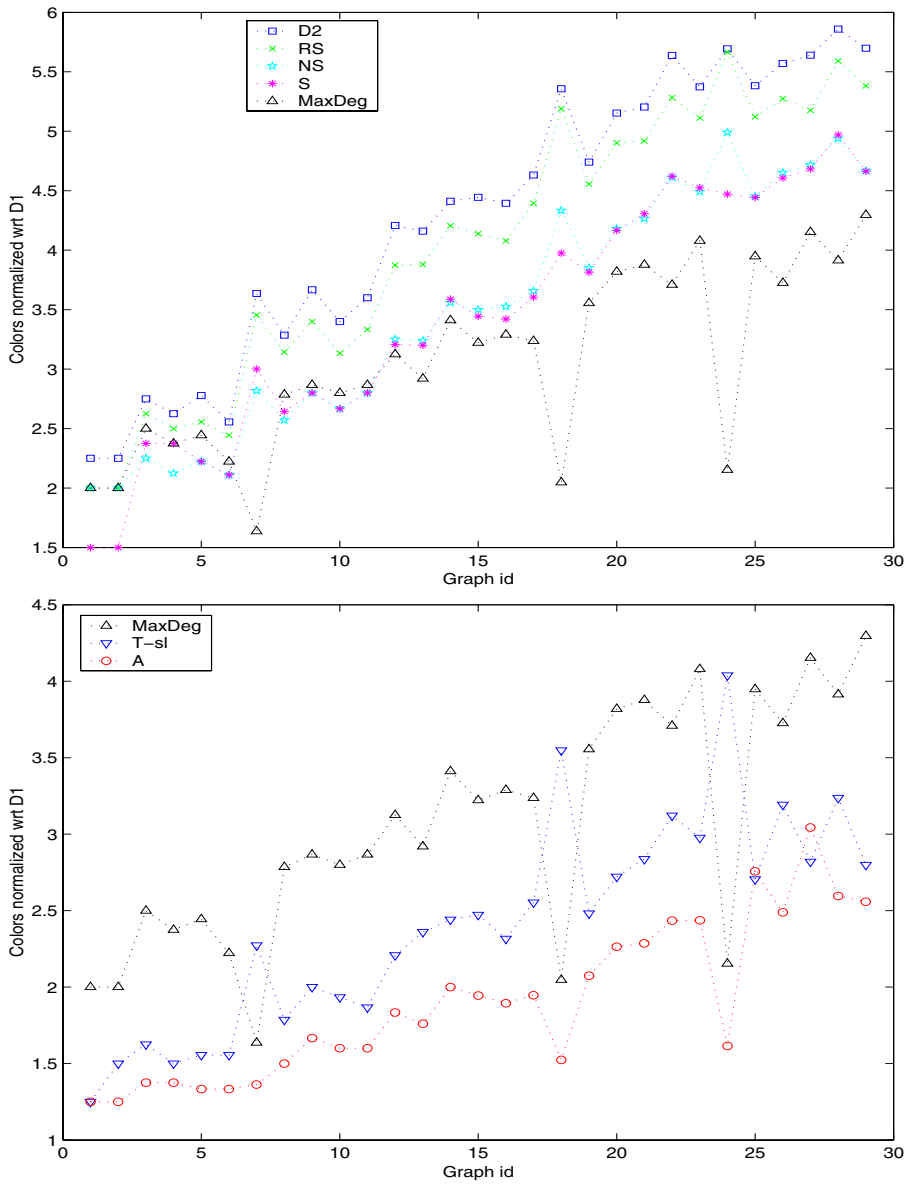


FIG. 7.1. *Normalized number of colors used by various greedy algorithms. In algorithm T-sl, a D1SL ordering of the vertices in G is used to both define and perform the coloring. Algorithm D1 uses a D1SL ordering for coloring. In algorithms D2, RS, NS, and S, vertices are colored in a D2SL ordering in G; in algorithm A, vertices are colored in the natural order they appear in G. The plots are normalized relative to the number of colors used by algorithm D1, i.e., the coloring numbers of the graphs. These numbers are listed in Table* 7.1.

7.3. Results. We will now describe a few figures and tables that show our experimental results. The figures will be discussed collectively in section 7.4.

The upper graph of Figure 7.1 displays plots of the number of colors used by algorithms $D2$, , , , , and , normalized relative to the number of colors used by algorithm $D1$, the coloring numbers of the graphs. The bottom graph shows analogous

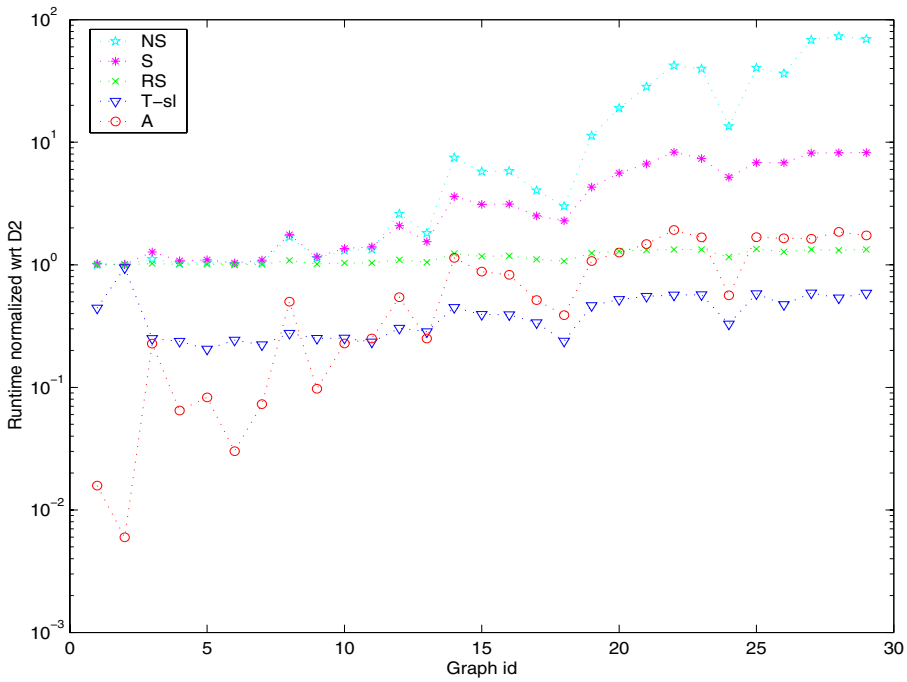


FIG. 7.2. *Execution time comparison. Vertices are colored in the orderings specified in Figure* 7.1. *The plots, in semilog scale, are normalized relative to the runtime of Algorithm D*2*, values listed under column D*2 *of Table* 7.1.

plots for algorithms and . The common curve in the two parts is the plot of the maximum degrees in the graphs, again normalized relative to the coloring numbers. The coloring numbers of the graphs, sorted in increasing order, are listed under column $col(G)$ in Table 7.1.

Figure 7.2 shows semilog plots of the runtimes of algorithms , , , , , , , and normalized relative to the runtime of algorithm $D2$, the values of which are listed in the column labeled $D2$ in Table 7.1. Column $D1$ in the same table shows the runtimes of algorithm $D1$. In these two columns as well as in execution times reported elsewhere in this section, the reported times include time spent on ordering, when applicable, but not time for reading test graph files from disk.

Figure 7.3 shows a comparison between algorithms and . The results reported in the figure concern the test graphs on which the latter could be run successfully, the first 23 of the 29 test graphs used.

Table 7.2 shows a summary of the results for the various algorithms; the upper row gives data summed over all test cases, and the lower row shows data summed over test cases on which algorithm was successfully run.

To show the reduction in number of colors that algorithms $D1$, $D2$, , , , and were able to get through their use of D1SL or D2SL orderings, we ran these algorithms with natural vertex ordering as well. Table 7.3 summarizes the results obtained. In addition to absolute figures, the table shows the savings in number of colors (as a percentage) and the relative time cost in comparison with a coloring that uses an appropriate SL ordering (data listed in Table 7.2).

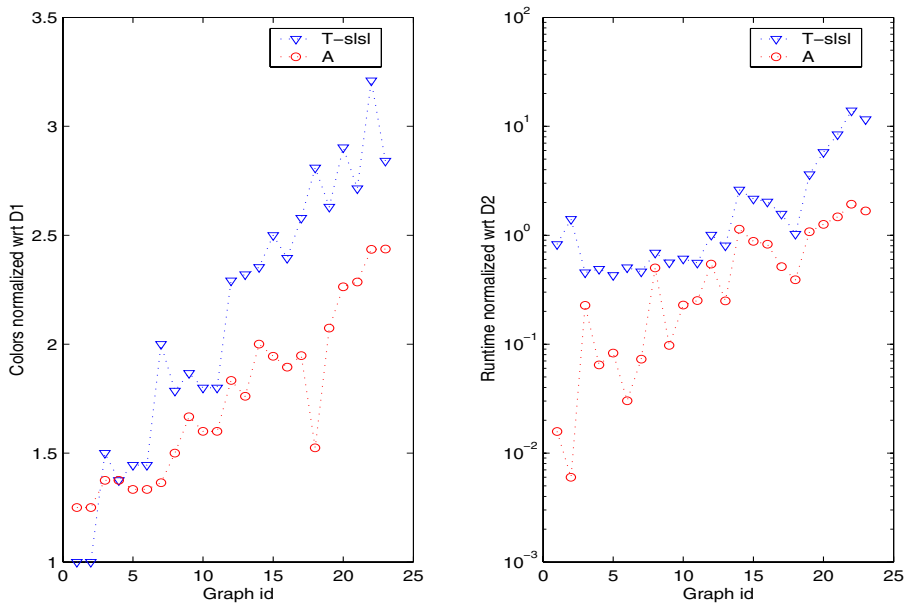


FIG. 7.3. *Comparison of normalized number of colors (left) and normalized runtimes (right) between algorithms A and T-slsl. The number of colors are normalized relative to that used by algorithm D1, and the runtimes are normalized relative to that of algorithm D2.*

TABLE 7.2

Top row: Total number of colors and runtime summed over all test graphs. Bottom row: Totals over test graphs on which algorithm T-slsl could be run successfully (i.e., graphs without asterisks in Table 7.1). The order in which vertices are colored is as specified in Figure 7.1.

	D2	*RS*	*NS*	*S*	*T-sl*	*T-slsl*	*A*	*D1*
Colors	9,240	8,749	7,636	7,558	5,065		4,110	1,757
Runtime (min)	28.2	34.4	930	162	12.4		32.5	0.04
Colors	3,951	3,734	3,207	3,191	2,189	2,149	1,709	823
Runtime (min)	9.6	11	128	34.5	3.8	41.8	7	0.02

TABLE 7.3

Top row: Total number of colors and runtime summed over all test graphs, when vertices are colored in their natural order in an input graph (Nat). Bottom row: Color savings and relative time cost in comparison with coloring using an appropriate SL ordering (numbers listed in Table 7.2).

	D2	*RS*	*NS*	*S*	*D1*
Colors	11,429	11,275	9,973	9,470	2,384
Runtime (min)	4.32	9.60	679	126	0.03
SL vs. Nat colors saved (%)	19	22	23	20	26
SL to Nat time ratio	6.5	3.6	1.4	1.3	1.3

7.4. Discussion. A reader can make several observations from Figures 7.1–7.3 and Tables 7.1–7.3. We highlight the following few points to supplement such observations in the context of Hessian computation.

- The advantage of exploiting sparsity using coloring cannot be overstated. If all of the Hessians (with the nonzero structures of the test graphs) were to be computed without exploiting sparsity or symmetry, the time required would be proportional to 1.5 million steps (the sum total of the vertices in Table 7.1).

If the strongest coloring model considered here—acyclic coloring—were to be used, all of these Hessians could be evaluated in about four thousand time steps (the number of colors, or compressed columns, in the column at the top half of Table 7.2). Moreover, in general, coloring constitutes only a small fraction (typically far less than 10%) of the overall computation time involved in obtaining numerical values in a derivative matrix [18, 19]. Note that the total coloring time required to acyclic-color of the test graphs, totaling over 88 million edges, is about 30 minutes.

- As the upper part of Figure 7.1 shows, among the three algorithms , , , and for direct Hessian computation where symmetry is exploited, algorithm uses the fewest colors. Algorithm uses slightly fewer colors than the symmetry-ignoring algorithm $D2$. The difference in the number of colors used by algorithms and is significant, whereas that between and is marginal.
- The maximum degree Δ in a graph is a lower bound for the distance-2 chromatic number; this fact is clearly reflected in Figure 7.1. One can also see that for a majority of our test graphs, algorithms , , and used more than Δ colors. However, there were cases, noticeably among the relatively sparse graphs of ID less than 10, where these algorithms used fewer than Δ colors.
- The more important advantage of algorithm over algorithm is its lower time complexity, a fact supported by the nearly sixfold reduction in observed running times, as Figure 7.2 and Table 7.2 attest. Algorithms and have the same theoretical complexity, and, in practice, algorithm is observed to be slower by a small factor.
- As the bottom graph of Figure 7.1 illustrates, the coloring algorithms for Hessian computation via substitution—algorithms and —use many fewer colors than the coloring algorithms for direct Hessian computation—algorithms , , , and . On the test graphs we used, our acyclic coloring algorithm consistently used fewer colors than the triangular coloring algorithm . Moreover, in of our test cases, algorithm used fewer than Δ colors, while for algorithm the statement holds true for a majority, but not all, of the test cases. Interestingly, the exceptions in the latter case are regular graphs (graphs 7, 18, and 24 in Table 7.1). Note that the reference against which the number of colors in Figures 7.1 and 7.3 are normalized is a lower bound for triangular coloring.
- As Figure 7.3 shows, our acyclic coloring algorithm also used fewer colors than algorithm . On the subset of the test cases on which algorithm could be run successfully, algorithm used 20% fewer colors than . In terms of runtime, algorithms and are comparable, whereas is much slower.
- In comparison with a natural ordering, an appropriate SL vertex (re)ordering for coloring reduced the number of colors used by algorithms $D1$, $D2$, , , and by roughly 20% (see Table 7.3). In contrast, the numbers listed in Table 7.2 show that the savings in number of colors rendered by algorithm compared to algorithm is only 1.8%, obtained at a cost of runtime that is larger by a factor of 11. The acyclic coloring algorithm did not benefit from an SL ordering.
- Disregarding time for computing vertex orderings, the practical runtimes of algorithms $D2$, , , , , and are expected to be similar since the time

complexity of each of these algorithms is nearly $O(n\bar{d}_2)$. In the observed runtimes in Figure 7.2, the plots for and for the most part lie below the line $y = 1$, which corresponds to the runtime of algorithm $D2$. This is mainly due to the variation in the orderings used: for algorithm $D2$ (as well as and) a D2SL ordering in G was used, for algorithm a D1SL in G was used, and for algorithm the natural ordering was used.

- Even when vertices were colored in natural order, the acyclic coloring algorithm was observed to be slightly faster than the star coloring algorithm . On the other hand, time complexity analyses of the two algorithms show to be slightly slower than . The most likely reason for being slower than in practice is that there are many more two-colored stars to be maintained in algorithm than there are two-colored trees to be maintained in algorithm , which in turn makes the former use more memory operations, and therefore more runtime.

8. Conclusion. We have presented new algorithms for star and acyclic coloring, models for efficient Hessian computation via direct and substitution methods. We experimentally demonstrated that the new algorithms outperform previously known methods. Our acyclic coloring algorithm runs faster, uses fewer colors, and requires less memory and storage space than the triangular coloring method of Coleman and Moré [13]. In comparison with our implementation of the triangular coloring algorithm—which unlike the original algorithm of Coleman and Moré works directly on an input graph G—the new acyclic coloring algorithm produces fewer colors at comparable runtime. Our new star coloring algorithm runs faster and uses fewer colors than our previous algorithm for the same problem [17]. In comparison with the previously known restricted star coloring algorithm, our new star coloring algorithm yields a significant reduction in number of colors at the cost of a slight increase in observed runtime.

Using an acyclic coloring, Coleman and Cai [9] showed that the nonzero entries of a Hessian can be evaluated via substitution by considering a pair of color classes at a time. Our new acyclic coloring algorithm has such structures (two-colored trees) in place, which facilitates an efficient implementation of a procedure for recovering a Hessian from a compressed representation. We have implemented such a routine. In collaboration with Andrea Walther at the Technical University of Dresden, our coloring and Hessian recovery codes have been incorporated in the automatic differentiation tool ADOL-C [20]. A paper that demonstrates the efficacy of the new star and acyclic coloring algorithms in the overall process of computing the Hessian of a given function using automatic differentiation has been submitted for publication elsewhere [18]. The results we obtained in our experiments using ADOL-C show that the new coloring algorithms render huge savings in overall runtime, and that Hessian computation via substitution using acyclic coloring is faster than a direct computation using star coloring, considering the overall process.

The underlying idea in the new algorithms, the exploitation of two-colored structures, can be extended to models in efficient computation of Jacobians [14]. We are currently working in this direction.

We conclude this paper by pointing out a few other directions for further work.

- As discussed in the paragraph immediately after Theorem 5.3, the relationship between the triangular and star chromatic numbers of a graph is not clear. Prove or disprove the claim $\chi_t(G) \leq \chi_s(G)$.

- The size of a largest clique in a graph is an obvious lower bound for the distance-1 chromatic number $\chi_1(G)$. Analogously, $\Delta + 1$ is a lower bound for the distance-2 chromatic number $\chi_2(G)$, since there exists a clique of that size in the square graph G^2. Finding nontrivial lower bounds for acyclic and star chromatic numbers is a worthwhile issue.
- Appropriate extensions of orderings such as [illegible] and [illegible] helped reduce the number of colors used by our star coloring algorithm, but not that of the acyclic coloring algorithm. It would be interesting to find orderings suitable for the latter algorithm.

Acknowledgments. We thank Jorge Moré for suggesting a comparison between acyclic and triangular coloring and for his helpful comments on a draft of this paper. We also thank Paul Hovland for making the test graphs available to us. Our sincere thanks go to the anonymous referees for their valuable comments on the content and presentation of this paper and for bringing related works in the literature to our attention.

REFERENCES

[1] G. Agnarsson, R. Greenlaw, and M. M. Halldórsson, *On powers of chordal graphs and their colorings*, Congr. Numer., 100 (2000), pp. 41–65.

[2] G. Agnarsson and M. M. Halldórsson, *Coloring powers of planar graphs*, SIAM J. Discrete Math., 16 (2003), pp. 651–662.

[3] G. Agnarsson and M. M. Halldórsson, *Coloring powers of planar graphs*, in Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, Philadelphia, 2000, pp. 654–662.

[4] M. O. Albertson, G. G. Chappell, H. A. Kierstead, A. Kündgen, and R. Ramamurthi, *Coloring with no 2-colored P_4's*, Electron. J. Combin., 11 (2004), article R26.

[5] S. Arora and C. Lund, *Hardness of approximations*, in Approximation Algorithms for NP-Hard Problems, D. S. Hochbaum, ed., PWS Publishing, Boston, MA, 1997, Chap. 10, pp. 399–446.

[6] H. Balakrishnan, C. L. Barrett, V. S. A. Kumar, M. V. Marathe, and S. Thite, *The distance-2 matching problem and its application to the MAC-layer capacity of ad hoc wireless networks*, IEEE J. Selected Areas in Commun., 22 (2004), pp. 1069–1079.

[7] O. V. Borodin, *On acyclic colorings of planar graphs*, Discrete Math., 25 (1979), pp. 211–236.

[8] B. Brooks, R. Bruccoleri, B. Olafson, D. States, S. Swaminathan, and M. Karplus, *CHARMM: A program for macromolecular energy, minimization, and dynamics calculations*, J. Comput. Chem., 4 (1983), pp. 187–217.

[9] T. F. Coleman and J.-Y. Cai, *The cyclic coloring problem and estimation of sparse Hessian matrices*, SIAM J. Algebraic Discrete Methods, 7 (1986), pp. 221–235.

[10] T. F. Coleman, B. Garbow, and J. J. Moré, *Software for estimating sparse Jacobian matrices*, ACM Trans. Math. Softw., 10 (1984), pp. 329–347.

[11] T. F. Coleman, B. Garbow, and J. J. Moré, *Software for estimating sparse Hessian matrices*, ACM Trans. Math. Softw., 11 (1985), pp. 363–377.

[12] T. F. Coleman and J. J. Moré, *Estimation of sparse Jacobian matrices and graph coloring problems*, SIAM J. Numer. Anal., 20 (1983), pp. 187–209.

[13] T. F. Coleman and J. J. Moré, *Estimation of sparse Hessian matrices and graph coloring problems*, Math. Program., 28 (1984), pp. 243–270.

[14] T. F. Coleman and A. Verma, *The efficient computation of sparse Jacobian matrices using automatic differentiation*, SIAM J. Sci. Comput., 19 (1998), pp. 1210–1233.

[15] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed., MIT Press, Cambridge, MA, 2001.

[16] A. R. Curtis, M. J. D. Powell, and J. K. Reid, *On the estimation of sparse Jacobian matrices*, J. Inst. Math. Appl., 13 (1974), pp. 117–119.

[17] A. H. Gebremedhin, F. Manne, and A. Pothen, *What color is your Jacobian? Graph coloring for computing derivatives*, SIAM Rev., 47 (2005), pp. 629–705.

[18] A. H. Gebremedhin, A. Pothen, A. Tarafdar, and A. Walther, *Efficient computation of sparse Hessians: An experimental study using ADOL-C*, submitted, 2006.

[19] A. Griewank, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, SIAM, Philadelphia, 2000.

[20] A. Griewank, D. Juedes, and J. Utke, *ADOL-C: A package for the automatic differentiation of algorithms written in C/C++*, ACM Trans. Math. Softw., 22 (1996), pp. 131–167.

[21] B. Grünbaum, *Acyclic colorings of planar graphs*, Israel J. Math., 14 (1973), pp. 390–408.

[22] S. Hossain and T. Steihaug, *Computing a sparse Jacobian matrix by rows and columns*, Optim. Methods Softw., 10 (1998), pp. 33–48.

[23] T. Jensen and B. Toft, *Graph Coloring Problems*, Wiley-Interscience, New York, 1995.

[24] S. O. Krumke, M. V. Marathe, and S. S. Ravi, *Models and approximation algorithms for channel assignment in radio networks*, Wireless Networks, 7 (2001), pp. 567 – 574.

[25] V. S. A. Kumar, M. V. Marathe, S. Parthasarathy, and A. Srinivasan, *End-to-end packet-scheduling in wireless ad-hoc networks*, in Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, Philadelphia, 2004, pp. 1014–1023.

[26] Y.-L. Lin and S. S. Skiena, *Algorithms for square roots of graphs*, SIAM J. Discrete Math., 8 (1995), pp. 99–118.

[27] S. T. McCormick, *Optimal approximation of sparse Hessians and its equivalence to a graph coloring problem*, Math. Program., 26 (1983), pp. 153–171.

[28] M. J. D. Powell and P. L. Toint, *On the estimation of sparse Hessian matrices*, SIAM J. Numer. Anal., 16 (1979), pp. 1060–1074.

[29] L. Stockmeyer and V. Vazirani, *NP-completeness of some generalizations of the maximum matching problem*, Inform. Process. Lett., 15 (1982), pp. 14–19.

[30] M. M. Strout and P. D. Hovland, *Metrics and models for reordering transformations*, in Proceedings of the 2004 Workshop on Memory System Performance, ACM, New York, 2004, pp. 23–34.

SIAM J. SCI. COMPUT.
Vol. 29, No. 3, pp. 1073–1092
© 2007 Society for Industrial and Applied Mathematics

OVERLAPPING SCHWARZ METHODS FOR FEKETE AND GAUSS–LOBATTO SPECTRAL ELEMENTS*

L. F. PAVARINO†, E. ZAMPIERI†, R. PASQUETTI‡, AND F. RAPETTI‡

Abstract. The classical overlapping Schwarz algorithm is here extended to the triangular/tetrahedral spectral element (TSEM) discretization of elliptic problems. This discretization, based on Fekete nodes, is a generalization to nontensorial elements of the tensorial Gauss–Lobatto–Legendre quadrilateral spectral elements (QSEM). The overlapping Schwarz preconditioners are based on partitioning the domain of the problem into overlapping subdomains, solving local problems on these subdomains, and solving an additional coarse problem associated with either the subdomain mesh or the spectral element mesh. The overlap size is generous, i.e., one element wide, in the TSEM case, while it is minimal or variable in the QSEM case. The results of several numerical experiments show that the convergence rate of the proposed preconditioning algorithm is independent of the number of subdomains N and the spectral degree p in case of generous overlap; otherwise it depends inversely on the overlap size. The proposed preconditioners are also robust with respect to arbitrary jumps of the coefficients of the elliptic operator across subdomains.

Key words. spectral elements, Fekete nodes, Gauss–Lobatto–Legendre nodes, overlapping Schwarz preconditioners

AMS subject classifications. 65N55, 65N35, 65F10

DOI. 10.1137/060663568

1. Introduction.

Spectral and *hp*-finite elements are among the most successful high-order methods for the numerical approximation of partial differential equations. These methods can achieve spectral convergence of the discrete solution by allowing both *h*-refinement of the elements mesh and *p*-refinement of the polynomial degree in each element. While hierarchical *hp*-finite elements employ modal basis functions (see [29, 27]), spectral element methods on quadrilateral elements (QSEM) employ a nodal approach based on the tensorial product of one-dimensional Gauss–Lobatto–Legendre (GLL) points (see, e.g., [1, 17, 7]). The need to extend spectral element formulations to complex geometries and unstructured meshes has recently led to the construction and study of triangular/tetrahedral spectral elements (TSEM); see, e.g., [15, 16, 31, 33, 12, 13] and the previous studies of interpolation nodes on triangles [5, 6, 14] and on *p*-version substructuring methods [2]. In this paper, we consider TSEM based on the Fekete nodes [3, 30], as in our previous works [19, 20, 21].

The discrete systems produced by these high-order methods are much more ill-conditioned than in the standard finite element or finite difference cases, and therefore the design and analysis of efficient iterative solvers is still a challenging research topic. In this paper, we construct and study overlapping Schwarz methods for Fekete–Gauss TSEM, obtaining solvers that are scalable (independent of the number of subdomains N) and optimal (independent of the spectral degree p). This last property holds because we employ generous overlap (at least one element wide) between subdomains. In the QSEM case it is possible to construct Schwarz preconditioners with minimal

*Received by the editors June 22, 2006; accepted for publication (in revised form) November 29, 2006; published electronically May 10, 2007.
http://www.siam.org/journals/sisc/29-3/66356.html
†Department of Mathematics, Università di Milano, Via Saldini 50, 20133 Milan, Italy (Luca.Pavarino@mat.unimi.it, Elena.Zampieri@mat.unimi.it).
‡Laboratoire J.-A. Dieudonné, CNRS & Université de Nice et Sophia-Antipolis, Parc Valrose, 06108 Nice cedex 02, France (rpas@math.unice.fr, frapetti@math.unice.fr).

overlap by extending each subdomain with a few rows of GLL nodes of the neighboring elements [4, 10, 11, 18], but the design of a similar technique in the TSEM case is still an open problem, since Fekete points are not distributed as a tensor product. The good convergence properties of our solver are retained also for heterogeneous problems with discontinuous coefficients across subdomain boundaries. Preliminary results with only one element per subdomain and constant coefficients have been reported in the conference paper [21]. A different approach involving preconditioners based on nonoverlapping methods of Neumann–Neumann type for TSEM has been recently studied in [22].

In the rest of the paper, we introduce the model elliptic problem and its QSEM and TSEM discretizations (section 2), the overlapping Schwarz preconditioners in additive and multiplicative form (section 3), and the results of several numerical experiments with the additive Schwarz preconditioners for TSEM and QSEM for homogeneous and heterogeneous problems (section 4).

2. The model problem and SEM formulation. Let $\Omega \in \mathbb{R}^d,\ d = 2, 3$, be a bounded Lipschitz domain with piecewise smooth boundary $\partial\Omega$. For simplicity, we consider a model elliptic problem in the plane ($d = 2$), with homogeneous Dirichlet boundary data, but the numerical methods and results presented in this paper can be applied equally well in three dimensions and to more general elliptic problems:

$$-\mathrm{div}(\alpha\,\mathbf{grad}\,u) + \beta u = f \quad \text{in } \Omega, \qquad u = 0 \quad \text{on } \partial\Omega, \tag{2.1}$$

where $\alpha,\ \beta > 0$ are piecewise constant in Ω and f is a given function in $L^2(\Omega)$. We denote by $L^2(\Omega)$ the space of square integrable measurable functions in Ω, and by $H^1(\Omega)$ the space of functions in $L^2(\Omega)$ whose gradient is in $[L^2(\Omega)]^2$. Then let V be the Sobolev space

$$V \equiv H^1_0(\Omega) = \{v \in H^1(\Omega) : v = 0 \text{ on } \partial\Omega\}.$$

The weak formulation of (2.1) reads (see, e.g., [26]): Find $u \in V$ such that

$$a(u, v) := \int_\Omega (\alpha \mathbf{grad} u \cdot \mathbf{grad} v + \beta\, u\, v)\, \mathrm{d}\mathbf{x} = (f, v) := \int_\Omega f\, v\, \mathrm{d}\mathbf{x} \quad \forall v \in V. \tag{2.2}$$

The variational problem (2.2) is discretized by the standard conforming spectral element method, either quadrilateral-based (QSEM) or triangle-based (TSEM). The method can be viewed as a nodal version of the *hp*-finite element method, which uses GLL points of the quadrilateral for the QSEM and the Fekete points of the triangle for the TSEM, and employs a discrete space consisting of continuous piecewise polynomials of degree p within each quadrilateral or triangular spectral element; see [1, 7, 17] for a general introduction and an analysis of the method. For the QSEM, let Q_{ref} be the reference square $(-1, 1)^2$ and let $\mathbb{Q}_p(Q_{\mathrm{ref}})$ be the set of polynomials on Q_{ref} of degree $\le p$ in each variable. Then, for the TSEM, let $T_{\mathrm{ref}} = \{(r, s) :\ -1 \le r, s,\ r + s \le 0\}$ be the reference triangle and let $\mathbb{P}_p(T_{\mathrm{ref}})$ be the set of polynomials on T_{ref} of total degree $\le p$. We assume that the original domain Ω is decomposed into K quadrilateral or triangular spectral elements T_k,

$$\overline{\Omega} = \bigcup_{k=1}^{K} \overline{T}_k.$$

This is a conforming finite element partition; in particular, the intersection between two distinct elements T_k is either the empty set or a common vertex or a common side.

We denote by h the maximum diameter of the elements T_k and by τ_h the associated finite element mesh. Each element T_k is the image of the reference square $Q_{\rm ref}$ or triangle $T_{\rm ref}$ by means of a suitable mapping $g_k,\ k = 1, \ldots, K$, i.e., $T_k = g_k(Q_{\rm ref})$ or $T_k = g_k(T_{\rm ref})$.

Finally, the space V is discretized by continuous piecewise polynomials of degree $\leq p$ in each variable for QSEM,

$$V_{K,p}^Q = \{v \in V : v|_{\Omega_k} \circ g_k \in \mathbb{Q}_p(Q_{\rm ref}),\ k = 1, \ldots, K\},$$

or of total degree $\leq p$ for TSEM,

$$V_{K,p}^T = \{v \in V : v|_{\Omega_k} \circ g_k \in \mathbb{P}_p(T_{\rm ref}),\ k = 1, \ldots, K\}.$$

The spectral element approximation of the variational elliptic problem (2.2) is obtained by replacing the L^2-inner product and the bilinear form defined in (2.2) with their approximations based either on GLL points for QSEM or on Gauss points for the Fekete–Gauss TSEM.

2.1. The GLL QSEM. We consider conforming QSEM based on GLL quadrature points, which also allows the construction of a tensor-product basis for $V_{K,p}^Q$. We denote by $\{\xi_j\}_{j=0}^p$ the set of GLL points of $[-1, 1]$ that are the $(p + 1)$ zeros of the polynomial

$$(1 - \zeta^2)\frac{\partial L_p(\zeta)}{\partial \zeta},$$

where L_p is the pth Legendre polynomial in $[-1, 1]$. Then we denote by σ_j the quadrature weight associated with ξ_j. Let $l_j(x)$ be the Lagrange interpolating polynomial of degree $\leq p$ which vanishes at all the GLL nodes except ξ_j, where it equals 1. The Lagrangian nodal basis functions on the reference square $Q_{\rm ref}$ are defined by building tensor products

$$l_j(x)l_\ell(y), \quad 0 \leq j, \ell \leq p.$$

Each function $u \in \mathbb{Q}_p(Q_{\rm ref})$ is expanded in this nodal GLL basis through its values at GLL nodes $u(\xi_j, \xi_\ell),\ 0 \leq j, \ell \leq p$, as

$$u(x, y) = \sum_{j=0}^{p}\sum_{\ell=0}^{p} u(\xi_j, \xi_\ell)l_j(x)l_\ell(y).$$

Then, on $Q_{\rm ref}$, the discrete L^2-inner product is

$$(u, v)_{Q_{\rm ref},p} = \sum_{j=0}^{p}\sum_{\ell=0}^{p} u(\xi_j, \xi_\ell)v(\xi_j, \xi_\ell)\sigma_j\sigma_\ell,$$

and in general on Ω,

$$(u, v)_{K,p}^Q = \sum_{k=1}^{K}\sum_{j,\ell=0}^{p} (u \circ g_k)(\xi_j, \xi_\ell)(v \circ g_k)(\xi_j, \xi_\ell)|J_k^Q|\sigma_j\sigma_\ell, \tag{2.3}$$

where $|J_k^Q|$ is the Jacobian of the mapping g_k. We then obtain the discrete variational problem: Find $u \in V_{K,p}^Q$ such that

$$a_{K,p}^Q(u, v) = (f, v)_{K,p}^Q \quad \forall v \in V_{K,p}^Q, \tag{2.4}$$

where $a^Q_{K,p}(\cdot,\cdot)$ is obtained by substituting each integral in (2.2) with the GLL quadrature rule (2.3).

The discrete problem (2.4) can be written equivalently in matrix form as a linear system $A_Q\mathbf{u} = \mathbf{b}$, where A_Q is here the assembled QSEM matrix, $\mathbf{b}$ is the load vector accounting for the contribution of f, and $\mathbf{u}$ is the vector of the unknown nodal values of the function u at the GLL nodes.

2.2. The Fekete–Gauss TSEM. For TSEM it is no longer possible to define spectral elements by tensor product as for QSEM. With $n = (p+1)(p+2)/2$, let $\{\psi_j\}_{j=1}^n$ be a polynomial basis of $\mathbb{P}_p(T_{\rm ref})$ for the usual $L^2(T_{\rm ref})$-inner product, e.g., the Koornwinder–Dubiner orthogonal basis [9]. The Fekete points on $T_{\rm ref}$ are defined as the points $\{\hat{\mathbf{x}}_i\}_{i=1}^n$ that maximize the determinant of the Vandermonde matrix V with elements

$$V_{ij} = \psi_j(\hat{\mathbf{x}}_i), \quad 1 \le i,j \le n,$$

and an algorithm to accurately solve the corresponding optimization problem is described in [30]. For TSEM introduced in [31], the Fekete points are used as approximation points; i.e., the Lagrange polynomials $\{\phi_i\}_{i=1}^n$ built on these points are used as basis functions. Among the main properties of Fekete points proved in [3, 30], we recall that for the cube, Fekete points and Gauss–Lobatto points coincide, thus providing a strong link with the usual QSEM.

Unlike GLL points, a quadrature formula based on Fekete points is exact only for integrands in $\mathbb{P}_p(T_{\rm ref})$. This fact has suggested for the TSEM that we might separate the sets of approximation and quadrature points, using the Fekete points $\{\hat{\mathbf{x}}_i\}_{i=1}^n$ for the first set and Gaussian points $\{\hat{\mathbf{y}}_i\}_{i=1}^m$ for the second set, obtained by imposing an exact integration of polynomials, e.g., in $\mathbb{P}_{2p}(T_{\rm ref})$; see [20]. Given the values of a polynomial $u_p \in \mathbb{P}_p(T_{\rm ref})$ at the Fekete points, one can set up interpolation and differentiation matrices to compute, at the Gauss points, the values of u_p and of its derivatives, respectively. For instance, denoting by $\underline{u}$ the vector of the $u_i = u_p(\hat{\mathbf{x}}_i)$, $1 \le i \le n$, and by $\underline{u}'$ that of the $u_p(\hat{\mathbf{y}}_i)$, $1 \le i \le m$, we have $\underline{u}' = V'V^{-1}\underline{u}$, where $V'_{ij} = \psi_j(\hat{\mathbf{y}}_i)$. On a generic triangle $T_k = g_k(T_{\rm ref})$, the same relation between $\underline{u}'$ and $\underline{u}$ holds true, provided that $u_i = (u_p \circ g_k)(\hat{\mathbf{x}}_i)$ and $u'_j = (u_p \circ g_k)(\hat{\mathbf{y}}_j)$. Similarly, in $T_{\rm ref}$ one has $(\underline{\partial u})' = W'V^{-1}\underline{u}$, with ∂ for differentiation with respect to any coordinates and where $W'_{ij} = \partial\psi_j(\hat{\mathbf{y}}_i)$. Then, using the chain rule, one can compute derivatives in the generic triangle.

The TSEM requires, of course, the use of highly accurate integration rules based on Gauss points, which are still an open subject of research, but in any case, we can use integration rules based on Gauss points for the quadrilateral and then map them to $T_{\rm ref}$; see [17]. On a generic triangle $T_k = g_k(T_{\rm ref})$,

$$(u,v)_{T_k,p} = \sum_{j=1}^m u'_j\, v'_j\, |J_k^T(\hat{\mathbf{y}}_j)|\, \omega_j,$$

where $\omega_j > 0$, $1 \le j \le m$, are the quadrature weights and $|J_k^T|$ is the Jacobian of the mapping g_k between $T_{\rm ref}$ and T_k, and in general on Ω,

$$(u,v)^T_{K,p} = \sum_{k=1}^K (u,v)_{T_k,p}. \tag{2.5}$$

As for (2.4), we obtain a discrete problem

$$a^T_{K,p}(u,v) = (f,v)^T_{K,p} \quad \forall v \in V^T_{K,p}, \tag{2.6}$$

which can be written in matrix form as a linear system $A_T\mathbf{u} = \mathbf{b}$. The TSEM matrix A_T is less sparse than the QSEM matrix A_Q and more ill-conditioned, since its condition number grows as $O(p^4h^{-2})$ rather than $O(p^3h^{-2})$ for $d = 2$ (see section 4).

3. Decomposition into subdomains and overlapping Schwarz preconditioners. We now apply some domain decomposition techniques and build some preconditioning techniques for the iterative solution of the QSEM or TSEM discrete systems $A\mathbf{u} = \mathbf{b}$ by the preconditioned CG (PCG) method, with $A = A_Q$ for QSEM or $A = A_T$ for TSEM. The preconditioner belongs to the family of overlapping Schwarz methods and is built from the solution of parallel independent local elliptic problems on overlapping subdomains, in addition to the solution of coarse problems on the coarse mesh, needed to ensure scalability. For a general introduction to domain decomposition methods and overlapping Schwarz preconditioners we refer to [8, 32, 28, 26] and the references therein.

Preliminary results on overlapping Schwarz preconditioners for Fekete TSEM have been reported in the conference paper [21], where we considered only the case of one spectral element per subdomain and constant coefficients in the elliptic operator. For the classical GLL QSEM case with only one element per subdomain, several works are available in the literature, both for the scalar and vector cases; see, e.g., [4, 10, 24, 25]. In this paper, we extend our investigation to the more general case where Ω is decomposed into subdomains Ω_i, which are in turn partitioned into spectral elements, for both the triangular and quadrilateral case. To this aim, we assemble the elements T_k, $k = 1, \ldots, K$, of the QSEM or TSEM partitions into N quadrilateral or triangular subdomains Ω_i, each consisting of K_i spectral elements T_k. With a local renumbering of the triangles we then have

$$\overline{\Omega}_i = \bigcup_{k=1}^{K_i} \overline{T}_k,$$

where $K = \sum_i K_i$ (see Figures 1–3 below for some examples). We assume also that the subdomain partition is conforming, i.e., that the intersection between two distinct subdomains is either the empty set or a common vertex or a common side, and we denote by H the maximum diameter of the subdomains Ω_i.

The coarse mesh τ_0 is defined to be either the subdomain mesh τ_H or the element mesh τ_h. Let τ_p be the fine discretization mesh determined by either the GLL or the Fekete nodes introduced in each element T_k in sections 2.1 and 2.2. We can define different discretizations and overlapping partitions, according to the spectral element domain decomposition of Ω and the set of nodes within each subdomain. In the special case where $H = h$, each element T_k coincides with a whole subdomain. When $H > h$, we associate groups of triangular or quadrilateral elements to each subdomain, and two different choices for the coarse problem are possible, one associated with the subdomain mesh τ_H, the other with the element mesh τ_h, and both using coarse basis functions with $p = 1$ polynomial degree. We will denote them as τ_H-coarse space and τ_h-coarse space.

The coarse triangulation τ_0 is given by quadrilateral subdomains Ω_i (resp., elements T_k) providing a coarse problem on $\tau_0 \equiv \tau_H$ (resp., $\tau_0 \equiv \tau_h$) with bilinear finite

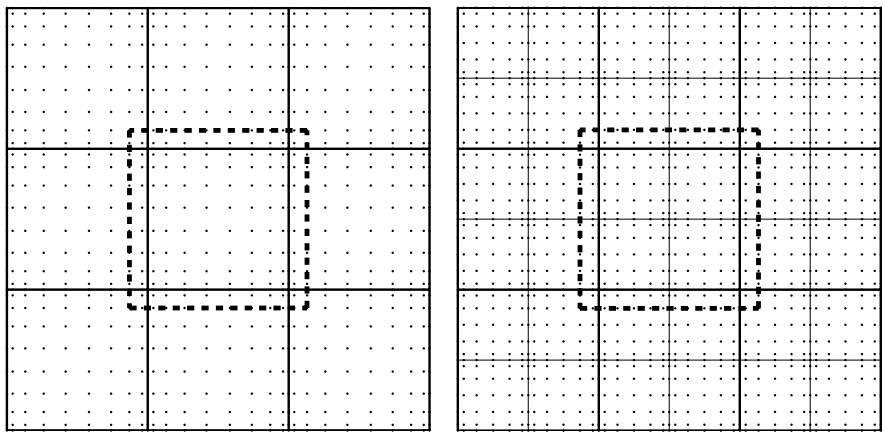


FIG. 1. *QSEM examples with $N = 3 \times 3$ subdomains Ω_i' corresponding to an overlap of $\delta = 2$ GLL points. The internal overlapping subdomain is depicted with thick dashed lines. Left: spectral degree $p = 9$ and $K_i = 1$; i.e., each subdomain consists of only one element. Right: spectral degree $p = 6$ and $K_i = 2 \times 2$; i.e., each subdomain consists of four elements.*

elements ($p = 1$ in each direction). Then the local fine discretization τ_p is determined by GLL nodes in each subdomain Ω_i, when we associate each quadrilateral finite element to a subdomain, or GLL nodes in each element T_k, when each subdomain Ω_i is in turn partitioned into K_i quadrilateral elements of smaller size h. We observe that τ_p is not shape-regular, because the distance between GLL points close to the endpoints of $[-1, 1]$ is on the order of $1/p^2$, while in the middle of the interval $[-1, 1]$ this distance is on the order of $1/p$ (see [1]).

We consider an overlapping partition of Ω by extending each subdomain Ω_i to a larger subdomain Ω_i', consisting of all GLL nodes of τ_p within a certain distance δ^* from Ω_i. Precisely, $\delta^* = \min_{i=1}^{N}\{\text{dist}(\partial\Omega_i, \partial\Omega_i')\}$. We also measure the size of the overlapping region by the number δ of GLL points extending Ω_i in each direction. For example, in the case of minimal overlap and a uniform partition of a square domain Ω, as in Figure 1, we have $\delta = 1$ and $\delta^* = (\xi_1 - \xi_0)h/2$, where ξ_0 and ξ_1 are respectively the first and second GLL node in $[-1, 1]$; in the case of generous overlap we have $\delta = p$ and $\delta^* = h$.

See Figure 1 (left) for a two-dimensional example on a square domain Ω corresponding to local polynomial degree $p = 9$ and one element per subdomain ($H = h$), and Figure 1 (right) for a two-dimensional example on a square domain Ω corresponding to local polynomial degree $p = 6$ and $K_i = 2 \times 2$ elements per subdomain ($H = 2h$). In both cases the original domain Ω is decomposed into $N = 3 \times 3$ subdomains, with overlap $\delta = 2$ GLL points.

The coarse triangulation τ_0 is given either by all quadrilateral or triangular subdomains Ω_i, providing a coarse problem on $\tau_0 \equiv \tau_H$ with, respectively, bilinear or linear finite elements ($p = 1$), or by all triangular elements T_k providing a richer coarse problem on $\tau_0 \equiv \tau_h$ with linear finite elements ($p = 1$). Then the local fine discretization τ_p is determined by Fekete nodes within each subdomain Ω_i in the first case, or within each element T_k in the second one. The overlapping partition of Ω is generated by extending each quadrilateral or triangle Ω_i to a large subdomain Ω_i' consisting of all triangular elements sharing with Ω_i either a vertex or an edge. See Figure 2 for a two-dimensional example on a square domain Ω corresponding to

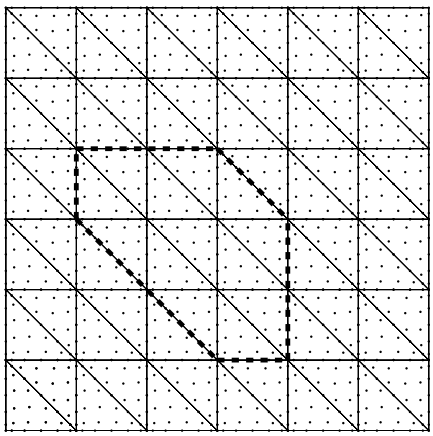


FIG. 2. *TSEM example with spectral degree* $p = 6$, $K = 6 \times 6 \times 2$ *triangular elements, and* $K_i = 1$*; i.e., each subdomain consists of only one element. In this case, an internal overlapping subdomain (depicted with thick dashed lines) consists of* 13 *elements.*

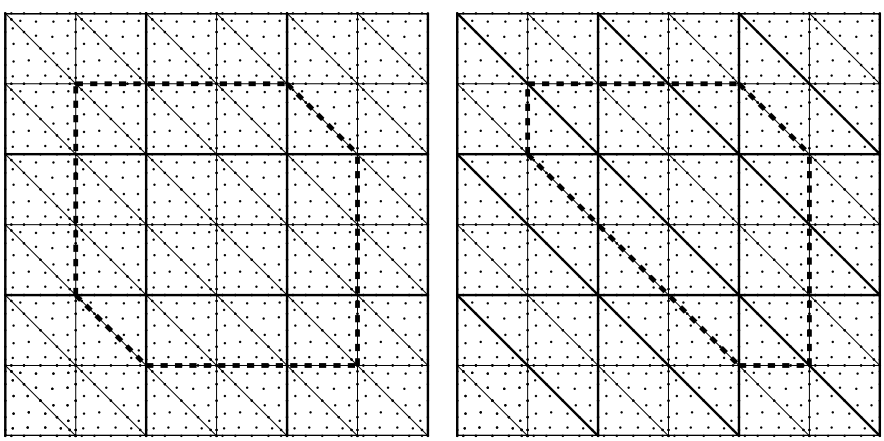


FIG. 3. *TSEM example with spectral degree* $p = 6$. *Left:* $N = 3 \times 3$ *square subdomains,* $K_i = 2 \times 2 \times 2$ *elements per subdomain; in this case, an internal overlapping subdomain (depicted with thick dashed lines) consists of* 30 *elements. Right:* $N = 3 \times 3 \times 2$ *triangular subdomains,* $K_i = 4$ *elements per subdomain; in this case, an internal overlapping subdomain (depicted with thick dashed lines) consists of* 22 *elements.*

$N = 6 \times 6 \times 2$ triangular subdomains ($H = 1/3$), polynomial degree $p = 6$, and one element per subdomain ($H = h$). Moreover, see Figure 3 (left) for a two-dimensional example corresponding to $N = 3 \times 3$ quadrilateral subdomains ($H = 2/3$), polynomial degree $p = 6$, and $K_i = 2 \times 2 \times 2$ triangular elements per subdomain, and Figure 3 (right) for a two-dimensional example corresponding to $N = 3 \times 3 \times 2$ triangular subdomains, polynomial degree $p = 6$, and $K_i = 2 \times 2$ triangular elements per subdomain. Overlapping techniques involving a smaller number of elements (e.g., sharing edges of T_k only) proved unsuccessful, whereas less generous overlapping partitions considering few nodes around Ω_i similarly to the QSEM case cannot be designed straightforwardly, since the internal Fekete nodes are not distributed regularly as in a tensor product.

Overlapping Schwarz methods can now be described either in matrix form or in terms of a space decomposition of the discrete spaces. Let us consider the matrix form first. The overlapping Schwarz preconditioner B^{-1} for A is based on the following:

(a) the solution of a coarse problem on the coarse mesh τ_0, with bilinear elements for QSEM and bilinear or linear ones for TSEM ($p = 1$);

(b) the solutions of local problems on the overlapping subdomains Ω_i'.

For the coarse solve, we need to define:

(a_1) a restriction matrix R_0; its transpose R_0^T interpolates coarse bilinear (resp., bilinear or linear) functions on τ_0 to spectral elements functions on the fine GLL (resp., Fekete) mesh τ_p;

(a_2) a coarse stiffness matrix $A_0 = R_0 A R_0^T$ needed for the solution of the coarse problem with $p = 1$ on τ_0.

For the local solves, we need to define the following:

(b_1) restriction matrices R_i (with 0,1 entries) returning only the degrees of freedom inside each subdomain Ω_i';

(b_2) local stiffness matrices $A_i = R_i A R_i^T$ needed for the solution of the ith local problem on Ω_i' with zero Dirichlet boundary conditions on $\partial\Omega_i'$.

Finally, we define

$$P_i = R_i^T A_i^{-1} R_i A, \qquad i = 0, 1, \ldots, N.$$

These are the building blocks of the proposed preconditioners. The additive form of the overlapping Schwarz preconditioner is

$$B_{add}^{-1} = R_0^T A_0^{-1} R_0 + \sum_{i=1}^{N} R_i^T A_i^{-1} R_i, \qquad \text{i.e.,} \quad P_{add} \equiv B_{add}^{-1} A = P_0 + \sum_{i=1}^{N} P_i, \tag{3.1}$$

while the multiplicative form is

$$B_{mul}^{-1} A = I - (I - P_N) \cdots (I - P_1)(I - P_0). \tag{3.2}$$

More general hybrid variants can be considered as well; see [28].

These preconditioners are associated with the space decomposition

$$V_{K,p} = V_0 + \sum_{i=1}^{N} V_i,$$

where either $V_{K,p} \equiv V_{K,p}^Q$ or $V_{K,p} \equiv V_{K,p}^T$. Here, V_0 is the subspace of $V_{K,p}$ consisting of piecewise bilinear or linear functions on the coarse mesh τ_0 and for QSEM

$$V_i = \{v \in V_{K,p}^Q : v = 0 \text{ at all the GLL nodes outside } \Omega_i' \text{ and on } \partial\Omega_i'\},$$

while for TSEM

$$V_i = \{v \in V_{K,p}^T : v = 0 \text{ at all the Fekete nodes outside } \Omega_i' \text{ and on } \partial\Omega_i'\}.$$

Defining the operators $T_i : V_{K,p} \longrightarrow V_i,\ i = 0, 1, \ldots, N$, by

$$a_{K,p}(T_i u, v) = a_{K,p}(u, v) \qquad \forall v \in V_i,\ i = 0, 1, \ldots, N,$$

where $a_{K,p} \equiv a^Q_{K,p}$ for QSEM and $a_{K,p} \equiv a^T_{K,p}$ for TSEM, it is then easy to check that $B^{-1}_{add}A$ given in (3.1) is exactly the matrix form of the additive Schwarz operator

$$T_{add} = T_0 + T_1 + \cdots + T_N. \tag{3.3}$$

Analogously, $B^{-1}_{mul}A$ given in (3.2) is the matrix form of the multiplicative Schwarz operator

$$T_{mul} = I - (I - T_N)\cdots(I - T_1)(I - T_0). \tag{3.4}$$

The spectral equivalence between the QSEM matrices (mass and stiffness) and the usual P_1 finite element matrices obtained when using the GLL nodal mesh (see Casarin [4]) allows us to transfer the main domain decomposition results from the finite element case to QSEM (see, e.g., Toselli and Widlund [32]).

THEOREM 3.1. $\kappa_2(T_{add}) = \lambda_{max}(T_{add})/\lambda_{min}(T_{add})$ T_{add}

$$\kappa_2(T_{add}) \le C\left(1 + \frac{H}{\delta^*}\right), \tag{3.5}$$

C p, H, h, δ^*

As in the standard finite element case, we cannot guarantee that in the case of QSEM this bound is independent of the coefficient jumps in the elliptic operator, in our case α, but the numerical experiments reported in the next section show that such independence holds.

The bound (3.5) implies that in case of minimal overlap $\delta = 1$ GLL points, i.e., $\delta^* \approx h/p^2$, we have

$$\kappa_2(T_{add}) \le C\left(1 + \frac{H}{h}p^2\right); \tag{3.6}$$

i.e., the number of iterations is expected to scale like p for fixed H/h, while it scales like $\sqrt{H/h}$ for fixed p. In case of generous overlap $\delta^* = h$, we have

$$\kappa_2(T_{add}) \le C\left(1 + \frac{H}{h}\right); \tag{3.7}$$

i.e., the number of iterations is expected to be independent of p and on the order of $\sqrt{H/h}$. This was already proved in Pavarino [23] for *hp*-finite elements and generous overlap. For unstructured *hp* elements on nontensorial elements, such as TSEM, the spectral equivalence used for QSEM in [4, 23] does not hold, and preconditioners with small overlap are not known. Nevertheless, we can build preconditioners with generous overlap as shown before, and the numerical results of the next section show that they are optimal and scalable. Hence we conjecture that a bound as in Theorem 3.1 also holds for TSEM.

We briefly examine here the computational costs of solving both the original TSEM discrete problem and the TSEM preconditioned problem in two dimensions on a serial computer. Analogous estimates can be derived in three dimensions and for QSEM.

We consider for simplicity the square domain $\Omega = (0,1)^2$ with a uniform subdivision in triangular elements as in Figure 3, where the total number of degrees of freedom

is $d_T = O(p^2 N K_i)$, since we have $O(p^2)$ degrees of freedom per element, K_i elements per subdomain, and N subdomains (in this geometry, we have $h^{-1} = \sqrt{NK_i/2}$ and $H^{-1} = \sqrt{N}$). The band of the stiffness matrix is $b_T = O(p^2\sqrt{NK_i})$, i.e., the number of degrees of freedom on one spectral element row (or column) of the square domain.

(a) The computational cost of solving the original TSEM system with a banded direct solver is $O(d_T b_T^2) = O(p^6 (NK_i)^2)$. The cost of solving the same system with a CG method would be proportional to the number of required CG iterations (i.e., the square root of the condition number $= O(p^2 h^{-1}) = O(p^2\sqrt{NK_i})$) times the cost of each CG iteration. The latter is dominated by the matrix-vector products, which for our banded matrix involve $O(d_T b_T) = O(p^4 (NK_i)^{3/2})$ operations. Hence the CG cost would be $O(p^6 (NK_i)^2)$, i.e., similar to the direct solver's one.

(b) The computational cost of solving the TSEM system with the Schwartz PCG is again given by the product of the required PCG iterations and the cost of each PCG iteration; see [28, Chapter 3.6]. From (3.7), the condition number of our two-level Schwarz preconditioner with generous overlap is $O(H/h) = O(\sqrt{K_i})$. Since H/h can be kept constant by decreasing h and H proportionally, we have that the PCG iterations are $O(1)$ (otherwise the extra cost would only be a factor $\sqrt{K_i}$). The cost per iteration is dominated by the matrix-vector products and the preconditioner solves. We assume that we have factored the local and coarse matrices before starting the iterations, so that the local and coarse solves are just banded back-substitutions, with band $b_i = O(p^2\sqrt{K_i})$ for the ith local matrix and band $b_0 = \sqrt{N}$ for the coarse matrix (linear elements on the coarse mesh $\tau_0 = \tau_H$). Denoting by $d_i = O(p^2 K_i)$ and $d_0 = O(N)$ the number of local and coarse degrees of freedom, we have that the cost of the local solves is $O(N d_i b_i) = O(p^4 N K_i^{3/2})$, while the cost of the coarse solve is $O(N^{3/2})$. The overlap costs amount to lower-order terms, since the factor $\sqrt{K_i}$ in the local band b_i is only increased by 1 (if the subdomain touches the domain boundary) or 2 (if the subdomain is internal), and similarly the local number of degrees of freedom d_i is only increased by a factor $O(p^2\sqrt{K_i})$ corresponding to the elements in the overlapping subregion Ω_i' outside subdomain Ω_i. Adding up the matrix-vector products and all the local and coarse solves of the preconditioner, we then estimate the total PCG cost as $O(p^4 (NK_i)^{3/2} + p^4 N K_i^{3/2} + N^{3/2})$. This cost is lower than the cost of solving the original TSEM system as estimated in (a). Better estimates could be obtained by considering better local and coarse solvers, matrix-vector products, etc. We remark that these serial estimates have a relative importance, since domain decomposition preconditioners should be considered on parallel architectures, where more involved complexity estimates need to include architecture-dependent quantities such as communication and memory costs (see [28, Chapter 3.6]), and are beyond the scope of this paper.

4. Numerical results. We tested the properties of the overlapping Schwarz preconditioned CG method for TSEM and QSEM partitions, by performing a p-, H-, and h-convergence study. We consider either a homogeneous or a heterogeneous material, given by the model problem (2.1) over the domain $\Omega = [-1, 1]^2$. The body force f is consistent with $u(x) = \sin(\pi x)\sin(\pi y)$ as an exact solution of (2.1). The discrete problems are solved by either the CG method or the PCG method, with zero initial guess and stopping criterion $\|\mathbf{r}^{(\nu)}\|_2 / \|\mathbf{b}^{(\nu)}\|_2 \leq 10^{-7}$, where $\mathbf{r}^{(\nu)}$ is the residual of the linear systems $A_T \mathbf{u} = \mathbf{b}$ or $A_Q \mathbf{u} = \mathbf{b}$ at the νth iterate. The Schwarz preconditioner is here applied in the additive version (3.1) without or with the coarse correction given by the term $R_0^T A_0^{-1} R_0$ in (3.1). We recall that the overlapping Schwarz preconditioners are built using generous overlap in the TSEM case and using minimal overlap ($\delta = 1$) in the QSEM case.

TABLE 1

TSEM: iteration counts, condition number, and extreme eigenvalues of the original and preconditioned (with $K_i = 1$) operators, fixing $p = 6$ and varying $1/h = 1/H$. See text for additional explanation.

	CG (no prec.)		PCG without coarse pb.				PCG with coarse pb.			
$1/H$	It.	κ_2	It.	λ_{max}	λ_{min}	κ_2	It.	λ_{max}	λ_{min}	κ_2
2	94	729.37	13	12.99	2.67	4.85	13	12.99	3.35	3.87
3	129	1641.55	14	12.99	1.42	9.14	14	12.99	2.35	5.52
4	161	2916.96	16	12.99	0.84	15.33	15	13.00	1.81	7.16
5	191	4542.20	18	12.99	0.55	23.34	16	13.00	1.53	8.46
6	221	6472.48	21	12.99	0.39	33.15	18	13.00	1.37	9.43
7	248	8470.90	24	12.99	0.23	56.47	19	13.00	1.28	10.15

TABLE 2

TSEM: iteration counts, condition number, and extreme eigenvalues of unpreconditioned and preconditioned (with $K_i = 1$) operators, fixing $N = 4 \times 4 \times 2$ triangular subdomains and varying p.

	CG (no prec.)		PCG without coarse pb.				PCG with coarse pb.			
p	It.	κ_2	It.	λ_{max}	λ_{min}	κ_2	It.	λ_{max}	λ_{min}	κ_2
3	28	84.34	12	12.99	2.66	4.87	13	13.00	3.34	3.88
6	85	729.37	13	12.99	2.67	4.85	13	12.99	3.35	3.87
9	206	4819.90	13	12.99	2.67	4.85	13	12.99	3.35	3.87
12	299	8899.07	13	12.99	2.67	4.85	13	12.99	3.35	3.87
15	456	21738.04	13	12.99	2.67	4.85	13	12.99	3.35	3.87
18	777	49837.97	13	12.99	2.67	4.85	13	12.99	3.35	3.87

4.1. The homogeneous case: The Fekete–Gauss TSEM. In this section, we report the results of numerical experiments for the model problem (2.1) discretized with triangular TSEM using the Fekete and Gauss nodes. We consider a homogeneous material with $\alpha = \beta = 1$.

4.1.1. Case $K_i = 1$. We start with the simple choice of associating each triangular element with a subdomain (i.e., $H = h$, or equivalently, $N = K$). The mesh is obtained by first dividing Ω into M^2 identical squares and then by dividing each of them into two triangles, thus providing a partition of $N = 2M^2$ triangular subdomains. The overlapping partition is described in Figure 2. The grid-size parameter H is equal to $2/M$.

In Table 1, we report the iteration counts (It.), spectral condition number (κ_2), and extreme eigenvalues (λ_{max}, λ_{min}), fixing $p = 6$ and varying the parameter $1/h = 1/H$ from 2 to 7, i.e., varying the number of subdomains from $N = 2 \times 4 \times 4$ to $N = 2 \times 14 \times 14$.

Columns 2–3 refer to unpreconditioned CG, columns 4–7 to PCG without coarse problem, and columns 8–11 to PCG with coarse problem ($\tau_h = \tau_H$ in this case).

The same quantities are reported in Table 2, now fixing $h = H = 1/2$ ($N = 2{\times}4{\times}4$ subdomains) and varying the degree p from 3 to 18.

The condition numbers and extreme eigenvalues (not reported in the tables) of the unpreconditioned Fekete matrix are plotted in Figure 4 as a function of H (left) and p (right). These results show that the very ill-conditioned TSEM original matrix A_T has a condition number that grows as $O(p^4 h^{-2})$. The maximum eigenvalue λ_{max} grows as $O(p^2)$, and $1/\lambda_{min}$ grows as $O(p^2 h^{-2})$. The results also show that the overlapping Schwarz preconditioned operator with or without coarse solver is optimal in p, since all the quantities are bounded by constants independent of p (Table 2). Without the coarse problem the preconditioner loses its scalability (Table 1), since

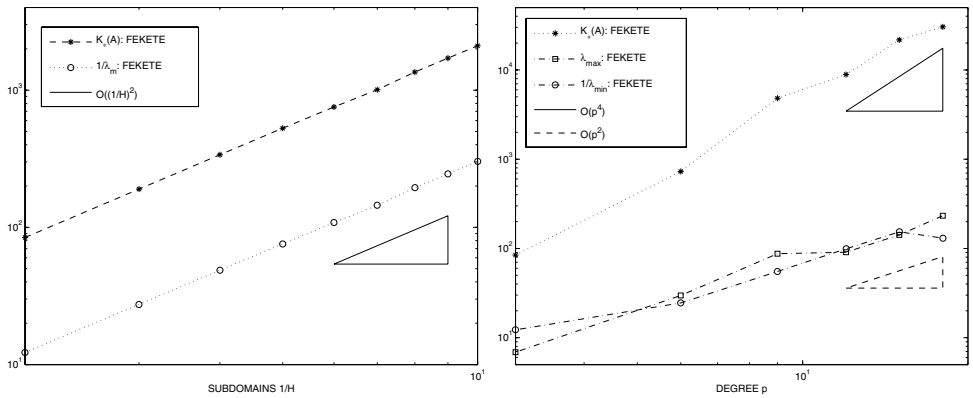


FIG. 4. *TSEM: log-log plot of the condition number of the unpreconditioned Fekete operator as a function of $h = H$ (left) and p (right) from the data of Tables* 1 *and* 2.

λ_{min} tends to 0 with increasing N, while the addition of a coarse problem seems to guarantee scalability (higher values of $1/H$ would be necessary in order to clearly asses scalability).

4.1.2. Case $K_i > 1$. We now consider the more general case where each subdomain Ω_i contains several triangular elements. The TSEM partition is built by first dividing the square computational domain Ω into $N = M^2$ identical square subdomains Ω_i, with $M = 2, \ldots, 6$. Then a fine local partition is considered within each subdomain, consisting of $2 \times 2, \ldots, 6 \times 6$ smaller squares, each of them being divided into two identical triangular elements, so that $K_i = 8, 18, \ldots, 72$. Each subdomain Ω_i is then extended into an overlapping subdomain Ω_i' by adding all the surrounding elements that intersect its boundary; see Figure 3(left). The numerical results with the overlapping Schwarz preconditioner are obtained with both the τ_h-coarse space and τ_H-coarse space.

First, we check the convergence properties of our TSEM preconditioners with respect to the number of subdomains, fixing $p = 6$ as total degree of the polynomials within each triangle and with $h = H/3$. The results are reported in Table 3 and plotted in Figure 5(left). As expected, one observes an increase of the condition number when no coarse solver is used. However, nice results are obtained when the preconditioner employs either one of the τ_h- or τ_H-coarse spaces, and the condition number appears to be bounded above by a constant of about 20.

Second, we vary the number K_i of triangular elements in each subdomain; i.e., we vary the ratio H/h. The results are reported in Table 4 and plotted in Figure 6(left). Again, one observes the efficiency of the overlapping Schwarz preconditioner, especially when a coarse solver is included. Here it should be remarked that the result is better than expected, since the condition number appears to be independent of H/h, i.e., K_i.

Third, we vary the TSEM polynomial degree p. The results are reported in Table 5 and plotted in Figure 7(left). Of course, the coarse solver has no influence here, and all the three preconditioners have slowly increasing condition numbers that seem to be bounded above by a constant independent of p. We note that such slow increase toward an upper bound was not observed in the case $K_i = 1$ (Table 2), where the independence of p was very clear.

TABLE 3
TSEM: iteration counts, condition number, and extreme eigenvalues of the original and preconditioned matrix, fixing $p = 6$, $K_i = 3 \times 3 \times 2$, and varying N.

	CG				PCG	without coarse pb.		
N	It.			κ_2	It.	λ_{max}	λ_{min}	κ_2
2×2	129			1641.54	37	7.81	0.43	18.07
3×3	176			3687.55	39	8.52	0.38	21.99
4×4	221			6471.91	47	8.71	0.22	39.36
5×5	261			9348.38	46	7.87	0.15	49.95
6×6	329			12152.89	54	7.52	0.11	64.86
	PCG	with τ_h-coarse pb.			PCG	with τ_H-coarse pb.		
N	It.	λ_{max}	λ_{min}	κ_2	It.	λ_{max}	λ_{min}	κ_2
2×2	37	7.82	0.43	18.13	37	7.81	0.43	18.12
3×3	37	8.54	0.45	18.67	37	8.52	0.45	18.66
4×4	40	8.72	0.43	19.85	41	8.72	0.44	19.89
5×5	37	7.94	0.45	17.42	37	7.87	0.45	17.30
6×6	37	7.55	0.45	16.53	39	7.52	0.45	16.60

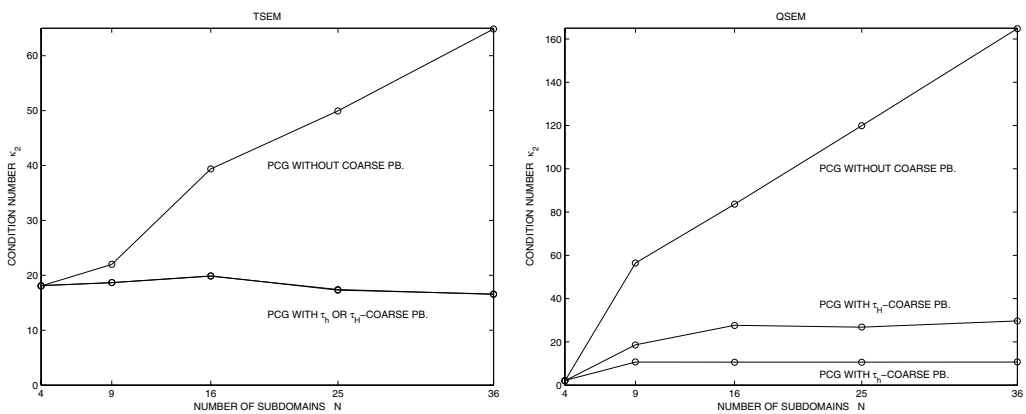


FIG. 5. *Condition number of PCG with and without coarse problem as a function of the number of subdomains N; TSEM data from Table 3 (left), QSEM data from Table 6 (right).*

These results show that the overlapping Schwarz preconditioner with a coarse problem is optimal and scalable also in the nontrivial case $H \neq h$, i.e., $K_i > 1$. The results are not as clear as in the trivial case with only one element per subdomain ($H = h$, i.e., $K_i = 1$), but we conjecture that numerical experiments with higher values of p and N would confirm these properties.

4.2. The homogeneous case: The GLL QSEM. Overlapping Schwarz methods for QSEM based on GLL points have been successfully applied to many elliptic and parabolic problems (see, e.g., [32, 7] and the references therein). Numerical results are usually carried out by considering the simpler choice $H = h$ with only one element per subdomain.

In this section, we report the results of numerical experiments with GLL QSEM for the same tests and model problem (2.1) of the previous section. We focus on the nontrivial case $H \neq h$ and perform analogous p-, H-, and h-convergence tests. The QSEM partition is built by dividing the square computational domain Ω into $N = M^2$ identical square subdomains Ω_i of characteristic diameter H. Then a local partition is considered within each subdomain, consisting of K_i square elements of

TABLE 4

TSEM: iteration counts, condition number, and extreme eigenvalues of the original and preconditioned matrix, fixing $p = 6$, $N = 3 \times 3$, and varying K_i.

	CG				PCG	without coarse pb.		
K_i	It.			κ_2	It.	λ_{max}	λ_{min}	κ_2
$2 \times 2 \times 2$	129			1641.54	36	8.48	0.45	18.54
$2 \times 3 \times 3$	176			3687.55	39	8.52	0.38	21.99
$2 \times 4 \times 4$	221			6471.91	40	7.88	0.28	27.34
$2 \times 5 \times 5$	261			9348.38	47	8.49	0.21	40.17
$2 \times 6 \times 6$	329			12152.89	48	8.53	0.18	46.14
	PCG	with τ_h-coarse pb.			PCG	with τ_H-coarse pb.		
K_i	It.	λ_{max}	λ_{min}	κ_2	It.	λ_{max}	λ_{min}	κ_2
$2 \times 2 \times 2$	36	8.49	0.45	18.54	36	8.48	0.46	18.47
$2 \times 3 \times 3$	37	8.54	0.45	18.67	37	8.52	0.45	18.66
$2 \times 4 \times 4$	36	7.95	0.46	17.27	37	7.88	0.45	17.20
$2 \times 5 \times 5$	38	8.50	0.45	18.60	39	8.49	0.45	18.61
$2 \times 6 \times 6$	38	8.54	0.45	18.69	40	8.53	0.45	18.69

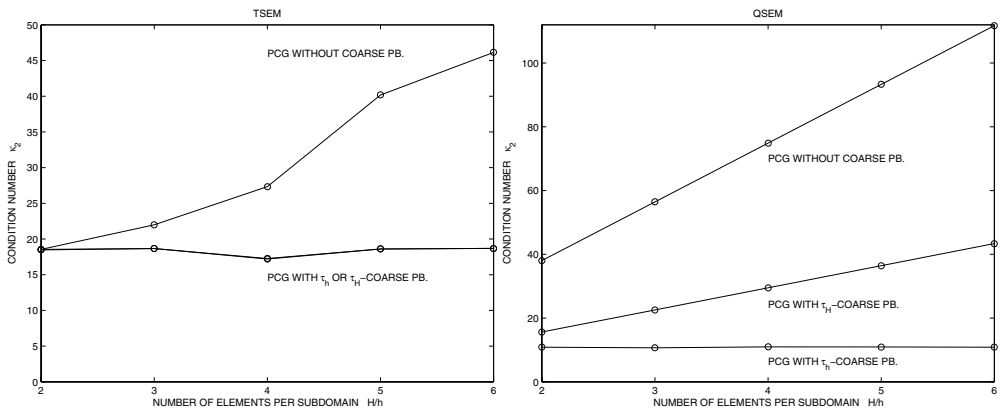


FIG. 6. *Condition number of PCG with and without coarse problem as a function of the number of elements per subdomain H/h; TSEM data from Table 4 (left), QSEM data from Table 7 (right).*

TABLE 5

TSEM: iteration counts, condition number, and extreme eigenvalues of the original and preconditioned matrix, fixing $N = 3 \times 3$, $K_i = 2 \times 3 \times 3$, and varying p.

	CG				PCG	without coarse pb.		
p	It.			κ_2	It.	λ_{max}	λ_{min}	κ_2
3	59			428.10	32	6.88	0.38	17.78
6	176			3687.55	39	8.52	0.38	21.99
9	437			24235.35	43	9.39	0.38	24.23
12	625			44816.79	45	9.81	0.38	25.30
15	940			109050.07	47	10.03	0.38	25.88
18	1460			236603.08	48	10.16	0.38	26.22
	PCG	with τ_h-coarse pb.			PCG	with τ_H-coarse pb.		
p	It.	λ_{max}	λ_{min}	κ_2	It.	λ_{max}	λ_{min}	κ_2
3	29	7.13	0.55	12.89	30	6.89	0.55	12.50
6	37	8.54	0.45	18.67	37	8.52	0.45	18.66
9	41	9.39	0.42	22.12	42	9.39	0.42	22.23
12	43	9.81	0.41	23.41	44	9.81	0.41	23.65
15	45	10.03	0.41	24.10	46	10.04	0.41	24.39
18	46	10.16	0.41	24.47	47	10.16	0.40	24.80

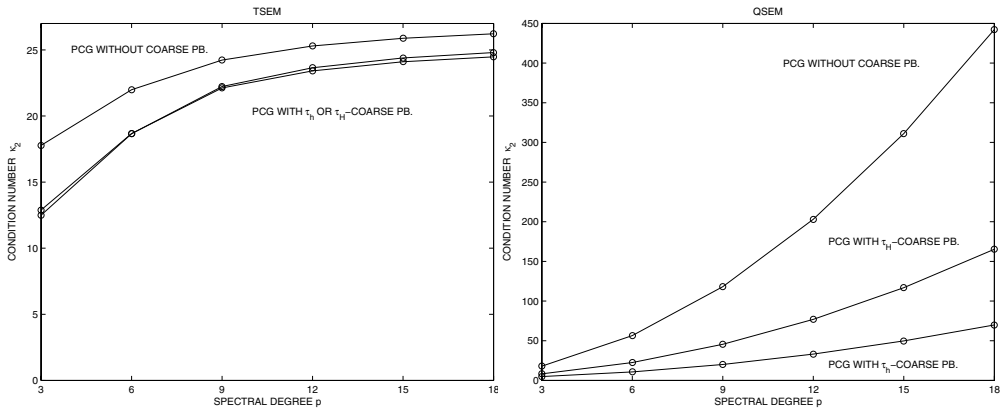


FIG. 7. *Condition number of PCG with and without coarse problem as a function of the spectral degree p; TSEM data from Table* 5 *(left), QSEM data from Table* 8 *(right).*

TABLE 6

QSEM: iteration counts, condition number, and extreme eigenvalues of the original and preconditioned matrix, fixing $p = 6$, $K_i = 3 \times 3$, *minimal overlap* $\delta = 1$, *and varying* N.

	CG				PCG	without coarse pb.		
N	It.			κ_2	It.	λ_{max}	λ_{min}	κ_2
2×2	67			270.78	8	1.93	1.00	1.93
3×3	106			603.09	25	4.00	0.07	56.45
4×4	141			1067.56	30	4.00	0.05	83.60
5×5	175			1667.71	41	4.00	0.03	119.93
6×6	213			2399.75	46	4.00	0.02	164.83
	PCG	with τ_h-coarse pb.			PCG	with τ_H-coarse pb.		
N	It.	λ_{max}	λ_{min}	κ_2	It.	λ_{max}	λ_{min}	κ_2
2×2	10	2.17	1.00	2.17	8	2.00	1.00	2.00
3×3	20	4.01	0.37	10.68	23	8.52	0.45	18.66
4×4	21	4.01	0.38	10.62	27	4.00	0.14	27.68
5×5	22	4.01	0.38	10.65	32	4.00	0.15	26.80
6×6	22	4.01	0.37	10.69	33	4.00	0.13	29.70

characteristic diameter h, and a tensorial GLL mesh is introduced in each element. We then extend each subdomain Ω_i to an overlapping subdomain Ω_i' with minimal overlap, consisting of just $\delta = 1$ layer of GLL nodes outside the boundary; see Figure 1(right) and section 3.

We first study the convergence properties of overlapping Schwarz preconditioners for QSEM with respect to the number of subdomains. We compare the overlapping Schwarz preconditioner operator based either on the τ_h- or the τ_H-coarse space. In Table 6 we report the same quantities of the previous section, and we plot them in Figure 5(right). We fix the local polynomial degree $p = 6$ and the number of elements per subdomain $K_i = 3 \times 3$, which corresponds to $H = 3h$, varying the number of subdomains from $N = 2 \times 2$ to $N = 6 \times 6$. While both unpreconditioned CG and PCG without coarse problem are not scalable, PCG with either the τ_h- or τ_H-coarse space is.

We then consider the convergence properties with respect to the number K_i of elements per subdomain in Table 7 and plot the results in Figure 6(right). We fix the local polynomial degree $p = 6$ and the number of subdomains $N = 3 \times 3$, varying the number of elements per subdomain from $K_i = 2 \times 2$ to $K_i = 6 \times 6$. In agreement

TABLE 7
QSEM: iteration counts, condition number, and extreme eigenvalues of the original and preconditioned matrix, fixing $p = 6$, $N = 3 \times 3$, minimal overlap $\delta = 1$, and varying K_i.

	CG				PCG	without coarse pb.		
K_i	It.			κ_2	It.	λ_{max}	λ_{min}	κ_2
2×2	67			270.78	20	4.00	0.10	38.03
3×3	106			603.09	25	4.00	0.07	56.45
4×4	141			1067.56	29	4.00	0.05	74.88
5×5	175			1667.71	32	4.00	0.04	93.31
6×6	213			2399.75	34	4.00	0.03	111.74
	PCG	with τ_h-coarse pb.			PCG	with τ_H-coarse pb.		
K_i	It.	λ_{max}	λ_{min}	κ_2	It.	λ_{max}	λ_{min}	κ_2
2×2	19	4.01	0.37	10.86	19	4.00	0.26	15.63
3×3	20	4.01	0.37	10.68	23	4.00	0.18	22.55
4×4	21	4.01	0.36	10.97	26	4.00	0.13	29.49
5×5	21	4.01	0.37	10.92	30	4.00	0.11	36.43
6×6	21	4.01	0.37	10.84	32	4.00	0.09	43.38

TABLE 8
QSEM: iteration counts, condition number, and extreme eigenvalues of the original and preconditioned matrix, fixing $N = 3 \times 3$, $K_i = 3 \times 3$, minimal overlap $\delta = 1$, and varying p.

	CG				PCG	without coarse pb.		
p	It.			κ_2	It.	λ_{max}	λ_{min}	κ_2
3	46			118.29	15	4.00	0.22	17.89
6	106			603.10	25	4.00	0.07	56.45
9	178			1627.80	35	4.00	0.03	118.18
12	263			3553.80	45	4.00	0.02	203.07
15	359			6707.30	55	4.00	0.01	311.11
18	464			11379.62	65	4.00	0.01	442.30
	PCG	with τ_h-coarse pb.			PCG	with τ_H-coarse pb.		
p	It.	λ_{max}	λ_{min}	κ_2	It.	λ_{max}	λ_{min}	κ_2
3	14	4.09	0.85	4.81	15	4.02	0.49	8.23
6	20	4.01	0.37	10.68	23	4.00	0.18	22.55
9	27	4.00	0.20	20.11	32	4.00	0.09	45.50
12	33	4.00	0.12	33.14	41	4.00	0.05	76.96
15	41	4.00	0.08	49.75	50	4.00	0.03	116.90
18	48	4.00	0.06	69.91	58	4.00	0.02	165.32

with the theoretical estimate of Theorem 3.1, the preconditioner with the τ_H-coarse space yields condition numbers that grow linearly with K_i. On the other hand, the τ_h-coarse space yields condition numbers that are independent of K_i.

We then study the convergence properties with respect to the polynomial degree p in Table 8 and plot the results in Figure 7(right). We fix the number of subdomains $N = 3 \times 3$, and the number of elements per subdomain $K_i = 3 \times 3$, varying the local polynomial degree from $p = 3$ to $p = 18$. In agreement with the bound of Theorem 3.1, all PCG iteration counts display a linear growth with p, corresponding to a quadratic growth of the condition number associated with a quadratic decay of λ_{min} (while λ_{max} has a constant value of about 4).

Finally, we consider the convergence properties of our preconditioner with respect to the overlap δ (number of GLL points extending Ω_i in each direction) in Table 9 and plot the same results with respect to the corresponding values of δ^* (size of the overlapping region) in Figure 8. We fix the number of subdomains $N = 3 \times 3$, the number of elements per subdomain $K_i = 2 \times 2$, the local polynomial degree $p = 9$, varying the overlap from $\delta = 1$ GLL points (minimal overlap) to $\delta = 9$ GLL

TABLE 9

QSEM: iteration counts, condition number, and extreme eigenvalues of P_{add}, fixing $N = 3 \times 3$, $K_i = 2 \times 2$, $p = 9$, and varying the overlap δ. Columns 2–5 refer to the coarse mesh $\tau_0 \equiv \tau_H$, and columns 6–9 refer to the richer coarse mesh $\tau_0 \equiv \tau_h$.

	PCGwith τ_H-coarse pb.				PCG with τ_h-coarse pb.			
δ	It.	λ_{max}	λ_{min}	κ_2	It.	λ_{max}	λ_{min}	κ_2
1	27	4.00	0.13	30.92	25	4.00	0.19	20.64
2	18	4.02	0.38	10.57	18	4.04	0.54	7.49
3	16	4.08	0.64	6.34	16	4.16	0.81	5.13
4	14	4.20	0.83	5.04	14	4.36	0.94	4.66
5	14	4.35	0.93	4.67	14	4.58	0.98	4.68
6	13	4.50	0.97	4.63	14	4.77	1.00	4.77
7	13	4.62	0.99	4.66	13	4.89	1.00	4.89
8	13	4.70	1.00	4.70	14	4.97	1.00	4.97
9	12	4.74	1.00	4.74	12	5.00	1.00	5.00

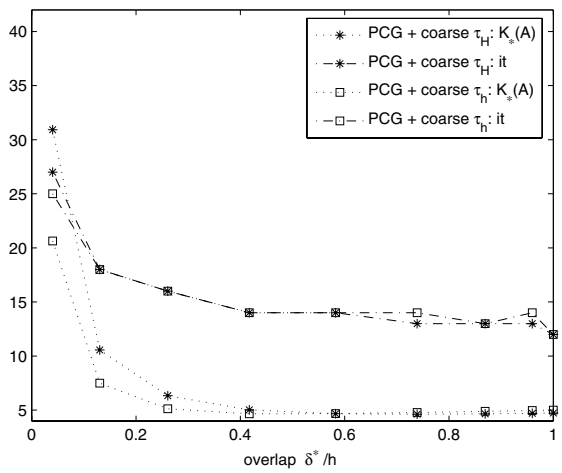


FIG. 8. *QSEM: condition number and iteration counts for the overlapping Schwarz preconditioned operator, as a function of the overlap size δ^* normalized with respect to h, from the corresponding data expressed in terms of δ in Table 9.*

points (generous overlap). Again in agreement with the theory, the condition numbers improve with increasing δ because λ_{min} improves considerably. The improvement is large in going from $\delta = 1$ to $\delta = 2$ and 3, while it becomes rapidly marginal (or negative) for larger values of δ, due to the slight growth of λ_{max}.

4.3. The heterogeneous case: The Fekete–Gauss TSEM. To check the efficiency of the additive Schwarz preconditioner for TSEM, we vary the diffusion coefficient α in the elliptic operator of our model problem (2.1). The computations are done for the reference case with $p = 6$, $N = 3 \times 3$, and $K_i = 2 \times 3 \times 3$.

First, we use two different values of α, $\alpha_1 = 1$ in some subdomains and $\alpha_2 = 10^t$, with $t = -3, -2, \ldots, 3$ in the others. These two values are distributed in a checkerboard layout on the subdomains, starting with α_1 on the top left subdomain; see Figure 9(left). The results are reported in Table 10. We remark that the efficiency of the overlapping Schwarz preconditioner is not affected by the jump of α at the interfaces of the subdomains, while unpreconditioned CG greatly suffers the severe

1	10^t	1
10^t	1	10^t
1	10^t	1

10	10^{-2}	10^5
10^4	10^6	1
10^{-3}	10^2	10^{-1}

FIG. 9. *Values of the discontinuous coefficient α of the elliptic operator in the heterogeneous test of Tables* 10 *and* 11*: checkerboard (left) and random (right) layout.*

TABLE 10

TSEM for heterogeneous case: iteration counts, condition number, and extreme eigenvalues of the original and preconditioned matrix, fixing $p = 6$, $K_i = 3 \times 3 \times 2$, $N = 3 \times 3$, and considering the coefficient α discontinuous across subdomains.

	CG				PCG	without coarse pb.		
t	It.			κ_2	It.	λ_{max}	λ_{min}	κ_2
-3	1023			31023.64	41	9.45	0.45	20.70
-2	798			16120.09	41	9.42	0.45	20.61
-1	387			8493.16	40	9.16	0.45	20.07
0	176			3687.55	39	8.52	0.38	21.99
1	443			8689.35	38	8.43	0.45	18.48
2	1164			49766.04	39	9.07	0.45	19.84
3	3058			285022.32	40	9.17	0.45	20.08
Random	*			*	49	9.46	0.40	23.58
	PCG	with τ_h-coarse pb.			PCG	with τ_H-coarse pb.		
t	It.	λ_{max}	λ_{min}	κ_2	It.	λ_{max}	λ_{min}	κ_2
-3	40	9.46	0.45	20.61	40	9.45	0.45	20.69
-2	40	9.43	0.45	20.49	41	9.42	0.45	20.59
-1	39	9.17	0.45	20.00	40	9.16	0.45	20.06
0	37	8.54	0.45	18.67	37	8.52	0.45	18.66
1	38	8.52	0.45	18.65	38	8.43	0.45	18.48
2	39	8.79	0.45	19.27	40	8.80	0.45	19.30
3	41	8.84	0.45	19.40	41	8.86	0.45	19.45
Random	48	9.47	0.45	20.64	50	9.46	0.45	20.74

ill-conditioning of the problems due to the increasing coefficient jumps.

We then consider a harder case with "random" distribution of α shown in Figure 9(right), and report the results in the last line of Table 10. Even in this very difficult case, where unpreconditioned CG does not seem to converge, the overlapping Schwarz preconditioner remains very efficient.

4.4. The heterogeneous case: The GLL QSEM. Analogous conclusions are obtained with GLL QSEM; see Table 11. Here the condition numbers of the preconditioned operator seem to even improve with increasing coefficient jumps, while their maximum is attained in absence of jumps for $t = 0$. For the difficult test case with the same random distribution of α, unpreconditioned CG again fails to converge, while the overlapping Schwarz preconditioner requires about the same iterations as in absence of jumps.

5. Conclusions. We have constructed and studied overlapping Schwarz preconditioners for the triangular/tetrahedral spectral element (TSEM) discretization of scalar elliptic problems based on Fekete nodes. The proposed preconditioners employ generous overlap among subdomains, since it is not currently know how to construct TSEM preconditioners with small overlap. We also compared the TSEM nontensorial

TABLE 11

QSEM: iteration counts, condition number, and extreme eigenvalues of the original and preconditioned matrix, fixing $p = 6$, $K_i = 3 \times 3$, $N = 3 \times 3$, and considering the coefficient α discontinuous across subdomains.

	CG				PCG	without coarse pb.		
t	It.			κ_2	It.	λ_{max}	λ_{min}	κ_2
-3	631			18069	11	4.00	0.60	6.65
-2	390			6576	14	4.00	0.46	8.76
-1	207			1445	19	4.00	0.16	25.01
0	106			603.10	25	4.00	0.07	56.45
1	241			2958.20	19	4.00	0.17	22.79
2	669			27712	13	4.00	0.47	8.58
3	>1000			275540	11	4.00	0.60	6.65
Random	*			*	24	4.00	0.03	114.17
	PCG	with τ_h-coarse pb.			PCG	with τ_H-coarse pb.		
t	It.	λ_{max}	λ_{min}	κ_2	It.	λ_{max}	λ_{min}	κ_2
-3	14	4.01	0.90	4.46	13	4.00	0.82	4.88
-2	15	4.01	0.78	5.13	15	4.00	0.66	6.07
-1	17	4.01	0.50	8.02	20	4.00	0.30	13.29
0	20	4.01	0.37	10.68	23	4.00	0.18	22.55
1	16	4.01	0.59	6.82	18	4.00	0.36	10.94
2	13	4.01	0.85	4.73	14	4.00	0.68	5.85
3	12	4.01	0.90	4.45	12	4.00	0.79	5.02
Random	23	4.01	0.45	8.84	25	4.00	0.32	12.59

discretization with the classical Gauss–Lobatto–Legendre QSEM discretization, where it is possible to construct Schwarz preconditioners with variable overlap (from generous to minimal). In spite of the more severe ill-conditioning of the resulting TSEM discrete problem (where $\kappa_2 \approx O(p^4 h^{-2})$), the convergence rate of the proposed preconditioning algorithm is independent of the number of subdomains N when a coarse problem is employed and of the spectral degree p in case of generous overlap; otherwise it depends inversely on the overlap size. Moreover, the proposed preconditioners are robust with respect to arbitrary jumps across subdomains of the coefficients of the elliptic operator. Our results are limited to regular structured meshes in the plane. Future work will extend this study to three dimensions and to unstructured meshes for more general domains.

REFERENCES

[1] C. BERNARDI AND Y. MADAY, *Spectral methods*, in Handbook of Numerical Analysis, Volume V, Techniques of Scientific Computing (Part 2), North–Holland, Amsterdam, 1997, pp. 209–485.

[2] I. BICA, *Iterative Substructuring Methods for the p-Version Finite Element Method for Elliptic Problems*, Ph.D. thesis, Courant Institute, New York University, New York, 1997; technical report 1997-743.

[3] L. BOS, M. A. TAYLOR, AND B. A. WINGATE, *Tensor product Gauss–Lobatto points are Fekete points for the cube*, Math. Comp., 70 (2001), pp. 1543–1547.

[4] M. A. CASARIN, *Quasi-optimal Schwarz methods for the conforming spectral element discretization*, SIAM J. Numer. Anal., 34 (1997), pp. 2482–2502.

[5] Q. CHEN AND I. BABUŠKA, *Approximate optimal points for polynomial interpolation of real functions in an interval and in a triangle*, Comput. Methods Appl. Mech. Engrg., 128 (1995), pp. 485–494.

[6] Q. CHEN AND I. BABUŠKA, *The optimal symmetrical points for polynomial interpolation of real functions in a tetrahedron*, Comput. Methods Appl. Mech. Engrg., 137 (1996), pp. 89–94.

[7] M. O. DEVILLE, P. F. FISCHER, AND E. H. MUND, *High-Order Methods for Incompressible Fluid Flow*, Cambridge University Press, London, 2002.

[8] M. Dryja and O. B. Widlund, *Domain decomposition algorithms with small overlap*, SIAM J. Sci. Comput., 15 (1994), pp. 604–620.
[9] M. Dubiner, *Spectral methods on triangles and other domains*, J. Sci. Comput., 6 (1991), pp. 345–390.
[10] P. F. Fischer, *An overlapping Schwarz method for spectral element solution of the incompressible Navier–Stokes equations*, J. Comput. Phys., 133 (1997), pp. 84–101.
[11] P. F. Fischer and J. W. Lottes, *Hybrid Schwarz–multigrid methods for the spectral element method: Extensions to Navier–Stokes*, J. Sci. Comput., 6 (2005), pp. 345–390.
[12] F. X. Giraldo and T. Warburton, *A nodal triangle-based spectral element method for the shallow water equations on the sphere*, J. Comput. Phys., 207 (2005), pp. 129–150.
[13] F. X. Giraldo, *High-order triangle-based discontinuous Galerkin methods for hyperbolic equations on a rotating sphere*, J. Comput. Phys., 214 (2006), pp. 447–465.
[14] J. S. Hesthaven, *From electrostatics to almost optimal nodal sets for polynomial interpolation in a simplex*, SIAM J. Numer. Anal., 35 (1998), pp. 655–676.
[15] J. S. Hesthaven and C. H. Teng, *Stable spectral methods on tetrahedral elements*, SIAM J. Sci. Comput., 21 (2000), pp. 2352–2380.
[16] J. S. Hesthaven and T. Warburton, *Nodal high-order methods on unstructured grids*, J. Comput. Phys., 181 (2002), pp. 186–221.
[17] G. E. Karniadakis and S. J. Sherwin, *Spectral/hp Element Methods for Computational Fluid Dynamics*, 2nd ed., Oxford University Press, London, 2005.
[18] A. Klawonn and L. F. Pavarino, *Overlapping Schwarz methods for mixed linear elasticity and Stokes problems*, Comput. Meth. Appl. Mech. Engrg., 165 (1998), pp. 233–245.
[19] R. Pasquetti and F. Rapetti, *Spectral element methods on triangles and quadrilaterals: Comparisons and applications*, J. Comput. Phys., 198 (2004), pp. 349–362.
[20] R. Pasquetti and F. Rapetti, *Spectral element methods on unstructured meshes: Comparisons and recent advances*, J. Sci. Comput., 27 (2006), pp. 377–387.
[21] R. Pasquetti, L. F. Pavarino, F. Rapetti, and E. Zampieri, *Overlapping Schwarz preconditioners for Fekete spectral elements*, in Domain Decomposition Methods in Science and Engineering 16, O. B. Widlund and D. Keyes, eds., Lecture Notes in Comput. Sci. Engrg. 55, Springer, New York, 2006, pp. 717–724.
[22] R. Pasquetti, F. Rapetti, L. F. Pavarino, and E. Zampieri, *Neumann–Neumann-Schur complement methods for Fekete spectral elements*, J. Engrg. Math., 56 (2006), pp. 323–335.
[23] L. F. Pavarino, *Additive Schwarz methods for the p-version finite element method*, Numer. Math., 66 (1994), pp. 493–515.
[24] L. F. Pavarino and T. Warburton, *Overlapping Schwarz methods for unstructured spectral elements*, J. Comput. Phys., 160 (2000), pp. 298–317.
[25] L. F. Pavarino and E. Zampieri, *Overlapping Schwarz and spectral element methods for linear elasticity and elastic waves*, J. Sci. Comput., 27 (2006), pp. 51–73.
[26] A. Quarteroni and A. Valli, *Domain Decomposition Methods for Partial Differential Equations*, Oxford Science Publications, London, 1999.
[27] C. Schwab, *p- and hp-Finite Element Methods. Theory and Applications in Solid and Fluid Mechanics*, Oxford University Press, New York, 1998.
[28] B. F. Smith, P. Bjørstad, and W. D. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, London, 1996.
[29] B. Szabó and I. Babuška, *Finite Element Analysis*, John Wiley & Sons, New York, 1991.
[30] M. A. Taylor, B. A. Wingate, and R. E. Vincent, *An algorithm for computing Fekete points in the triangle*, SIAM J. Numer. Anal., 38 (2000), pp. 1707–1720.
[31] M. A. Taylor and B. A. Wingate, *A generalized diagonal mass matrix spectral element method for nonquadrilateral elements*, Appl. Numer. Math., 33 (2000), pp. 259–265.
[32] A. Toselli and O. B. Widlund, *Domain Decomposition Methods: Algorithms and Theory*, Comput. Math. 34, Springer-Verlag, Berlin, 2004.
[33] T. Warburton, L. F. Pavarino, and J. S. Hesthaven, *A pseudo-spectral scheme for the incompressible Navier–Stokes equation using unstructured nodal elements*, J. Comput. Phys., 164 (2000), pp. 1–21.

SIAM J. SCI. COMPUT.
Vol. 29, No. 3, pp. 1093–1126
© 2007 Society for Industrial and Applied Mathematics

ON WELL-BALANCED FINITE VOLUME METHODS FOR NONCONSERVATIVE NONHOMOGENEOUS HYPERBOLIC SYSTEMS*

M. J. CASTRO DÍAZ†, T. CHACÓN REBOLLO‡, E. D. FERNÁNDEZ-NIETO§, AND CARLOS PARÉS†

Abstract. In this work we introduce a general family of finite volume methods for nonhomogeneous hyperbolic systems with nonconservative terms. We prove that all of them are "asymptotically well-balanced": they preserve all smooth stationary solutions in all the domain except for a set whose measure tends to zero as Δx tends to zero. This theory is applied to solve the bilayer shallow-water equations with arbitrary cross-section. Finally, some numerical tests are presented for simplified but meaningful geometries, comparing the computed solution with approximated asymptotic analytical solutions.

Key words. well-balanced finite volume method, upwinding, shallow water, source terms, two-layer flows

AMS subject classifications. 65N06, 76B15, 76M20, 76N99

DOI. 10.1137/040607642

1. Introduction.

This work deals with finite volume solvers for nonhomogeneous hyperbolic systems with nonconservative terms. Our goal is to design numerical schemes that calculate with high accuracy the stationary solutions of the system.

The accurate computation of stationary solutions of hyperbolic systems with source terms has been found in recent years to be closely related to the accurate computation of transient solutions. Numerical schemes that do not solve steady solutions up to second order, at least, yield transient solutions that present large errors, unacceptable from the physical point of view. To avoid this difficulty, Bermúdez and Vázquez [1] introduced the so-called C-property in the context of shallow-water equations. They ask the numerical schemes to exactly calculate at grid nodes the stationary solution corresponding to water at rest. They also present some numerical evidence of the large errors presented by schemes that do not verify this property. In that work, the scheme of Roe (cf. [19]) is generalized to nonhomogeneous hyperbolic systems, in such a way that it verifies the C-property.

This property has given fruitful extensions to more general situations: numerical models, hyperbolic equations, and extensions of the condition itself.

For instance, an extension to kinetic schemes was introduced by Perthame and Simeoni in [17]. This type of scheme is based upon the discretization of an equation for the density of particles. Also, Kurganov and Levy in [14] introduced the central upwind schemes for shallow-water equations, characterized by including only scalar numerical diffusion. No upwinding matrix is needed. Both schemes ensure the

*Received by the editors April 30, 2004; accepted for publication (in revised form) November 1, 2006; published electronically May 14, 2007. This research was partially supported by Spanish Government research projects BFM2003-07530-C02-01 and BFM2003-07530-C02-02.
http://www.siam.org/journals/sisc/29-3/60764.html

†Departamento Análisis Matemático, Universidad de Málaga, Campus Teatinos s/n, Málaga, Spain (castro@anamat.cie.uma.es, pares@anamat.cie.uma.es).

‡Departamento de Ecuaciones Diferenciales y Análisis Numérico, Universidad de Sevilla, C/Tarfia s/n, 41080 Sevilla, Spain (chacon@us.es).

§Departamento de Matemática Aplicada I, Universidad de Sevilla, 41012 Sevilla, Spain (edofer@us.es).

positiveness of the water height and verify the C-property.

The relevance of this property as a consistency condition was pointed out in the work of Perthame and Simeoni [18]. They prove the convergence, in the sense of distributions, of the numerical solution computed by a finite volume scheme for a nonhomogeneous equation under some hypothesis of consistency of the numerical scheme. This hypothesis is fulfilled if the numerical scheme preserves certain equilibrium states of the system.

An extension of the C-condition to a more general condition was introduced in Greenberg and Leroux [13]: the concept of a "well-balanced"scheme, as a scheme that preserves all equilibria of the system at grid nodes. In this work, a numerical scheme for nonhomogeneous scalar equations satisfying this condition was introduced.

An extension to a class of nonconservative hyperbolic system of equations was introduced in [2] by Castro, Macías, and Parés by means of a generalized Roe scheme. The systems considered include a nonconservative product with the structure $B(W)\frac{\partial W}{\partial x}$, where B is a matrix and W is the unknown. The numerical stability of this scheme is achieved by using upwinding matrices defined not only from the Jacobian of the flux, but also from the matrix B involved by the nonconservative terms. The scheme introduced is applied to the one-dimensional two-layer shallow-water system for channels with constant width. The scheme exactly computes water at rest. More recently, in [4] this Roe scheme has been extended to the more general case of channels with irregular geometry and not necessarily rectangular cross-section. This extended numerical scheme was applied to the simulation of the Strait of Gibraltar.

The derivation of systematic techniques to build numerical schemes satisfying the C-property is also a relevant issue, as this is far from being a straightforward condition to obtain. In [5] Chacón Rebollo, Domínguez Delgado, and Fernández-Nieto introduced the idea of separately compensating the centered and decentered components of the numerical flux by centered and decentered discretizations of the source terms. In [6] the same authors introduced the idea of adding an additional source term to the parabolic equivalent equation. This term is specifically designed to compensate for the first order error generated by the numerical diffusion term on steady solutions. Then centered discretizations automatically provide second order accurate approximations of all steady solutions.

In this work, we extend the preceding analysis by considering a general nonhomogeneous hyperbolic system of equations with nonconservative terms, which also depends upon the space variable. We derive a general class of numerical schemes that we prove to be "asymptotically well-balanced" in the following sense: they calculate, with at least order two, all stationary solutions of the system in all the domain except those on a set whose measure tends to zero as Δx tends to zero. One of the free parameters of this class is the diffusion matrix, so we include both flux-splitting and flux-difference or Roe methods as particular cases.

We also give general sufficient conditions under which the schemes exactly calculate a given stationary solution. These conditions may be read as meaning on one hand that the numerical flux of the scheme must compensate the centered part of the numerical source term, and on another hand that the numerical diffusion must compensate the decentered part of the numerical source term.

This paper is organized as follows: In section 2, a family of numerical schemes for a general nonhomogeneous hyperbolic systems of equations with nonconservative terms is introduced. In section 3, we introduce the concept of the "asymptotically well-balanced" scheme and prove that the introduced numerical schemes satisfy this

property. Section 4 is devoted to the construction of numerical diffusion matrices that ensure the stability of our schemes for linear systems, while ensuring the asymptotically well-balanced computation of all steady solutions. In section 5, we apply the theory developed to the bilayer shallow-water equations with arbitrary cross-sections to construct a family of asymptotically well-balanced numerical schemes. The Roe-type scheme introduced in [4] is a particular case. In section 6, we present some numerical tests for simplified geometries, where exact solutions can be determined. The purpose of these experiments is to compare the properties of the numerical schemes introduced here for the two-layer shallow water system. More precisely, we focus on the well-balanced property and the quality of the numerical approximations for smooth transitions and shocks. We present our conclusions in section 7. In particular we conclude that the use of pointwise adapted discretizations of the centered part of the numerical flux helps to construct accurate solvers for nonconservative equations.

2. Nonhomogeneous nonconservative hyperbolic systems. We shall consider the following system:

$$(1)\quad \begin{cases} \dfrac{\partial W(x,t)}{\partial t} + \dfrac{\partial}{\partial x}[F(x,W(x,t))] = G(x,W(x,t)) + B(x,W(x,t))\dfrac{\partial W(x,t)}{\partial x}, \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad x \in (0,L), \quad t \in (0,T); \\ \\ W(x,0) = W_0(x), \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad x \in (0,L), \end{cases}$$

where by $F : [0,L] \times \mathbf{R}^N \to \mathbf{R}^N$ we denote the physical flux function and by $G : [0,L] \times \mathbf{R}^N \to \mathbf{R}^N$ the source term, and where $B : [0,L] \times \mathbf{R}^N \to \mathbf{R}^N \times \mathbf{R}^N$ is a regular matrix function of x and W. We will suppose that F, G, and B are C^2 functions. By L we denote the length of the domain and by T the final time. We will complete (1) with appropriate boundary conditions that we set in each problem.

System (1) includes the nonconservative product

$$B(x,W)\frac{\partial W}{\partial x},$$

which is not a distribution if W is discontinuous. Its presence requires an adaptation of the usual concept of weak solutions: Across a discontinuity of W, the nonconservative product can be defined along paths connecting the two states, following Dal Masso, LeFloch, and Murat [8].

System (1) also includes a dependency upon the space variable of the flux F. Both characteristics appear in two-layer shallow-water equations in channels of variable breadth, as we shall see later on. However, these characteristics also appear in other flow models such as boiling flows and two-phase flows, among others (see Fowler [11]).

We suppose that the system (1) is hyperbolic in the sense that $\forall x \in [0,L]$ and $\forall W \in \mathbf{R}^d$ the matrix

$$(2)\qquad M(x,W) = A(x,W) - B(x,W)$$

can be diagonalized and all its eigenvalues are real and different, where $A(x,W) = \frac{\partial F}{\partial W}(x,W)$ is the Jacobian matrix of the flux function F.

We consider a partition $\{x_i\}_{i=0}^{M+1}$ of the interval $[0,L]$, and a partition $\{t^n\}_{n=0}^{m+1}$ of $[0,T]$. By W_i^n we define an approximation to the average of the solution in the control volume, $(x_{i-1/2}, x_{i+1/2})$ at $t = t^n$.

In this work, for the space discretization, we develop finite volume schemes that can be written in viscous form. For the homogeneous conservative system, these schemes are second order approximations in space of a parabolic equivalent system:

$$\frac{\partial W}{\partial t} + \frac{\partial}{\partial x} F(x, W) - \nu \frac{\partial}{\partial x} \left(\mathcal{D}(x, W) \frac{\partial}{\partial x} W \right) = 0,$$

where ν is equal to half of the space step ($\nu = \Delta x/2$) and $\mathcal{D}(x, W)$ is the numerical diffusion matrix: The term

$$-\nu \frac{\partial}{\partial x} \left(\mathcal{D}(x, W) \frac{\partial W}{\partial x} \right) \tag{3}$$

represents the numerical diffusion introduced by the scheme, stemming from the upwinding of the convection terms. The diffusion matrix $\mathcal{D}(x, W)$ must take into account that the effective transport term in system (1) is $M(x, W)\frac{\partial W}{\partial x}$, where M is the matrix defined by (2).

If a centered discretization of the remaining terms in the nonhomogeneous and nonconservative system (1) is performed, then the equivalent parabolic system is

$$\frac{\partial W}{\partial t} + \frac{\partial}{\partial x} F(x, W) - \nu \frac{\partial}{\partial x} \left(\mathcal{D}(x, W) \frac{\partial}{\partial x} W \right) = G(x, W) + B(x, W) \frac{\partial W}{\partial x}. \tag{4}$$

Notice that if $\bar{W}$ is a stationary solution of (1), then it verifies

$$\frac{\partial}{\partial x} [F(x, \bar{W}(x))] = G(x, \bar{W}(x)) + B(x, \bar{W}(x)) \frac{\partial \bar{W}(x)}{\partial x}. \tag{5}$$

So, it is clear that $\bar{W}$ is not a stationary solution of (4). There is a first order error due to the numerical diffusion term (3) that is not compensated.

In order to obtain numerical schemes that calculate up to a second order of accuracy all smooth stationary solutions, we propose compensating the numerical diffusion term by an additional source term, which we denote by $C(x, W)$ (correction term), that is, the numerical scheme must be a second order approximation of

$$\frac{\partial W}{\partial t} + \frac{\partial}{\partial x} \left[F(x, W) \right] - \nu \frac{\partial}{\partial x} \left(\mathcal{D}(x, W) \frac{\partial W}{\partial x} \right) = G(x, W) + B(x, W) \frac{\partial W}{\partial x} + C(x, W).$$

In order to define the term $C(x, W)$, we consider a smooth stationary solution $\bar{W}$ of (1). We want the identity

$$-\nu \frac{\partial}{\partial x} \left(\mathcal{D}(x, \bar{W}) \frac{\partial \bar{W}}{\partial x} \right) = C(x, \bar{W}) \tag{6}$$

to hold. As $\bar{W}$ verifies (5), then

$$\frac{\partial F}{\partial x}(x, \bar{W}(x)) + A(x, \bar{W}(x)) \frac{\partial \bar{W}}{\partial x}(x) = \frac{\partial}{\partial x} [F(x, \bar{W}(x))] = G(x, \bar{W}(x)) + B(x, \bar{W}(x)) \frac{\partial \bar{W}}{\partial x}.$$

Thus,

$$M(x, \bar{W}(x)) \frac{\partial \bar{W}}{\partial x}(x) = G(x, \bar{W}(x)) - F_x(x, \bar{W}(x)), \tag{7}$$

where $F_x(x,W) = \frac{\partial F}{\partial x}(x,W)$. Moreover, if $M(x,\bar{W})$ is nonsingular, we have

$$\frac{\partial \bar{W}(x)}{\partial x} = M^{-1}(x,\bar{W}(x))S(x,\bar{W}(x)),$$

where

$$S(x,W) = G(x,W) - F_x(x,W).$$

Therefore,

$$-\nu\frac{\partial}{\partial x}\left(\mathcal{D}(x,\bar{W}(x))\frac{\partial \bar{W}}{\partial x}(x)\right) = -\nu\frac{\partial}{\partial x}\left(\mathcal{D}(x,\bar{W}(x))M^{-1}(x,\bar{W}(x))S(x,\bar{W}(x))\right). \tag{8}$$

Comparing (8) with (6), we propose to define

$$C(x,W) = -\nu\frac{\partial}{\partial x}\left(\mathcal{D}(x,W)M^{-1}(x,W)S(x,W)\right) \quad \forall x \in [0,L],$$

$\forall W \in \mathbf{R}^N$ such that $M(x,W)$ is nonsingular. The equivalent system for the hyperbolic system with source term and nonconservative terms is

$$\begin{aligned}&\frac{\partial W}{\partial t} + \frac{\partial}{\partial x}F(x,W) - \nu\frac{\partial}{\partial x}\left(\mathcal{D}(x,W)\frac{\partial W}{\partial x}\right)\\ &= G(x,W) + B(x,W)\frac{\partial W}{\partial x} - \nu\frac{\partial}{\partial x}\left(\mathcal{D}(x,W)M^{-1}(x,W)S(x,W)\right).\end{aligned} \tag{9}$$

Every stationary solution of (1) is also a stationary solution of this system at points where $M(x,W(x))$ is nonsingular.

We conclude then that asymptotically well-balanced numerical schemes for the hyperbolic system (1) can be built by the use of approximations of second order in space of (9).

Notice that system (9) has a conservative structure when the original system (1) is conservative (i.e., $B = 0$). Indeed, it can be rewritten as

$$\frac{\partial W}{\partial t} + \frac{\partial}{\partial x}\left[F(x,W) - \nu\mathcal{D}(x,W)\left(\frac{\partial W}{\partial x} - M^{-1}S(x,W)\right)\right] = G(x,W) + B(x,W)\frac{\partial W}{\partial x}. \tag{10}$$

Based upon this structure, we propose general three-point schemes as follows:

$$\begin{aligned}&\frac{W_i^{n+1} - W_i^n}{\Delta t} + \frac{\phi_R^S(x_i,x_{i+1},W_i^n,W_{i+1}^n) - \phi_L^S(x_{i-1},x_i,W_{i-1}^n,W_i^n)}{\Delta x}\\ &= G_C(x_{i-1},x_i,x_{i+1},W_{i-1}^n,W_i^n,W_{i+1}^n),\end{aligned} \tag{11}$$

where

$$\phi_R^S(x_i,x_{i+1},W_i,W_{i+1}) = \phi^S(x_i,x_{i+1},W_i,W_{i+1}) - \frac{1}{2}\mathcal{B}_R(x_i,x_{i+1},W_i,W_{i+1})\,(W_{i+1}-W_i), \tag{12}$$

$$\phi_L^S(x_i,x_{i+1},W_i,W_{i+1}) = \phi^S(x_i,x_{i+1},W_i,W_{i+1}) + \frac{1}{2}\mathcal{B}_L(x_i,x_{i+1},W_i,W_{i+1})(W_{i+1}-W_i), \tag{13}$$

and

$$\begin{aligned}\phi^S(x_i,x_{i+1},W_i,W_{i+1}) &= F_C(x_i,x_{i+1},W_i,W_{i+1}) - \nu D(x_i,x_{i+1},W_i,W_{i+1})\\ &\times\left(\frac{W_{i+1}-W_i}{\Delta x} - \widetilde{M^{-1}}(x_i,x_{i+1},W_i,W_{i+1})S_D(x_i,x_{i+1},W_i,W_{i+1})\right).\end{aligned} \tag{14}$$

Here, $\mathcal{B}_L$ and $\mathcal{B}_R$ are approximations of B at $x = x_{i+1/2}$, respectively. Moreover, G_C is an approximation of $G(x,W)$ in $x = x_i$. By F_C, D, $\widetilde{M^{-1}}$, and S_D we denote the approximations of F, $\mathcal{D}$, M^{-1}, and S in $x = x_{i+1/2}$, respectively.

Let us suppose, for instance, that the singularities of M are concentrated on a continuous hypersurface $\gamma \subset \mathbf{R}^{n+1}$, where, say the jth eigenvalue λ_j vanishes. In this case, we define $\widetilde{M^{-1}}(x,y,U,V)$ as

$$\widetilde{M^{-1}}(x,y,U,V) = X((x+y)/2,\widetilde{W})\widetilde{\Lambda^{-1}}(x,y,U,V)X^{-1}((x+y)/2,\widetilde{W}), \tag{15}$$

where $\widetilde{W} = \widetilde{W}(x,y,U,V)$ is an intermediate state between U and V, by X we denote the matrix defined by the eigenvectors of $M((x+y)/2,\widetilde{W})$, and

$$\widetilde{\Lambda^{-1}}(x,y,U,V) = \mathrm{Diag}(\widetilde{\lambda_i^{-1}}(x,y,U,V), i = 1,\dots,N)$$

with

$$\widetilde{\lambda_i^{-1}} = \lambda_i^{-1} \quad \text{if } i \neq j$$

and

$$\widetilde{\lambda_j^{-1}} = \begin{cases} \lambda_j^{-1} & \text{if } L[(x,U),(y,V)] \subset \mathbf{R}^{N+1} \setminus \gamma,\\ 0 & \text{otherwise,}\end{cases} \tag{16}$$

where the λ_j are the eigenvalues of $M((x+y)/2,\widetilde{W})$ and $L[(x,U),(y,V)]$ is the closed segment in $\mathbf{R}^{N+1}$ that connects (x,U) and (y,V). It is straightforward to extend this definition to a more general case.

This definition of $\widetilde{M^{-1}}$ implies that when an eigenvalue vanishes, no upwinding of the source term is performed at the direction defined by the corresponding eigenvector. For nonhomogeneous scalar hyperbolic equations we may prove that this leads to stable schemes.

This definition of $\widetilde{M^{-1}}$ allows the schemes to be asymptotically well-balanced even for a class of stationary solutions for which matrix M is singular and, for convenient choices of the diffusion matrix D, for every stationary solution. We shall prove this in the next section.

. . . . 1. In [7] it is proved that for homogeneous conservative systems like (10) both flux-difference and flux-splitting schemes fit into the general structure of fluxes given by (12) and (13) with convenient choices of the centered numerical flux F_C and the numerical diffusion matrix D. We point out that for nonconservative systems, it makes no sense to consider flux-splitting schemes, as these are intrinsically based upon the conservative formulation. Indeed, these schemes use a splitting of the flux into (we assume $B = 0$)

$$F(W) = \mathcal{A}^*(W)\,W$$

for some matrix $\mathcal{A}^*$. However, in general we may use flux-difference schemes, as these are based upon the use of the velocities and speeds of propagation of the total flux, given by the matrix M.

In addition, other kind of schemes fit into the general framework set by the general structure (11). We shall introduce some of them in section 5.

3. Analysis of the "asymptotically well-balanced" property. The purpose of this section is to analyze the well-balanced property of the numerical schemes of the form (10)–(14). This analysis is performed by considering an arbitrary stationary solution and using Taylor expansions. As a consequence, the results are valid for every stationary solution (and not only for particular cases of "water at rest" solutions for the shallow-water equations), but we shall only be able to analyze the well-balanced property in terms of orders of accuracy, and not exactly at grid nodes.

Due to the lack of regularity of M^{-1} (and also to the low regularity of D, usually only Lipschitz continuous) we cannot hope to balance steady solutions of system (1) up to second order of accuracy in the whole domain $(0, L)$. However, we shall prove that this property holds on a subset of $(0, L)$ except for a subset whose measure tends to zero as Δx tends to zero for a large class of stationary solutions. We will say in this case that the numerical scheme is "asymptotically well-balanced."

Based upon these observations, we introduce the following definitions.

DEFINITION 1 (second order well-balanced schemes). [illegible] (11)–(14) [illegible] $W(x)$ [illegible] (1) [illegible] x_i [illegible] $(0, L)$

The following definition was introduced in [6].

DEFINITION 2 (asymptotically well-balanced schemes). [illegible] (11)–(14) [illegible] $W(x)$ [illegible] (1) [illegible] $\{K_n\}_n$ [illegible] $[0, L]$ [illegible]

(1) $\mu((0, L) \setminus \cup_n K_n) = 0$ [illegible] μ [illegible] $\mathbf{R}$

(2) [illegible] n [illegible] $\delta_n > 0$ [illegible], $0 < \Delta x < \delta_n$ [illegible] (1) [illegible] K_n

To perform our analysis, we will assume some hypotheses on the different elements defining the numerical scheme (11)–(14). Although these are somewhat cumbersome, this will allow us to extend the analysis to a rather large class of schemes:

HYPOTHESIS 1. (a) [illegible] C^1 2 [illegible] $F_1 : [0, L] \times \mathbf{R}^N \to \mathcal{L}(\mathbf{R}^{n+1}, \mathbf{R}^n)$ [illegible] $\forall U, V \in \mathbf{R}^N \quad \forall x, y \in [0, L]$

$$F_C(x, y, U, V) = F\left(\frac{x+y}{2}, \frac{U+V}{2}\right) + F_1\left(\frac{x+y}{2}, \frac{U+V}{2}\right)\begin{pmatrix} y-x \\ V-U \end{pmatrix} + o(|x-y|^2 + |U-V|^2).$$

(b) [illegible] C^1 3 [illegible] $B_{1,L}$ [illegible] $B_{1,R} : [0, L] \times \mathbf{R}^N \to \mathcal{L}(\mathbf{R}^{n+1}, \mathcal{L}(\mathbf{R}^n, \mathbf{R}^n))$ [illegible] $\forall U, V \in \mathbf{R}^N \quad \forall x, y \in [0, L]$

$$\mathcal{B}_{L/R}(x, y, U, V) = B\left(\frac{x+y}{2}, \frac{U+V}{2}\right) + B_{1,L/R}\left(\frac{x+y}{2}, \frac{U+V}{2}\right)\begin{pmatrix} y-x \\ V-U \end{pmatrix} + o(|x-y|^2 + |U-V|^2).$$

(c) $\forall x, y, z, \in [0, L] \quad \forall U, V, W \in R^N$

$$G_C(x, y, z, U, V, W) = G(y, V) + \mathcal{O}(|y - x|^2 + |z - y|^2 + |V - U|^2 + |W - V|^2).$$

(d) [illegible] C^1 [illegible] $S_1 : [0, L] \times \mathbf{R}^N \to \mathcal{L}(\mathbf{R}^{n+1}, \mathbf{R}^n)$ [illegible] $\forall x, y \in [0, L] \quad \forall U, V, \in \mathbf{R}^N$

$$S_D(x, y, U, V) = S\left(\frac{x+y}{2}, \frac{U+V}{2}\right) + S_1\left(\frac{x+y}{2}, \frac{U+V}{2}\right)\begin{pmatrix} y-x \\ V-U \end{pmatrix} + \mathcal{O}(|x-y|^2 + |V-U|^2).$$

(e) [illegible] $\gamma \subset \mathbf{R}^{N+1}$ [illegible] M [illegible] C^2 [illegible] $\mathbf{R}^{N+1} \setminus \gamma$ [illegible] 3 [illegible] $M_1 : \mathbf{R}^{N+1} \setminus \gamma \to \mathcal{L}(\mathbf{R}^{n+1}, \mathcal{L}(\mathbf{R}^n, \mathbf{R}^n))$ [illegible] C^1 [illegible] $\mathbf{R}^{N+1} \setminus \gamma$ [illegible]

$$\widetilde{M^{-1}}(x, y, U, V) = M^{-1}\left(\frac{x+y}{2}, \frac{U+V}{2}\right) + M_1\left(\frac{x+y}{2}, \frac{U+V}{2}\right)\begin{pmatrix} y-x \\ V-U \end{pmatrix} + \mathcal{O}(|x-y|^2 + |V-U|^2)$$

[illegible] (x, U) (y, V) [illegible] $L[(x, U), (y, V)] \subset \mathbf{R}^{N+1} \setminus \gamma$

(f) [illegible] $\widetilde{\gamma} \subset \mathbf{R}^{N+1}$ [illegible] 3 [illegible] $D_1 : \mathbf{R}^{N+1} \setminus \widetilde{\gamma} \to \mathcal{L}(\mathbf{R}^{n+1}, \mathcal{L}(\mathbf{R}^n, \mathbf{R}^n))$ [illegible] C^1 [illegible] $\mathbf{R}^{N+1} \setminus \widetilde{\gamma}$ [illegible] (x, U) (y, V) [illegible] $L[(x, U), (y, V)] \subset \mathbf{R}^{N+1} \setminus \widetilde{\gamma}$

$$D(x, y, U, V) = \mathcal{D}\left(\frac{x+y}{2}, \frac{U+V}{2}\right) + D_1\left(\frac{x+y}{2}, \frac{U+V}{2}\right)\begin{pmatrix} y-x \\ V-U \end{pmatrix} + \mathcal{O}(|y-x|^2 + |V-U|^2),$$

[illegible] E F $\mathcal{L}(E, F)$ [illegible]

Hypotheses (a), (b), and (c) state that F_C, $(\mathcal{B}_L + \mathcal{B}_R)/2\Delta x$, and G_C, respectively, are second order approximations of F, $B\partial W/\partial x$, and G. Hypothesis (e) formalizes the fact that M^{-1} presents singularities at points where one of its eigenvalues vanishes. Hypothesis (f) takes into account the fact that usually matrix D will present only Lipschitz regularity when one of its eigenvalues vanishes. A typical choice is, for example, $D = |M|$.

To state our main result, we need another technical definition.

DEFINITION 3. [illegible] $W : [0, L] \to \mathbf{R}^N$ [illegible] "W [illegible] (11)–(14) [illegible] $[0, L]$ [illegible]

$$\omega(W) = \{x \in (0, L) \text{ [illegible] } W(x) \in (\mathbf{R}^N \setminus \gamma) \cap (\mathbf{R}^N \setminus \widetilde{\gamma})\},$$

$$\sigma(W) = (0, L) \setminus \omega(W).$$

Then we have the following.

THEOREM 1. [illegible] $W : [0, L] \to \mathbf{R}^N$ [illegible] C^2 [illegible] (1) [illegible]

(a) [illegible] $M(x, W)$ [illegible] (11)–(14) [illegible] (1) [illegible] W [illegible]

(b) [illegible] W $\sigma(W)$ [illegible] (11)–(14) [illegible] (1) [illegible] W.

(c) [illegible] D [illegible] M [illegible] D [illegible] M [illegible] (11)–(14) [illegible] (1) [illegible] W

The proof of this result is given in the appendix, as it essentially uses Taylor expansions and finite-dimensional compactness arguments.

We also have the following result that yields sufficient conditions for the exact calculation of stationary solutions at grid points.

THEOREM 2. [illegible] W [illegible] (1) [illegible] F_C G_C S_D $\mathcal{B}_L$ $\mathcal{B}_R$ [illegible] W [illegible]

$$\frac{F_C(x_i, x_{i+1}, W_i, W_{i+1}) - F_C(x_{i-1}, x_i, W_{i-1}, W_i)}{\Delta x} = G_C(x_{i-1}, x_i, x_{i+1}, W_{i-1}, W_i, W_{i+1})$$

$$+ \frac{1}{2}\left(\mathcal{B}_L(x_{i-1}, x_i, W_{i-1}, W_i)\frac{W_i - W_{i-1}}{\Delta x} + \mathcal{B}_R(x_i, x_{i+1}, W_i, W_{i+1})\frac{W_{i+1} - W_i}{\Delta x}\right), \tag{17}$$

$$\frac{W_{i+1} - W_i}{\Delta x} = \widetilde{M^{-1}}(x_i, x_{i+1}, W_i, W_{i+1}) S_D(x_i, x_{i+1}, W_i, W_{i+1}). \tag{18}$$

[illegible] (11)–(14) [illegible] (1) [illegible] $D(x, y, U, V)$

To prove this theorem it is enough to verify that under these hypotheses the numerical scheme is reduced to $W_i^{n+1} = W_i^n$.

These results mean that the general structure of our schemes allows us to separately balance the numerical flux and the numerical diffusion terms, in a way largely independent of the discretization parameters appearing in each of them.

In fact, the role of the matrix D is mainly to give some stability to the scheme and not to contribute to the balancing of source terms. It remains, however, to construct matrices D that satisfy Hypothesis 1(f) of Theorem 1 in order to ensure the well-balanced property for all stationary solutions. The construction of such matrices is discussed in section 4.

[illegible] 2. The definition of $\widetilde{M^{-1}}$ given by (15) satisfies Hypothesis 1(e) if the intermediate state $\widetilde{W}$ satisfies for (x, U) and (y, V) close enough

$$\widetilde{W}(U, V) = \frac{U + V}{2} + \rho\left(\frac{U + V}{2}\right)(V - U) + \mathcal{O}(|V - U|^2), \tag{19}$$

where $\rho : \mathbf{R}^N \to \mathbf{R}$ is a C^1 function. Indeed, if $L[(x, U), (y, V)] \subset \mathbf{R}^{N+1} \setminus \gamma$, as $(\frac{x+y}{2}, \widetilde{W}(U, V)) \in \mathbf{R}^{N+1} \setminus \gamma$ and M^{-1} is C^2 in $\mathbf{R}^{N+1} \setminus \gamma$, we deduce

$$\widetilde{M^{-1}}(x, y, U, V) = M^{-1}\left(\frac{x + y}{2}, \widetilde{W}(U, V)\right) = M^{-1}\left(\frac{x + y}{2}, \frac{U + V}{2}\right)$$

$$+ \partial_W(M^{-1})\left(\frac{x + y}{2}, \frac{U + V}{2}\right)\left(\widetilde{W}(U, V) - \frac{U + V}{2}\right) + \mathcal{O}\left(\left|\widetilde{W}(U, V) - \frac{U + V}{2}\right|^2\right)$$

$$= M^{-1}\left(\frac{x + y}{2}, \frac{U + V}{2}\right) + \partial_W(M^{-1})\left(\frac{x + y}{2}, \frac{U + V}{2}\right)\rho\left(\frac{U + V}{2}\right)(V - U) + \mathcal{O}(|V - U|^2).$$

We thus obtain Hypothesis 1(e).

REMARK 3. Hypothesis 1(f) is also recovered if the diffusion matrix is defined similarly to $\widetilde{M^{-1}}$:

$$D(x,y,U,V) = X((x+y)/2, \widetilde{W})\widetilde{\Lambda}_D(x,y,U,V)X^{-1}((x+y)/2, \widetilde{W}),$$

where X denotes the matrix composed by the eigenvectors of $\mathcal{D}$, and

$$\widetilde{\Lambda}_D(x,y,U,V) = \mathrm{Diag}(\widetilde{d}_j(x,y,U,V), j = 1,\dots,N)$$

with

$$\widetilde{d}_j(x,y,U,V) = \begin{cases} d_j & \text{if } L[(x,U),(y,V)] \subset \mathbf{R}^N \setminus \gamma; \\ 0 & \text{if not,} \end{cases} \tag{20}$$

where d_j denotes the eigenvalues of $\mathcal{D}((x+y)/2, \widetilde{W})$. Notice that this is the usual way of defining the diffusion matrix.

REMARK 4. Property (19) is satisfied by intermediate states of usual solvers such as Van Leer's or Roe's for shallow-water equations.

REMARK 5. Equalities (17) and (18) are discrete versions of the identities

$$\frac{\partial}{\partial x}F(x,W(x)) = G(x,W(x)) + B(W(x))\frac{\partial W}{\partial x}$$

and

$$\frac{\partial W}{\partial x}(x) = M^{-1}(x,W(x)) \cdot S(x,W(x)).$$

4. Some possible choices of matrix D. In this section we discuss some possible choices of the diffusion matrix D with a double purpose: to ensure both the asymptotically well-balanced property for all steady smooth solutions and the stability of the numerical scheme at least for the linear homogeneous case.

To start with this last purpose, we consider the linear system

$$\frac{\partial W}{\partial t} + A\frac{\partial W}{\partial x} = B\frac{\partial W}{\partial x}, \tag{21}$$

where A and B are constant matrices. We suppose that the system is hyperbolic, that is, the matrix $M = A - B$ has N real different eigenvalues $m_j \in \mathbf{R}$, $j = 1,\dots,m$.

We study under which conditions the family of schemes previously introduced is L^2 stable. To do this, we observe that these schemes can be rewritten in viscous form, so the L^2 stability condition is given from the following result (cf. Godlewsky and Raviart [12])

THEOREM 3. [illegible] $d_j \in \mathbf{R}$ $j = 1,\dots,N$ [illegible] D [illegible] D [illegible] M [illegible]

$$\left(\frac{\Delta t}{\Delta x}m_j\right)^2 \le \frac{\Delta t}{\Delta x}d_j \le 1, \quad j = 1,\dots,N, \tag{22}$$

[illegible] ^{2} [illegible]

We look for choices of D satisfying the hypothesis of the previous theorem (and therefore those of statement (c) of Theorem 1 and Hypothesis 1(f)). Some possible choices of matrix D satisfying these conditions are the following:

If $M = X\Lambda X^{-1}$ with $\Lambda = \mathrm{diag}(m_j, j = 1, \ldots, N)$, then

$$D = X\Lambda_D X^{-1} \quad \text{with} \quad \Lambda_D = \mathrm{diag}(d_j, j = 1, \ldots, N)$$

and

- $d_j = |m_j|$,
- $d_j = \frac{\Delta t}{\Delta x} m_j^2$,
- $d_j = (1 - \varphi(r))|m_j| + \varphi(r)\frac{\Delta t}{\Delta x} m_j^2$, where $\varphi(r)$ is a flux-limiter (cf. [12], [15]).

Moreover, (22) gives the corresponding CFL condition to set the time step in terms of the solution and the space step.

. . . . 6. We observe that for previous definitions, the singularities of matrix $\widetilde{M^{-1}}$ and D are located at same points, so the hypersurfaces γ and $\widetilde{\gamma}$ coincide. Moreover, as matrices M and D have the same eigenvectors, we obtain

$$\widetilde{M^{-1}}(x, y, U, V) D(x, y, U, V) = X((x + y)/2, \widetilde{W})\, \Lambda(x, y, U, V)\, X^{-1}((x + y)/2, \widetilde{W})$$

with $\Lambda = \mathrm{Diag}(\widetilde{\lambda_j^{-1}}\, \widetilde{d}_j, j = 1, \ldots, N)$, where by (16) and (20) we have

$$\widetilde{\lambda_j^{-1}}\widetilde{d}_j(x, y, U, V) = \begin{cases} \lambda_j^{-1} d_j & \text{if } L[(x, U), (y, V)] \subset \mathbf{R}^N \setminus \gamma; \\ 0 & \text{if not.} \end{cases} \tag{23}$$

Finally, we also observe that in the last case we obtain a zero in the jth diagonal entry because $\widetilde{\lambda_j^{-1}} = 0$, and $\widetilde{d}_j = 0$. In practice, in this case we define directly $\widetilde{M^{-1}D}$ by (23).

5. Application to the two-layer shallow-water system. In this section we apply the general discretization technique previously introduced to the numerical solution of the one-dimensional two-layer shallow-water system for channels with arbitrary cross-sections. We start by describing the model equations. Next, we propose a family of numerical schemes by defining the different components of the general scheme (11)–(14) in such a way that Hypotheses 1(a)–(f) are satisfied. We also discuss some possible choices for the diffusion matrix D.

5.1. Equations. Let us consider the flow of a stratified fluid along an open channel. The channel is supposed to have a straight axis and be symmetric with respect to a vertical plane passing through its axis. The channel shape of cross-sections is given by the function $\sigma(x, z)$ and the bottom is given by the function $z_b(x)$. The fluid is assumed to be composed of two shallow layers of immiscible fluids of constant density, ρ_1 and ρ_2. Moreover, we assume that the flow is one-dimensional, i.e., at every layer the velocities $v_i(x, t)$, $i = 1, 2$, are uniform over the cross-section, and the thickness depends only on the x coordinate and the time. In the following sections, index 1 makes reference to the upper layer and index 2 to the lower one (see Figure 1).

Let r be the ratio of densities $r = \rho_1/\rho_2$. Let $Q_i(x, t) = v_i(x, t)A_i(x, t)$ be the discharge of the ith layer, with $A_i(x, t)$ the wetted cross-section of the ith layer at position x and time t. The thickness of layer i is given by $h_i(x, t)$. Therefore, A_i, $\sigma(x, z)$, h_i, and z_b are related through the equations

$$A_1 = \int_{z_b+h_2}^{z_b+h_2+h_1} \sigma(x, z)dz, \quad A_2 = \int_{z_b}^{z_b+h_2} \sigma(x, z)dz.$$

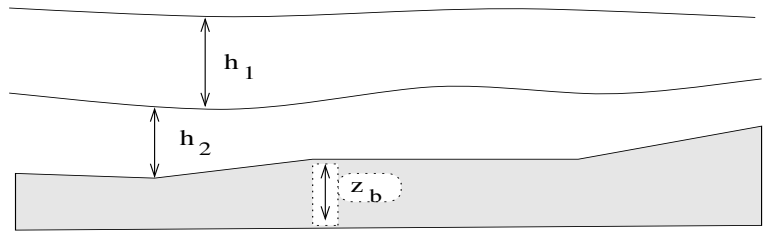


FIG. 1. *Sketch of a bilayer fluid.*

In [3] a PDE system modeling such a flow was deduced by imposing the mass and momentum conservations principles. Here, we consider the reformulation introduced in [4]:

$$\frac{\partial \mathrm{W}}{\partial t}(x,t) + \frac{\partial \mathrm{F}}{\partial x}(\varsigma(x,t), \mathrm{W}(x,t)) = G(x, \varsigma(x,t), \sigma^b(x), \mathrm{W}(x,t)) + B(\varsigma(x,t), \mathrm{W}(x,t))\frac{\partial \mathrm{W}}{\partial x} \quad \forall x \in (0,L),\ \ t \in (0,T), \tag{24}$$

where the unknowns are

$$\mathrm{W}(x,t) = \begin{bmatrix} W_1(x,t) \\ W_2(x,t) \end{bmatrix}, \quad W_j = \begin{bmatrix} A_j \\ Q_j \end{bmatrix}, \quad j = 1,2,$$

and $\varsigma = \left[\begin{smallmatrix} \sigma_1 \\ \sigma_2 \end{smallmatrix}\right]$, where σ_1 and σ_2 are defined by

$$\sigma_1(x,t) = \sigma(x, z_b(x) + h_1(x,t) + h_2(x,t)),$$

$$\frac{1}{\sigma_2(x,t)} = \frac{1-r}{\sigma_3(x,t)} + \frac{r}{\sigma_1(x,t)},$$

with

$$\sigma_3(x,t) = \sigma(x, z_b(x) + h_2(x,t)).$$

We also define

$$\sigma^b(x) = \sigma(x, z_b(x)).$$

The flux function, the coupling (nonconservative) terms, and the source terms are given, respectively, by

$$\mathrm{F}(\varsigma, \mathrm{W}) = \begin{bmatrix} F_1(\sigma_1, W_1) \\ F_2(\sigma_2, W_2) \end{bmatrix}, \quad F_j(\sigma_j, W_j) = \begin{bmatrix} Q_j \\ \dfrac{Q_j^2}{A_j} + \dfrac{g}{2\sigma_j} A_j^2 \end{bmatrix}, \quad j = 1,2,$$

$$B(\varsigma, \mathrm{W}) = \begin{bmatrix} 0 & B_1(\varsigma, W_1) \\ B_2(\varsigma, W_2) & 0 \end{bmatrix},$$

with

$$B_1(\varsigma, W_1) = \begin{bmatrix} 0 & 0 \\ -g\dfrac{1}{\sigma_1} A_1 & 0 \end{bmatrix}, \quad B_2(\varsigma, W_2) = \begin{bmatrix} 0 & 0 \\ -g\dfrac{r}{\sigma_1} A_2 & 0 \end{bmatrix},$$

and

$$G(x,\varsigma,\sigma^b,\mathrm{W}) = \mathrm{V}(\varsigma,\mathrm{W}) + \mathbf{S}(x,\varsigma,\sigma^b,\mathrm{W}),$$

with

$$\mathrm{V}(\varsigma,\mathrm{W}) = \begin{bmatrix} V_1(\sigma_1,W_1) \\ V_2(\sigma_2,W_2) \end{bmatrix}, \quad V_j(\sigma_j,W_j) = \begin{bmatrix} 0 \\ \dfrac{g}{2}\left(\dfrac{1}{\sigma_j}\right)_x A_j^2 \end{bmatrix}, \quad j=1,2.$$

We denote by $\mathbf{S}$ the source terms due to the variations of the geometry (depth and cross-section variations). If we define $\omega = z_b + h_1 + h_2$ and $\omega_r = z_b + h_2 + r\,h_1$, then $\mathbf{S}$ can be written

$$\mathbf{S} = \begin{pmatrix} 0 \\ g\dfrac{A_1}{\sigma_1}\dfrac{\partial}{\partial x}(A_1 + A_2 - \omega) \\ 0 \\ gA_2\dfrac{1}{\sigma_2}\dfrac{\partial A_2}{\partial x} + gA_2\dfrac{r}{\sigma_1}\dfrac{\partial A_1}{\partial x} - gA_2\dfrac{\partial \omega_r}{\partial x} \end{pmatrix}. \tag{25}$$

In order to easily describe the discretization of the terms V_j, we rewrite them as follows:

$$V_j(\sigma_j,W_j) = \begin{bmatrix} 0 \\ \dfrac{g}{2}\left(\dfrac{A_j^2}{\sigma_j}\right)_x - g\dfrac{A_j}{\sigma_j}(A_j)_x \end{bmatrix}, \quad j=1,2.$$

These terms appear due to the dependence of the flux F on the spatial variable x. In fact, $V = \partial_\varsigma F\ \partial_x \varsigma$. Therefore

$$G(x,\varsigma,\sigma^b,\mathrm{W}) - \mathrm{F}_\varsigma(\varsigma,\mathrm{W})\,\partial_x\varsigma(x) = \mathbf{S}(x,\varsigma,\sigma^b,\mathrm{W}).$$

It is easy to verify that this system can be written in matrix form as follows:

$$\frac{\partial W}{\partial t} + \mathcal{A}(\varsigma,\mathrm{W})\frac{\partial W}{\partial x} = \mathbf{S}(x,\varsigma,\sigma^b,\mathrm{W}), \tag{26}$$

where $\mathcal{A}$ is the 4×4 matrix whose expression is given by

$$\mathcal{A}(\sigma,\mathrm{W}) = J(\varsigma,\mathrm{W}) - B(\varsigma,\mathrm{W}),$$

with

$$J(\varsigma,\mathrm{W}) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\dfrac{Q_1^2}{A_1^2} + \dfrac{g}{\sigma_1}A_1 & 2\dfrac{Q_1}{A_1} & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\dfrac{Q_2^2}{A_2^2} + \dfrac{g}{\sigma_2}A_2 & 2\dfrac{Q_2}{A_2} \end{bmatrix}.$$

The eigenvalues of $\mathcal{A}$ can be classified in two external and two internal eigenvalues. The external eigenvalues, $\lambda^{e\pm}$, are related to the propagation speed of barotropic

perturbations and the internal ones, $\lambda^{i\pm}$, to the propagation of baroclinic perturbations. In the case $r \cong 1$, first order approximations of the eigenvalues were given in [20]. In most of the applications to geophysical flows, one has $\lambda^{e-} < 0$ and $\lambda^{e+} > 0$. Therefore, the transitions depend on the sign of the internal eigenvalues: the flow is said to be [illegible] if the sign of the internal eigenvalues differs, [illegible] if one of them takes the value zero; otherwise the flow is called [illegible]. It can be deduced from the expression of the matrix $\mathcal{A}$ that critical, supercritical, and subcritical sections can be characterized by $G^2 = 1$, $G^2 > 1$, $G^2 < 1$, respectively, where

$$G^2 = F_1^2 + F_2^2 - (1-r)\frac{\sigma_2}{\sigma_3}F_1^2F_2^2 \tag{27}$$

and

$$F_1^2 = \frac{v_1^2}{g'\left(\frac{\sigma_2}{\sigma_3}\right)\left(\frac{A_1}{\sigma_1}\right)}, \quad F_2^2 = \frac{v_2^2}{g'\left(\frac{A_2}{\sigma_3}\right)}. \tag{28}$$

In these definitions, $g' = g(1-r)$ is the reduced gravity, and G and F_i, $i = 1, 2$, are the appropriate definitions for this case of the [illegible] and the [illegible], respectively.

The internal eigenvalues may become complex corresponding to the development of shear instabilities. Nevertheless, in the description of the numerical schemes only the case in which the matrix $\mathcal{A}$ has 4 different real eigenvalues is considered, i.e., the flow is supposed to be stable and the system strictly hyperbolic.

5.2. Discretization of the system. In this subsection we introduce a one-parameter numerical scheme to solve system (24), with the structure (11)–(14). In order to obtain asymptotically well-balanced schemes, we must define the different components of the numerical scheme to fit the general Hypotheses 1(a)–(f).

We will use the following notation:

$$\mathrm{W}_{i+\alpha} = (1-\alpha)\mathrm{W}_i + \alpha\mathrm{W}_{i+1}, \quad \alpha \in [0,1].$$

We also consider an analogous notation for ς:

$$\varsigma_{i+\alpha} = \begin{bmatrix} \sigma_{1,i+\alpha} \\ \sigma_{2,i+\alpha} \end{bmatrix},$$

with

$$\sigma_{1,i+\alpha} = (1-\alpha)\sigma_{1,i} + \alpha\sigma_{1,i+1}, \quad \sigma_{3,i+\alpha} = (1-\alpha)\sigma_{3,i} + \alpha\sigma_{3,i+1}$$

and $\sigma_{2,i+\alpha}$ defined by

$$\frac{1}{\sigma_{2,i+\alpha}} = \frac{1-r}{\sigma_{3,i+\alpha}} + \frac{r}{\sigma_{1,i+\alpha}}.$$

With this notation, following [7] we define the centered approximation of the flux function by

$$F_C(x_i, x_{i+1}, W_i, W_{i+1}) = \frac{F(\varsigma_{i+\alpha}, \mathrm{W}_{i+\alpha}) + F(\varsigma_{i+1-\alpha}, \mathrm{W}_{i+1-\alpha})}{2}, \tag{29}$$

with $\alpha \in [0,1]$. This approximation satisfies Hypothesis 1(a).

We propose to use this approximation of F_C due to our former experience with a similar one-parameter family of numerical schemes for solving the one-layer shallow-water system. In this case, a good choice of the parameter α allows us to correctly approximate solutions including sonic rarefaction waves without the use of entropy corrections. In particular, in [7] the value $\alpha = 1/8$ is proposed.

Matrix $\widetilde{M^{-1}}$ is defined by (15). In practice we set $\widetilde{\lambda_j^{-1}} = 0$ if the absolute value of λ_j is smaller than a certain ϵ nearly to zero. The decentered source terms $\mathcal{B}_L$ and $\mathcal{B}_R$ are given by

(30)
$$\mathcal{B}_L(x_{i-1}, x_i, W_{i-1}, W_i) = \alpha \mathbf{B}\left(\varsigma_{i-\alpha/2}, \mathrm{W}_{i-\alpha/2}\right) + (1-\alpha)\mathbf{B}\left(\varsigma_{i-(1-\alpha)/2}, \mathrm{W}_{i-(1-\alpha)/2}\right)$$

and

(31)
$$\mathcal{B}_R(x_i, x_{i+1}, W_i, W_{i+1}) = \alpha \mathbf{B}\left(\varsigma_{i+\alpha/2}, \mathrm{W}_{i+\alpha/2}\right) + (1-\alpha)\mathbf{B}\left(\varsigma_{i+(1-\alpha)/2}, \mathrm{W}_{i+(1-\alpha)/2}\right).$$

To complete the definition of ϕ_L^S and ϕ_R^S we must define a discretization of S in $x = x_{i+1/2}$, $S_D(x_i, x_{i+1}, W_i, W_{i+1})$, satisfying Hypothesis 1(c). We propose the following one:

(32)
$$\mathcal{S}_D(x_i, x_{i+1}, W_i, W_{i+1}) = \frac{1}{\Delta x}\begin{pmatrix} 0 \\ g\dfrac{A_{1,i+1/2}}{\sigma_{1,i+1/2}}(A_{1,i+1} + A_{2,i+1} - \omega_{i+1} - A_{1,i} - A_{2,i} + \omega_i) \\ 0 \\ g\dfrac{A_{2,i+1/2}}{\sigma_{2,i+1/2}}(A_{2,i+1} - A_{2,i}) + gr\dfrac{A_{2,i+1/2}}{\sigma_{1,i+1/2}}(A_{1,i+1} - A_{1,i}) \\ -gA_{2,i+1/2}(\omega_{r,i+1} - \omega_{r,i}) \end{pmatrix}.$$

We define a centered discretization of G in $x = x_i$, G_C:

$$G_C = \mathbf{S}_i + V_i,$$

where $\mathbf{S}_i$ and V_i are centered approximations of $\mathbf{S}$ and V at $x = x_i$, respectively. They are constructed as follows:

$$\mathbf{S}_i = \frac{1}{2}\Big(\mathbf{S}_{i,L} + \mathbf{S}_{i,R}\Big), \tag{33}$$

where, if we denote by $[\cdot]_j$ the jth component of the vector$[\cdot]$, then

$$[\mathcal{S}_{i,L}]_1 = 0, \tag{34}$$

$$\Delta x[\mathcal{S}_{i,L}]_2 = g\frac{A_{1,i-\alpha/2}}{\sigma_{1,i-\alpha/2}}(A_{1,i} + A_{2,i} - \omega_i - A_{1,i-\alpha} - A_{2,i-\alpha} + \omega_{i-\alpha})$$

$$+ g\frac{A_{1,i-(1-\alpha)/2}}{\sigma_{1,i-(1-\alpha)/2}}(A_{1,i} + A_{2,i} - \omega_i - A_{1,i-1+\alpha} - A_{2,i-1+\alpha} + \omega_{i-1+\alpha}), \tag{35}$$

$$[\mathcal{S}_{i,L}]_3 = 0, \tag{36}$$

and

$$\Delta x[\mathcal{S}_{i,L}]_4 = g\frac{A_{2,i-\alpha/2}}{\sigma_{2,i-\alpha 1/2}}(A_{2,i}-A_{2,i-\alpha}) + g\,r\frac{A_{2,i-\alpha/2}}{\sigma_{1,i-\alpha/2}}(A_{1,i}-A_{1,i-\alpha})$$
$$- gA_{2,i-\alpha/2}(\omega_{r,i}-\omega_{r,i-\alpha}) + g\frac{A_{2,i-(1-\alpha)/2}}{\sigma_{2,i-(1-\alpha)/2}}(A_{2,i}-A_{2,i-1+\alpha})$$
$$+ g\,r\frac{A_{2,i-(1-\alpha)/2}}{\sigma_{1,i-(1-\alpha)/2}}(A_{1,i}-A_{1,i-1+\alpha}) - gA_{2,i-(1-\alpha)/2}(\omega_{r,i}-\omega_{r,i-1+\alpha}). \tag{37}$$

$\mathcal{S}_{i,R}$ is given by

$$[\mathcal{S}_{i,R}]_1 = 0, \tag{38}$$

$$\Delta x[\mathcal{S}_{i,R}]_2 = g\frac{A_{1,i+\alpha/2}}{\sigma_{1,i+\alpha/2}}(A_{1,i+\alpha}+A_{2,i+\alpha}-\omega_{i+\alpha}-A_{1,i}-A_{2,i}+\omega_i)$$
$$+ g\frac{A_{1,i-(1-\alpha)/2}}{\sigma_{1,i-(1-\alpha)/2}}(A_{1,i+1-\alpha}+A_{2,i+1-\alpha}-\omega_{i+1-\alpha}-A_{1,i}-A_{2,i}+\omega_i), \tag{39}$$

$$[\mathcal{S}_{i,R}]_3 = 0, \tag{40}$$

and

$$\Delta x[\mathcal{S}_{i,R}]_4 = g\frac{A_{2,i+\alpha/2}}{\sigma_{2,i+\alpha/2}}(A_{2,i+\alpha}-A_{2,i}) + g\,r\frac{A_{2,i-\alpha/2}}{\sigma_{1,i-\alpha/2}}(A_{1,i+\alpha}-A_{1,i})$$
$$- gA_{2,i-\alpha/2}(\omega_{r,i+\alpha}-\omega_i) + g\frac{A_{2,i+(1-\alpha)/2}}{\sigma_{2,i+(1-\alpha)/2}}(A_{2,i+1-\alpha}-A_{2,i})$$
$$+ g\,r\frac{A_{2,i+(1-\alpha)/2}}{\sigma_{1,i+(1-\alpha)/2}}(A_{1,i+1-\alpha}-A_{1,i}) - gA_{2,i+(1-\alpha)/2}(\omega_{r,i+1-\alpha}-\omega_{r,i}). \tag{41}$$

The centered discretization of the term V at point $x = x_i$, V_i, is given by

$$V_i = \begin{bmatrix} \mathcal{V}_1 \\ \mathcal{V}_2 \end{bmatrix}, \tag{42}$$

with

$$[\mathcal{V}_j]_1 = 0 \quad \text{for} \quad j = 1,2 \tag{43}$$

and

$$2\Delta x[\mathcal{V}_j]_2 = g\frac{A_{j,i+\alpha}A_{j,i+\alpha}}{2\sigma_{j,i+\alpha}} - g\frac{A_{j,i-1+\alpha}A_{j,i-1+\alpha}}{2\sigma_{j,i-1+\alpha}}$$
$$+ g\frac{A_{j,i+1-\alpha}A_{j,i+1-\alpha}}{2\sigma_{j,i+1-\alpha}} - g\frac{A_{j,i-\alpha}A_{j,i-\alpha}}{2\sigma_{j,i-\alpha}}$$
$$- g\frac{A_{j,i+\alpha/2}}{\sigma_{j,i+\alpha/2}}(A_{j,i+\alpha}-A_{j,i}) - g\frac{A_{j,i+(1-\alpha)/2}}{\sigma_{j,i+(1-\alpha)/2}}(A_{j,i+1-\alpha}-A_i)$$
$$- g\frac{A_{j,i-\alpha/2}}{\sigma_{j,i-\alpha/2}}(A_i-A_{j,i-\alpha}) - g\frac{A_{j,i-(1-\alpha)/2}}{\sigma_{j,i-(1-\alpha)/2}}(A_{j,i}-A_{j,i-1+\alpha}) \tag{44}$$

for $j = 1,2$.

It remains to define a matrix D satisfying Hypothesis 1(f). This may be done as indicated in section 4. In all the numerical tests discussed in section 6, we have chosen $D = |M|$, where M is the Roe-type matrix introduced in [4]. With this choice, the numerical scheme proposed in [4] is the particular case of the previous one corresponding to $\alpha = 0$.

5.3. Balance properties. As the different components defining the numerical scheme verify Hypotheses 1(a)–(f), we can apply Theorem 1 and conclude the following result.

THEOREM 4. (1) [illegible] D [illegible] 4 [illegible] (11)–(14), (29)–(44) [illegible] (24) [illegible]

(2) [illegible] D [illegible] $W(x)$ [illegible] $M(x, W(x))$ [illegible] $\forall x \in [0, L]$ [illegible] $M(x, W(x))$ [illegible] (24) [illegible] $W(x)$

We can also apply Theorem 2 to the stationary solutions corresponding to water at rest:

$$\overline{W} = \begin{bmatrix} A_1 \\ 0 \\ A_2 \\ 0 \end{bmatrix}, \quad \text{with} \quad A_1 = \int_{\overline{h}_2}^{\overline{h}_2+\overline{h}_1} \sigma(x,z)dz, \quad A_2 = \int_{z_b}^{\overline{h}_2} \sigma(x,z)dz, \tag{45}$$

with $\overline{h}_1$ constant and $\overline{h}_2 = h_2 + z_b(x)$ with $\overline{h}_2$ constant. In this solution the free surface of the fluid, given by $\overline{h}_1 + \overline{h}_2 + z_b$, is constant, as well as the interface given by $\overline{h}_2 + z_b$.

We have the following result.

THEOREM 5. [illegible] (29)–(44) [illegible] (24) [illegible] (45) [illegible] α [illegible] D

To prove this theorem it is enough to verify conditions (17) and (18) on the stationary solution (45).

[illegible] 7. As we previously mentioned, solving the one-layer shallow-water system with centered numerical fluxes F_C as (29) with a good choice of the parameter α ($\alpha = 1/8$) allows us to correctly approximate sonic rarefaction waves without the need for entropy-correction techniques. Also, in the case of stationary shocks the best choice is given by $\alpha = 0$. This suggests that a better method can be obtained if the parameter α is replaced by a function $\alpha(x)$ taking the value 0 at intercells where a stationary shock develops and 1/8 elsewhere.

Taking in mind that transitions depend on the sign of the internal eigenvalues, a suitable approximation $\alpha_{j+1/2}$ to $\alpha(x_{j+1/2})$ can be given by

$$\begin{aligned} &\alpha_{j+1/2} = \min(\alpha^-_{j+1/2}, \alpha^+_{j+1/2}), \quad \text{with} \\ &\alpha^\pm_{j+1/2} = \begin{cases} \bar{\alpha} & \text{if } \operatorname{sgn}(\lambda^{i\pm}_j) = \operatorname{sgn}(\lambda^{i\pm}_{j+1}), \\ \bar{\alpha}(1 - 0.5(1 + \operatorname{sgn}(\lambda^{i\pm}_j - \lambda^{i\pm}_{j+1}))) & \text{otherwise,} \end{cases} \end{aligned} \tag{46}$$

where $\bar{\alpha}$ is a constant value ($\bar{\alpha} = 1/8$) and $\lambda^{i\pm}_j$ are the internal eigenvalues of the matrix $M(x_j, W_j)$.

In most applications, this numerically defined function $\alpha(x)$ will be constant in all the domain but for a set whose measure tends to zero (a bounded number of cells), so that the balance properties are preserved.

6. Numerical tests. In this section, some experiments illustrating the properties of the proposed family of numerical schemes are presented. The purposes of these experiments are mainly to validate the numerical schemes and to investigate the

influence of the parameter α when the numerical viscosity vanishes. For this reason in all our tests we have chosen a unique numerical diffusion matrix D, given by the simplest choice in section 4, i.e., $D = |M|$.

The first experiment has been designed to verify whether the still-water solution is exactly computed in practice, even for very irregular geometries. The second one provides a numerical study of the order of the numerical scheme. This study confirms the theoretical results. In section 6.3 the influence of the parameter α in smooth transitions is investigated: we observe that any positive value yields good results. Finally, the last test has been designed to analyze the influence of the parameter α for a nonsmooth solution. In this case we see that the best results are obtained if the constant parameter α is replaced by the function α defined in Remark 6.

6.1. Water at rest in a channel with irregular geometry. In this example, a numerical test is presented to verify how accurately the solution (45), in practice, is computed for a channel with very irregular geometry. To do this, we have considered a 4-meter long channel with rectangular cross-section and for which depth and breadth were randomly generated. The numerical scheme was applied for the values of the parameter $\alpha = 0$, $1/8$, $1/4$, and $3/8$ (only the values for $\alpha = 1/8$ are shown, because all results are very similar), taking as initial condition a steady solution representing water at rest. The CFL parameter is set to 0.95 and $\Delta x = 0.05$.

Figures 2(a) and 2(b) show the bottom topography and the channel breadth, respectively. Figures 2(c) and 2(d) depict the free surface and the water interface at the initial time and after 15 seconds of simulation. Finally, Figures 2(e) and 2(f) show the value of the discharge at each layer at the initial state and after 15 seconds of simulation, respectively. As predicted in Theorem 5, for any value of the parameter, the water elevation at the free surface and the interface as well as the discharges at each layer are accurately computed: only round-off errors are present, and they do not grow with time.

6.2. Well-balancing test. In this section, we test the order of the numerical scheme when it is applied to approach a stationary solution of the two-layer shallow-water system with constant breadth function. The regular stationary solutions of this system satisfy the following:

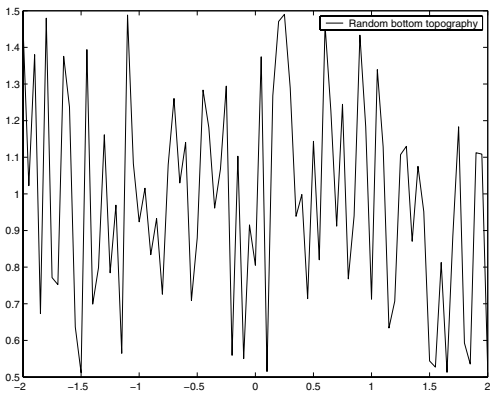
$$\begin{cases} Q_1 = \text{constant}, \\ \dfrac{v_1^2}{2} - \dfrac{v_2^2}{2} + g' h_1 = \text{constant}, \\ Q_2 = \text{constant}, \\ \dfrac{v_1^2}{2} + g(h_1 + h_2 + z_b) = \text{constant}, \end{cases} \tag{47}$$

where $g' = (1 - r)g$ is the reduced gravity.

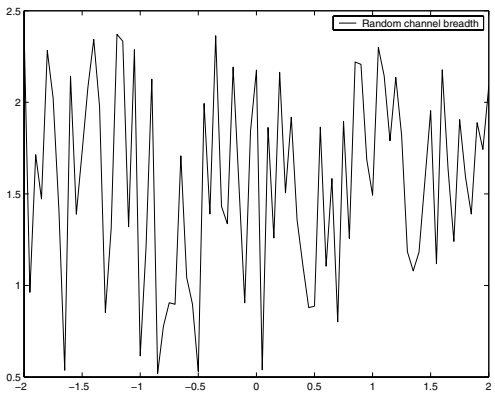
In this numerical test, we consider $\sigma(x, z) = 1.0$, $r = 0.98$, and the depth function

$$z_b(x) = 0.5e^{-x^2} - 2.0, \quad x \in [-3, 3].$$

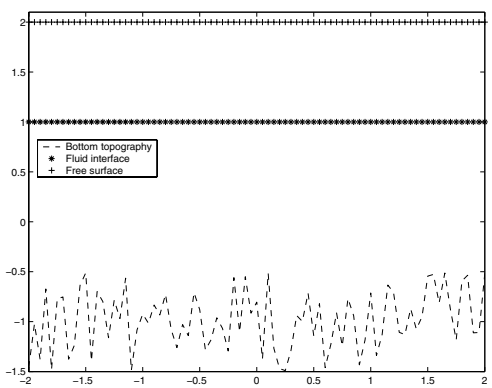
In order to obtain a particular stationary solution, we impose first its values at $x = -3$: $h_1(-3, 0) = 0.5$, $h_2(-3, 0) = z_b(-3) + 1.5$, $Q_1(-3, 0) = 0.15$, and $Q_2(-3, 0) = -0.15$. From these values, we compute the constants in (47). Once the constants are known,



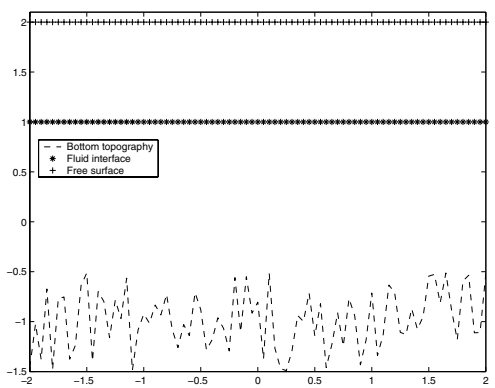
(a) Channel bottom topography (random).



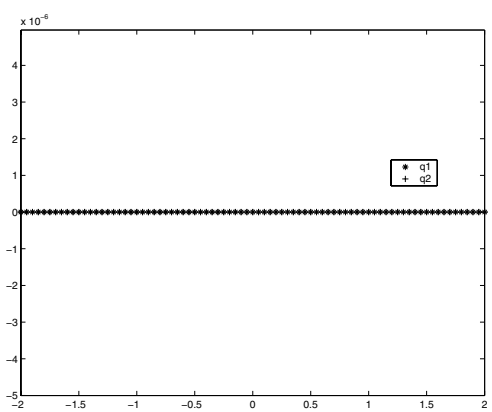
(b) Channel breadth (random).



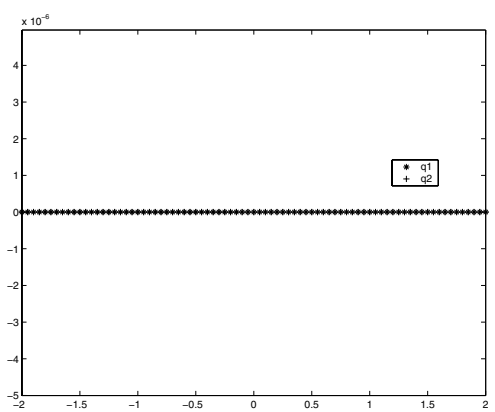
(c) Free surface and interface at initial state.



(d) Free surface and interface after 15s of simulation.



(e) Discharge at each layer at initial state.



(f) Discharge at each layer after 15s of simulation.

FIG. 2. *Water at rest in an irregular channel ($\alpha = 1/8$).*

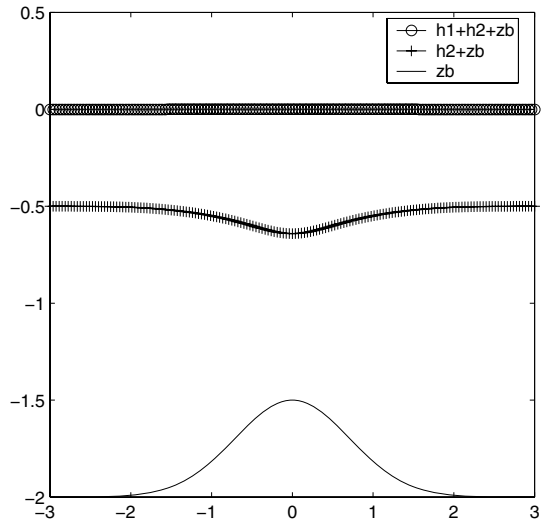


FIG. 3. *Stationary solution in the test case of section* 6.2. *Elevations* $\eta_1 = h_1 + h_2 + z_b(x)$, $\eta_2 = h_2 + z_b(x)$ *and bottom topography* $z_b(x)$.

the values of the solution at any point are obtained by solving (47). Figure 3 shows the computed elevations $\eta_1(x) = h_1(x) + h_2(x) + z_b(x)$ and $\eta_2(x) = h_2(x) + z_b(x)$ corresponding to this particular stationary solution.

In the numerical experiment, this stationary solution is taken as the initial condition. As the stationary solution chosen does not correspond to water at rest, the numerical scheme should be second order accurate.

We have applied the numerical scheme for different values of the parameter $\alpha = 0$, 1/8, 1/4, and 3/8 (only the values for $\alpha = 1/8$ are shown, because all results are very similar). The CFL coefficient has been taken as 0.9. The results obtained for $\alpha = 1/8$ are shown in Tables 1 and 2. As expected, the order obtained is 2.

6.3. Rectangular channel with a bump. The validation of numerical schemes for solving the two-layer shallow-water system with transcritical solutions is not a simple task, as exact solutions with smooth transitions cannot be easily obtained for this system. Nevertheless, Farmer and Armi introduced in [9], in the context of the study of exchange flows through channels, an asymptotic technique to obtain steady solutions for channels with simplified geometries. In this particular case, they consider a channel of uniform width with a sill on the floor, separating two broad and deep reservoirs.

Let us first briefly recall the main aspects of the theory developed by these authors. In their works, smooth steady state solutions are studied. The equations of the model are obtained by replacing the fourth equation in (47) by the ,,, , assumption:

$$h_1 + h_2 + z_b = \text{constant}. \tag{48}$$

The equations are first written in nondimensional form using some dimensionless variables Q_i', h_i', z_b' that are such that $h_1' + h_2' + z_b' = 1$ at the section of minimal depth. Then all the variables are expressed in terms of the internal Froude numbers, $F_i = v_i/\sqrt{(g' h_i)}$, $i = 1, 2$. Using the approximation $r = 1$ the following 2×2 nonlinear

TABLE 1
Test case of section 6.2 *solved with* $\alpha = 1/8$. h_1 *and* h_2.

No. cells	L^1 error h_1	L^1 order h_1	L^1 error h_2	L^1 order h_2
25	5.20×10^{-3}	–	5.50×10^{-3}	–
50	1.66×10^{-3}	1.65	1.87×10^{-3}	1.56
100	4.20×10^{-4}	1.98	4.83×10^{-4}	1.95
200	1.04×10^{-4}	2.02	1.15×10^{-4}	2.07
400	2.38×10^{-5}	2.12	2.72×10^{-5}	2.08
800	5.67×10^{-6}	2.07	6.48×10^{-6}	2.07

TABLE 2
Test case of section 6.2 *solved with* $\alpha = 1/8$. Q_1 *and* Q_2.

No. cells	L^1 error Q_1	L^1 order Q_1	L^1 error Q_2	L^1 order Q_2
25	8.42×10^{-4}	–	6.31×10^{-4}	–
50	2.30×10^{-4}	1.87	1.84×10^{-4}	1.78
100	5.96×10^{-5}	1.95	4.89×10^{-5}	1.91
200	1.38×10^{-5}	2.11	1.23×10^{-5}	1.99
400	3.11×10^{-6}	2.14	2.97×10^{-6}	2.05
800	6.91×10^{-7}	2.17	6.79×10^{-7}	2.13

system is obtained from the second equation of (47) with the assumption (48):

$$F_1^{-\frac{2}{3}} + \frac{1}{2}F_1^{\frac{4}{3}} - \frac{1}{2}q_r^{-\frac{2}{3}}F_2^{\frac{4}{3}} = \Delta H'(Q_1')^{-\frac{2}{3}}, \tag{49}$$

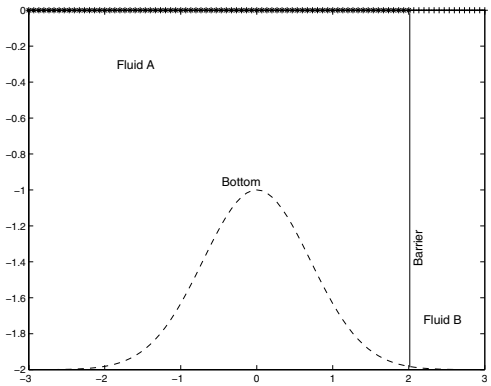
$$F_1^{-\frac{2}{3}} + q_r^{-\frac{2}{3}}F_2^{-\frac{2}{3}} = \left[\frac{Q_1'}{(1-z_b')^{\frac{3}{2}}}\right]^{-\frac{2}{3}}, \tag{50}$$

where $\Delta H'$ is a constant corresponding to the dimensionless energy difference between the two layers and $q_r = Q_1/Q_2$. Here, only the case $|q_r| = 1$ will be considered.

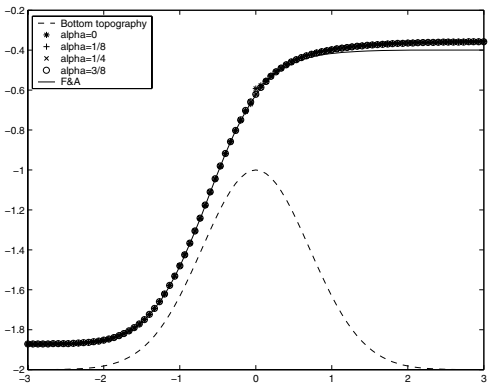
The solutions of the equations can be identified with the family of curves in the (F_1^2, F_2^2)-plane defined by (49). These curves have been plotted in detail in [9].

If the reservoirs are sufficiently deep, except in the neighborhood of the sill, the speed of the thicker layer is negligible away from the influence of the sill. In this case, the Froude numbers F_1 and F_2 have to be near zero far from the sill at the reservoir corresponding to water of densities ρ_1 and ρ_2, respectively. Observe that there is only one family of curves for which F_1 and F_2 are close to zero at the reservoir corresponding to water densities ρ_1 and ρ_2, respectively. Among the curves of this family, there is one for which the value of the exchanged flow Q_1' is maximal. This curve corresponds to $\Delta H' = 1.5$ and is called the [illegible] : it represents the steady state limit that occurs when the fluids are free to move in opposite directions across the sill. If the reservoir corresponding to water of density ρ_1 is placed to the left, this solution is critical at the section of minimal depth, subcritical to the right of this section, and supercritical to the left. If the condition $F_2 = 0$ is imposed at the exit section of the channel (that is, if the depth of the reservoir to the left is assumed to be infinity), the flow is critical again at this section.

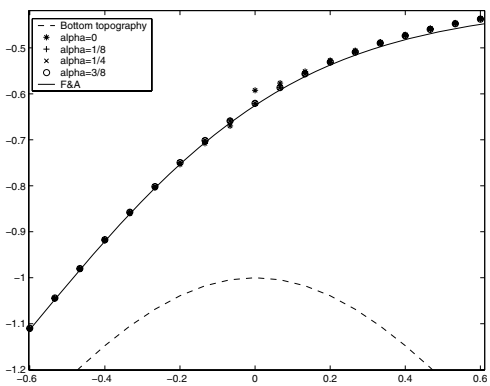
In the experiment, we consider a rectangular channel 1 meter wide and 6 meters long. Its bottom topography is defined by the function $z_b(x) = \exp(-x^2) - 2$, where $x \in [-3, 3]$. The ratio between fluxes is set to 1 and the ratio between densities is set



(a) Initial condition.



(b) Interface at the stationary state and comparison with F&A interface.



(c) Interface at the stationary state and comparison with F&A interface (zoom).

FIG. 4. *Maximal exchange flow through a rectangular channel with a single bump. Initial condition and comparison with F&A stationary interface* ($\bar{\alpha} = 0,\ 1/8,\ 1/4,\ 3/8$).

to $r = \rho_1/\rho_2 = 0.98$. In order to reproduce the maximal two-way exchange with the numerical model, we solve the classical [illegible]. This problem consists in taking as initial state the two fluids separated by a vertical "artificial barrier" that, in this case, is located at $x = 2$ (see Figure 4(a)). This barrier is dropped out at time $t = 0$ and the fluids are left to evolve until a stationary state is reached (see Figure 4(b)). The CFL parameter is set to 0.9 and $\Delta x = 1/15$.

The boundary conditions in this test have been designed to simulate the presence of the infinite reservoirs at both ends of the channel and the fact that $q_r = 1$: the two layers must be allowed to go out of the channel as freely as possible, with the only constraint $Q_1 = -Q_2$. To do this, we have modified the usual numerical treatment based on the introduction of two ghost cells and the duplication of states, which is used very often in practice to simulate open boundaries. The idea is as follows: let us suppose that, at the nth time iteration, the approximation obtained at the first cell is W_1^n. If the corresponding value of Q_2, $Q_{2,1}^n$, is equal to zero, the lower layer has not

yet reached the left boundary. In this case, the state is duplicated at the ghost cell,

$$\mathrm{W}_0^n = \mathrm{W}_1^n,$$

and the first cell is treated as an internal one during the calculation of the approximations at time t^{n+1}. If, instead, $Q_{2,1}^n$ is not equal to zero, then the state at the ghost cell is modified so that the relation $Q_1 = -Q_2$ holds. More precisely, we define

$$\mathrm{W}_0^n = \mathrm{W}_1^n - \frac{\Delta Q_1^n}{|Q_{1,1}^n| + |Q_{2,1}^n|} \begin{bmatrix} 0 \\ |Q_{1,1}^n| \\ 0 \\ |Q_{2,1}^n| \end{bmatrix},$$

with

$$\Delta Q_1^n = Q_{1,1}^n + Q_{2,1}^n.$$

Again, the first cell is treated as an internal one. A similar treatment is performed at the right boundary.

During the computations, complex eigenvalues appear. Following [10], when the appearance of complex eigenvalues is detected, a linear friction term is added to the momentum equations of both layers. Concretely, a source term of the form

$$-CQ_k, \quad k = 1, 2,$$

is added which is intended to simulate the loss of momentum induced by the instabilities in real flows. The local friction coefficient C is large enough to make the numerical scheme linearly stable. More precisely, we take $C = (\sqrt{1 + 2|m_2|\Delta t/\Delta x} - 1)/\Delta t$, where m_2 is the complex part of the eigenvalue. In [10] it has been shown that this choice ensures the L^2 stability in a simplified case.

Figures 4(b) and 4(c) show the interface at the steady state reached for the different values of the parameter $\alpha = 0$, 1/8, 1/4, and 3/8. These steady states are compared with the interface corresponding to the maximal two-way exchange obtained in [9] (in what follows, this solution will be called the F&A solution, after Farmer and Armi). To do this, once the value Q_1' has been calculated, the 2×2 nonlinear system (49)–(50) is solved numerically for every value of z_b' corresponding to the intercells of the mesh (the graphical study of the solutions performed in [9] gives the necessary hints about the adequate choice of the initial guess for the numerical algorithms).

Notice that all the numerical solutions differ slightly from the F&A solution at the right part of the channel (see Figure 4(b)). These differences are due mainly to the rigid-lid hypothesis. The main differences among the numerical solutions are located near the critical section, as expected. The parameter $\bar{\alpha}$ has practically no influence in regular zones, but it is important in transitions (regular or not). So, we focus only in a neighborhood of $x = 0$ ($x \in [-0.4, 0.4]$). Figure 4(c) shows the computed interfaces and the F&A interface in this zone. Notice that if the parameter α is set to 0, the numerical scheme corresponds to a Roe-type scheme and produces a nonentropy solution, as expected. Table 3 shows the L^∞ and L^1 errors corresponding to the comparison of the numerical interfaces with the analytical one. Observe that the values $\alpha = 1/8$, $\alpha = 1/4$, $\alpha = 3/8$ produce similar results.

The discharges computed by the model are $Q_1 = -Q_2 = 9.38\,10^{-2}$ m^3/s for $\alpha = 1/8$, 1/4, and 3/8, and $Q_1 = -Q_2 = 9.33\,10^{-2}$ m^3/s for $\alpha = 0$, while the F&A model gives $Q_1 = -Q_2 = 9.213\,10^{-2}$ m^3/s.

TABLE 3
L^∞ and L^1 errors (F&A interface against numerical computed ones) ($x = [-0.4, 0.4]$).

α	L^∞-error	L^1-error
0	0.0331	0.0074
1/8	0.0085	0.0045
1/4	0.0085	0.0045
3/8	0.0085	0.0046

TABLE 4
Test case of section 6.3. *h_1 and h_2.*

No. cells	L^1 error h_1	L^1 order h_1	L^1 error h_2	L^1 order h_2
45	6.47×10^{-3}	–	4.17×10^{-3}	–
90	2.50×10^{-3}	1.37	1.60×10^{-3}	1.38
180	8.46×10^{-4}	1.56	5.10×10^{-4}	1.65
360	2.35×10^{-4}	1.85	1.44×10^{-4}	1.83
720	6.20×10^{-5}	1.92	3.77×10^{-5}	1.93

TABLE 5
Test case of section 6.3. *Q_1 and Q_2.*

No. cells	L^1 error Q_1	L^1 order Q_1	L^1 error Q_2	L^1 order Q_2
45	4.36×10^{-3}	–	4.47×10^{-3}	–
90	1.88×10^{-3}	1.21	1.93×10^{-3}	1.21
180	7.60×10^{-4}	1.31	7.74×10^{-4}	1.32
360	2.74×10^{-4}	1.47	2.79×10^{-4}	1.47
720	8.62×10^{-5}	1.67	8.72×10^{-5}	1.68

TABLE 6
Test case of section 6.3. *L^1 errors: F&A solution against numerical stationary solution on a finer mesh (*11520 *cells).*

L^1 error h_1	L^1 error h_2	L^1 error Q_1	L^1 error Q_2
1.14×10^{-2}	8.60×10^{-2}	8.16×10^{-3}	8.16×10^{-3}

In order to test the convergence, a numerical stationary solution over a finer mesh of 11520 cells has been computed. The L^1 errors for the variables h_1, h_2, Q_1, and Q_2 are shown in Tables 4 and 5 for several meshes with increasing numbers of cells using the numerical scheme, with the parameter α replaced by the function $\alpha(x)$ defined in Remark 7 with $\bar{\alpha} = 1/8$. As expected, the order obtained is nearly 2. Notice that the computed stationary solutions are not expected to converge to the solution provided by F&A due to the rigid-lid assumption. In Table 6 we show the difference in L^1-norm between the F&A solution and the numerical solution corresponding to the mesh of 11520 cells.

6.4. Nonsmooth solution. The purpose of this numerical test is to compare the behavior of the numerical schemes in the neighborhood of a discontinuity. We consider the two-layer shallow-water system for channels with rectangular cross-section with constant breadth ($\sigma = 1$) and a flat bottom topography. In this case, the system

reduces to

$$\frac{\partial W}{\partial t} + \mathcal{A}(\mathrm{W})\frac{\partial W}{\partial x} = 0, \tag{51}$$

where $\mathcal{A}$ is the 4×4 matrix whose expression is given by

$$\mathcal{A}(\mathrm{W}) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\dfrac{Q_1^2}{A_1^2} + gA_1 & 2\dfrac{Q_1}{A_1} & gA_1 & 0 \\ 0 & 0 & 0 & 1 \\ rgA_2^2 & 0 & -\dfrac{Q_2^2}{A_2^2} + gA_2 & 2\dfrac{Q_2}{A_2} \end{bmatrix}.$$

The difficulty in defining a discontinuous solution of this system comes from the fact that the nonconservative products do not make sense within the distribution theory for such a solution. Nevertheless, there are several mathematical theories allowing us to make sense of these products in a weaker manner. In particular we consider here the definition of nonconservative products as Borel measures introduced by Volpert in [21]. This definition can be considered as a particular case of the concept of nonconservative products developed in [8]. In this latter work, the definition of the nonconservative product depends on the choice of a family of viscous paths. Volpert's definition is equivalent to the choice of the family of segments.

According to this definition, across a discontinuity with speed ξ a weak solution of the system must satisfy the generalized Rankine–Hugoniot condition

$$\int_0^1 \mathcal{A}\left(\mathrm{W}^- + s(\mathrm{W}^+ - \mathrm{W}^-)\right) \cdot (\mathrm{W}^+ - \mathrm{W}^-)\, ds = \xi(\mathrm{W}^+ - \mathrm{W}^-), \tag{52}$$

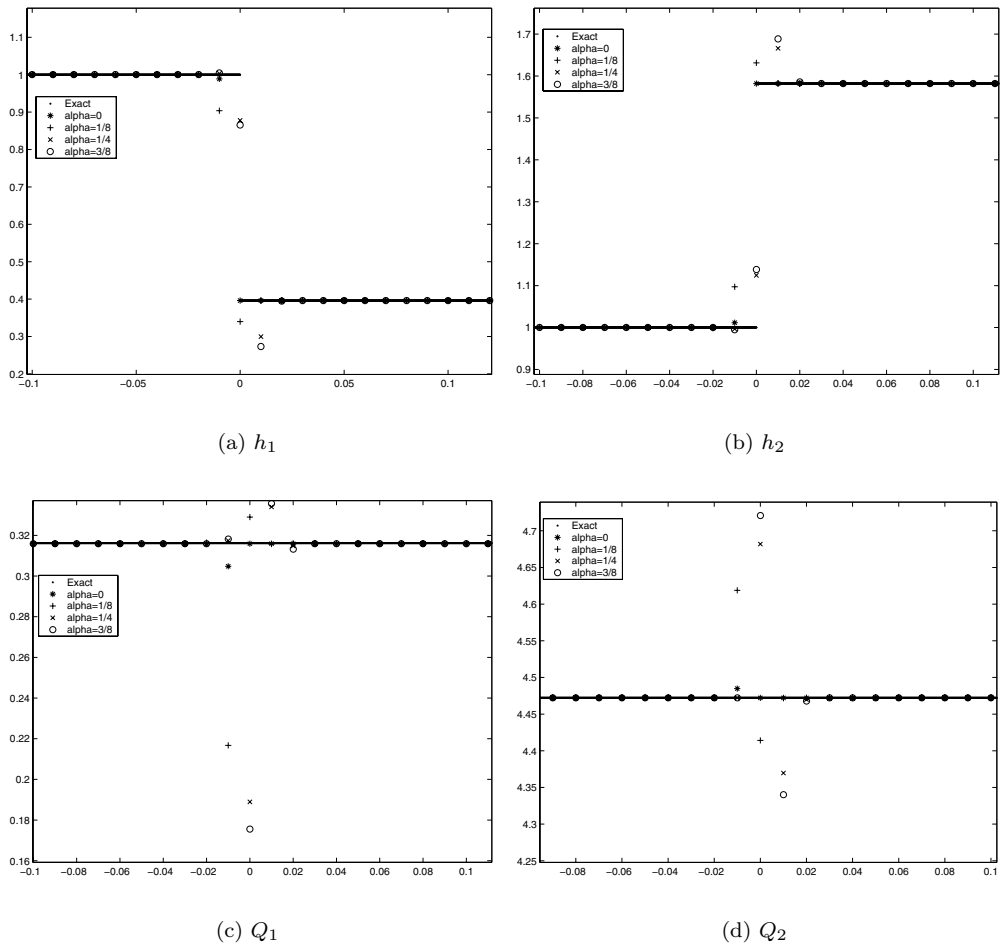
where W^-, W^+ are the left and right limits of the solution at the discontinuity.

In [16], a Roe-type numerical scheme for the two-layer shallow-water system was studied which is the particular case of the family introduced here that corresponds to the choice $\alpha = 0$. In this reference it was shown that this numerical scheme is consistent with the notion of weak solutions associated to Volpert's definition of nonconservative products in the following sense: Let us suppose that the approximations W_i^n and W_{i+1}^n obtained by the numerical scheme in two neighbor cells can be linked by a shock of speed ξ satisfying the jump condition (52). Then ξ is an eigenvalue of the intermediate matrix of the scheme, $\mathcal{A}_{i+1/2}$, and $\mathrm{W}_{i+1}^n - \mathrm{W}_i^n$ is an associated eigenvector. As a consequence, the solution of the linear approximate Riemann solver used by the numerical scheme to advance in time is equal to that of the exact Riemann problem. In particular, this implies that the numerical scheme solves exactly stationary shocks. In order to numerically verify this property, in [16] the authors proposed a test consisting in solving the system with values $g = 10$ and $r = 0.02$, and the initial condition

$$W(x,t) = \begin{cases} W_L & \text{if } x < 0, \\ W_R & \text{if } x > 0, \end{cases}$$

with

$$W_L = \begin{bmatrix} 1 \\ \sqrt{0.1} \\ 1 \\ \sqrt{20} \end{bmatrix}, \quad W_R = \begin{bmatrix} 0.3961560 \\ \sqrt{0.1} \\ 1.5820186 \\ \sqrt{20} \end{bmatrix}.$$



(a) h_1 (b) h_2

(c) Q_1 (d) Q_2

FIG. 5. *Internal hydraulic jump: water depths and discharges at each layer of fluid for* $\bar{\alpha} = 0$, $1/8$, $1/4$, *and* $3/8$.

These states satisfy the generalized jump condition (52) with an error of about 10^{-5}.

We have applied to this test problem the numerical scheme with values of the parameter $\alpha = 0$, $1/8$, $1/4$, and $3/8$. If the numerical schemes were consistent with the jump condition (52), they should obtain a stationary discontinuity placed at $x = 0$ with values of the unknowns close to the initial conditions. Figure 5 depicts the steady state solutions (water depths and discharges) for the different values of the parameter $\bar{\alpha}$ obtained with $\Delta x = 0.01$ and CFL $= 0.90$ for a rectangular channel 1 meter long. The errors are shown in Tables 7 and 8. Notice that in all the cases the numerical scheme behaves as expected: a stationary discontinuity placed at $x = 0$ and close to the initial condition is captured.

From Tables 7 and 8 and Figure 5 it is clear that the smallest errors correspond to $\alpha = 0$. For the other values an overshooting near the stationary shock occurs (see Figure 5), becoming larger for increasing values of $\bar{\alpha}$.

We have applied to the test problem the numerical scheme with the parameter α replaced by the function $\alpha(x)$ defined in Remark 7 with $\bar{\alpha} = 1/8$. Figure 6 depicts the steady state solution (water depths and discharges) obtained with again $\Delta x = 0.01$

TABLE 7
L^∞ and L^1 errors (water depths).

h_1		
α	L^∞-error	L^1-error
0	0.0111	2.24×10^{-4}
1/8	0.0963	0.0017
1/4	0.4693	0.0059
3/8	0.4817	0.0061

h_2		
α	L^∞-error	L^1-error
0	0.0114	1.14×10^{-4}
1/8	0.0972	0.0015
1/4	0.4434	0.0055
3/8	0.4574	0.0056

TABLE 8
L^∞ and L^1 errors (discharges).

Q_1		
α	L^∞-error	L^1-error
0	0.0114	4.33×10^{-4}
1/8	0.0995	0.0014
1/4	0.1273	0.0018
3/8	0.1406	0.0020

Q_2		
α	L^∞-error	L^1-error
0	0.0127	1.28×10^{-4}
1/8	0.1467	0.0021
1/4	0.2098	0.0031
3/8	0.2487	0.0039

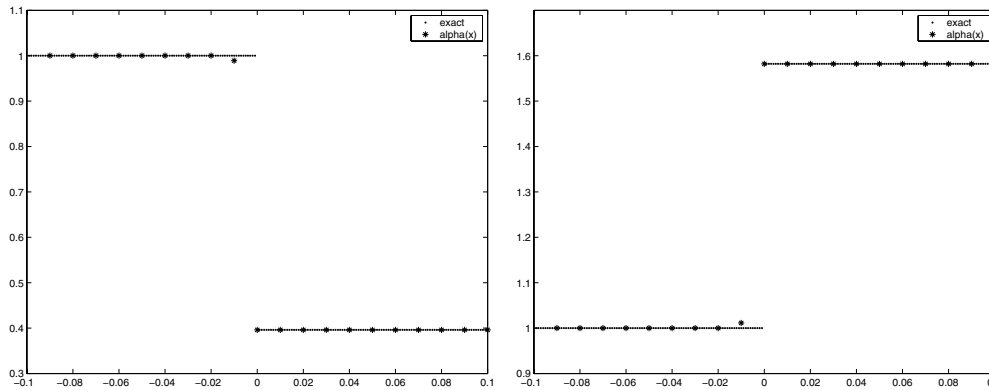
and CFL = 0.90. It coincides with that obtained with $\alpha = 0$. In this case, once the stationary solution is reached the corresponding function $\alpha(x)$ takes the value 0 only at the intercell where the discontinuity stands and 1/8 at the others: Figure 7 shows the values of $\alpha(x)$ close to the discontinuity.

Finally, the L^1 errors for the variables h_1, h_2, Q_1, and Q_2 are shown in Tables 9 and 10 for several meshes with increasing numbers of cells using the numerical scheme with the function $\alpha(x)$ with $\bar{\alpha} = 1/8$. As expected, the order obtained is one, due to the presence of the stationary discontinuity.

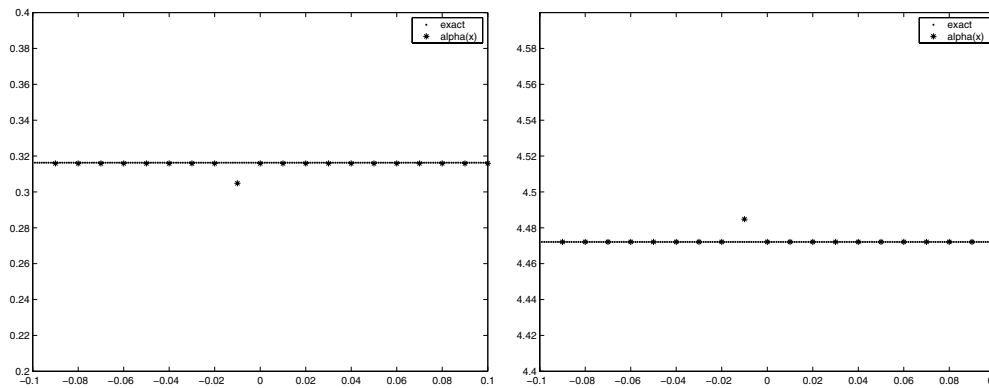
7. Conclusion. We have addressed in this paper the accurate numerical solution of nonconservative nonhomogeneous hyperbolic systems in one space dimension by finite volume solvers. We have introduced a general class of schemes that we have proved to be "asymptotically well-balanced" in the sense that they calculate, with at least order two, all stationary solutions of the system in all the domain except those on a set whose measure tends to zero as Δx tends to zero. One of the free parameters of this class is the diffusion matrix, so we include both flux-splitting (for conservative equations) and flux-difference solvers as particular cases.

We also give general sufficient conditions under which the schemes exactly calculate a given stationary solution. These conditions may be read as meaning on one hand that the numerical flux of the scheme must compensate the centered part of the numerical source term, and on another hand that the numerical diffusion must compensate the decentered part of the numerical source term.

In the case of two-layer shallow-water flows, we finally have introduced a specific solver to deal with the numerical difficulties associated to nonconservative products combined with nonlinear convection and source terms effects. On one hand, this solver is able to exactly compute water at rest. On the other hand, it yields accurate computation of subcritical to supercritical transitions without need of entropy corrections, and also of nonsmooth solutions due to the presence of the nonconservative products. Our main technical innovation is an adaptive discretization of the centered part of the numerical flux which, in our opinion, may be extended to construct accurate solvers for more general nonconservative nonhomogeneous equations.



(a) h_1 (b) h_2



(c) Q_1 (d) Q_2

FIG. 6. *Internal hydraulic jump: water depths and discharges at each layer of fluid for* $\alpha(x)$ *with* $\bar{\alpha} = 1/8$ *defined in Remark* 7.

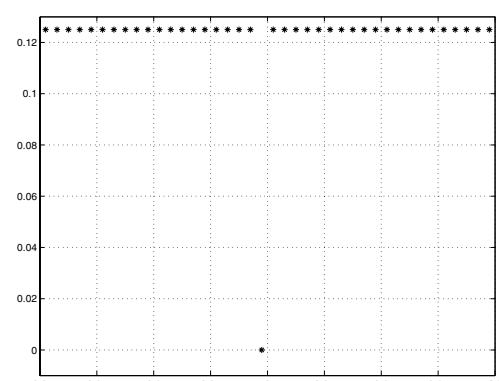


FIG. 7. $\alpha(x)$ *near the discontinuity placed at* $x = 0$.

TABLE 9
Test case 6.4. h_1 *and* h_2.

No. cells	L^1 error h_1	L^1 order h_1	L^1 error h_2	L^1 order h_2
25	7.78×10^{-4}	–	4.00×10^{-4}	–
50	4.15×10^{-4}	0.91	2.13×10^{-4}	0.91
100	2.24×10^{-4}	0.89	1.14×10^{-4}	0.90
200	1.18×10^{-4}	0.92	6.15×10^{-5}	0.89
400	6.04×10^{-5}	0.97	3.18×10^{-5}	0.95
800	3.09×10^{-5}	0.97	1.64×10^{-5}	0.96

TABLE 10
Test case 6.4. Q_1 *and* Q_2.

No. cells	L^1 error Q_1	L^1 order Q_1	L^1 error Q_2	L^1 order Q_2
25	1.50×10^{-3}	–	4.55×10^{-4}	–
50	8.19×10^{-4}	0.87	2.39×10^{-4}	0.93
100	4.33×10^{-4}	0.92	1.28×10^{-4}	0.9
200	2.27×10^{-4}	0.93	6.86×10^{-5}	0.9
400	1.18×10^{-4}	0.95	3.63×10^{-5}	0.92
800	6.01×10^{-5}	0.97	1.88×10^{-5}	0.95

Appendix. Proof of Theorem 1. We consider first a regular stationary solution W of (1) and a compact set, $K \subset \omega(W)$.

Let T be the continuous operator appearing in system (1),

$$T(W) = \frac{\partial}{\partial x}[F(x,W)] - G(x,W) - B(x,W)\frac{\partial W}{\partial x},$$

and T_h the discrete operator appearing in the numerical scheme (11)–(14),

$$T_h(W) = \frac{\phi_R^S - \phi_L^S}{\Delta x} - G_C$$

(where we omit the arguments for brevity). We must prove that

$$T_h(W)(x_i) = T(W)(x_i) + \mathcal{O}((\Delta x)^2) \quad \forall x_i \in K.$$

The proof is based on identity (8):

$$\widetilde{T}(W) = \frac{\partial}{\partial x}\left[F(x,W) - \nu\mathcal{D}(x,W)\left(\frac{\partial W}{\partial x} - M^{-1}S(x,W)\right)\right] - G(x,W) - B(x,W)\frac{\partial W}{\partial x}.$$

It is enough to prove that every term appearing in T_h,

$$\begin{aligned} \alpha_i &= \frac{F_C(x_i, x_{i+1}, W_i, W_{i+1}) - F_C(x_{i-1}, x_i, W_{i-1}, W_i)}{\Delta x}, \\ \beta_i &= \nu\frac{D(x_i, x_{i+1}, W_i, W_{i+1})(W_{i+1} - W_i) - D(x_{i-1}, x_i, W_{i-1}, W_i)(W_i - W_{i-1})}{(\Delta x)^2}, \end{aligned} \tag{53}$$

$$\begin{aligned} \eta_i = \nu\frac{1}{\Delta x}\Big[&D(x_i, x_{i+1}, W_i, W_{i+1})\widetilde{M^{-1}}(x_i, x_{i+1}, W_i, W_{i+1})S_D(x_i, x_{i+1}, W_i, W_{i+1}) \\ &- D(x_{i-1}, x_i, W_{i-1}, W_i)\widetilde{M^{-1}}(x_{i-1}, x_i, W_{i-1}, W_i)S_D(x_{i-1}, x_i, W_{i-1}, W_i)\Big], \end{aligned} \tag{54}$$

$$\delta_i = G_C(x_{i-1}, x_i, x_{i+1}, W_{i-1}, W_i, W_{i+1}),$$

$$\lambda_i = \frac{1}{2\Delta x}\Big[\mathcal{B}_R(x_i, x_{i+1}, W_i, W_{i+1})\,(W_{i+1} - W_i) - \mathcal{B}_L(x_{i-1}, x_i, W_{i-1}, W_i)\,(W_i - W_{i-1})\Big],$$

balances up to second order the corresponding term in $T(W)$. We prove the case of the more complex term η_i. The proof is similar for the other terms.

We define

$$R(x,U) = \mathcal{D}(x,U)M^{-1}(x,U)S(x,U),$$

$$R_C(x,y,U,V) = D(x,y,U,V)\widetilde{M^{-1}}(x,y,U,V)S_D(x,y,U,V),$$

and we want to prove that

$$\begin{aligned}\Delta R_C(x_i) &= R_C(x_i, x_{i+1}, W_i, W_{i+1}) - R_C(x_{i-1}, x_i, W_{i-1}, W_i)\\ &= \Delta x \frac{\partial}{\partial x}[R(x, W(x))]_{|_{x=x_i}} + \mathcal{O}((\Delta x)^2). \qquad (55)\end{aligned}$$

As K is a compact set and $\mathbf{R}^{N+1} \setminus \gamma$ is an open set, there exists $\delta_K > 0$ such that if

$$|U - W(x)| + \Delta x < \delta_K,$$

then $(x + \Delta x, U) \subset \mathbf{R}^{N+1} \setminus \gamma \quad \forall x \in K$.

On the other hand, if $W_{i+\theta} = (1-\theta)W_i + \theta W_i$, then using the mean value theorem we prove

$$|W_{i+\theta} - W(x_{i+\theta})| \le \frac{\Delta x}{2}\left\|\frac{\partial W}{\partial x}\right\|_{L^\infty(K)} \qquad \forall\, x_i \in K.$$

So, if

$$\Delta x < \delta_K / \left(1 + \left\|\frac{\partial W}{\partial x}\right\| /2\right),$$

we obtain that $(x_{i+\theta}, W_{i+\theta}) \subset \mathbf{R}^{N+1} \setminus \gamma$ for $\theta \in [0,1]$. Then if Δx verifies the previous inequality, we obtain that $L[(x_i, W_i), (x_{i+1}, W_{i+1})] \subset \mathbf{R}^{N+1} \setminus \gamma \ \forall x_i \in K$. Then we can apply Hypotheses 1(c), (d), and (f) and conclude that

$$\begin{aligned}R_C(x_i, x_{i+1}, W_i, W_{i+1}) &= R\left(x_{i+1/2}, \frac{W_i + W_{i+1}}{2}\right)\\ &+ R_1\left(x_{i+1/2}, \frac{W_i + W_{i+1}}{2}\right)\begin{pmatrix} x_{i+1} - x_i \\ W_{i+1} - W_i \end{pmatrix} + \mathcal{O}(|x_{i+1} - x_i|^2 + |W_{i+1} - W_i|^2)\end{aligned}$$

for some C^1 matrix function R_1.

As W is of class C^2, we have that

$$\begin{aligned} R_1\left(x_{i+1/2}, \frac{W_i+W_{i+1}}{2}\right)\begin{pmatrix} x_{i+1}-x_i \\ W_{i+1}-W_i \end{pmatrix} - R_1\left(x_{i-1/2}, \frac{W_{i-1}+W_i}{2}\right)\begin{pmatrix} x_i-x_{i-1} \\ W_i-W_{i-1} \end{pmatrix} \\ = \Delta x \left[R_1(x_{i+1/2}, W(x_{i+1/2}))\begin{pmatrix} 1 \\ \dfrac{\partial W}{\partial x}(x_{i+1/2}) \end{pmatrix} \right. \\ \left. - R_1(x_{i-1/2}, W(x_{i-1/2}))\begin{pmatrix} 1 \\ \dfrac{\partial W}{\partial x}(x_{i-1/2}) \end{pmatrix}\right] + \mathcal{O}((\Delta x)^2) \\ = (\Delta x)^2 \frac{\partial}{\partial x}\left[R_1(x, W(x))\begin{pmatrix} 1 \\ \dfrac{\partial W}{\partial x} \end{pmatrix}\right]_{\Big|_{x=x_i}} + \mathcal{O}((\Delta x)^2) = \mathcal{O}((\Delta x)^2). \end{aligned} \tag{56}$$

Moreover,

$$\begin{aligned} R\left(x_{i+1/2}, \frac{W_i+W_{i+1}}{2}\right) - R\left(x_{i-1/2}, \frac{W_{i-1}+W_i}{2}\right) \\ = \Delta x \frac{\partial}{\partial x}\Big[R(x, W(x))\Big]_{\Big|_{x=x_i}} + \mathcal{O}((\Delta x)^2). \end{aligned} \tag{57}$$

From (56) and (57) we conclude that (55) is verified.

We can now prove statements (a) and (b) of Theorem 1:

(a) If W has no singular points, then $\omega(W) = (0, L)$, and it is enough to use the preceding argument for some compact set $K \subset \omega(W)$ containing all the internal nodes of the discretization.

(b) If $\mathrm{meas}(\sigma(W)) = 0$, since $\omega(W)$ is an open set, then we may find an increasing sequence of compact sets $\{K_n\}_{n\geq 1}$ such that $\cup_{n\geq 1} K_n = \omega(W)$. Then $\mathrm{meas}((0, L) \setminus \cup_{n\geq 1} K_n) = 0$. It is enough to use the preceding argument for each K_n.

To prove statement (c) let us remark that, by the previous argument, the scheme is asymptotically well-balanced in $\omega(W)$. Thus, it only remains to prove this property in $\sigma(W)$ when this set has nonzero measure.

Let us denote by $\mathrm{int}(\sigma(W))$ the interior of $\sigma(W)$. As

$$\mathrm{meas}(\sigma(W) - \mathrm{int}(\sigma(W))) = 0,$$

it is enough to prove the property in $\mathrm{int}(\sigma(W))$. And, as we shall see, this occurs because in $\mathrm{int}(\sigma(W))$ $\mathcal{D}$ and M^{-1} coincide with smooth functions.

For the sake of simplicity, we shall assume $N = 2$ and that the first eigenvalue of M vanishes in $\sigma(W)$.

Let us define the operator

$$\begin{aligned} \widetilde{T}(W) = \frac{\partial}{\partial x}\left[F(x, W) - \mathcal{H}(x, W)\left(\frac{\partial W}{\partial x} - \mathcal{C}(x, W)S(x, W)\right)\right] \\ - G(x, W) - B(x, W)\frac{\partial W}{\partial x}, \end{aligned}$$

where

$$\mathcal{H}(x, W) = X(x, W)\begin{pmatrix} 0 & 0 \\ 0 & d_2(x, W) \end{pmatrix} X^{-1}(x, W),$$

$$\mathcal{C}(x,W) = X(x,W) \begin{pmatrix} 0 & 0 \\ 0 & \lambda_2^{-1}(x,W) \end{pmatrix} X^{-1}(x,W).$$

By d_2 and λ_2 we denote the second eigenvalue of $\mathcal{D}$ and M, respectively.

As W is a steady solution of (1),

$$M(x,W)\frac{\partial W}{\partial x}(x) = S(x,W).$$

Then

$$\mathcal{C}(x,W)S(x,W) = \mathcal{C}(x,W)M(x,W)\frac{\partial W}{\partial x}(x).$$

But

$$\begin{aligned}
&\mathcal{C}(x,W)M(x,W)\frac{\partial W}{\partial x}(x) = X(x,W) \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} X^{-1}(x,W)\frac{\partial W}{\partial x}(x) \\
&= X(x,W)\left[\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \right] X^{-1}(x,W)\frac{\partial W}{\partial x}(x) \\
&= \frac{\partial W}{\partial x} - \left[X^{-1}(x,W)\frac{\partial W}{\partial x}(x) \right]_1 X_1(x,W),
\end{aligned}$$

where by $[\,\cdot\,]_1$ we denote the first component of a vector and by X_1 the first eigenvector of M.

Then, as X_1 is also the first eigenvector of $\mathcal{H}$,

$$\mathcal{H}(x,W)\left[\frac{\partial W}{\partial x}(x) - \mathcal{C}(x,W)S(x,W) \right] = \lambda_{\mathcal{H},1}(x,W)\left[X^{-1}(x,W)\frac{\partial W}{\partial x}(x) \right]_1 X_1(x,W) = 0,$$

as by construction the first eigenvalue $\lambda_{\mathcal{H},1}$ of $\mathcal{H}$ is zero.

We deduce that

$$\widetilde{T}(W) = T(W).$$

It is then enough to prove that for any compact set $K \subset \mathrm{int}(\sigma(W))$ there exists $\delta_K > 0$ such that if $0 < \Delta x < \delta_K$, then

$$T_h(W)(x_i) = \widetilde{T}(W)(x_i) + \mathcal{O}((\Delta x)^2) \quad \forall\, x_i \in K. \tag{58}$$

Let us assume

$$\delta_K \leq \mathrm{dist}(K, \partial\sigma(W)).$$

If $x_i \in K$, then $x_{i-1},\ x_{i+1} \in \mathrm{int}(\sigma(W))$ and we may ensure that $L[(x_i,W_i),(x_{i+1},W_{i+1})]$ cuts $\gamma = \widetilde{\gamma}$. Then, using our definition of $\widetilde{M^{-1}}$ and D, we have

$$\begin{aligned}
&\widetilde{M^{-1}}(x_i, x_{i+1}, W_i, W_{i+1}) \\
&= X(x_{i+1/2}, \widetilde{W}_{i+1/2}) \begin{pmatrix} 0 & 0 \\ 0 & \lambda_2^{-1}(x_{i+1/2}, \widetilde{W}_{i+1/2}) \end{pmatrix} X^{-1}(x_{i+1/2}, \widetilde{W}_{i+1/2}) \\
&= \mathcal{C}(x_{i+1/2}, \widetilde{W}_{i+1/2}),
\end{aligned}$$

where we denote $\widetilde{W}_{i+1/2} = \widetilde{W}(x_i, x_{i+1}, W_i, W_{i+1})$. Also,

$$\begin{aligned} &D(x_i, x_{i+1}, W_i, W_{i+1}) \\ &= X(x_{i+1/2}, \widetilde{W}_{i+1/2}) \begin{pmatrix} 0 & 0 \\ 0 & d_2(x_{i+1/2}, \widetilde{W}_{i+1/2}) \end{pmatrix} X^{-1}(x_{i+1/2}, \widetilde{W}_{i+1/2}) \\ &= \mathcal{H}(x_{i+1/2}, \widetilde{W}_{i+1/2}). \end{aligned}$$

Now, T_h can be written as in the previous case by replacing D by $\mathcal{H}$ and $\widetilde{M^{-1}}$ by $\mathcal{C}$ in the definitions of β_i, η_i, and the proof is similar. ∎

REFERENCES

[1] A. Bermúdez and M. E. Vázquez Cendón, *Upwind methods for hyperbolic conservation laws with source terms*, Comput. & Fluids, 23 (1994), pp. 1049–1071.

[2] M. J. Castro, J. Macías, and C. Parés, *A Q-scheme for a class of systems of coupled conservation laws with source term. Application to a two-layer* 1*-d shallow water system*, M2AN Math. Model. Numer. Anal., 35 (2001), pp. 107–127.

[3] M. J. Castro, J. Macías, C. Parés, J. A. Rubal, and M. E. Vázquez, *Two-layer numerical model for solving exchange flows through channels with irregular geometry*, in Proceedings of ECCOMAS 2001, Swansea, Wales, UK, 2001, pp. 241–247.

[4] M. J. Castro, J. A. García, J. M. González, J. Macías, C. Parés, and M. E. Vázquez, *Numerical simulation of two-layer shallow water flows in channels with irregular geometry*, J. Comput. Phys., 195 (2004), pp. 202–235.

[5] T. Chacón Rebollo, A. Domínguez Delgado, and E. D. Fernández Nieto, *A family of stable numerical solvers for shallow water equations with source terms*, Comput. Methods Appl. Mech. Engrg., 192 (2003), pp. 203–225.

[6] T. Chacón Rebollo, A. Domínguez Delgado, and E. D. Fernández Nieto, *An entropy correction-free solver for non-homogeneous shallow-water equations*, M2AN Math. Model. Numer. Anal., 37 (2003), pp. 755–772.

[7] T. Chacón Rebollo, A. Domínguez Delgado, and E. D. Fernández Nieto, *Asymptotically balanced schemes for non-homogeneous hyperbolic systems. Application to shallow water equations*, C. R. Math. Acad. Sci. Paris, 338 (2004), pp. 85–90.

[8] G. Dal Maso, P. G. LeFloch, and F. Murat, *Definition and weak stability of non-conservative products*, J. Math. Pures Appl., 74 (1995), pp. 483–548.

[9] D. Farmer and L. Armi, *Maximal two-layer exchange over a sill and through a combination of a sill and contraction with barotropic flow*, J. Fluid Mech., 164 (1986), pp. 53–76.

[10] E. D. Fernández Nieto, *Aproximación numérica de leyes de conservación hiperbólicas no homogéneas. Aplicación a las ecuaciones de Aguas Someras*, Ph.D. thesis, Universidad de Sevilla, Spain, 2003.

[11] A. C. Fowler, *Mathematical Model in the Applied Sciences*, Cambridge University Press, Cambridge, UK, 1997.

[12] E. Godlewski and P. A. Raviart, *Numerical Approximation of Hyperbolic Systems of Conservation Laws*, Springer-Verlag, New York, 1996.

[13] J. M. Greenberg and A. Y. Leroux, *A well-balanced scheme for the numerical processing of source terms in hyperbolic equations*, SIAM J. Numer. Anal., 33 (1996), pp. 1–16.

[14] A. Kurganov and D. Levy, *Central-upwind schemes for the Saint-Venant system*, M2AN Math. Model. Numer. Anal., 36 (2002), pp. 397–425.

[15] R. J. LeVeque, *Numerical Methods for Conservation Laws*, Birkhäuser-Verlag, Basel, 1990.

[16] C. Parés and M. J. Castro, *On the well-balanced property of Roe's method for non-conservative hyperbolic systems. Applications to shallow-water systems*, M2AN Math. Model. Numer. Anal., 38 (2004), pp. 821–852.

[17] B. Perthame and C. Simeoni, *A kinetic scheme for the Saint-Venant system with a source term*, Calcolo, 38 (2001), pp. 201–231.

[18] B. Perthame and C. Simeoni, *Convergence of the upwind interface source method for hyperbolic conservation laws*, in Hyperbolic Problems: Theory, Numerics, Applications. Proceedings of the 9th International Conference on Hyperbolic Problems, T. Thou and E. Tadmor, eds., Springer-Verlag, New York, 2003, pp. 61–78.

[19] P. L. Roe, *Upwind differencing schemes for hyperbolic conservation laws with source terms*, in Nonlinear Hyperbolic Problems, C. Carraso, P.-A. Raviart, and D. Serre, eds., Lecture

Notes in Math. 1270, Springer-Verlag, Berlin, 1986, pp. 41–51.

[20] J. B. Schijf and J. C. Schonfeld, *Theoretical considerations on the motion of salt and fresh water*, in Proceedings of the Minnesota International Hydraulic Convention, Joint meeting IAHR and Hyd. Div. ASCE., 1953, pp. 321–333.

[21] A. I. Volpert, *The space BV and quasilinear equations*, Math. USSR Sb., 73 (1967), pp. 225–267.

SIAM J. SCI. COMPUT.
Vol. 29, No. 3, pp. 1127–1159
© 2007 Society for Industrial and Applied Mathematics

CONSERVATIVE COUPLING BETWEEN FINITE ELEMENTS AND RETARDED POTENTIALS. APPLICATION TO VIBROACOUSTICS*

P. GROB† AND P. JOLY‡

Abstract. The numerical simulation of vibroacoustics is concerned with the radiation of sound emitted by thin vibrating mechanical structures. We present a numerical method that combines boundary elements (integral equation) for the acoustic wave equation with standard finite elements for the mechanics. The originality of our work is that we consider the time domain problem and use retarded potentials for writing the integral equation. We establish a nonstandard variational formulation of this new problem, in space-time for the acoustic equation, and in space only for the mechanic equation. The basic ideas for the discretization are the following: (i) Space finite elements and finite differences with time step Δt (a θ-scheme, $0 \leq \theta \leq 1/2$) are used for the discretization of the mechanic equation. (ii) Space-time finite elements are used for the discretization of the acoustic equation, which is "projected" two times on two staggered time grids of time step $2\Delta t$. The use of staggered twice larger time grids for the discretization of the acoustic equation (see (ii) above) plays a key role in the cancellation of the "coupling terms" (between the two equations), which is crucial in the energy analysis.

Key words. coupling method, retarded potentials, stability

AMS subject classifications. 45A05, 65M12, 65R99, 74J20, 74K20

DOI. 10.1137/050647141

1. Introduction.

Vibroacoustics is the science that studies the radiation of sound emitted by thin vibrating mechanical structures (a membrane, a plate, or a shell). Under the assumption of small vibrations, usual mathematical models couple the standard acoustic wave equation with dynamic membrane, plate, or shell models. A lot of work has been devoted to the time harmonic problem [26]. In this work we are more specifically interested in the time dependent case. A first natural approach consists in coupling three-dimensional (3D) finite element approaches for the sound radiation with appropriate two-dimensional (2D) finite element approaches for the thin structure. In such a case, it is important to preserve the explicit nature of the time stepping procedure, especially for the acoustic part. This is what has been done, for instance, in [12], [16], and [14].

In the present work, we investigate an alternative approach which consists in coupling a finite element method for the thin structure with an integral equation method for the acoustics. This is made possible by the homogeneous structure of the surrounding fluid: the Green's function of the exterior problem is known explicitly. Such an approach has been extensively developed in the time harmonic case [8], [23], [34], and we investigate an extension to the time dependent case, This is more delicate especially because of the additional time discretization that may be the cause of some instability phenomena. The a priori advantages of the use of integral equations are quite clear and have already been largely investigated in the case of pure diffraction problems (which corresponds to the case where the thin structure does not vibrate):

*Received by the editors December 9, 2005; accepted for publication (in revised form) September 14, 2006; published electronically May 22, 2007.
http://www.siam.org/journals/sisc/29-3/64714.html

†INRIA, B.P. 105, 78153 Le Chesnay Cedex, France and Department of Math and Statistics, Laval University, G1K 7P4 Quebec, Canada (pgrob@mat.ulaval.ca).

‡INRIA, B.P. 105, 78153 Le Chesnay Cedex, France and EADS-CCR, 12 rue Pasteur, 92150 Suresnes, France (patrick.joly@inria.fr).

(i) The calculations are restricted to the thin structure: one makes the economy of one space dimension.
(ii) One avoids the question of the truncation of the exterior domain (absorbing conditions, perfectly matched layers, etc.): the unboundedness of this domain is treated exactly.
(iii) One avoids (at least partially) the dispersion (propagation) errors induced by the discretization of the 3D wave equation.

The counterparts of these advantages are also well known:

(a) The time discretization of the integral equation may induce instability phenomena (this is of course the major question).
(b) One has to deal with full matrices, contrary to what happens with 3D finite elements. In particular, in the time domain, the resulting scheme cannot be completely explicit: a (full) band matrix has to be inverted.
(c) In the time domain, one needs to store the story of the solution in the past.

However, if point (c) cannot be overcome, the negative impact of points (a) and (b) is minimized by some recent progress of the research:

- Instability phenomena are essentially observed with collocation methods [28], [33]. The development of variational methods leads to the construction of unconditionally stable schemes [19], [22], [4], [6], [20], [21] (even though they appear to be more difficult to implement).
- A lot of progress has been made in the conception of rapid algorithms (multiple methods) for the evaluation of the product's matrix-vector for the matrices that appear after the discretization of integral equations. Initially developed for the time hamonic case [18], they are now also available for the time dependent case [17]. Coupled with iterative solvers, they permit us to efficiently solve large scale diffraction problems with integral equations.

With the coupling of time domain integral equations with finite elements, new difficulties appear, linked to the objective we wish to achieve:

- We want to couple the methods that would be naturally usually employed for discretizing the "uncoupled problem" made of the integral equation only (acoustic part of the problem, assuming that the structure does not vibrate) and the dynamic thin structure equation only (mechanic part of the problem). These are by nature different: one uses space-time finite elements for the acoustic part, and space finite elements and finite differences in time for the mechanic part.
- We want to find a robust coupling algorithm, namely, an algorithm that guarantees a priori, from the theoretical point of view, the stability of the resulting method, if possible without affecting the stability of the method in the uncoupled case.

It is difficult to find, in the mathematical literature, works that address these questions in a rigorous way. A first work in this direction can be found in [2] and [3] in the case of electromagnetism (Maxwell's equations), where retarded potentials are used as a transparent boundary condition for diffraction problems with locally heterogeneous media. One of the major differences from our work is that a space-time variational formulation is used for both the surfacic part and the volumic part of the equations, which leads to a nonstandard discretization, necessarily nonexplicit, of Maxwell's equations in the volume. From this point of view, our work is closer to some recent research in the engineering literature [33], [28], where finite differences are used for the mechanic part of the problem but where the central question of stability is not

treated.

The work we present in this paper has been precisely guided by the stability constraint, which has led us to propose a new method. The key idea is to ensure the stability via some discrete energy identity, which is a kind of equivalent of the classical energy conservation for the full coupled problem. In this sense, our method can be seen as a conservative coupling method in a sense similar to the one developed in [24] and used in [15], [5] for space-time mesh refinement methods. To get such a discrete energy identity, assuming that Δt is the time step for the discretization of the mechanic part of the problem, we are led to distinguish two staggered discretization grids, corresponding to even and odd instants, for the discretization of the acoustic part. These two grids are coupled only through the coupling with the mechanic part.

The outline of the article is as follows. The major part of the article (sections 2 to 6) is devoted to the presentation and the analysis of our method on the membrane problem that we present as our model problem in section 2. In section 3, we describe and give the main properties of the so-called Dirichlet to Neumann operator N_Σ. This operator is used to reduce our original 2D-3D problem into a 2D one posed only on the membrane (section 4). Sections 5 and 6 are the two main sections of this article. We describe our numerical method in section 5 and analyze its main properties in section 6, where the fundamental stability result (Theorem 6.2) is proven as a consequence of the energy identities of Theorem 6.1. In section 7, we explain how the formalism of sections can be extended to the vibrations of a plate of Kirchoff–Love type (section 7.1) or Mindlin–Reissner type (section 7.2). Finally, section 8 is devoted to the numerical results. We consider successively the vibrations of a membrane (section 8.1) and the vibrations of a Mindlin–Reissner plate (section 8.2). We validate our new method through a comparison with computations done with a full finite element code (coupling between 3D finite elements for the acoustics and 2D finite elements for the mechanics).

2. A model problem.

2.1. Notation. In what follows $\mathbf{x} = (x, x_3)$, with $x = (x_1, x_2) \in \mathbb{R}^2$ and $x_3 \in \mathbb{R}$, will denote the current point of $\mathbb{R}^3$, with associated canonic basis $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$. For simplicity, x will also denote the point $(x, 0)$ of $\mathbb{R}^3$. We consider the (small) vibrations of a membrane Σ located in the plane $x_3 = 0$ so that it can de identified to a bounded open or $\mathbb{R}^2$. This membrane is fixed along its boundary and coupled to a homogeneous fluid occupying the exterior domain $\Omega = \mathbb{R}^3 \setminus \Sigma$. For what follows, in particular, to define traces of functions in Ω, it will be useful to distinguish the two "lips" of Σ that we can define as $\Sigma^+ = \partial\Omega^+ \cap \Sigma$ and $\Sigma^- = \partial\Omega^- \cap \Sigma$, where $\Omega^\pm = \{\mathbf{x} = (x, x_3) \in \Omega \ / \ \pm x_3 > 0\}$. We shall define an "orientation" of Σ by defining the unit normal vector to Σ as $n = \mathbf{e}_3$, which points from Ω^- toward Ω^+ (see Figure 2.1). Finally, for any function $g : \Omega \mapsto \mathbb{R}$ sufficiently smooth (typically in $H^1(\Omega)$), we denote by $[g]$ its jump across Σ, defined in agreement with the above orientation, namely,

$$[g] = g_{|\Sigma_+} - g_{|\Sigma_-} \quad (\in H_{00}^{\frac{1}{2}}(\Sigma) \quad \text{if} \quad g \in H^1(\Omega)) \,. \tag{2.1}$$

We shall denote by Δ the 2D Laplacian operator (with respect to x) and by $\mathbf{\Delta} = \Delta + \partial^2/\partial x_3^2$ the 3D Laplacian operator. In the same way we shall denote ∇ and $\mathbf{\nabla}$, respectively, the 2D and 3D gradient operators.

2.2. Description of the model. The unknowns of our mathematical model will be as follows.

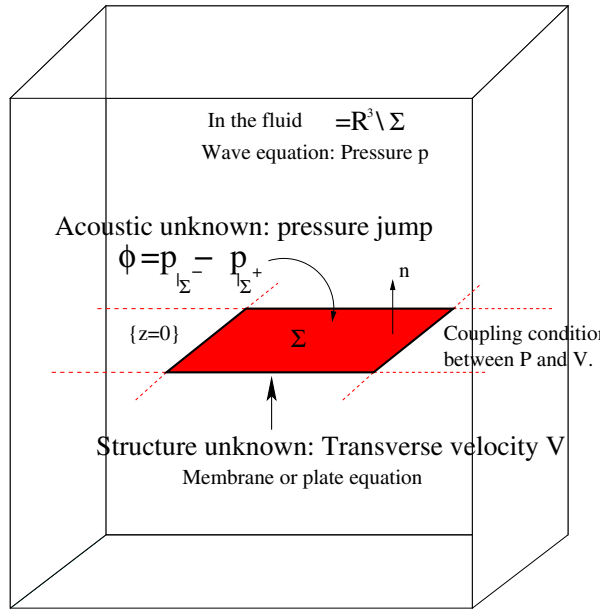


FIG. 2.1. *The problem geometry.*

- The pressure field distribution in the fluid:

$$p(\mathbf{x}, t), \quad \mathbf{x} \in \Omega, \quad t > 0.$$

- The distribution over the membrane of the normal velocity (as usual for membrane models, we assume that the deformation of the membrane can be described only with normal displacements in the x_3 direction):

$$v(x, t), \quad x \in \Sigma, \quad t > 0.$$

In what follows, we shall denote by ρ and c, respectively, the fluid density and sound speed, which are supposed to be positive constants (homogeneity of the fluid, a fundamental assumption). In the same way $\rho_m > 0$ will denote the surface density of the membrane and $T > 0$ its tension, so that the propagation velocity of waves along the membrane is $c_m = (\rho_m/\sigma)^{1/2}$. For simplicity, we shall also assume that these quantities are also constant, but this assumption is not essential here: we could consider a heterogeneous membrane. Finally, we assume that the membrane is submitted to a loading force described by its surfacic density $f(x, t), x \in \Sigma, t \geq 0$. The equations of the mathematical model are (we refer the reader to [9] for more physical insight)

$$\begin{cases} \rho_m \dfrac{\partial^2 v}{\partial t^2} - T\, \Delta v = f + \left[\dfrac{\partial p}{\partial t}\right], & x \in \Sigma, \quad t > 0, \\ \dfrac{1}{c^2}\dfrac{\partial^2 p}{\partial t^2} - \boldsymbol{\Delta} p \quad = 0, & \mathbf{x} \in \Omega, \quad t > 0, \end{cases} \tag{2.2}$$

coupled by the fluid structure interaction condition, that expresses in terms of our unknowns, the continuity across Σ of the transverse displacement (in the fluid and on the membrane)

$$\left.\frac{\partial p}{\partial n}\right|_{\Sigma_+} = \left.\frac{\partial p}{\partial n}\right|_{\Sigma_-} = -\rho \frac{\partial v}{\partial t}, \quad x \in \Sigma, \quad t > 0, \tag{2.3}$$

and completed by a boundary condition along $\partial\Sigma$,

$$v(x,t) = 0, \quad x \in \Sigma, \quad t > 0, \tag{2.4}$$

and homogeneous initial conditions:

$$\begin{cases} v(x,0) = \dfrac{\partial v}{\partial t}(x,0) = 0, & x \in \Sigma, \\ p(\mathbf{x},0) = \dfrac{\partial p}{\partial t}(\mathbf{x},0) = 0, & \mathbf{x} \in \Sigma. \end{cases} \tag{2.5}$$

2.3. A mathematical result. Let us first introduce a notation for spaces of functions of the variable $t \in [0,T]$, with values Hilbert space X, which are restrictions of functions in Sobolev spaces on $\mathbb{R}$ that have support in $\mathbb{R}^+$ (causal functions). More precisely, given $s \in \mathbb{R}^+$, we shall set

$$\begin{aligned} \mathcal{H}^s(0,T;X) = \{u \in L^2(0,T;X) \,/\, \exists\, \widetilde{u} \in H^s_+(\mathbb{R};X) \\ \text{such that } u(t) = \widetilde{u}(t) \text{ if } 0 \le t \le T\}, \end{aligned} \tag{2.6}$$

where by definition

$$H^s_+(\mathbb{R};X) = \{u \in H^s(\mathbb{R};X) \mid u(t) = 0 \text{ for } t < 0\}.$$

$\mathcal{H}^s(0,T;X)$ is a Hilbert space equipped with the norm

$$\|u\|_{\mathcal{H}^s(0,T;X)} = \inf_{u \in H^s(\mathbb{R};X),\, \widetilde{u}_{|[0,T]} = u} \|u\|_{H^s(\mathbb{R};X)}, \tag{2.7}$$

and we shall denote by $\mathcal{H}^{-s}(0,T;X)$ the dual of $\mathcal{H}^s(0,T;X)$.

Remark 1. Of course the lower bound 0 of the interval $[0,T]$ plays a special role. One can show, for instance [27], that

- for $0 \le s < \frac{1}{2}$, $\mathcal{H}^s(0,T;X) = H^s(0,T;X)$,
- for $s = \frac{1}{2}$, $\mathcal{H}^{\frac{1}{2}}(0,T;X) = \{u \in H^s(0,T;X) / \int_0^T \|u(t)\|^2_X \frac{dt}{t} < +\infty\}$,
- for $\frac{1}{2} < s < \frac{3}{2}$, $\mathcal{H}^s(0,T;X) = \{u \in H^s(0,T;X) / u(0) = 0\}$.

On the other hand, for $T_2 > T_1$, $\mathcal{H}^s(0,T_1;X)$ is nothing but the space generated by the restrictions to $]0,T_1[$ of functions in $\mathcal{H}^s(0,T_2;X)$.

Next, we state an existence and uniqueness result for the problem (2.2)–(2.5).

THEOREM 2.1.

$$f \in C^1(\mathbb{R}^+;L^2(\Sigma)) \tag{2.8}$$

(2.2)–(2.5) (v,p)

$$\begin{cases} v \in C^2(\mathbb{R}^+;L^2(\Sigma)) \cap C^1(\mathbb{R}^+;H^1_0(\Sigma)) \cap C^0(\mathbb{R}^+;H^2(\Sigma)), \\ p \in C^2(\mathbb{R}^+;L^2(\Omega)) \cap C^1(\mathbb{R}^+;H^1(\Omega)), \quad \Delta p \in C^0(\mathbb{R}^+;L^2(\Omega)). \end{cases} \tag{2.9}$$

The proof of the theorem is an exercise on the application of Hille–Yosida's theorem [35]. □

3. A Dirichlet to Neumann operator and its properties. In this section, we introduce an operator that maps functions defined on Σ onto functions defined on Σ and is of Dirichlet to Neumann nature. This operator will allow us (section 4) to reduce our problem to a new one posed only on Σ.

All the material of this section is essentially already known. It is, however, essential in understanding the rest of the paper. We choose to present it in a manner adapted to our purpose.

3.1. Formal definition of the operator. In this paragraph we shall be interested in the properties of a (very classical) intermediate problem which consists in solving the wave equation in Ω, with 0 initial data, with a prescribed jump φ across Σ, assuming that the normal derivative of the solution is continuous across Σ.

$$\begin{cases} \dfrac{1}{c^2}\dfrac{\partial^2 u_\varphi}{\partial t^2} - \Delta u_\varphi = 0 & \text{on } \Omega, \\ \left[\dfrac{\partial u_\varphi}{\partial n}\right] = 0, \quad [u_\varphi] = \varphi & \text{on } \Sigma, \\ u_\varphi(x,0) = \dfrac{\partial u_\varphi}{\partial t}(x,0) = 0. \end{cases} \tag{3.1}$$

Formally, we define the operator N_Σ by

$$N_\Sigma \varphi = \frac{\partial u_\varphi}{\partial n}|_\Sigma, \tag{3.2}$$

which is a well-defined quantity since $\frac{\partial u_\varphi}{\partial n}$ is continuous across Σ.

In the forthcoming subsections, we shall

- present a functional framework for the definition of $N_\Sigma\varphi$ (section 3.2),
- give its main properties (section 3.2), and
- give an explicit expression for $N_\Sigma\varphi$, useful for numerical computations (section 3.3).

3.2. Functional framework and properties of N_Σ. Our first objective is to give a functional framework in which the operator N_Σ can be defined rigorously. This subject has already been substantially investigated and it is not our purpose to present all the needed functional analysis. However, we wish to provide the minimum information in order to make our exposition self-contained.

The first step is to define precisely $N_\Sigma\varphi$ for sufficiently smooth φ. As usual, the notion of the Neumann boundary condition has to be understood in the weak sense. Using standard semigroup theory, it is easy to show that if

$$\varphi \in C^3(\mathbb{R}^+; H^{\frac{1}{2}}_{00}(\Sigma)), \tag{3.3}$$

the problem (3.1) admits a unique solution,

$$u_\varphi \in C^1(\mathbb{R}^+; H^1(\Omega)) \cap C^2(\mathbb{R}^+; L^2(\Omega)), \tag{3.4}$$

with, moreover,

$$\Delta u_\varphi \in C^0(\mathbb{R}^+; L^2(\Omega)). \tag{3.5}$$

By the trace theorem, one deduces that

$$\frac{\partial u_\varphi}{\partial n}|_{\Sigma^+} \equiv \frac{\partial u_\varphi}{\partial n}|_{\Sigma^-} = \frac{\partial u_\varphi}{\partial n}|_\Sigma \in C^0(\mathbb{R}^+; H^{\frac{1}{2}}_{00}(\Sigma)'). \tag{3.6}$$

This allows us to define an operator:

$$\left| \begin{array}{rccl} N_\Sigma : & C^3(\mathbb{R}^+; H^{\frac{1}{2}}_{00}(\Sigma)) & \longrightarrow & C^0(\mathbb{R}^+; H^{\frac{1}{2}}_{00}(\Sigma)'), \\ & \varphi & \longrightarrow & N_\Sigma\varphi = \dfrac{\partial u_\varphi}{\partial n}|_\Sigma. \end{array} \right. \tag{3.7}$$

The results (3.3) to (3.5) are far from optimal, in terms of the required regularity for φ and in terms of the resulting regularity of the solution u_φ. Sharper results can be obtained by Fourier–Laplace techniques. Such techniques naturally lead to results in weighted (in time) Sobolev spaces in $\Omega \times \mathbb{R}^+$ and $\Sigma \times \mathbb{R}^+$ [1], [19]. However, by truncation and causality arguments, these results can be translated into results over finite time intervals. This is the point of view that we shall adopt here. For a given $T > 0$, we denote $\mathcal{Q}_T = \Omega \times]0, T[$ and $\mathcal{S}_T = \Omega \times]0, T[$.

We introduce the functional space

$$V(\mathcal{S}_T) = L^2(0, T; H^{\frac{1}{2}}_{00}(\Sigma)) \cap \mathcal{H}^{\frac{1}{2}}(0, T; L^2(\Sigma)) \tag{3.8}$$

that we equip with its natural norms $\| \cdot \|_{V(\mathcal{S}_T)}$. Then, one can show that there exists a constant $C_T > 0$ such that

$$\forall\, \varphi \in C^3(\mathbb{R}^+; H^{\frac{1}{2}}_{00}(\Sigma)), \quad \|N_\Sigma \varphi\|_{V(\mathcal{S}_T)'} \leq C_T \, \|\varphi\|_{V(\mathcal{S}_T)}. \tag{3.9}$$

Remark 2. The estimate is valid for a general manifold Σ. In our case, since Σ is a part of a plane $x_3 = 0$, this estimate could be obtained using a Fourier–Laplace transform in time and a Fourier transform in space,

$$\widetilde{\varphi}(x, t) \longrightarrow \widehat{\varphi}(k, s), \quad k \in \mathbb{R}^2, \quad \mathcal{R}e\ s > 0,$$

where $\widetilde{\varphi}$ is the extension of φ by 0 outside Σ. One can compute (with obvious notation)

$$\widehat{\frac{\partial u_\varphi}{\partial n}}(k, s, x_3 = 0) = \frac{1}{2}\, (k^2 + s^2/c^2)^{\frac{1}{2}}\, \hat{\varphi}(k, s) \quad (\text{with } \mathcal{R}e\, (k^2 + s^2/c^2)^{\frac{1}{2}} > 0).$$

The conclusion follows from standard arguments (Plancherel's theorem, definition of Sobolev spaces via the Fourier–Laplace transform, and causality).

We refer the reader to [25] for a more thorough study of this operator.

As a consequence, by the density (modulo truncation to $]0, T[$) of $C^3(\mathbb{R}^+; H^{\frac{1}{2}}_{00}(\Sigma))$ in $V(\mathcal{S}_T)$), the operator N_Σ can be uniquely extended as an operator, still denoted N_Σ, such that

$$N_\Sigma \ \in \ \mathcal{L}(V(\mathcal{S}_T), V(\mathcal{S}_T)'), \tag{3.10}$$

and we shall denote by $b_T(\cdot, \cdot)$ the associated bilinear form

$$b_T(\varphi, \psi) \ = \ \langle N_\Sigma \varphi, \psi \rangle_{\mathcal{S}_T} \quad \forall\, (\varphi, \psi) \ \in \ V(\mathcal{S}_T)^2, \tag{3.11}$$

where $\langle \cdot, \cdot \rangle_{\mathcal{S}_T}$ denotes the usual duality pairing between $V(\mathcal{S}_T)$ and $V(\mathcal{S}_T)'$, which we can also rewrite as

$$b_T(\varphi, \psi) \ = \ \int_0^T \langle N_\Sigma \varphi(\cdot, t), \psi(\cdot, t) \rangle_\Sigma \, dt,$$

where $\langle \cdot, \cdot \rangle_\Sigma$ denotes the duality product between $H^{\frac{1}{2}}_{00}(\Sigma)'$ and $H^{\frac{1}{2}}_{00}(\Sigma)$.

Moreover, thanks to Remark 1 (second part), for any $t \leq T$ one can define

$$b_t(\varphi, \psi) = b_T(\varphi, \chi_t \psi) \quad \forall\, (\varphi, \psi) \ \in \ V(\mathcal{S}_T)^2.$$

Next, we state and prove a fundamental positivity property of N_Σ. For this, we introduce

$$W(\mathcal{S}_T) = \mathcal{H}^1(0, T; H^{\frac{1}{2}}_{00}(\Sigma)) \cap \mathcal{H}^{\frac{3}{2}}(0, T; L^2(\Sigma)). \tag{3.12}$$

Since $\partial/\partial t$ maps $W(\mathcal{S}_T)$ in $V(\mathcal{S}_T)$, we can introduce, for any $t \leq T$, the bilinear form

$$p_t(\varphi,\psi) \;=\; b_t\left(\varphi, \frac{\partial \psi}{\partial t}\right) \quad \forall\, (\varphi,\psi) \;\in\; V(\mathcal{S}_T)\times W(\mathcal{S}_T). \tag{3.13}$$

THEOREM 3.1. [illegible] $0 < t \leq T$ [illegible] $\varphi \in W(\mathcal{S}_T)$ [illegible]

$$|\varphi|^2_{N,\mathcal{S}_t} := p_t(\varphi,\varphi) \tag{3.14}$$

[illegible]

$$\varphi \;\in\; W(\mathcal{S}_T) \longrightarrow \|\varphi\|_{N,\mathcal{S}_T} := \int_0^T p_t(\varphi,\varphi)\, dt \equiv \int_0^T |\varphi|^2_{N,\mathcal{S}_t}\, dt \tag{3.15}$$

[illegible] $\|\cdot\|_{N,\mathcal{S}_T}$ [illegible] $W(\mathcal{S}_T)^2$ [illegible] α_T [illegible]

$$\|\varphi\|^2_{N,\mathcal{S}_T} \;\geq\; \alpha_T\, \|\varphi\|^2_{V(\mathcal{S}_T)}. \tag{3.16}$$

[illegible]. Obviously, it suffices to work with $\varphi \in C^3(0,T;H^{\frac{1}{2}}_{00}(\Sigma))$ and to conclude by density. First writing that

$$\iint_{\mathcal{Q}_t} \left(\frac{1}{c^2}\frac{\partial^2 u_\varphi}{\partial t^2} - \Delta u_\varphi \right) \frac{\partial u_\varphi}{\partial t}\, dxds = 0,$$

we easily deduce from Green's formula that

$$\int_0^t \left\langle \left[\frac{\partial u_\varphi}{\partial n}\right], \frac{\partial u_\varphi}{\partial t} \right\rangle_\Sigma ds = \frac{1}{2}\int_\Omega \left(\frac{1}{c^2}\left|\frac{\partial u_\varphi}{\partial t}\right|^2 + |\nabla u_\varphi|^2 \right)(x,t)\, dx,$$

that is, according to the definitions of N_Σ, $b_t(\cdot,\cdot)$, and $p_t(\cdot,\cdot)$,

$$p_t(\varphi,\varphi) = \int_0^t \left\langle N_\Sigma \varphi, \frac{\partial u_\varphi}{\partial t} \right\rangle_\Sigma ds = \frac{1}{2}\int_\Omega \left(\frac{1}{c^2}\left|\frac{\partial u_\varphi}{\partial t}\right|^2 + |\nabla u_\varphi|^2 \right)(x,t)\, dx,$$

which proves the first part of the theorem. Next, after time integration over $t \in [0,T]$ of the previous identity (in which T has been replaced by t), we get

$$\int_0^T p_t(\varphi,\varphi)\, dt = \frac{1}{2}\int_\Omega \left(\frac{1}{c^2}\left|\frac{\partial u_\varphi}{\partial t}\right|^2 + |\nabla u_\varphi|^2 \right) dxdt,$$

and the estimate (3.16) follows from the trace theorem in $H^1(\mathcal{Q}_T)$.

3.3. An explicit representation for N_Σ. The definition and properties of N_Σ as defined in section 3.2 remain valid in the case of nonconstant coefficients (for instance, with c a function of x). We can exploit the fact that we worked in a homogeneous medium (c is constant) by getting an explicit representation for N_Σ, based on the following integral representation (also called double layer representation) of the solution u_φ.

LEMMA 1. [illegible] u_φ [illegible] (3.1) [illegible]

$$u_\varphi(\mathbf{x},t) = -\int_\Sigma \frac{x_3}{|\mathbf{x}-y|}\left(\frac{\varphi(y,t-\tau(\mathbf{x},y))}{|\mathbf{x}-y|^2} + \frac{1}{c}\,\frac{\dot\varphi(y,t-\tau(\mathbf{x},y))}{|x-y|} \right) dy, \tag{3.17}$$

[illegible] $\tau(\mathbf{x},\mathbf{y}) = |\mathbf{x}-y|/c$ [illegible] $\ddot{\varphi}$ [illegible] φ [illegible]. One can find in [19] the general formula for any manifold Σ. Equation (3.17) is nothing but this formula specified to the case where Σ is a part of a plane. □

The evaluation of $N_\Sigma\varphi$ requires us to differentiate (3.17) with respect to x_3 and take the trace for $x_3 = 0$. This is made delicate by the presence of the singularity of the fundamental solution of the wave equation, since $|\mathbf{x} - y|$ may tend to 0 when $\mathbf{x}$ approaches Σ. The resulting formula is a so-called hypersingular integral whose meaning has to be precisely defined and which is not practical for numerical computation. This difficulty can be overcome by choosing to evaluate the bilinear form $b_T(\varphi,\psi)$ associated to N_Σ, at least for sufficiently smooth (φ,ψ). This is, moreover, the quantity that we need for finite element approximation. The main idea is to use integration by parts to transfer space derivatives from the Green's kernel to the test functions.

THEOREM 3.2. [illegible] $\varphi \in H^2(0,T;L^2(\Sigma)) \cap L^2(0,T;H_0^1(\Sigma))$ [illegible] $\varphi(x,0) = \frac{\partial\varphi}{\partial t}(x,0) = 0$

$$
\left|
\begin{aligned}
b_T(\varphi,\psi) &= \frac{1}{4\pi}\int_0^T \int_\Sigma\int_\Sigma \frac{\ddot{\varphi}(y,t-\tau(x,y))\;\psi(x,t)}{c^2|x-y|}\;dxdydt \\
&\quad + \frac{1}{4\pi}\int_0^T \int_\Sigma\int_\Sigma \frac{\nabla\varphi(y,t-\tau(x,y))\cdot\nabla\psi(x,t)}{|x-y|}\;dxdydt
\end{aligned}
\right. \tag{3.18}
$$

[illegible] $\psi \in V(\mathcal{S}_T)\cap L^2(0,T;H_0^1(\Sigma))$ $\ddot{\varphi}$ [illegible] φ

[illegible]. This is a well-known result. For completeness, we give a simple proof adapted to our particular case (plane geometry). Let $\widetilde{\varphi} \in H^1(\mathbb{R}^2\times\mathbb{R}) \cap H^2(\mathbb{R};L^2(\mathbb{R}^2))$ such that

$$\widetilde{\varphi}(x,t) = 0 \quad \forall\, x \notin \overline{\Sigma} \quad \text{and} \quad \widetilde{\varphi}(x,t) = \varphi(x,t) \quad \forall\, x \in \Sigma, \quad \forall\, t \in [0,T].$$

Let $\psi \in H^2(0,T;L^2(\Sigma)) \cap L^2(0,T;H_0^1(\Sigma))$ and $\widetilde{\psi}$ its extension (to $\mathbb{R}^2\times\mathbb{R}$) by 0 outside $\mathcal{S}_T$. We use successively the Fourier transform in time and the Fourier transform in space (each normalized such that the Fourier transform of the Dirac distribution is 1):

$$(\widetilde{\varphi},\widetilde{\psi})(x,t) \xrightarrow{\mathcal{F}_t} (\widehat{\Phi},\widehat{\Psi})(x,\omega) \xrightarrow{\mathcal{F}_x} (\widehat{\varphi},\widehat{\psi})(k,\omega.$$

By definition of $b_T(\cdot,\cdot)$, we can write, thanks to the regularity of φ and ψ,

$$b_T(\varphi,\psi) = \int_0^T\int_\Sigma \frac{\partial u_\varphi}{\partial n}(x,0,t)\;\psi(x,t)\;dxdt = \int_{\mathbb{R}}\int_{\mathbb{R}^2} \frac{\partial u_\varphi}{\partial n}(x,0,t)\;\widetilde{\psi}(x,t)\;dxdt.$$

On the other hand, one computes easily that

$$\widehat{\frac{\partial u_\varphi}{\partial n}}(k,\omega,x_3=0) = \frac{1}{2}\left(k^2 - \frac{\omega^2}{c^2}\right)^{\frac{1}{2}}\;\hat{\varphi}(k,\omega),$$

$$\text{with}\quad \left(k^2 - \frac{\omega^2}{c^2}\right)^{\frac{1}{2}} = \lim_{\varepsilon\longrightarrow 0}\left(k^2 - \frac{(\omega+i\varepsilon)^2}{c^2}\right)^{\frac{1}{2}}.$$

Thus, using Plancherel's theorem in x and t, we have

$$\begin{aligned} b_T(\varphi,\psi) &= \int_{\mathbb{R}}\int_{\mathbb{R}^2} (k^2-\omega^2/c^2)^{\frac12}\ \widehat{\varphi}\ \overline{\widehat{\psi}}\, dkd\omega \\ &= \int_{\mathbb{R}}\int_{\mathbb{R}^2} \frac{ik\widehat{\varphi}\ \overline{ik\widehat{\psi}}}{(k^2-\omega^2/c^2)^{\frac12}}\, dkd\omega + \frac{1}{c^2}\int_{\mathbb{R}}\int_{\mathbb{R}^2} \frac{(i\omega)^2\widehat{\varphi}\ \overline{\widehat{\psi}}}{(k^2-\omega^2/c^2)^{\frac12}}\, dkd\omega. \end{aligned}$$

Next, we use the fact that

$$\mathcal{F}_x^{-1}\left[\frac12\,(k^2-\omega^2/c^2)^{-\frac12}\right] = \frac{e^{-i\frac{\omega}{c}|x|}}{4\pi|x|}$$

so that, using the standard properties of the Fourier transform,

$$\mathcal{F}_x^{-1}\left(\frac{ik\widehat{\varphi}}{(k^2-\omega^2/c^2)^{\frac12}}\right) = \int_{\mathbb{R}^2} \frac{e^{-i\frac{\omega}{c}|x-y|}}{4\pi|x-y|}\,\nabla\Phi(y,\omega)\,dy.$$

Then, Plancherel's theorem in x (using that the functions have support in Σ) leads to

$$\int_{\mathbb{R}^2} \frac{ik\widehat{\varphi}\ \overline{ik\widehat{\psi}}}{(k^2-\omega^2/c^2)^{\frac12}}\, dk = \int_\Sigma\int_\Sigma e^{-i\frac{\omega}{c}|x-y|}\,\frac{\nabla\Phi(y,\omega)\,\overline{\nabla\Psi(x,\omega)}}{4\pi|x-y|}\,dxdy.$$

Next, we observe that

$$\mathcal{F}_t^{-1}\left[e^{-i\frac{\omega}{c}|x-y|}\ \Phi(y,\omega)\right] = \varphi(y,t-\tau(x,y))$$

so that Plancherel's theorem in the t variable (plus the fact that supp $\widetilde{\psi}\subset\overline{\Sigma}\times[0,T]$) leads to

$$\int_{\mathbb{R}}\int_{\mathbb{R}^2} \frac{ik\widehat{\varphi}\ \overline{ik\widehat{\psi}}}{(k^2-\omega^2/c^2)^{\frac12}}\, dkd\omega = \frac{1}{4\pi}\int_0^T\int_\Sigma\int_\Sigma \frac{\nabla\varphi(y,\tau(x,y,t))\cdot\nabla\psi(x,t)}{|x-y|}\,dxdydt.$$

In the same way

$$\int_{\mathbb{R}}\int_{\mathbb{R}^2} \frac{(i\omega)^2\widehat{\varphi}\ \overline{ik\widehat{\psi}}}{(k^2-\omega^2/c^2)^{\frac12}}\, dkd\omega = \frac{1}{4\pi}\int_0^T\int_\Sigma\int_\Sigma \frac{\ddot{\varphi}(y,\tau(x,y,t))\,\psi(x,t)}{|x-y|}\,dxdydt.$$

Then, summing the two last equations and using the inverse Fourier transform, we obtain 3.18. □

3. The integrals appearing in (3.18) are well defined because

$$(x,y)\longrightarrow |x-y|^{-1}\ \in\ L^2(\Sigma\times\Sigma).$$

The formula (3.18) also shows that $b_T(\varphi,\psi)$ is well defined with less regular ψ when

$$(\varphi,\psi)\in\mathcal{H}^2(0,T;H_0^1(\Sigma))\times L^2(0,T;H_0^1(\Sigma)).$$

Moreover, for any $t\le T$ one has the formula

$$b_t(\varphi,\psi) = b_T(\varphi,\chi^t\psi)\quad\forall\,(\varphi,\psi)\in\mathcal{H}^2(0,T;H_0^1(\Sigma))\times L^2(0,T;H_0^1(\Sigma)), \tag{3.19}$$

where χ^t denotes the characteristic function of the interval $[0,t]$.

4. There is a more general formula for a nonplane manifold Σ (see [30]),

$$b_T(\varphi,\psi) = \frac{1}{4\pi}\int_0^T \int_\Sigma \int_\Sigma \frac{\ddot{\varphi}(y,\tau(x,y,t))\ \psi(x)}{c^2|x-y|}\ d\gamma_x d\gamma_y dt + \frac{1}{4\pi}\int_0^T \int_\Sigma \int_\Sigma \frac{\overrightarrow{\mathrm{rot}}_\Sigma\ \varphi(y,\tau(x,y,t))\cdot \overrightarrow{\mathrm{rot}}_\Sigma\ \psi(x)}{|x-y|}\ d\gamma_x d\gamma_y dt,$$

where n_x denotes the unit vector to Σ at point x, $d\gamma$ the surfacic measure along Σ, and $\overrightarrow{\mathrm{rot}}_\Sigma$ the surface curl operator [31]. It is straightforward to recover (3.18) from the above formula.

4. Reduction to a problem on the surface.

4.1. An equivalent problem posed on Σ. Let $(v,p) \in W_v(\mathcal{Q}_T) \times W_p(\mathcal{Q}_T)$, where

(4.1)
$$\begin{cases} W_v(0,T) = C^2(0,T;L^2(\Sigma))\ \cap\ C^1(0,T;H_0^1(\Sigma))\ \cap\ C^0(0,T;H^2(\Sigma)), \\ W_p(0,T) = \{p \in C^2(0,T;L^2(\Omega))\ \cap\ C^1(0,T;H^1(\Omega))\ /\ \Delta p\ \in\ C^0(0,T;L^2(\Omega))\} \end{cases}$$

is the unique solution of (2.2)–(2.5) (see Theorem 2.1).

Let us introduce the new unknown function

$$\varphi := [p] \quad (\Sigma \times \mathbb{R}^+ \longrightarrow \mathbb{R}). \tag{4.2}$$

From the standard trace theorem in $H^1(\Omega)$, we deduce that

$$\varphi\ \in\ C^1(0,T;H_{00}^{\frac{1}{2}}(\Sigma)). \tag{4.3}$$

Assuming that $p \in \mathcal{H}^3(0,T;L^2(\Omega))$, the trace operator $p \mapsto [p]$ maps $\mathcal{H}^3(0,T;L^2(\Omega)) \cap \mathcal{H}^1(0,T;H^1(\Omega))$ into $\mathcal{H}^{\frac{3}{2}}(0,T;L^2(\Sigma))$. Therefore, we also have

$$\varphi\ \in\ \mathcal{H}^{\frac{3}{2}}(0,T;L^2(\Sigma)). \tag{4.4}$$

Finally, $p(x,0) = 0$ implies that

$$\varphi(x,0) = 0. \tag{4.5}$$

Then, it is straightforward to infer from (4.3), (4.4), and (4.5) that

$$\varphi\ \in\ W(\mathcal{S}_T). \tag{4.6}$$

Using (2.3) and the definition of N_Σ, one sees that $(v,\varphi) \in W_v(\mathcal{Q}_T) \times W(\mathcal{S}_T)$ is a solution of

(4.7)
$$\begin{cases} \rho_m \dfrac{\partial^2 v}{\partial t^2} - T\ \Delta v - \dfrac{\partial \varphi}{\partial t} = f, & x \in \Sigma, \quad t > 0, \\ N_\Sigma \varphi + \rho \dfrac{\partial v}{\partial t} = 0 & \text{in } V(\mathcal{S}_T), \\ v(x,0) = \dfrac{\partial v}{\partial t}(x,0) = 0, & x \in \Sigma. \end{cases}$$

More precisely, problems (2.2)–(2.5) and (4.7) are "equivalent" in the following sense.

THEOREM 4.1. [illegible] $(v,p) \in W_v(\mathcal{Q}_T) \times W_p(\mathcal{Q}_T)$ [illegible] (2.2)–(2.5) [illegible] 2.1 [illegible] $\varphi = [p]$ $(v,\varphi) \in W_v(\mathcal{Q}_T) \times W(\mathcal{S}_T)$ [illegible] (4.7).

[illegible] $(v,\varphi) \in W_v(\mathcal{Q}_T) \times W(\mathcal{S}_T)$ [illegible] (4.7) [illegible] $(v, p = u_\varphi$ [illegible] (3.1)) $\in W_v(\mathcal{Q}_T) \times W_p(\mathcal{Q}_T)$ [illegible] (2.2)–(2.5).

[illegible]. We have already proven the first part of the theorem. For the second claim, it is sufficient to prove the uniqueness result of (v,p) which is a consequence of Lemma 2 below. □

LEMME 2. [illegible] $(v,\varphi) \in W_v(\mathcal{Q}_T) \times W(\mathcal{S}_T)$ [illegible] (4.7), [illegible] $0 \le t \le T$

$$\frac{1}{2}\int_\Sigma \left(\rho_m \left|\frac{\partial v}{\partial t}\right|^2 + T\,|\nabla v|^2\right)(x,t)\,dx + \frac{1}{\rho}\,|\varphi|^2_{N,\mathcal{S}_t} = \iint_{\mathcal{S}_t} f(x,s)\frac{\partial v}{\partial t}(x,s)\,dxds. \tag{4.8}$$

[illegible]. We multiply the first equation of (4.7) by $\frac{\partial v}{\partial t}$ and integrate over $\mathcal{S}_t$ to obtain, thanks to Green's formula,

$$\frac{1}{2}\int_\Sigma \left(\rho_m \left|\frac{\partial v}{\partial t}|^2 + T\,\right|\nabla v|^2\right)\,dx - \iint_{\mathcal{S}_t} \frac{\partial \varphi}{\partial t}\,\frac{\partial v}{\partial t}\,dxds = \iint_{\mathcal{S}_t} f\,\frac{\partial v}{\partial t}\,dxds. \tag{4.9}$$

On the other hand, from the second equation of (4.7) (multiply by $\frac{\partial \varphi}{\partial t}$ and integrate over $\mathcal{S}_t$), we obtain, by definition of $b_t(\cdot,\cdot)$,

$$\frac{1}{\rho}\,b_t\left(\varphi, \frac{\partial \varphi}{\partial t}\right) + \iint_{\mathcal{Q}_t} \frac{\partial v}{\partial t}\frac{\partial \varphi}{\partial t}\,dxds = 0. \tag{4.10}$$

One concludes by summing (4.9) and (4.10) and employing (3.13) and (3.14). □

The identity (4.8) is fundamental in the sense that it not only leads to the uniqueness result but also to a priori estimates. Indeed, after time integration of (4.8), we get

$$\frac{1}{2}\iint_{\mathcal{S}_T} \left(\rho_m \left|\frac{\partial v}{\partial t}\right|^2 + T\,|\nabla v|^2\right)\,dxdt + \frac{1}{\rho}\|\varphi\|^2_{N,\mathcal{S}_T} = \iint_{\mathcal{Q}_T} (T-t)\,f\,\frac{\partial v}{\partial t}\,dxdt, \tag{4.11}$$

which yields, using in particular Theorem 3.1,

$$\begin{cases} \|v\|_{H^1(\mathcal{S}_T)} \le C_T\,\|f\|_{L^2(\mathcal{S}_T)}, \\ \|\varphi\|_{H^{\frac{1}{2}}(\mathcal{S}_T)} \le C_T\,\|f\|_{L^2(\mathcal{S}_T)}. \end{cases} \tag{4.12}$$

That is why the two identities (4.8) and (4.12) are the ones that we shall try to respect, at least in some sense, after discretization. This will allow us to prove a stability theorem.

4.2. A variational formulation. We give in this section a variational formulation of (4.7) which is preparatory to its discretization in space and time. This formulation is, as we shall see, somewhat hybrid and guided by the following remarks:

- If φ were considered as a given source term in the first equation of (4.7), then one would like to compute an approximation of v by using a finite element in space and finite differences in time, as this is standard for a pure membrane problem (see, for instance, [12]). That is why we shall make a variational formulation in space only on this equation.

- If v were considered as a given source term in the first equation of (4.7), then one would like to compute an approximation of φ by using space-time finite elements as we would do for a pure crack diffraction problem formulated with retarded potential (see, for instance, [22], [1], [4]). That is why we shall make a variational formulation of this equation in space and time.

Therefore, the variational formulation we shall work with is

(4.13)
$$\begin{cases} \text{Find } (v,\varphi) \in W_v(\mathcal{Q}_T) \times W(\mathcal{S}_T) \quad \text{such that} \quad \forall\, w \in H^1_0(\Sigma), \quad \forall\, \psi \in V(\mathcal{S}_T), \\ \\ \dfrac{d^2}{dt^2}(v(t),w) + a(v(t),w) - \displaystyle\int_\Sigma \frac{\partial \varphi}{\partial t}(x,t)\; w(x)\; dx = \int_\Sigma f(x,t)\; w(x)\; dx, \\ \\ \dfrac{1}{\rho}\; b_T(\varphi,\psi) + \displaystyle\iint_{\mathcal{S}_T} \frac{\partial v}{\partial t}\; \psi\; dxdt = 0, \\ \\ v(0) = \dfrac{dv}{dt}(0) = 0, \end{cases}$$

where we have set

(4.14)
$$\begin{cases} (v,w)_\rho = \displaystyle\int_\Sigma \rho_m\; v\; w\; dx \qquad \forall\, (v,w)\ \in\ L^2(\Sigma)^2, \\ \\ a(v,w) = \displaystyle\int_\Sigma T\; \nabla v \cdot \nabla w\; dx \quad \forall\, (v,w)\ \in\ H^1(\Sigma)^2. \end{cases}$$

REMARK 5. The solution (v,φ) of (4.7) is obviously a solution of (4.13). Proceeding as in Lemma 2, one shows the uniqueness of the solution of (4.13). Therefore, (4.7) and (4.13) are equivalent.

REMARK 6. Note that, in the second equation of (4.13), the space $W(\mathcal{S}_T)$ for test functions is different from the space $V(\mathcal{S}_T)$ in which the solution is searched.

5. The space-time discretization.

5.1. Space-time finite element spaces. The most nontrivial part of the approximation process is the construction of suitable finite element spaces for the approximation of the spaces $W(\mathcal{S}_T)$ and $V(\mathcal{S}_T)$ appearing in (4.13). In this goal, we introduce suitable notation.

Given some Hilbert space X and some time step $\Delta t = T/2N$, we introduce spaces of piecewise linear and piecewise constant functions of time with values in X:

(5.1)
$$\begin{cases} \mathcal{P}^1_{\Delta t}(0,T;X) = \{f \in C^0([0,T],H)\ /\ \forall\, n \le 2N-1,\ f\ |_{[t^n,t^{n+1}]} \in P^1(X)\}, \\ \\ \mathcal{P}^0_{\Delta t}(0,T;X) = \{f \in L^2([0,T],X)\ /\ \ \forall\, n \le 2N-1, f\ |_{[t^n,t^{n+1}]} \in P^0(X)\}, \end{cases}$$

where $t^n = n\Delta t$ and

(5.2)
$$\begin{cases} P^1(X) = \{\psi(t) = \psi_0 + t\psi_1, \quad (\psi_0,\psi_1) \in X^2\}, \\ \\ P^0(X) = \{\psi(t) = \psi_0, \quad \psi_0 \in X\}. \end{cases}$$

In what follows, we shall be led to distinguish two time grids of space step $2\Delta t$, respectively, the even grid associated to instants t^{2n} and the odd grid associated to

instants t^{2n+1}. This leads to a natural decomposition of the spaces $\mathcal{P}^k_{\Delta t}(0,T;X),\ k=0,1$, as

$$\mathcal{P}^k_{\Delta t}(0,T;X) = \mathcal{P}^{k,ev}_{\Delta t}(0,T;X) + \mathcal{P}^{k,od}_{\Delta t}(0,T;X), \quad k=0,1\ , \tag{5.3}$$

(5.4)
$$\begin{cases} \mathcal{P}^{k,ev}_{\Delta t}(0,T;X) = \{f\ \in\ \mathcal{P}^k_{\Delta t}(0,T;X)\ /\ \forall\ n \le N-1,\ f\ |_{[t^{2n},t^{2n+2}]} \in P^k(X)\}, \\ \mathcal{P}^{k,od}_{\Delta t}(0,T;X) = \{f\ \in\ \mathcal{P}^k_{\Delta t}(0,T;X)\ /\ \forall\ n \le N-1,\ f\ |_{[t^{2n-1},t^{2n+1}]} \in P^k(X)\}. \end{cases}$$

To the decomposition (5.3) for $k=1$, we associate two interpolation operators,

$$\Pi^{1,ev}_{\Delta t} : \mathcal{P}^1_{\Delta t}(0,T;X) \longrightarrow \mathcal{P}^{1,ev}_{\Delta t}(0,T;X), \quad \Pi^{1,od}_{\Delta t} : \mathcal{P}^1_{\Delta t}(0,T;X) \longrightarrow \mathcal{P}^{1,od}_{\Delta t}(0,T;X),$$

which are completely characterized by

$$\begin{cases} \left[\Pi^{1,ev}_{\Delta t}\ f\right](t^{2n}) = f(t^{2n}), & 0 \le n \le N-1, \\ \left[\Pi^{1,od}_{\Delta t}\ f\right](t^{2n+1}) = f(t^{2n+1}), & 0 \le n \le N-1. \end{cases} \tag{5.5}$$

In the same way, for k=0, we define the $L^2(0,T;X)$ projection operators

$$\Pi^{0,ev}_{\Delta t} : \mathcal{P}^0_{\Delta t}(0,T;X) \longrightarrow \mathcal{P}^{k,ev}_{\Delta t}(0,T;X), \quad \Pi^{0,od}_{\Delta t} : \mathcal{P}^0_{\Delta t}(0,T;X) \longrightarrow \mathcal{P}^{0,od}_{\Delta t}(0,T;X).$$

Assuming that $f_{n+\frac12}$ is the constant value of some $f \in \mathcal{P}^0_{\Delta t}(0,T;X)$, we simply have

$$\begin{cases} \left[\Pi^{0,ev}_{\Delta t}\ f(t)\right] = \frac{f_{2n+\frac12}+f_{2n+\frac32}}{2} & \text{if}\quad t^{2n} < t < t^{2n+2}, \qquad 0 \le n \le N-1, \\ \left[\Pi^{0,od}_{\Delta t}\ f(t)\right] = \frac{f_{2n+\frac12}+f_{2n-\frac12}}{2} & \text{if}\quad t^{2n-1} < t < t^{2n+1}, \quad 0 \le n \le N-1. \end{cases} \tag{5.6}$$

The effect of the interpolation and projection operators is illustrated in Figure 5.1. Next, we introduce a suitable family $(\mathcal{M}_h, h>0)$ of finite dimensional subspaces of $H^1_0(\Sigma)$, and we consider the finite element spaces for approximating $W(\mathcal{S}_T)$ and $V(\mathcal{S}_T)$:

$$\begin{cases} W_{h,\Delta t}(\mathcal{S}_T) = \{\varphi_{h,\Delta t} \in \mathcal{P}^1_{\Delta t}(0,T;\mathcal{M}_h)\ /\ \varphi_{h,\Delta t}(\cdot,0) = 0\ \}, \\ V_{h,\Delta t}(\mathcal{S}_T) = \mathcal{P}^0_{\Delta t}(0,T;\mathcal{M}_h). \end{cases} \tag{5.7}$$

These spaces can also be defined as follows. Let $w_n(t),\ n\ \in\ \mathbb{N}$, be the functions defined by

$$w_0(t) = 1 - \frac{|t|}{\Delta t} \quad \text{if } |t| \le \Delta t, \quad = 0 \text{ if not}, \quad w_n(t) = w_0(t-t_n). \tag{5.8}$$

Then $\varphi_{h,\Delta t} \in W_{h,\Delta t}(\mathcal{S}_T)$ if and only if there exists $(\varphi_1,\dots,\varphi_{2N}) \in \mathcal{M}_h^{2N}$ such that

$$\varphi_{h,\Delta t}(x,t) = \sum_{n=1}^{2N} \varphi_n(x)\ w_n(t). \tag{5.9}$$

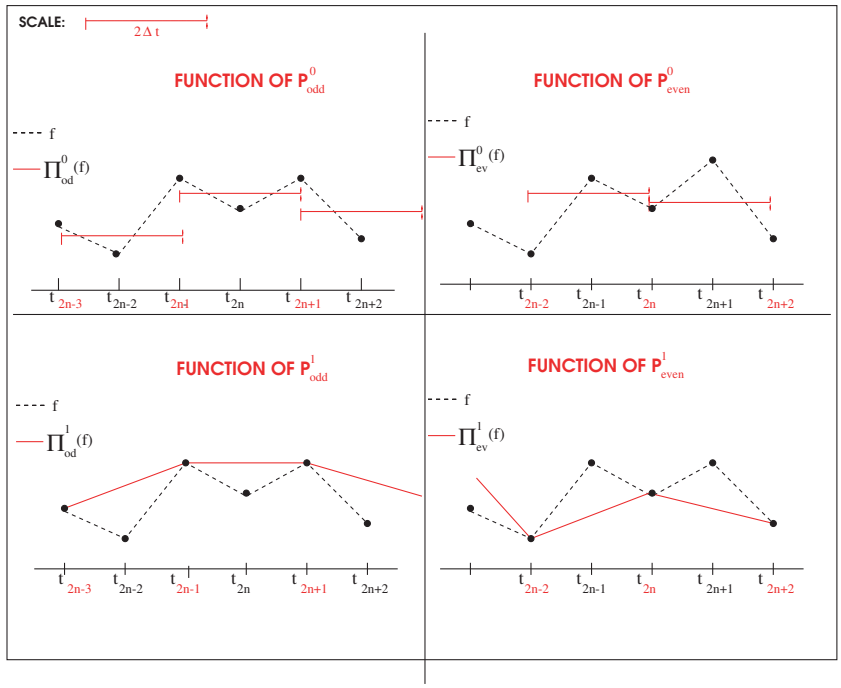


FIG. 5.1. *The interpolation and projection operators.*

In the same way, let $\zeta_n(t)$, $n \in \mathbb{N}^*$, be the functions defined by

$$\zeta_n(t) = 1 \quad \text{if } t \in [t_{n-1}, t_n], \quad = 0 \text{ if not.} \tag{5.10}$$

For any $\psi_{h,\Delta t} \in W_{h,\Delta t}(\mathcal{S}_T)$, there exists $(\psi_1, \ldots, \varphi_{2N}) \in \mathcal{M}_h^{2N}$ such that

$$\psi_{h,\Delta t}(x,t) = \sum_{n=1}^{2N} \psi_n(x)\, \zeta_n(t). \tag{5.11}$$

One can notice the following:

- The space $W_{h,\Delta t}(\mathcal{S}_T))$ is stable by "integer" truncation:

$$\forall\, n \le N, \quad \forall\, \psi_{h,\Delta t} \in W_{h,\Delta t}(\mathcal{S}_T), \quad \chi_{n\Delta t}\psi_{h,\Delta t} \in W_{h,\Delta t}(\mathcal{S}_T). \tag{5.12}$$

- The time derivative $\partial/\partial t$ maps $V_{h,\Delta t}(\mathcal{S}_T)$ into $W_{h,\Delta t}(\mathcal{S}_T)$.

Moreover, one easily checks the commutation property

$$\begin{aligned} \forall\, \varphi_{h,\Delta t} \in W_{h,\Delta t}(\mathcal{S}_T), \quad \Pi^0_{ev} \frac{\partial \varphi_{h,\Delta t}}{\partial t} &= \frac{\partial}{\partial t}\left[\Pi^1_{ev}\varphi_{h,\Delta t}\right], \\ \Pi^0_{od} \frac{\partial \varphi_{h,\Delta t}}{\partial t} &= \frac{\partial}{\partial t}\left[\Pi^1_{od}\varphi_{h,\Delta t}\right]. \end{aligned} \tag{5.13}$$

For the construction of our scheme, since $V_{h,\Delta t}(\mathcal{S}_T)$ (resp., $W_{h,\Delta t}(\mathcal{S}_T)$) is not a subspace of $V(\mathcal{S}_T)$ (resp., $W(\mathcal{S}_T)$), there is a problem, a priori, for defining $b_T(\varphi_{h,\Delta t}, \psi_{h,\Delta t})$ for $\varphi_{h,\Delta t}$ in $W_{h,\Delta t}(\mathcal{S}_T)$ and $\psi_{h,\Delta t}$ in $V_{h,\Delta t}(\mathcal{S}_T)$. Fortunately, this difficulty can be overcome as follows.

For any $0 < \varepsilon < 1/2$, let $w_0^\varepsilon \in C^2(\mathbb{R})$ such that

$$\begin{cases} w_0^\varepsilon(t) = w_0(t) \quad \text{if} \quad |t| \notin [0, \varepsilon\Delta t[\cup]\Delta t(1-\varepsilon), \Delta t(1+\varepsilon)[\,, \\ |w_0^\varepsilon(t)| \le 1 \quad \forall\, t \in \mathbb{R}. \end{cases} \tag{5.14}$$

We set $w_n^\varepsilon(t) = w_0^\varepsilon(t - t_n)$ and introduce the regularization operator

$$\begin{aligned} \varphi_{h,\Delta t} = \sum_{n=1}^{2N} \varphi_n(x)\, w_n(t) \;\in\; W_{h,\Delta t}(\mathcal{S}_T) &\longrightarrow J_\varepsilon \varphi_{h,\Delta t} \\ = \sum_{n=1}^{2N} \varphi_n(x)\, w_n^\varepsilon(t) &\in \mathcal{H}^2(0,T;\mathcal{M}_h). \end{aligned} \tag{5.15}$$

In particular, one can define (cf. Remark 3)

$$b_T(J_\varepsilon \varphi_{h,\Delta t}, \psi_{h,\Delta t}) \quad \forall \psi_{h,\Delta t} \in V_{h,\Delta t}(\mathcal{S}_T).$$

We state without proof the precise result. The main ingredients for the computation will be given in the proof of Lemma 4, which provides an explicit expression useful for numerical computations. The rigorous justification relies on Lebesgue's theorem.

LEMMA 3. [illegible] $(\varphi_{h,\Delta t}, \psi_{h,\Delta t}) \in W_{h,\Delta t}(\mathcal{S}_T) \times V_{h,\Delta t}(\mathcal{S}_T)$ [illegible]

$$b_T(\varphi_{h,\Delta t}, \psi_{h,\Delta t}) = \lim_{\varepsilon \to 0} b_T(J_\varepsilon \varphi_{h,\Delta t}, \psi_{h,\Delta t}), \tag{5.16}$$

[illegible] (6.12) [illegible] 4. [illegible] $\alpha_T > 0$ [illegible] (3.16)

$$\begin{aligned} \forall\, \varphi_{h,\Delta t} \in W_{h,\Delta t}(\mathcal{S}_T), \quad p_T(\varphi_{h,\Delta t}, \varphi_{h,\Delta t}) &\equiv b_T\left(\varphi_{h,\Delta t}, \frac{\partial \varphi_{h,\Delta t}}{\partial t}\right) \\ &\ge \alpha_T\, \|\varphi_{h,\Delta t}\|^2_{V(\mathcal{S}_T)}. \end{aligned} \tag{5.17}$$

[illegible] 7. The inequality (5.17) is obtained by passing to the limit in the inequality (3.16) applied to $J_\varepsilon\, \varphi_{h,\Delta t}$.

5.2. The numerical scheme. We consider a second family $\mathcal{V}_h$ of finite element subspaces of $H_0^1(\Sigma)$ for the approximation of the unknown v. Our approximate solution is

$$\begin{cases} v_{h,\Delta t} \in \mathcal{P}^1_{\Delta t}(0,T;\mathcal{V}_h), \text{ the approximation of } v, \text{ with } v_h^n = v_{h,\Delta t}(t^n), \\ \varphi_{h,\Delta t} \in W_{h,\Delta t}(\mathcal{S}_T), \text{ the approximation of } \varphi, \text{ with } \varphi_h^n = \varphi_{h,\Delta t}(t^n). \end{cases} \tag{5.18}$$

The way we approximate the first equation of (4.13) is completely standard. We use a Galerkin approximation in space (in $\mathcal{V}_h$) and a θ finite difference scheme in time, where θ denotes some parameter in $[0,1]$:

(5.19)

$$\begin{cases} \text{For any } w_h \in \mathcal{V}_h \quad \text{and} \quad n = 1, 2, \dots, 2N-1, \quad v_h^0 = v_h^1 = 0, \text{ and} \\ \left(\dfrac{v_h^{n+1} - 2v_h^n + v_h^{n-1}}{\Delta t^2}, w_h \right) + a(v_h^{n,\theta}, w_h \Big) - \displaystyle\int_\Sigma \frac{\varphi_h^{n+1} - \varphi_h^{n-1}}{2\Delta t}\, w_h\, dx = \int_\Sigma f^n\, w_h\, dx, \end{cases}$$

where

$$f^n = f(t^n) \quad \text{and} \quad v_h^{n,\theta} = \theta\, v_h^{n+1} + (1-2\theta)\, v_h^n + \theta\, v_h^{n-1}. \tag{5.20}$$

For the approximation of the second equation of (4.13), the most natural process would be

$$\frac{1}{\rho}\, b_T\Big(\varphi_{h,\Delta t}, \psi_{h,\Delta t}\Big) + \iint_{\mathcal{S}_T} \frac{\partial v_{h,\Delta t}}{\partial t}\, \psi_{h,\Delta t}\, dxdt = 0 \quad \forall\, \psi_{h,\Delta t} \in V_{h,\Delta t}(\mathcal{S}_T). \tag{5.21}$$

However, this choice does not allow us to guarantee the stability of the resulting method (at least, we did not find a proof—see also Remark 9) by ensuring a discrete equivalent of the identities (4.8) or (4.12).

To make this possible, it appears that a solution consists in writing the same type of equation after "projection" on the two (even and odd) time grids. More precisely, we write that

$$\begin{cases} \dfrac{1}{\rho}\, b_T\Big(\left[\Pi^1_{ev}\varphi_{h,\Delta t}\right], \Pi^0_{ev}\psi_{h,\Delta t}\Big) + \displaystyle\iint_{\mathcal{S}_T} \frac{\partial v_{h,\Delta t}}{\partial t}\ \Pi^0_{\mathrm{ev}}\psi_{h,\Delta t}\, dxdt = 0, \\ \dfrac{1}{\rho}\, b_T\Big(\left[\Pi^1_{od}\varphi_{h,\Delta t}\right], \Pi^0_{od}\psi_{h,\Delta t}\Big) + \displaystyle\iint_{\mathcal{S}_T} \frac{\partial v_{h,\Delta t}}{\partial t}\ \Pi^0_{\mathrm{od}}\psi_{h,\Delta t}\, dxdt = 0 \end{cases} \tag{5.22}$$

for any $\psi_{h,\Delta t} \in V_{h,\Delta t}(\mathcal{S}_T)$.

Remark 8. The reader will note that, in some sense, (5.22) amounts to writing two times (5.21) using two staggered time grids of time step $2\Delta t$.

The numerical method we propose is made of (5.18), (5.19), and (5.22).

6. Analysis of the numerical method. In this section, we first present the calculations that lead to the practical implementation of the method (subsection 6.1). Then we prove the main theoretical result of this paper, namely, the stability theorem, Theorem 6.1. This is the major step to the convergence analysis of the method and the derivation of error estimates, which are delayed to a future work.

6.1. Matrix formulation. Let us introduce a basis $\{\eta_j, 1 \le j \le N_h^v\}$ (resp., $\{\xi_j^h, 1 \le j \le N_h^m\}$) of $\mathcal{V}_h$ (resp., $\mathcal{M}_h$). We can decompose the discrete unknowns (5.18) as

$$v_h^n = \sum_{j=1}^{N_h^v} v_j^n\, \eta_j, \quad \varphi_h^n = \sum_{j=1}^{N_h^v} \varphi_j^n\, \xi_j, \quad 0 \le n \le 2N. \tag{6.1}$$

We are going to rewrite (5.19), (5.22) as a linear problem in the unknown vectors

$$V_h^n = \left(v_j^n\right)_{1\le j\le N_h^v} \in \mathbb{R}^{N_h^v}, \quad \Phi_h^n = \left(\varphi_j^n\right)_{1\le j\le N_h^m} \in \mathbb{R}^{N_h^m}, \quad n = 1, 2, \ldots, 2N. \tag{6.2}$$

We denote by $\mathbf{M}_h$ the $N_h^v \times N_h^v$ mass matrix defined by

$$\mathbf{M}_h = \left((\mathbf{M}_{ij})\right)_{1\le i,j\le N_h^v}, \quad \mathbf{M}_{ij} = (\eta_i, \eta_j), \tag{6.3}$$

by $\mathbf{A}_h$ the $N_h^v \times N_h^v$ stiffness matrix defined by

$$\mathbf{A}_h = \left((\mathbf{A}_{ij})\right)_{1\le i,j\le N_h^v}, \quad \mathbf{A}_{ij} = a(\eta_i, \eta_j), \tag{6.4}$$

and by $\mathbf{C}_h$ the $N_h^v \times N_h^m$ coupling matrix defined by

$$\mathbf{C}_h = ((\mathbf{C}_{ij}))_{1\le i\le N_h^v,\ 1\le j\le N_h^m}, \quad \mathbf{C}_{ij} = \int_\Sigma \eta_i\ \xi_j\ dx. \tag{6.5}$$

Then (5.19) is equivalent to

$$\mathbf{M}_h\ \frac{V_h^{n+1} - 2V_h^n + V_h^{n-1}}{\Delta t^2} + \mathbf{A}_h\ V_h^{n,\theta} - \mathbf{C}_h \frac{\Phi_h^{n+1} - \Phi_h^{n-1}}{2\Delta t} = F_h^n, \tag{6.6}$$
$$n = 1, 2, \dots, 2N-1,$$

where $F_h^n = \left(F_j^n\right)_{1\le j\le N_h^v}$, with $F_j^n = \int_\Sigma f(x,t^n)\ \eta_j\ dx$.

The matrix formulation of (5.22) is slightly more complicated. It is based on Lemma 4. To state this lemma, we need some notation. Let $\delta > 0$ (homogeneous to a time). We introduce the subdomains of $\mathbb{R}^4$:

$$\mathcal{C}_k(\delta) = \{\ (x,y) \in \Sigma^2\ /\ c\ k\delta \le |x-y| \le c\ (k+1)\delta\ \}, \quad k \ge 1. \tag{6.7}$$

Note that these are empty sets for k large enough:

$$k > 1 + \frac{diam\ \Sigma}{c\ \delta} \Longrightarrow \mathcal{C}_k(\delta) = \emptyset.$$

We then define the $N_h^m \times N_h^m$ matrices $\mathbf{P}_h^k(\delta)$ as

$$\mathbf{P}_{ij}^k(\delta) = \delta^2 \int_{C_k(\delta)} \frac{\xi_i(x)\xi_j(y)}{4\pi|x-y|}\ dxdy. \tag{6.8}$$

Next, setting

$$\sigma_k^\delta(x,y) = \frac{1}{2}(k\ - |x-y|/c\delta)^2 \ \text{ and } \ G_{ij}(x,y) = \frac{\nabla\xi_i(x)\cdot\nabla\xi_j(y)}{4\pi|x-y|}\ ,$$

we then define $N_h^m \times N_h^m$ matrices $\mathbf{Q}_h^k(\delta)$:

$$\left|\begin{array}{l} \mathbf{Q}_{ij}^k(\delta) = \delta^2 \displaystyle\int_{C_k(\delta)} G_{ij}\ dxdy - \delta^2 \int_{C_k(\delta)} G_{ij}\ \left[\ \sigma_k^\delta + \sigma_{k-1}^\delta\ \right]\ dxdy \\ \qquad\qquad + \delta^2 \displaystyle\int_{C_{k-1}(\delta)} G_{ij}\ \sigma_{k-2}^\delta\ dxdy\ + \delta^2 \int_{C_{k+1}(\delta)} G_{ij}\ \sigma_{k+1}^\delta\ dxdy. \end{array}\right. \tag{6.9}$$

Finally, we introduce the matrices $\mathbf{N}_h^k(\delta)$ defined by

$$\mathbf{N}_h^k(\delta) = \frac{\mathbf{P}_h^{k+1}(\delta) - 2\ \mathbf{P}_h^k(\delta) + \mathbf{P}_h^{k-1}(\delta)}{\delta^2} - \mathbf{Q}_h^k(\delta). \tag{6.10}$$

Note that the matrices $\mathbf{P}_h^{k+1}(\delta)$, $\mathbf{Q}_h^k(\delta)$, and $\mathbf{N}_h^k(\delta)$ are all symmetric.

LEMMA 4. [illegible] $(\varphi_{h,\Delta t}, \psi_{h,\Delta t}) \in W_{h,\Delta t}(\mathcal{S}_T) \times V_{h,\Delta t}(\mathcal{S}_T)$ [illegible] $\Phi^n = (\varphi_j^n)_{1\le j\le N_h^m}$ $\Psi^n = (\psi_j^n)_{1\le j\le N_h^m}$ [illegible] N_h^m [illegible]

$$\varphi_{h,\Delta t} = \sum_{n=0}^{2N} \varphi_j^n\ w^n \otimes \xi_j, \quad \psi_{h,\Delta t} = \sum_{n=0}^{2N} \psi_j^n\ \zeta^n \otimes \xi_j, \tag{6.11}$$

[illegible] $\Phi^0 = \Psi^0 = 0$ [illegible]

$$b_T(\varphi_{h,\Delta t}, \psi_{h,\Delta t}) = \sum_{n=0}^{2N} \sum_{m=0}^{n} \mathbf{N}_h^{n-m}(\Delta t)\ \Phi_h^m \cdot\ \Psi_h^n, \tag{6.12}$$

[illegible] $\Phi_h \cdot\ \Psi_h$ [illegible] $\mathbb{R}^{N_h^m}$.

[illegible] We sketch the proof by presenting the formal computations that lead to formulas and that can be rigorously justified (this is omitted here) according to Lemma 3.

By linearity, we have

$$\left| \begin{aligned} b_T(\varphi_{h,\Delta t}, \psi_{h,\Delta t}) &= \sum_{m=0}^{2N} \sum_{n=0}^{2N} \sum_{i=1}^{N_h^m} \sum_{j=1}^{N_h^m} b_T(w^m \otimes \xi_i\ ,\ \zeta^n \otimes \xi_j)\ \varphi_i^m\ \psi_j^n \\ &= \sum_{m=0}^{2N} \sum_{n=0}^{2N} \mathbf{B}_T^{m,n} \Phi_h^m \cdot\ \Psi_h^n, \end{aligned} \right. \tag{6.13}$$

where $\mathbf{B}_T^{m,n}$ is the matrix given by $(\mathbf{B}_T^{m,n})_{ij} = b_T(w^m \otimes \xi_i\ ,\ \zeta^n \otimes \xi_j)$.

Using formula (3.18), we have formally

$$\left| \begin{aligned} \mathbf{B}_T^{m,n} \quad &= \mathbf{B}_{T,1}^{m,n}\ +\ \mathbf{B}_{T,2}^{m,n}, \\ \left(\mathbf{B}_{T,1}^{m,n}\right)_{ij} &= \frac{1}{2\pi} \int_\Sigma \int_\Sigma \frac{\xi_j(y)\ \xi_i(x)}{c^2|x-y|}\ J_{mn}^1(x,y)\ dxdy, \\ \left(\mathbf{B}_{T,2}^{m,n}\right)_{ij} &= \frac{1}{2\pi} \int_\Sigma \int_\Sigma \frac{\nabla\xi_j(y) \cdot \nabla\xi_i(x)}{c^2|x-y|}\ J_{mn}^2(x,y)\ dxdy, \end{aligned} \right. \tag{6.14}$$

where

$$\begin{cases} J_{mn}^1(x,y) = \displaystyle\int_0^T \ddot{w}^m\big(t - \tau(x,y)\big)\ \zeta^n(t)\ dt, \\ J_{mn}^2(x,y) = \displaystyle\int_0^T w^m\big(t - \tau(x,y)\big)\ \zeta^n(t)\ dt. \end{cases}$$

In particular, since $\operatorname{supp} w^m(\cdot - \tau(x,y)) \subset [t_{m-1}, +\infty\,[$, one sees that

$$m > n \quad \Longrightarrow \quad J_{mn}^1(x,y) = J_{mn}^2(x,y) = 0 \quad \forall (x,y)\ \in\ \Sigma \times \Sigma \quad \Longrightarrow \quad \mathbf{B}_T^{m,n} = 0.$$

For $m \le n$, the integral J_{mn}^1 has to be understood in a generalized sense since one has, in the sense of distributions ($\delta(t-a)$ denoting the Dirac distribution at time $t = a$),

$$\ddot{w}^n = \frac{1}{\Delta t}\ [\delta(t - t^{n+1}) - 2\ \delta(t - t^n) + \delta(t - t^{n-1})]. \tag{6.15}$$

Therefore,

$$J_{mn}^1(x,y) = \frac{1}{\Delta t}\left[\zeta^m\big(t^{n+1} + \tau(x,y)\big) - 2\ \zeta^m\big(t^n + \tau(x,y)\big) + \zeta^m\big(t^{n-1} + \tau(x,y)\big)\right].$$

The reader will notice that the above expression makes sense for almost every $(x,y) \in \Sigma \times \Sigma$ (as soon as $\tau(x,y)$ is not a multiple of Δt). This is what makes the rigorous

justification of these computations possible. Substituting this expression in (6.14) allows us to show that (simply note that looking at the support of the functions $(x,y) \mapsto \chi^m\big(t^n + \tau(x,y)\big)$ leads us to introduce the domains $\mathcal{C}_k(\Delta t)$)

$$\mathbf{B}_{T,1}^{m,n} = \frac{\mathbf{P}_h^{n-m+1}(\Delta t) - 2\,\mathbf{P}_h^{n-m}(\Delta t) + \mathbf{P}_h^{n-m-1}(\Delta t)}{\Delta t^2}. \tag{6.16}$$

In the same way, rather tedious but simple computations show that

$$\mathbf{B}_{T,2}^{m,n} = -\mathbf{Q}_h^{n-m}(\Delta t). \tag{6.17}$$

One concludes by regrouping (6.13), (6.14), (6.16), and (6.17). ∎

Since our scheme (5.22) amounts to working with two staggered grids of space step $2\Delta t$ (see Remark 8), it is quite easy to deduce from Lemma 4 that (5.22) is equivalent to the following discrete convolution equations:

(6.18)
$$\begin{cases} \dfrac{1}{2\rho\Delta t}\displaystyle\sum_{m=0}^{n} N_h^{n-m}(2\Delta t)\ \Phi_h^{2m+2} + \ C_h^*\ \frac{V_h^{2n+2} - V_h^{2n}}{2\Delta t} = 0, \quad n = 0,1,\dots,N-1, \\ \dfrac{1}{2\rho\Delta t}\displaystyle\sum_{m=0}^{n} N_h^{n-m}(2\Delta t)\ \Phi_h^{2m+1} + \ C_h^*\ \frac{V_h^{2n+1} - V_h^{2n-1}}{2\Delta t} = 0, \quad n = 0,1,\dots,N-1. \end{cases}$$

Remark 9. The "natural" process (5.21) would have led to (instead of (6.18))

$$\frac{1}{\rho\Delta t}\sum_{m=0}^{n} N_h^{n-m}(\Delta t)\ \Phi_h^{m+1} + \ C_h^*\ \frac{V_h^{n+1} - V_h^n}{\Delta t} = 0, \quad n = 0,1,\dots,2N-1. \tag{6.19}$$

Looking at the two coupling terms in (6.6) and (6.18), one sees that the time derivative of v is discretized differently—with a time step Δt in (6.6) and a time step $2\Delta t$ in (6.18). This is what causes trouble for establishing a stability theorem.

6.2. The computational algorithm. In this section, we describe the algorithm that permits us to compute iteratively the solution (V_h^n, Φ_h^n) of (6.6), (6.18). This also gives a proof of the existence and uniqueness of the solution of the scheme (5.18), (5.19), (5.22).

Let us introduce, for each $n \geq 0$, the vector of dimension $N_h^m + N_h^r$:

$$U_h^n = (V_h^n, \Phi_h^n)^T. \tag{6.20}$$

Assuming that U_h^k has been computed up to $k = 2n$, we explain how to compute U_h^{2n+1} and U_h^{2n+2}. From (6.6), (6.18), it is easy to see that one has to solve successively two linear systems of the same nature, namely,

$$\mathbb{K}_h(\Delta t)\ U_h^{2n+1} = B_h^{2n}, \quad \mathbb{K}_h(\Delta t)\ U_h^{2n+2} = B_h^{2n+1}, \tag{6.21}$$

where the matrix $\mathbb{K}_h(\Delta t)$ is given by the block decomposition

$$\mathbb{K}_h(\Delta t) = \begin{pmatrix} (\mathbf{M}_h + \theta\Delta t^2 \mathbf{A}_h) & -\dfrac{\Delta t}{2}\mathbf{C}_h \\ \dfrac{\Delta t}{2}\mathbf{C}_h^* & \mathbf{N}_h^0(2\Delta t) \end{pmatrix},$$

where the right-hand sides B_h^{2n} and B_h^{2n+1} are known linear combinations (that we omit to develop here) of the previous U_h^k's. B_h^{2n} depends on V_h^{2n} and $\Phi_h^{2k}, k \leq N$, and

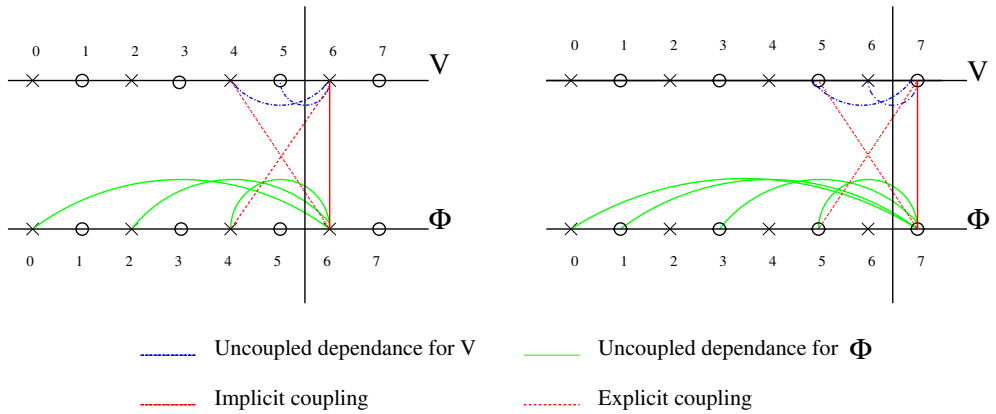


FIG. 6.1. *The computational algorithm.*

B_h^{2n+1} depends on V_h^{2n+1} and $\Phi_h^{2k+1}, k \leq N$. Let us mention that the $\mathbf{N}_h^k(2\Delta t)$'s for $k \geq 1$ appear only in the expressions of B_h^{2n} and B_h^{2n+1}.

The invertibility of S_h is ensured as soon as Δt is small enough (independently of h). Indeed, it is proven in [22] that, for small Δt, the matrix N_h^0 is symmetric positive definite. Figure 6.1 illustrates the computational algorithm.

6.3. Stability analysis. The stability analysis of the method relies on a discrete equivalent of the energy identity (4.11). In the following, we adopt the notation

$$|v|^2 = (v, v) \quad \text{and} \quad v_h^{n+\frac{1}{2}} := \frac{v_h^{n+1} + v_h^n}{2},$$

and we introduce the discrete energy of the membrane at time $t^{n+\frac{1}{2}}$:

(6.22)
$$E^{n+\frac{1}{2}} = \frac{1}{2}\left|\frac{v_h^{n+1} - v_h^n}{\Delta t}\right|^2 + \frac{1}{2}\, a(v_h^{n+\frac{1}{2}}, v_h^{n+\frac{1}{2}}) + \left(\theta - \frac{1}{4}\right)\, a\left(\frac{v_h^{n+1} - v_h^n}{\Delta t}, \frac{v_h^{n+1} - v_h^n}{\Delta t}\right).$$

This is nothing but a discrete equivalent of the continuous membrane energy (which appears as the first term in the left-hand side of the identity (4.8))

$$E(t) = \frac{1}{2}\int_\Sigma \left(\frac{1}{c_m^2}\left|\frac{\partial v}{\partial t}\right|^2 + |\nabla v|^2\right)(x,t)\, dx. \tag{6.23}$$

This discrete energy is the energy which is naturally associated to the θ-scheme for the membrane equation and one has the well-known identity

$$\begin{aligned}\frac{E^{n+\frac{1}{2}} - E^{n-\frac{1}{2}}}{\Delta t} &= \left(\rho_p \frac{v_h^{n+1} - 2v_h^n + v_h^{n-1}}{\Delta t^2}, \frac{v_h^{n+1} - v_h^{n-1}}{2\Delta t}\right) \\ &\quad + a\left(v_h^{n,\theta}, \frac{v_h^{n+1} - v_h^{n-1}}{2\Delta t}\right).\end{aligned} \tag{6.24}$$

The main result of this section is the following theorem.

THEOREM 6.1. [illegible] $(v_h^n, \varphi_{h,\Delta t})$ [illegible] (5.18), (5.19), (5.22) [illegible] n

$$E_h^{2n-\frac{1}{2}} + \frac{1}{2\rho}\left[\, |\Pi^1_{\mathrm{od}}\varphi_{\mathrm{h},\Delta\mathrm{t}}|^2_{\mathrm{N},\mathcal{S}_{\mathrm{t}2\mathrm{n}}} + |\Pi^1_{\mathrm{ev}}\varphi_{\mathrm{h},\Delta\mathrm{t}}|^2_{\mathrm{N},\mathcal{S}_{\mathrm{t}2\mathrm{n}-1}} \,\right] = \sum_{k=0}^{2n-1} \Delta t \int_\Sigma f^k \, \frac{v_h^{k+1} - v_h^{k-1}}{2\Delta t}\, dx, \tag{6.25}$$

$$E_h^{2n+\frac{1}{2}} + \frac{1}{2\rho}\left[\, |\Pi^1_{\mathrm{od}}\varphi_{\mathrm{h},\Delta\mathrm{t}}|^2_{\mathrm{N},\mathcal{S}_{\mathrm{t}2\mathrm{n}}} + |\Pi^1_{\mathrm{ev}}\varphi_{\mathrm{h},\Delta\mathrm{t}}|^2_{\mathrm{N},\mathcal{S}_{\mathrm{t}2\mathrm{n}+1}} \,\right] = \sum_{k=0}^{2n+1} \Delta t \int_\Sigma f^k \, \frac{v_h^{k+1} - v_h^{k-1}}{2\Delta t}\, dx. \tag{6.26}$$

[illegible]. We restrict ourselves to the proof of (6.25). The proof of (6.26) is almost identical.

Taking $w_h = \frac{v_h^{k+1} - v_h^{k-1}}{2\Delta t}$ in (5.19) (with $n = k$), and using identity (6.24), we obtain

$$\frac{E_p^{k+\frac{1}{2}}(v_h) - E_p^{k-\frac{1}{2}}(v_h)}{\Delta t} - \int_\Sigma \frac{\varphi_h^{k+1} - \varphi_h^{k+1}}{2\Delta t}\, \frac{v_h^{k+1} - v_h^{k-1}}{2\Delta t}\, dx = \int_\Sigma f^k \, \frac{v_h^{k+1} - v_h^{k-1}}{2\Delta t}\, dx.$$

After summation over k from 0 to $2n-1$, we get

$$\left|\begin{aligned} E_p^{2n-\frac{1}{2}}(v_h) &- \Delta t \sum_{k=0}^{n-1} \int_\Sigma \frac{\varphi_h^{2k+1} - \varphi_h^{2k-1}}{2\Delta t}\, \frac{v_h^{2k+1} - v_h^{2k-1}}{2\Delta t}\, dx \\ &- \Delta t \sum_{k=0}^{n-1} \int_\Sigma \frac{\varphi_h^{2k+2} - \varphi_h^{2k}}{2\Delta t}\, \frac{v_h^{2k+2} - v_h^{2k}}{2\Delta t}\, dx \\ &= \Delta t \sum_{k=0}^{2n-1} \int_\Sigma f^k \, \frac{v_h^{k+1} - v_h^{k-1}}{2\Delta t}\, dx. \end{aligned}\right. \tag{6.27}$$

Next, in the first equation of (5.22), we take $\psi_{h,\Delta t} = \chi^{2n\Delta t}\, \frac{\partial}{\partial t}\, \varphi_{h,\Delta t}$, which is possible since $\chi^{2n\Delta t}\, \partial/\partial t$ maps $W_{h,\Delta t}(\mathcal{S}_T)$ in $V_{h,\Delta t}(\mathcal{S}_T)$. Using the fact that

$$\Pi^0_{ev}\left[\chi^{2n\Delta t}\, \frac{\partial}{\partial t}\, \varphi_{h,\Delta t}\right] = \chi^{2n\Delta t}\, \frac{\partial}{\partial t}\, \Pi^1_{ev}\varphi_{h,\Delta t}$$

and the properties of the bilinear form $b_T(\cdot,\cdot)$ (see, in particular, (3.19)), we get

$$\left|\begin{aligned} b_T\Big(\left[\Pi^1_{ev}\varphi_{h,\Delta t}\right], \Pi^0_{ev}\left[\chi^{2n\Delta t}\, \tfrac{\partial}{\partial t}\, \varphi_{h,\Delta t}\right]\Big) &= b_T\Big(\left[\Pi^1_{ev}\varphi_{h,\Delta t}\right], \chi^{2n\Delta t}\, \tfrac{\partial}{\partial t}\, \Pi^1_{ev}\varphi_{h,\Delta t}\Big) \\ &= |\Pi^1_{ev}\varphi_{h,\Delta t}|^2_{N,\mathcal{Q}_{t2n}}. \end{aligned}\right. \tag{6.28}$$

On the other hand, one computes easily that

$$\int_{\mathcal{S}_T} \frac{\partial v_{h,\Delta t}}{\partial t} \cdot \chi^{2n\Delta t}\, \frac{\partial}{\partial t}\, \varphi_{h,\Delta t}\, dx\, dt = 2\Delta t \sum_{k=0}^{n} \int_\Sigma \frac{\varphi_h^{2k+2} - \varphi_h^{2k}}{2\Delta t}\, \frac{v_h^{2k+2} - v_h^{2k}}{2\Delta t}\, dx. \tag{6.29}$$

Finally, we obtain

$$\frac{1}{\rho\Delta t}\,|\Pi^1_{ev}\varphi_{h,\Delta t}|^2_{N,\mathcal{S}_{t^{2n}}} + 2\Delta t\sum_{k=0}^{n-1}\int_\Sigma \frac{\varphi_h^{2k+2}-\varphi_h^{2k}}{2\Delta t}\,\frac{v_h^{2k+2}-v_h^{2k}}{2\Delta t}\,dx = 0. \tag{6.30}$$

In the same way, substituting into the second equation of (5.22), $\psi_{h,\Delta t} = \chi^{(2n-1)\Delta t}\cdot\frac{\partial}{\partial t}\,\varphi_{h,\Delta t}$, we get

$$\frac{1}{\rho\Delta t}\,|\Pi^1_{od}\varphi_{h,\Delta t}|^2_{N,\mathcal{S}_{t^{2n-1}}} + 2\Delta t\sum_{k=0}^{n-1}\int_\Sigma \frac{\varphi_h^{2k+1}-\varphi_h^{2k-1}}{2\Delta t}\,\frac{v_h^{2k+1}-v_h^{2k-1}}{2\Delta t}\,dx = 0. \tag{6.31}$$

Finally, the combination (6.27) + $\frac{1}{2}$ (6.30) + $\frac{1}{2}$ (6.31) leads to the desired result. □

We are now in a position to state our main stability theorem.

THEOREM 6.2. [illegible] $\theta \geq 1/4$ [illegible]

$$\begin{cases} \displaystyle\sup_{t^{n+1}\leq T}\Big|\frac{v_h^{n+1}-v_h^n}{\Delta t}\Big| + \sup_{t^{n+1}\leq T}|\nabla v_h^{n+\frac{1}{2}}|_{L^2(\Sigma)} \;\leq\; C\,T\,\|f\|_{L^\infty(0,T;L^2(\Sigma))}, \\[2ex] \displaystyle\sup_{t^n\leq T}|\Pi^1_{ev}\varphi_{h,\Delta t}|_{S_{t^n}} + \sup_{t^n\leq T}|\Pi^1_{od}\varphi_{h,\Delta t}|_{S_{t^n}} \;\leq\; C\,T\,\|f\|_{L^\infty(0,T;L^2(\Sigma))}, \end{cases} \tag{6.32}$$

[illegible] C [illegible] Δt [illegible] $\theta < 1/4$ [illegible]

$$\exists\,\alpha<1,\quad \frac{\Delta t^2}{4}\,(1-4\theta)\,\|a_h\| \leq \alpha, \tag{6.33}$$

[illegible]

$$\|a_h\| = \sup_{v_h\in\mathcal{V}_h}\frac{a(v_h,v_h)}{|v_h|^2}. \tag{6.34}$$

[illegible] (4.12) [illegible] C [illegible] α [illegible]. We restrict ourselves to the case $\theta \geq 1/4$. The case $\theta < 1/4$ differs only in technical details. Due to $\theta \geq 1/4$, we have

$$\Big|\frac{v_h^{n+1}-v_h^n}{\Delta t}\Big|^2 \leq 2\,E_h^{n+\frac{1}{2}}.$$

Thus, from the two identities (6.25) and (6.26), we deduce the existence of a positive constant C (that depends only on σ) such that

$$\Big|\frac{v_h^{n+1}-v_h^n}{\Delta t}\Big|^2 \;\leq\; C\sum_{k=0}^{n-1}\|f^k\|_{L^2}\Big|\frac{v_h^{k+1}-v_h^{k-1}}{\Delta t}\Big|\,\Delta t.$$

Then, thanks to a discrete Gronwall lemma, we deduce

$$\Big|\frac{v_h^{n+1}-v_h^n}{\Delta t}\Big| \;\leq\; C\sum_{k=0}^{n}\|f^n\|_{L^2(\Sigma)}\,\Delta t \;\leq\; C\,t^n\,\|f\|_{L^\infty(0,T;L^2(\Sigma))}. \tag{6.35}$$

Using (6.35) in (6.25) and (6.26), we easily obtain

$$E_h^{n+\frac{1}{2}} \ \leq \ C \ (t^n)^2 \ \|f\|^2_{L^\infty(0,T;L^2(\Sigma))},$$

from which the first inequality of (6.32) follows immediately.

Next, (6.25) and (6.26) also lead to

$$|\Pi^1_{od}\varphi_{h,\Delta t}|^2_{N,\mathcal{S}_{t^n}} + |\Pi^1_{ev}\varphi_{h,\Delta t}|^2_{N,\mathcal{S}_{t^n}} \ \leq \ C \ (t^n)^2 \ \|f\|^2_{L^\infty(0,T;L^2(\Sigma))}$$

for any $t^n \leq T$. This yields the second inequality of (6.32). □

. . . . 10. The CFL condition is nothing but the standard CFL condition for the θ-scheme applied to the plate equation only.

7. Extension to the vibrations of a plate. We present the extension of the numerical method of section 4 to the vibrations of a plate, considering the two classical plate models: Kirchhoff–Love [10], [7] and Mindlin–Reissner [10], [32]. For simplicity, we shall assume that the material constituting the plate is an isotropic solid characterized by its Young modulus E and its Poisson's coefficient ν. However, the extension to the anisotropic case is straightforward (see [29], for instance). The surface density of the plate will be denoted ρ_p and its thickness will be denoted δ. What is important to notice is that, for both models, the normal velocity v is, as for the case of the membrane, one of the unknowns of the problems and, consequently, the kinematic coupling fluid-structure condition (2.3) is preserved. The changes of the models affect essentially the first equation of system (2.2): the wave equation has to be replaced by a plate equation. However, we show in this section how the coupled model can be put in the abstract variational framework (4.13) in such a way that the method of section (5) (and the results of section 6) can be applied directly. For the two models, the unknown φ has the same meaning as for the case of the membrane: it represents the pressure jump across Σ.

7.1. The case of a Kirchhoff–Love plate. In this case, the normal velocity is the only unknown of the mechanical model. The equations for the movement of the plate are

$$\begin{cases} \rho_p \ \delta \ \dfrac{\partial^2 v}{\partial t^2} + \dfrac{\delta^3 E}{12(1-\nu^2)} \ \Delta^2 v = f + \left[\dfrac{\partial p}{\partial t}\right], & x \in \Sigma, \quad t > 0, \\ v(x,t) = \dfrac{\partial v}{\partial \nu}(x,t) = 0, & x \in \partial\Sigma, \quad t > 0, \end{cases} \tag{7.1}$$

where f and $[p]$ have the same meaning as in section 2. We have considered the case of an isotropic plate, clamped along its boundary, which explains the boundary conditions in (7.1) ($\nu \in \mathbb{R}^2$ is the unit vector of the plane $z = 0$, normal to $\partial\Sigma$, outgoing with respect to Σ).

Proceeding as in section 4, it is not difficult to show that the problem (7.1) can be reduced in a problem of the form (4.13), where only the meanings of the inner product $(\cdot,\cdot)$, of the bilinear form $a(\cdot,\cdot)$, and of the space $\mathcal{V}$ are changed with respect to section 4. More precisely,

$$\begin{cases} \mathcal{V} = H^2_0(\Sigma), \\ (v,w) = \rho_p \ \delta \ \displaystyle\int_\Sigma v \ w \ dx & \forall \ (v,w) \ \in \ L^2(\Sigma)^2, \\ a(v,w) = \dfrac{\delta^3 E}{12(1-\nu^2)} \ \displaystyle\int_\Sigma \Delta v \ \Delta w \ dx & \forall \ (v,w) \ \in \ \mathcal{V}^2. \end{cases} \tag{7.2}$$

7.2. The case of a Mindlin–Reissner plate. The Mindlin–Reissner model is slightly more complicated than in the Kirchhoff–Love model in the sense that, in addition to the normal velocity v, a new unknown has to be introduced for the description of the movements of the plate, namely, the rotation vector

$$\theta = (\theta_x, \theta_y)^t. \tag{7.3}$$

The governing equation is a linear hyperbolic system of second order, namely,

(7.4)
$$\begin{cases} \rho_p\ \delta\ \dfrac{\partial^2 v}{\partial t^2} - \delta\ \mathrm{div}[\ G\ (\nabla v + \theta)\ ] & = f + \left[\dfrac{\partial p}{\partial t}\right], \quad x \in \Sigma, \quad t > 0, \\ \rho_p\ \dfrac{\delta^3}{12}\ \dfrac{\partial^2 \theta}{\partial t^2} - \dfrac{\delta^3}{12}\ \mathrm{Div}\left(\underline{\underline{C}}\ \underline{\underline{\varepsilon}}(\theta)\right) + \delta\ G\ (\nabla v + \theta) & = 0, \qquad x \in \Sigma, \quad t > 0, \\ v(x,t) = \theta(x,t) & = 0, \qquad x \in \partial\Sigma, \quad t > 0, \end{cases}$$

which depends on E and ν, where $\underline{\underline{\varepsilon}}$ is the linearized plane deformation tensor defined by

$$\underline{\underline{\varepsilon}}(\theta) := \underline{\underline{\varepsilon}}(\theta)_{i,j} = \frac{1}{2}\left(\frac{\partial \theta_i}{\partial x_j} + \frac{\partial \theta_j}{\partial x_i}\right) \quad \forall\ i,j = \{1,2\}.$$

$G = \frac{E}{2(1+\nu)}$ and $\underline{\underline{C}}$ is the stiffness tensor given by (the indices below belong to $\{1,2\}$ and $\delta_{\alpha\beta}$ is the usual Kronecker symbol)

$$C^{\alpha\beta\lambda\mu} = \frac{E}{2(1+\nu)}\left(\ \delta_{\alpha\lambda}\ \delta_{\beta\mu} + \delta_{\alpha\mu}\ \delta_{\beta\lambda} + \frac{2\nu}{1-\nu}\delta_{\alpha\beta}\ \delta_{\lambda\mu}\ \right). \tag{7.5}$$

After reduction to Σ, one can write a variational formulation of the type (4.13), provided that one has made the substitution

$$v \to (v, \theta), \quad w \to (w, \eta) \tag{7.6}$$

and one has defined

(7.7)
$$\begin{cases} \mathcal{V} = H_0^1(\Sigma) \times H_0^1(\Sigma) \quad \text{and for any } ((v,\theta),(w,\eta)) \text{ in } \mathcal{V}^2, \\ \big((v,\theta),(w,\eta)\big) = \rho_p\ \delta\ \displaystyle\int_\Sigma v\ w\ dx + \rho_p\ \frac{\delta^3}{12}\ \int_\Sigma \theta \cdot \eta\ dx, \\ a\big((v,\theta),(w,\eta)\big) = \dfrac{\delta^3}{12}\displaystyle\int_\Sigma \underline{\underline{C}}\ \underline{\underline{\varepsilon}}(\theta) : \varepsilon(\eta) + \delta \int_\Sigma G\ (\nabla v + \theta) \cdot (\nabla w + \eta). \end{cases}$$

8. Numerical results.

- Concerning the continuous model:
  - The sound speed in the fluid is $c = 1$.
  - The membrane (or plate) Σ is the square $[-5,5] \times [-5,5]$.
  - The loading force f is localized at the center of Σ and its time dependence is a Ricker-type function with a central frequency f_0:

$$\left|\begin{array}{l} f(t,x) = g(x)h(t), \\ h(t) = -2\pi^2 f_0^2\ [2\pi^2(f_0 t - 1)]\ e^{-\pi^2(f_0 t - 1)^2}, \\ g(x) = e^{-\pi^2 |x|^2} \quad \text{if } |x| < 1, \quad 0 \text{ if not.} \end{array}\right.$$

- Concerning the discretization:
  - For the discretization of the pressure jump φ, i.e., for the space $\mathcal{M}_h$, we use P_1 continuous Lagrange elements [11] associated with a triangular mesh of Σ.
  - The discretization grid for φ is constructed by partitioning Σ into equal $N \times N$ squares of size $h = 0.2 \Longleftrightarrow N = 50$ and dividing each small square via its diagonal into two square triangles.
  - We take $\Delta t = 0.07$ to have a CFL: $c\ (2\Delta t)/h = 0.7$ for the time discretization of retarded potentials.
  - For the 3D-2D computations, the computational box for the 3D part of the computation is a parallelepipedic box that surrounds Σ in a symmetric way. It is chosen large enough in order to be sure that the computation is not affected by the presence of the exterior artificial boundary for the duration of the experiment.

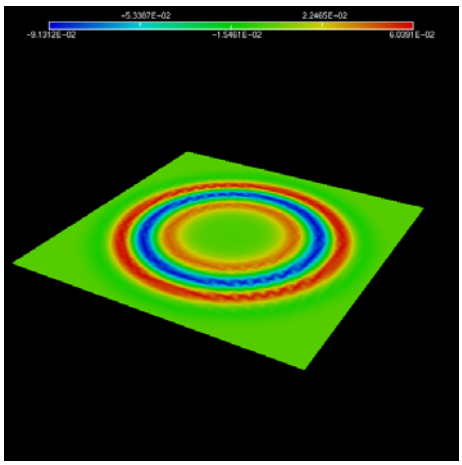
The numerical results that we present in this section are not supposed to reproduce realistic physical experiments but are essentially devoted to validate our numerical method by comparing the results of the computations with those obtained with another code based on the coupling between 3D and 2D finite elements. Such a code has already been validated for both cases of the membrane [9] and of the plate [7], [13]. We shall consider successively the case of the membrane (section 8.1) and of the Mindlin–Reissner plate (section 8.2). Both experiments will have common features. Moreover, in section 8.2, we shall also compute a reference solution computed with the 3D-2D method with very high accuracy (high order finite elements). This will allow us to compare the respective accuracy of the BEM-FEM and 3D-2D methods.

8.1. Vibrations of a membrane. We feature in this section some numerical results that illustrate the pressure jump $\varphi_{h,\Delta t}$ and the transverse velocity $v_{h,\Delta t}$, solutions of the system (5.19)–(5.22) where bilinear forms $(.,.)$ and $a(.,.)$ are given in (4.14). Toward this end, the speed velocity in the membrane is $c_m = 1$ and the central frequency f_0 is taken equal to 0.5 such that the central wavelength in each media is $\lambda = 2$. Parameters for the spatial discretization are well appropriated for this computation case.

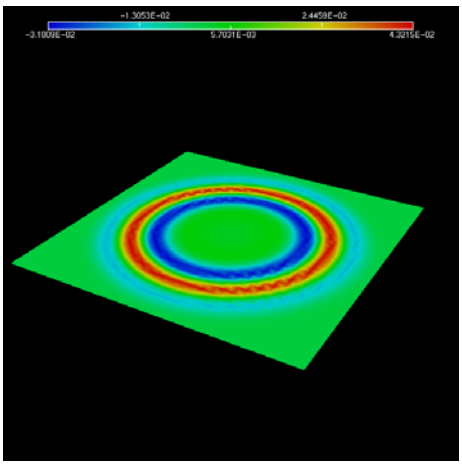
In order to illustrate the physical phenomenon we want to simulate, we represent in Figure 8.1 two snapshots of $\varphi_{h,\Delta t}$ and $v_{h,\Delta t}$ at times $t = 4.2s$ and $t = 8.4s$. Due to the fact that the two propagation velocities c and c_m are here equal and to the fact that the source is centered at the middle of the membrane, this phenomenon is particularly simple: propagation at velocity 1 along the membrane and reflection of the four boundaries in a symmetric way.

In Figures 8.2 and 8.3, we observe and compare $\varphi_{h,\Delta t}$ and $v_{h,\Delta t}$ as functions of the time at the center of the membrane during 30s. We have successively computed these solutions with our new BEM-FEM method and a 2D-3D finite element code. We can see that both solutions are very closed, which brings to the fore that our BEM-FEM method is stable and computes the good solution with a very good precision.

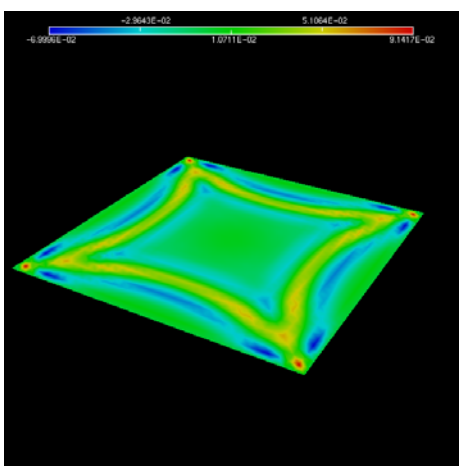
8.2. Vibrations of a plate. In this section, we consider a Reissner–Mindlin plate model instead of the membrane model. We feature here some results related to the pressure jump $\varphi_{h,\Delta t}$ and the transverse velocity $v_{h,\Delta t}$ that satisfy the system (5.19)–(5.22) where bilinear forms are given in (7.7). We have chosen to keep the same computational mesh as in the previous section (in particular to avoid recomputing the matrices $\mathbf{N}_h^M$—one of the more expensive steps of the computations). Consequently,



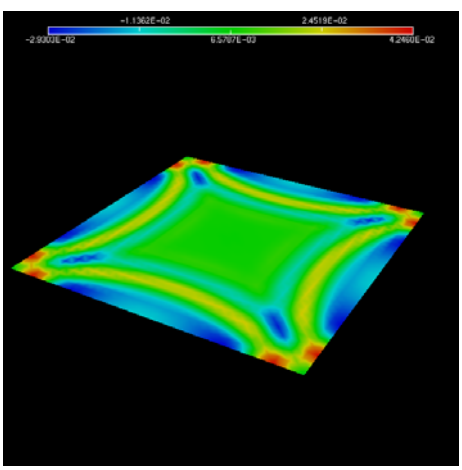
(a) Pressure jump at $t = 4.2s$



(b) Transverse velocity at $t = 4.2s$



(c) Pressure jump at $t = 8.4s$



(d) Transverse velocity at $t = 8.4s$

FIG. 8.1. *Snapshots of the solutions.*

we have chosen the following set of values for the physical parameters:

$$\rho_p = 1,\ E = 0.1297,\ \nu = 0.3,\ f_0 = 1,$$

and have been able to check numerically that, for this choice, our computational mesh is fine enough. Note that such parameters do not correspond to an existing physical material and are just used for an academic benchmark.

As in the previous subsection, we first give some snapshots (cf. Figure 8.4) that illustrate the involved physical phenomenon. As one can expect, this phenomenon is much more complex than for the membrane: in particular, the dispersive nature of the waves along the plate is clearly visible.

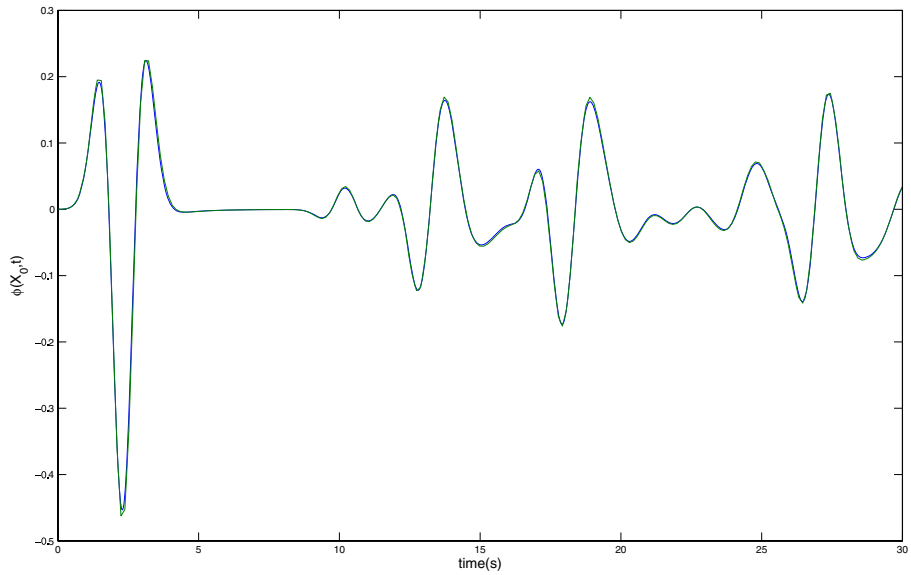


FIG. 8.2. *The pressure jump at the center of the plate. The solid line represents the* 2*D*-3*D method, and the dotted line represents the BEM-FEM method.*

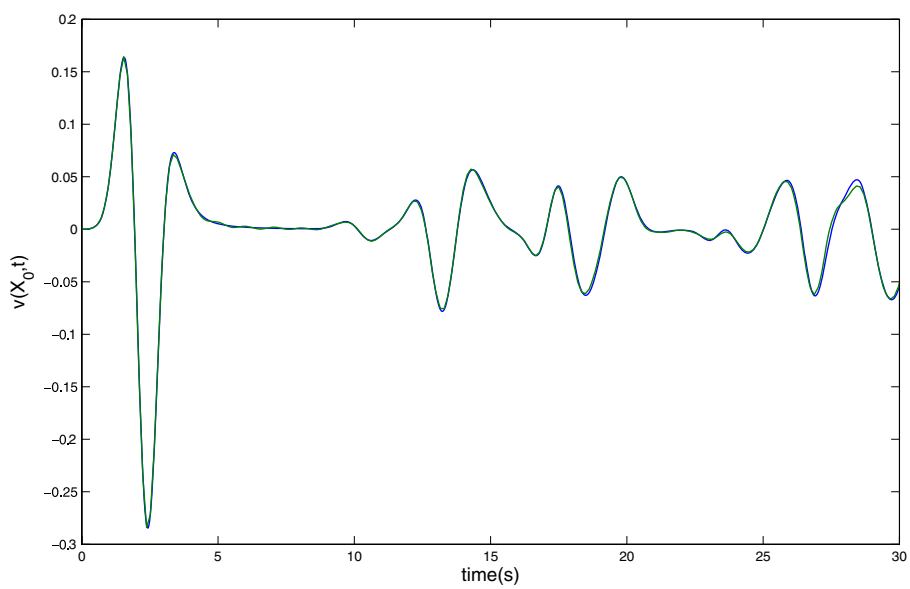
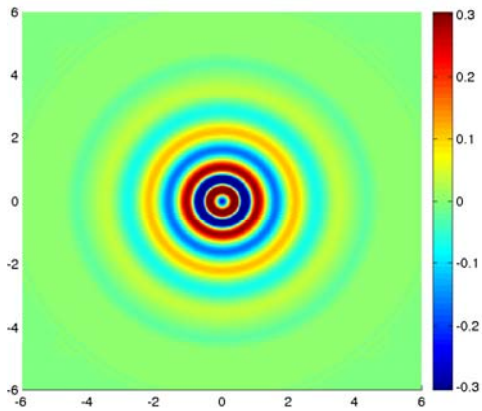
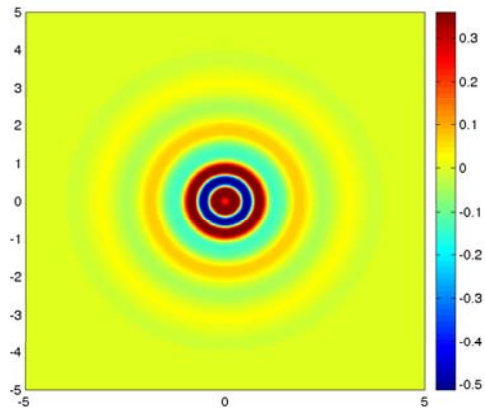


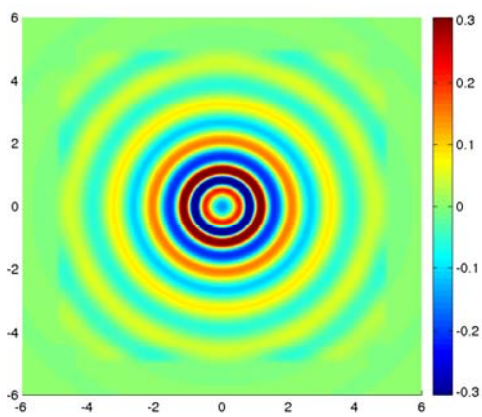
FIG. 8.3. *The transverse velocity at the center of the plate. The solid line represents the* 2*D*-3*D method, and the dotted line represents the BEM-FEM method.*



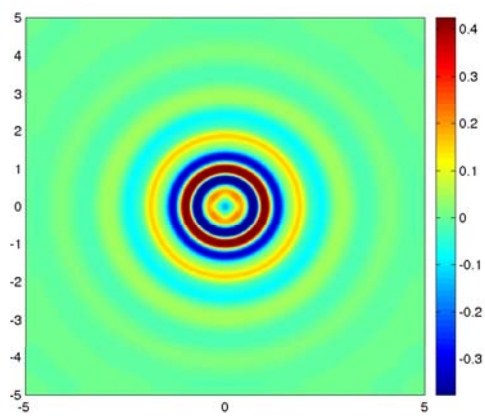
(a) Pressure jump at $t = 5s$



(b) Transverse velocity at $t = 5s$



(c) Pressure jump at $t = 7.5s$



(d) Transverse velocity at $t = 7.5s$

FIG. 8.4. *Snapshots of the solutions.*

Next we observe $\varphi_{h,\Delta t}$ and $v_{h,\Delta t}$ as functions of the time but at point $x_1 = (-2, 5, 0)$ (this is no longer the center of the plate), during 50s. Figures 8.5 and 8.6 show the comparison between the reference solution and the solution computed with our BEM-FEM method. The results are quite convincing with respect to the validation of our method.

Finally, in order to show the advantage of our BEM-FEM method in terms of accuracy (the use of retarded potential minimizes numerical dispersion), we show the transverse velocity and the pressure jump obtained with the 2D-3D finite element code with exactly the same parameters as for the spatial discretization. Figures 8.7 and 8.8 seem to acknowledge the fact that, with the same computational mesh, the BEM-FEM method is more accurate than the 2D-3D finite element one. Our interpretation is that this is probably due to the fact that the acoustic propagation is treated in a more exact way via retarded potentials: one eliminates the dispersion effects due to the discretization of the 3D wave equation.

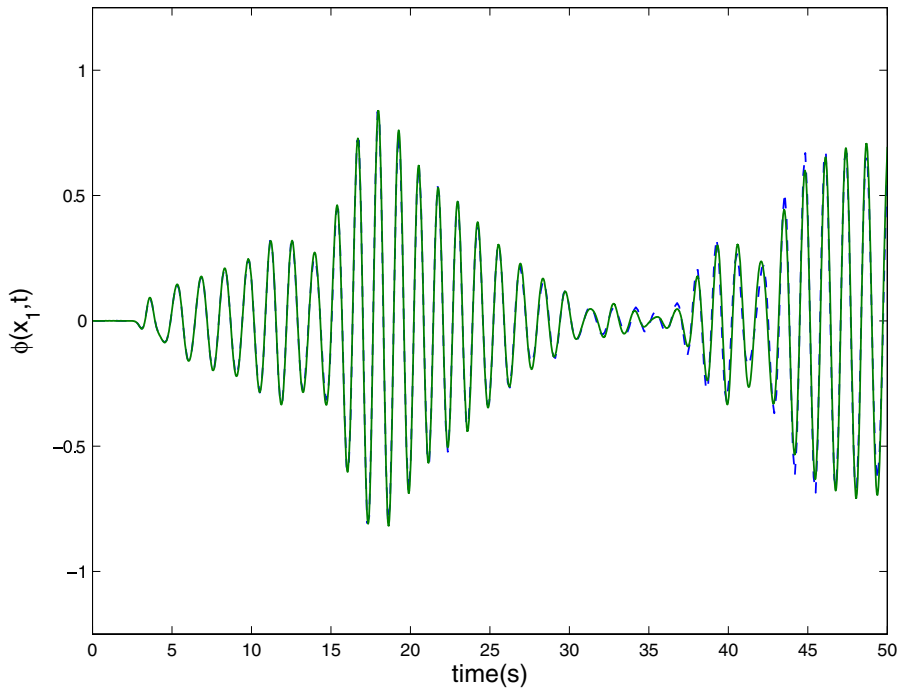


FIG. 8.5. *The pressure jump with the BEM-FEM method. The dashed line represents the BEM-FEM method, and the dotted line represents the reference solution.*

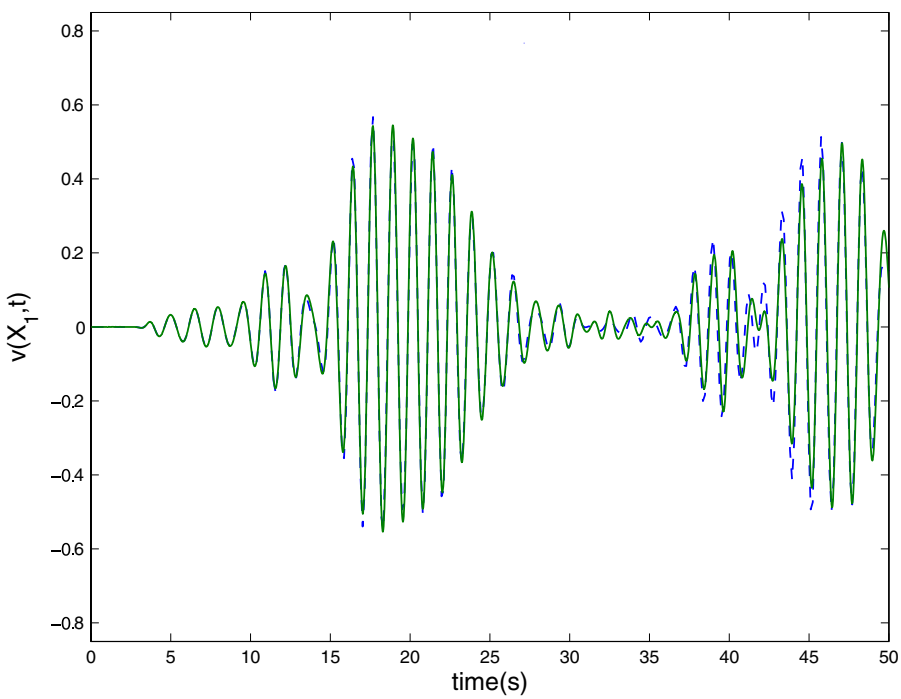


FIG. 8.6. *The transverse velocity with the BEM-FEM method. The dashed line represents the BEM-FEM method, and the dotted line represents the reference solution.*

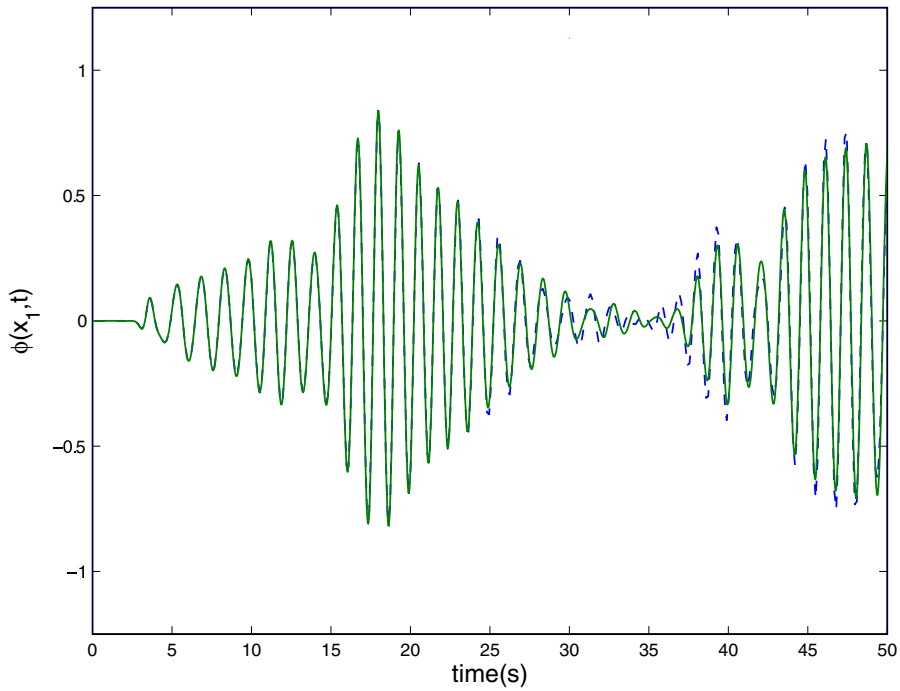


FIG. 8.7. *The pressure jump obtained with the 2D-3D finite element code. The dashed line represents the 2D-3D method, and the dotted line represents the reference solution.*

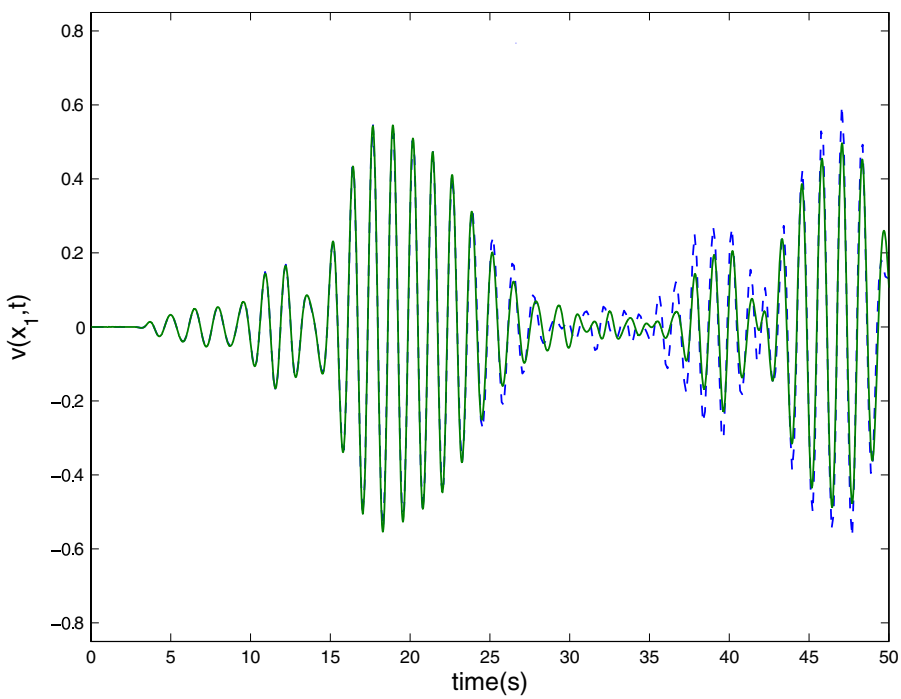


FIG. 8.8. *The transverse velocity obtained with the 2D-3D finite element code. The dashed line represents the 2D-3D method, and the dotted line represents the reference solution.*

Acknowledgments. The authors would like to thank Toufic Abboud, Eliane Bécache, and Isabelle Terrasse for stimulating discussions. They also thank Toufic Abboud for having provided the computations of the matrices $\mathbf{N}_h^m$.

REFERENCES

[1] T. Abboud, J. El Gharib, and B. Zhou *Retarded potentials for acoustic impedance problems*, in Mathematical and Numerical Aspects of Wave Propagation (Santiago de Compostela, 2000), SIAM, Philadelphia, 2000, pp. 703–708.

[2] A. Bachelot and V. Lubet, *On the coupling of boundary element and finite element methods for a time problem*, in Mathematical and Numerical Aspects of Wave Propagation (Mandelieu-La Napoule, 1995), SIAM, Philadelphia, 1995, pp. 130–139.

[3] A. Bachelot, L. Bounhoure, and A. Pujols, *Couplage éléments finis–potentiels retardés pour la diffraction électromagnétique par un obstacle hétérogène*, Numer. Math., 89 (2001), pp. 257–306.

[4] E. Bécache, *Variational integral method in the time domain for elastic scattering problems*, in Boundary Elements in Mechanical and Electrical Engineering (Nice, 1990), Comput. Mech., Southampton, UK, Springer-Verlag, Berlin, 1990, pp. 219–230.

[5] E. Bécache, P. Joly, and J. Rodriguez, *Space-time mesh refinement for elastodynamics. Numerical results*, Comput. Methods Appl. Mech. Engrg., 194 (2005), pp. 355–366.

[6] E. Bécache, *A variational boundary integral equation method for an elastodynamic antiplane crack*. Internat. J. Numer. Methods Engrg., 36 (1993), pp. 969–984.

[7] E. Bécache, G. Derveaux, and P. Joly, *An efficient numerical method for the resolution of the Kirchhoff-Love dynamic plate equation*, Numer. Methods Partial Differential Equations, 21 (2005), pp. 323–348.

[8] A. De La Bourdonnaye, *Some formulations coupling finite element and integral equation methods for Helmholtz equation and electromagnetism*, Numer. Math., 69 (1995), pp. 257–268.

[9] A. Chaigne, P. Joly, and L. Rhaouti, *Time-domain modeling and numerical simulation of a kettledrum*, J. Acoustical Society of America, 105 (1999), pp. 3545–3562.

[10] D. Chapelle and K. J. Bathe, *The Finite Element Analysis of Shells—Fundamentals*, Computational Fluid and Solid Mechanics, Springer-Verlag, Berlin, 2003.

[11] P. G. Ciarlet, *The Finite Element Method for Elliptic Problems*, Studies in Mathematics and Its Applications 4, North–Holland, Amsterdam, 1978.

[12] G. Cohen, *Higher-order numerical methods for transient wave equations (with a foreword by R. Glowinski)*, Scientific Computation, Springer-Verlag, Berlin, 2002.

[13] G. C. Cohen and P. Grob, *Mixed higher order spectral finite elements for Reissner–Mindlin equations*, SIAM J. Sci. Comput., to appear.

[14] G. Cohen and S. Fauqueux, *Mixed spectral finite elements for the linear elasticity system in unbounded domains*, SIAM J. Sci. Comput., 26 (2005), pp. 864–884.

[15] F. Collino, T. Fouquet, and P. Joly, *Conservative space-time mesh refinement methods for the FDTD solution of Maxwell's equations*, J. Comput. Phys., 211 (2006), pp. 1, 9–35.

[16] J. Diaz and P. Joly, *Robust high order non conforming finite element formulation for time domain fluid structure interaction*, J. Comput. Acoust., 13 (2005), pp. 403–431.

[17] A. Ergin, S. Arif, B. Shanker, and E. Michielssen, *Fast evaluation of three-dimensional transient wave fields using diagonal translation operators*, J. Comput. Phys., 146 (1998), pp. 157–180.

[18] L. Greengard and V. Rokhlin, *A new version of the fast multipole method for the Laplace equation in three dimensions*, in Acta Numerica 1997, Acta Numer. 6, Cambridge University Press, Cambridge, UK, 1997, pp. 229–269.

[19] T. Ha-Duong, *On retarded potential boundary integral equations and their discretisation*, in Topics in Computational Wave Propagation, Lect. Notes Comput. Sci. Eng. 31, Springer-Verlag, Berlin, 2003, pp. 301–336.

[20] T. Ha-Duong, *Space-time variational formulas and calculations of retarded potential*, in Boundary Elements VII, Vol. I, II (Como, 1985), Comput. Mech., Southampton, Springer-Verlag, Berlin, 1985, pp. 5.51–5.60.

[21] T. Ha Duong, *A finite element method for the double-layer potential solutions of the Neumann exterior problem*, Math. Methods Appl. Sci., 2 (1980), pp. 191–208.

[22] T. Ha-Duong, B. Ludwig, and I. Terrasse, *A Galerkin BEM for transient acoustic scattering by an absorbing obstacle*, Internat. J. Numer. Methods Engrg., 57 (2003), pp. 1845–1882.

[23] R. Hiptmair, *Coupling of finite elements and boundary elements in electromagnetic scattering*, SIAM J. Numer. Anal., 41 (2003), pp. 919–944.
[24] P. Joly, *Variational methods for time-dependent wave propagation problems*, in Topics in Computational Wave Propagation, Lect. Notes Comput. Sci. Eng. 31, Springer-Verlag, Berlin, 2003, pp. 201–264.
[25] G. Lebeau and M. Schatzman, *A wave problem in a half-space with a unilateral constraint at the boundary*, J. Differential Equations, 53 (1984), pp. 309–361.
[26] C. Lesueur, *Rayonnement Acoustique des Structures*, Edition Eyrolles, Paris, 1988.
[27] J.-L. Lions and E. Magenes, *Problémes aux limites non homogénes et applications*, Vol. 1, Travaux et Recherches Mathématiques 17, Dunod, Paris, 1968.
[28] W. J. Mansur, G. Yu, J. A. M. Carrer, S. T. Lie, and E. F. N. Siqueira, *The θ scheme for time-domain BEM/FEM coupling applied to the 2-D scalar wave equation*, Comm. Numer. Methods Engrg., 16 (2000), pp. 439–448.
[29] P. M. Naghdi, *On the theory of thin elastic shells*, Quart. Appl. Math., 14 (1957), pp. 369–380.
[30] J.-C. Nédélec, *Integral equations with nonintegrable kernels*, Integral Equations Operator Theory, 5 (1982), pp. 562–572.
[31] J.-C. Nédélec, *Curved finite element methods for the solution of singular integral equations on surfaces in R^3*, Comput. Methods Appl. Mech. Engrg., 8 (1976), pp. 61–80.
[32] E. Reissner, *The effect of transverse shear deformation on the bending of elastic plates*, J. Appl. Mech., 12 (1945), pp. 69–77.
[33] D. Soares, Jr. and W. J. Mansur, *An efficient time-domain BEM/FEM coupling for acoustic-elastodynamic interaction problems*, CMES Comput. Model. Eng. Sci., 8 (2005), pp. 153–164.
[34] E. P. Stephan and M. Maischak, *A posteriori error estimates for FEM-BEM couplings of three-dimensional electromagnetic problems*, Comput. Methods Appl. Mech. Engrg., 194 (2005), pp. 441–452.
[35] K. Yosida, *Functional analysis. Reprint of the 6th edition*, Classics in Mathematics, Springer-Verlag, Berlin, 1995.

SIAM J. SCI. COMPUT.
Vol. 29, No. 3, pp. 1160–1178
© 2007 Society for Industrial and Applied Mathematics

AN ACCELERATED KERNEL-INDEPENDENT FAST MULTIPOLE METHOD IN ONE DIMENSION*

P. G. MARTINSSON† AND V. ROKHLIN‡

Abstract. A version of the fast multipole method (FMM) is described for charge distributions on the line. Previously published schemes of this type relied either on analytical representations of the potentials to be evaluated (multipoles, Legendre expansions, Taylor series, etc.) or on tailored representations that were constructed numerically (using, e.g., the singular value decomposition (SVD), artificial charges, etc.). The algorithm of this paper belongs to the second category, utilizing the matrix compression scheme described in [H. Cheng, Z. Gimbutas, P. G. Martinsson, and V. Rokhlin, SIAM J. Sci. Comput. 26 (2005), pp. 1389–1404]. The resulting scheme exhibits substantial improvements in the CPU time requirements. Furthermore, the scheme is applicable to a wide variety of potentials; in this respect, it is similar to the SVD-based FMMs. The performance of the method is illustrated with several numerical examples.

Key words. fast multipole method, fast summation, matrix interpolation, skeletonization

AMS subject classifications. 65F30

DOI. 10.1137/060662253

1. Introduction.

We consider the problem of rapidly evaluating the sum

$$u_m = \sum_{n=1}^{N} K(x_m, x_n)\, q_n, \qquad \text{for } m = 1, \ldots, N, \tag{1.1}$$

given a set of points $(x_n)_{n=1}^N$ in $\mathbb{R}$, a kernel K that is smooth away from the diagonal, and a set of real or complex numbers $(q_n)_{n=1}^N$. We refer to the numbers q_n as "sources," the numbers u_n as "potentials," and the points x_n as "source locations."

The sum (1.1) can be straightforwardly evaluated using $O(N^2)$ floating point operations. In many applications, this cost is prohibitively large, and a number of methods that reduce the cost to $O(N)$ or $O(N \log^\kappa N)$ have been developed, [2, 10]. In this paper, we describe an $O(N)$ method that is a development of the fast multipole method (FMM) [10, 9]. The changes introduced enable the application of the method to a large class of kernels and makes the method significantly faster than existing techniques for the case where the sum (1.1) has to be evaluated several times for a fixed set of locations $(x_n)_{n=1}^N$ (see section 6).

There are two principal differences between the method presented here and the original FMM of [10]. The first concerns the way sources and potentials are represented. While the FMM relies on analytic properties of the kernel, the method presented here relies on a precomputation step that adaptively constructs representations that are optimized for the given kernel and the given source locations. The second difference concerns the treatment of the interactions between adjacent subintervals on the finest level of subdivision. While the original FFM evaluates such

*Received by the editors June 6, 2006; accepted for publication (in revised form) September 25, 2006; published electronically May 22, 2007.
http://www.siam.org/journals/sisc/29-3/66225.html
†Department of Applied Mathematics, University of Colorado, Boulder, CO 80309-0526 (martinss@colorado.edu)
‡Department of Mathematics, Yale University, New Haven, CT 06520-8283 (vladimir.rokhlin@yale.edu)

interactions directly, the representations used by the method of this paper enable the compression of such interactions.

The method presented here is similar to the methods of [8, 15, 25] in that they all rely on adaptively computed representations, and thus work for a wide range of kernels. While the methods of [8, 25] rely on the singular value decomposition (SVD) to compress the kernel and [15] relies on the QR-decomposition, this paper is based on a technique described in [5, 18, 17] that we refer to as "skeletonization." Similar techniques were previously used in [19, 20, 24]. We also note that the method presented here bears some similarity to the kernel-independent FMM of [26] (which can in turn be viewed as a development of [1, 16]). However, while the method of this paper relies only on certain smoothness properties of the kernel, the method of [26] requires the kernel to satisfy certain Green-type identities (as kernels associated with fundamental solutions of constant coefficient partial differential operators tend to do).

This paper is structured as follows: Section 2 lists some results from numerical linear algebra that will be of use. Section 3 describes the adaptive technique for the representation of sources and potentials that the current method relies on. Section 4 describes a fast summation scheme based on the same data structures as the original FMM, but using the representation described in section 3. Section 5 describes an additional acceleration of the method based on compressing interactions between [illegible] boxes, including adjacent ones. Section 6 reports the results of several numerical experiments, and section 7 summarizes the results.

[illegible] 1. This paper deals with the one-dimensional case. Extensions of the method to higher dimensions are under investigation and will be reported at a later date.

2. Preliminaries. The fast summation technique described in this paper achieves acceleration by approximating off-diagonal blocks of the matrix $[K(x_m, x_n)]_{m,n=1}^{N}$ in (1.1) by low-rank matrices. The particular matrix factorization we use to represent the low-rank matrix was described in [13] and [5]. The following lemma summarizes the facts needed for our purposes.

LEMMA 1. [illegible] A [illegible] $M \times N$ [illegible] k [illegible] $C_1, \ldots, C_N$ [illegible] $R_1, \ldots, R_M$ [illegible]

$$A = [C_1, \ldots, C_N] = \begin{bmatrix} R_1 \\ \\ R_M \end{bmatrix}.$$

[illegible]

$$A = A_{\text{col}} \circ \text{proj}, \tag{2.1}$$

[illegible] proj [illegible] $k \times N$ [illegible] $k \times k$ [illegible] A_{col} [illegible] $M \times k$ [illegible] k [illegible] A

$$A_{\text{col}} = [C_{n_1}, \ldots, C_{n_k}].$$

[illegible]

$$A = \text{eval} \circ A_{\text{row}}, \tag{2.2}$$

eval $M \times k$ $k \times k$
A_{row} $k \times N$ k A

$$A_{\text{row}} = \begin{bmatrix} R_{m_1} \\ \\ R_{m_k} \end{bmatrix}.$$

eval proj 1

2. Since the matrix proj contains a $k \times k$ identity matrix it can be applied to a vector using $k \times (N-k)$ multiplications and additions. Similarly, the matrix eval can be applied to a vector using $k \times (M-k)$ multiplications and additions.

3. As a direct consequence of Lemma 1, the condition number of proj is bounded by $\sqrt{1+k(N-k)}$ and the condition number of eval is bounded by $\sqrt{1+k(M-k)}$.

4. For simplicity, Lemma 1 is stated only for the case where the matrix A has exact rank k. Similar factorizations can be constructed for matrices of approximate rank k; see [5].

5. Methods for computing the factorizations in Lemma 1 are described in [5, 13]. The computational cost is typically $O(MNk)$ but can in rare cases be slightly higher.

3. Representation of functions via tabulation.

3.1. Outline. At the core of the original FMM is a technique for compactly representing sources and potentials. A source distribution inside a box is represented by a multipole expansion about the center of the box. From this expansion, the potential caused by the source distribution at distant target points can be evaluated. For a given accuracy, only a small number of terms in the expansion are needed, regardless of how many sources made up the original distribution. In the same way that source distributions are efficiently represented via multipole expansions, potentials are represented by giving the expansion coefficients in an expansion in harmonic polynomials.

In this section, we describe an alternative technique for representing sources and potentials. In order to describe the technique, we consider a simple model problem: Assume that we are given N source locations $(y_n)_{n=1}^N$ in a set b, M target locations $(x_m)_{m=1}^M$ in a set a, an interaction kernel $K(x,y)$, and that for a given vector of sources $q^b = (q_n)_{n=1}^N$, we wish to determine the vector of potentials $u^a = (u_m)_{m=1}^M$ given by

$$u^a = \text{direct}(a,b)\, q^b, \tag{3.1}$$

where $\text{direct}(a,b)$ is the $M \times N$ matrix with entries $K(x_m, y_n)$. We demonstrate that if the matrix $\text{direct}(a,b)$ has rank k (to within some precision ε), then it is possible to choose a subset of k source locations $(y_{n_j})_{j=1}^k$ with the property that the potential u^a can be replicated at all source points by placing some "proxy" sources on the points y_{n_j}. These proxy sources form the representation of the original source distribution. Similarly, it is possible to choose a subset of k target locations $(x_{m_i})_{i=1}^k$ with the property that if the potential is known at these k points, then it can be interpolated to all the remaining points. The potentials $(u_{m_i})_{i=1}^k$ form the representation for the potential u^a.

The representation technique described in this section has two principal advan-

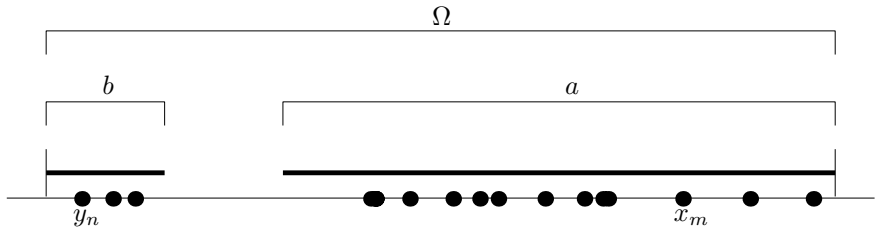


FIG. 1. *The computational domain described in section* 3.2.

tages over techniques based on analytic methods, such as multipole expansions: (1) It is cheaper to construct the proxy sources that represent a source distribution than it is to compute the corresponding multipole expansion (or any other similar representation). (2) The operator that maps a representation for a source distribution to a representation for a potential (sometimes called a "translation operator") is a submatrix of the original matrix of interaction direct(a, b). As a result, this operator need never be separately constructed or stored.

3.2. Notation. We assume that we are given a computational domain Ω, containing a number of source locations. For a given subset $b \subset \Omega$, we let $(y_n)_{n=1}^N$ denote the locations inside b, and we let $q^b = (q_n)_{n=1}^N$ denote a set of sources located at the points $(y_n)_{n=1}^N$. We let a denote the set of all points x that are [illegible] from b, meaning that $\text{dist}(x, b) \geq \text{diam}(b)$, where $\text{diam}(b) = \sup_{y,y' \in b} |y - y'|$. We let $(x_m)_{m=1}^M$ denote the locations inside a, and let $u^a = (u_m)_{m=1}^M$ denote the potential on a induced by the charges q^b (so that u^a and q^b satisfy (3.1)); see Figure 1.

3.3. Construction of the representation. We first consider the task of evaluating a potential in a caused by a set of charges in b. To this end, we form the $M \times N$ matrix $A = \text{direct}(a, b)$ with entries $K(x_m, y_n)$, we determine its ε-rank k, form a $k \times N$ matrix $\text{proj}(b)$, and an index vector $(n_j)_{j=1}^k$ such that, cf. (2.1),

$$A = A_{\text{col}} \circ \text{proj}(b) + O(\varepsilon),$$

where A_{col} is the $M \times k$ matrix whose jth column is the n_jth column of A. Then given any charge distribution q^b on b, we form the vector

$$\psi^b = \text{proj}(b)\, q^b \in \mathbb{C}^k. \tag{3.2}$$

The vector ψ^b has the property that the potential in a, caused by the charge distribution q^b, can within precision ε be reconstructed from ψ^b, since

$$u^a = A\, q^b = \big(A_{\text{col}} \circ \text{proj}(b) + O(\varepsilon)\big)\, q^b = A_{\text{col}}\, \psi^b + O(\varepsilon). \tag{3.3}$$

We say that ψ^b is the [illegible] of q^b, and that the points $(y_{n_j})_{j=1}^k$ form the [illegible] of b.

We next consider the task of evaluating a potential in b caused by a charge distribution in a. To this end, we form the $N \times M$ matrix $B = \text{direct}(b, a)$ with entries $K(y_n, x_m)$, we determine its ε-rank k, and form an $N \times k$ matrix $\text{eval}(b)$ and an index vector $(n_i)_{i=1}^k$ such that

$$B = \text{eval}(b) \circ B_{\text{row}} + O(\varepsilon),$$

where B_{row} is the $k \times M$ matrix whose ith row is the n_ith row of B. Then given any distribution q^a of charges on a, we form the vector

$$\phi^b = B_{\text{row}}\, q^a \in \mathbb{C}^k. \tag{3.4}$$

The vector ϕ^b has the property that the potential on b can within precision ε be reconstructed from ϕ^b only, since

$$u^b = B\, q^a = \big(\text{eval}(b) \circ B_{\text{row}} + O(\varepsilon)\big)\, q^a = \text{eval}(b)\, \phi^b + O(\varepsilon). \tag{3.5}$$

We say that ϕ^b is the [illegible] of u^b, and that the points $(y_{n_i})_{i=1}^k$ form the [illegible] of b.

[illegible] 6. The numbers in the vectors ψ^b and φ^b admit simple heuristic interpretations: Writing out equation (3.3) componentwise, we find that

$$u^a_m = \sum_{j=1}^{k} K(x_m, y_{n_j})\, \psi^b_j + O(\varepsilon), \qquad m = 1, \ldots, M. \tag{3.6}$$

In other words, the numbers ψ^b_j can be interpreted as charges that replicate the potential u^a when placed at the skeleton points y_{n_j}. Analogously, writing out (3.4) componentwise, we find that

$$\phi^b_i = \sum_{m=1}^{M} K(y_{n_i}, x_m)\, q^a_m + O(\varepsilon) = u^b_{n_i} + O(\varepsilon), \qquad i = 1, \ldots, k. \tag{3.7}$$

In other words, the number ϕ^b_i is simply the potential at the point y_{n_i}.

[illegible] 7. For many kernels it is possible to prove that when two sets a and b are separated by some finite distance, there exist basis functions $\{f_j\}_{j=1}^p$ and $\{g_j\}_{j=1}^p$ such that

$$\sup_{(x,y)\in a\times b} |K(x,y) - \sum_{j=1}^{p} f_j(x) g_j(y)| \leq \varepsilon, \tag{3.8}$$

where p tends to scale as a small power of $|\log \varepsilon|$ as $\varepsilon \to 0$. When (3.8) holds, the number p provides an upper bound on the numerical rank of the matrix of interaction direct(a, b), regardless of the number of target and source points and their locations. Moreover, when a formula like (3.8) can be constructed using analytical properties of the kernel, it can be used to accelerate the computation of proj and eval; cf. section 3.6.

[illegible] 8. In most environments, it is possible to construct the representations (3.3) and (3.5) in such a manner that the outgoing and the incoming skeletons are identical; see [5]. This causes only a minor increase in the number p.

3.4. Converting outgoing to incoming representations. In this section, we construct a matrix that converts the outgoing representation for a set of charges in one box, to an incoming representation for the potential induced by this set of charges in a different box.

Given two well-separated boxes a and b in Ω, suppose that we are given the outgoing representation $\psi^b = (\psi_j)_{j=1}^{k_b}$ for a charge distribution q^b in b, and that we wish to determine the incoming representation ϕ^a for the potential u^a induced by q^b.

We let $(x_m)_{m=1}^M$ and $(y_n)_{n=1}^N$ denote the locations in a and b, and we let $(x_{m_i})_{i=1}^{k_a}$ and $(y_{n_j})_{j=1}^{k_b}$ denote the corresponding skeletons. Then (3.7) implies that $\phi_i^a = u_{m_i}^a + O(\varepsilon)$, and (3.6) implies that $u_{m_i}^a = \sum_{j=1}^{k_b} K(x_{m_i}, y_{n_j})\, \psi_j^b + O(\varepsilon)$. In other words, letting $\mathrm{oi}(a,b)$ denote the $k_a \times k_b$ matrix with entries $K(x_{m_i}, y_{n_j})$, we find that

$$\phi^a = \mathrm{oi}(a,b)\, \psi^b + O(\varepsilon). \tag{3.9}$$

We refer to the matrix $\mathrm{oi}(a,b)$ as an [illegible] translation operator. This operator is a submatrix of $\mathrm{direct}(a,b)$. One ramification of this fact is that $\mathrm{oi}(a,b)$ need not be explicitly constructed. Since the kernel $K(x,y)$ is known, $\mathrm{oi}(a,b)$ is uniquely defined by the incoming and outgoing skeletons of a and b.

[illegible] 9. The matrix $\mathrm{direct}(a,b)$ is related to the matrices $\mathrm{eval}(a)$, $\mathrm{oi}(a,b)$, and $\mathrm{proj}(b)$ via the relation

$$\mathrm{direct}(a,b) = \mathrm{eval}(a) \circ \mathrm{oi}(a,b) \circ \mathrm{proj}(b) + O(\varepsilon). \tag{3.10}$$

To prove (3.10), we first note that $\mathrm{direct}(a,b)$ is defined by the relation

$$u^a = \mathrm{direct}(a,b)\, q^b. \tag{3.11}$$

Moreover, (3.5), (3.9), and (3.2) imply that

$$u^a = \mathrm{eval}(a)\, \phi^a + O(\varepsilon), \tag{3.12}$$

$$\phi^a = \mathrm{oi}(a,b)\, \psi^b + O(\varepsilon), \tag{3.13}$$

$$\psi^b = \mathrm{proj}(b)\, q^b, \tag{3.14}$$

respectively. Since (3.11) must hold for every q^b, the equations (3.11), (3.12), (3.13), and (3.14) together imply (3.10). (We remark that (3.10) is analogous to (3.1) in [5].)

3.5. Merging the representations of two boxes. In this section we describe a procedure for the construction of the outgoing representation of a set b if the outgoing representations are known for two sets b_1 and b_2 such that $b = b_1 \cup b_2$.

We let a denote the set of points that are well-separated from b, and for $j = 1, 2$, we similarly let a_j denote the set of points that are well-separated from b_j. It follows that $a \subseteq a_j$. Typically, a is strictly smaller than a_j.

We let u^a denote the potential induced on the locations $(x_m)_{m=1}^M \subset a$ by two given charge distributions in b_1 and b_2 with outgoing representations $\psi^{(1)}$ and $\psi^{(2)}$. Furthermore, we let $y^{(1)}$ and $y^{(2)}$ be the corresponding outgoing skeletons. (We recall that $y^{(1)}$ and $y^{(2)}$ are subsets of the original sets of locations in b_1 and b_2.) Merging these two pairs of vectors, we form $\hat{\psi} = [\psi^{(1)}, \psi^{(2)}]$ and $\hat{y} = [y^{(1)}, y^{(2)}]$. Since a is wholly contained in $a_1 \cap a_2$, (3.6) implies that

$$u_m^a = \sum_{j=1}^{k_1+k_2} K(x_m, \hat{y}_j)\, \hat{\psi}_j + O(\varepsilon) \qquad \text{for } m = 1, \ldots, M, \tag{3.15}$$

where k_1 and k_2 are the lengths of $\psi^{(1)}$ and $\psi^{(2)}$. We rewrite (3.15) as a matrix-multiplication by introducing the $M \times (k_1+k_2)$ matrix A with entries $A_{mj} = K(x_m, \hat{y}_j)$,

$$u^a = A\, \hat{\psi} + O(\varepsilon). \tag{3.16}$$

Equation (3.15) says that $\hat{\psi}$ is a valid outgoing representation for b with an associated outgoing skeleton $\hat{y}$. However, when a contains fewer target locations than $a_1 \cup a_2$ (which is typically the case), this representation is likely to be longer than necessary. To be precise, the length of an optimal representation equals the ε-rank of the matrix A in (3.16), which we denote by k. To obtain an optimal representation, we factor A as in (2.1),

$$A = \tilde{A}_{\text{col}} \circ Z + O(\varepsilon),$$

where Z is a $k \times (k_1 + k_2)$ matrix, and $\tilde{A}_{\text{col}}$ consists of k columns of A. We denote the indices of these by $(j_i)_{i=1}^k$. An optimal outgoing representation is then the vector $\psi^b = Z\,\hat{\psi}$; it is associated with the outgoing skeleton $(\hat{y}_{j_i})_{i=1}^k$. Letting $\text{oo}(b, b_1)$ denote the matrix formed by the first k_1 columns of Z and letting the remaining columns form $\text{oo}(b, b_2)$, we find that

$$\psi^b = \text{oo}(b, b_1)\,\psi^{(1)} + \text{oo}(b, b_2)\,\psi^{(2)}.$$

Analogously, we construct matrices $\text{ii}(b_1, b)$ and $\text{ii}(b_2, b)$ that construct the incoming representations $\phi^{(1)}$ and $\phi^{(2)}$ for b_1 and b_2 from the incoming representation of b via the formulas

$$\phi^{(1)} = \text{ii}(b_1, b)\,\phi^b \qquad \text{and} \qquad \phi^{(2)} = \text{ii}(b_2, b)\,\phi^b. \tag{3.17}$$

We refer to the matrix $\text{oo}(b, b_1)$ as an [illegible] translation operator, while the matrix $\text{ii}(b_1, b)$ is referred to as an [illegible] translation operator.

3.6. Efficient construction of translation operators. The techniques for constructing translation operators that were given in sections 3.3, 3.4, and 3.5 could potentially be quite expensive. For instance, when computing the operator $\text{proj}(b)$ in section 3.3, we considered the interaction between the set of locations in b and the set of all locations in Ω that are well-separated from b. The second set is typically very large. However, in most instances of practical interest, the process can be accelerated by replacing this very large set by a small set of predetermined locations in a that act as proxies for the actual charge locations. This acceleration technique works for kernels satisfying the following assumption.

Assumption I. [illegible] b [illegible] Ω [illegible] a [illegible] Ω [illegible] b [illegible] ε [illegible] $(z_i)_{i=1}^p \subset a$ [illegible] $(f_i)_{i=1}^p$ [illegible]

$$\sup_{x \in a} \sup_{y \in b} \left| K(x, y) - \sum_{i=1}^{p} f_i(x)\, K(z_i, y) \right| \leq \varepsilon. \tag{3.18}$$

[illegible] p [illegible]

$$p \leq C |\log \varepsilon|,$$

[illegible] $\varepsilon \to 0$ [illegible] $(w_i)_{i=1}^p \subset a$ [illegible] $(g_i)_{i=1}^p$ [illegible]

$$\sup_{x \in a} \sup_{y \in b} \left| K(y, x) - \sum_{i=1}^{p} K(y, w_i)\, g_i(x) \right| \leq \varepsilon. \tag{3.19}$$

REMARK 10. It is shown in [17] that Assumption I holds for any kernel that is separable in the sense described in Remark 7.

Suppose now that the kernel K satisfies Assumption I. Then for for any given set $b \subset \Omega$, one can quite inexpensively construct an outgoing representation that is valid for evaluating potentials at the points z_i. Assumption I then assures us that this representation is also valid at any other point that is well-separated from b, since the potential there can be interpolated locally from the potential at the z_i's. Incoming representations can be constructed analogously.

REMARK 11. In general, the cost of constructing the matrix $\text{proj}(b)$ and determining the outgoing skeleton $(y_{n_j})_{j=1}^{k}$ is $O(kNM)$, where k is the ε-rank of interaction, N is the number of points in b, and M is the number of locations that are well-separated from b. When the kernel K satisfies Assumption I, this cost can be reduced to $O(kNp)$ with no dependence on M.

3.7. Representations with extended domains of validity. For a given box b, we have so far constructed outgoing and incoming representations that are valid with respect to locations in the computational domain that are separated from a box b by at least the diameter of b. This requirement of a "buffer" zone is a standard feature of fast summation techniques. However, the particular technique for representing functions described in section 3.3 can also be used to compress the interaction between adjacent boxes. The resulting representations are very similar to the ones constructed for the interactions between separated boxes. However, the expansions are longer, and should not be used unless necessary. We call such unbuffered representations "rich representations," and denote the outgoing and the incoming rich representations by Ψ^b and Φ^b, respectively. The corresponding projection and evaluation operators are called Proj and Eval (capitalized), respectively, so that for a given box b

$$\Psi^b = \text{Proj}(b)\, q^b, \tag{3.20}$$

$$u^b = \text{Eval}(b)\, \Phi^b. \tag{3.21}$$

The representation Ψ^b in (3.20) can be used to compute the potential induced by q^b at any location outside b, while the representation Φ^b may be used to reconstruct any potential u^b induced by a distribution of charges at any of the locations outside b.

REMARK 12. In the original FMM and its variants, the existence of a buffer between source and target boxes was necessary to assure that the classical representations (multipole series, etc.) are valid and have controlled convergence rates. Here, we eliminate the need for the buffers by constructing the representations via numerical techniques (as opposed to using analytic properties of the kernel). Such representations are possible because they are not required to be valid everywhere in the boxes, but only at a finite number of source and target points.

REMARK 13. If for a given box b, we wish to compute both its rich and its regular outgoing representations, the most efficient way for doing so is to first compute the rich representation Ψ^b, and then compute the regular one, ψ^b, from it.

In order to construct the matrix that maps Ψ^b to ψ^b, we first construct the matrix A with entries $A_{mj} = K(x_m, y_j)$, where $(y_j)_{j=1}^{K}$ are the points in the rich skeleton and $(x_m)_{m=1}^{M}$ is a set of well-separated target points. Letting k denote the ε-rank of A, we factor A as in (2.1),

$$A = A_{\text{col}} \circ \overline{\text{proj}(b)} + O(\varepsilon),$$

where A_{col} is the $M \times k$ matrix consisting of the columns of A with indices $(j_i)_{i=1}^k$, and $\overline{\text{proj}(b)}$ is a $k \times K$ matrix. If Ψ^b is an outgoing representation associated with the rich outgoing skeleton $(y_j)_{j=1}^K$, then

$$\psi^b = \overline{\text{proj}(b)}\, \Psi^b$$

is a regular outgoing representation for b associated with the outgoing skeleton $(y_{j_i})_{i=1}^k$.

Analogously, we construct an operator $\overline{\text{eval}(b)}$ that constructs a rich incoming representation from a regular one:

$$\Phi^b = \overline{\text{eval}(b)}\, \phi^b. \tag{3.22}$$

Other types of translation operators involving rich representations can be constructed, but are not needed for the purposes of this paper.

4. A kernel-independent fast summation technique.

4.1. Problem formulation. In this section, we consider the problem of evaluating the sum

$$u_i = \sum_{j=1}^{N} K(x_i, x_j)\, q_j, \qquad i = 1, \ldots, N, \tag{4.1}$$

given a set of real numbers $(x_i)_{i=1}^N$, a set of real or complex numbers $(q_j)_{j=1}^N$, and a kernel $K(x, y)$. We call the numbers x_i "locations," the numbers q_i "charges," and the numbers u_i "potentials." The numbers x_i are all contained in an interval Ω called the "computational domain."

We focus on the situation where the potentials $(u_i)_{i=1}^N$ are to be evaluated for a sequence of charge distributions $(q_i)_{i=1}^N$ associated with a single set of locations $(x_i)_{i=1}^N$. In this environment, we spend a moderate amount of computational effort on optimizing the representations and the various translation operators used for the given set of locations. Once this has been done, each potential evaluation can be performed rapidly.

The summation technique presented in this section follows the same template as earlier versions of the fast multipole method [10, 11, 8, 6]. The only difference between these methods and the method presented in this section is that a different representation for potentials and sources is used. In section 5, a further acceleration of the method is described. This acceleration is obtained by compressing the interactions between adjacent regions.

4.2. Tree structure. Given a computational domain $\Omega = [x_{\text{left}}, x_{\text{right}}) \subset \mathbb{R}$ and a set of locations $(x_n)_{n=1}^N \subset \Omega$, we construct an adaptive partitioning of the domain into subintervals in such a way that no interval contains more than N_0 locations, for some given (small) integer N_0. We do this via a hierarchical subdivision process in which any interval holding more than N_0 points is split into two halves, and then the process is continued with each half that, in turn, contains more than N_0 points. If $a = [x^a_{\text{left}}, x^a_{\text{right}})$ is an interval resulting from this process, then

$$\begin{aligned} x^a_{\text{left}} &= x_{\text{left}} + (j-1)\, 2^{-l} \left(x_{\text{right}} - x_{\text{left}}\right), \\ x^a_{\text{right}} &= x_{\text{left}} + j\, 2^{-l} \left(x_{\text{right}} - x_{\text{left}}\right), \end{aligned}$$

for some number $l = 0, 1, 2, \ldots$, and some number $j = 1, 2, \ldots, 2^l$. The number l is called the "level" of the box a and denotes how many times Ω has been cut in half to reach a.

For every box a, we construct the following lists of other boxes:

$\mathrm{L}_{\mathrm{children}}(\mathrm{a})$: For a nonleaf box a, this is a list of all boxes b that are contained in a and that are separated from a by one level only. If $b \in \mathrm{L}_{\mathrm{children}}(a)$, we say that b is a "child" of a and that a is a "parent" of b. For a leaf box, $\mathrm{L}_{\mathrm{children}}(a)$ is empty.

$\mathrm{L}_{\mathrm{close}}(a)$: For a leaf-box a, this is a list of all leaf boxes b that are not well-separated from a. For a nonleaf box, $\mathrm{L}_{\mathrm{close}}(a)$ is empty.

$\mathrm{L}_2(a)$: The list of all boxes b on the same level as a that are well-separated from a, but whose parents are not separated from the parent of a. (This list is known as the "interaction list" in the original FMM.)

4.3. Outgoing and incoming representations. For any box a, we let the vector of charges inside the box be denoted by q^a, and with $\mathrm{proj}(a)$ the matrix defined in section 3.3, we let the vector

$$\psi^a = \mathrm{proj}(a)\, q^a \tag{4.2}$$

denote the outgoing representation of a. Similarly, we let u^a_{far} denote the potential on a caused by all charges that are well-separated from a. An incoming representation for a is a vector ϕ^a such that

$$u^a_{\mathrm{far}} = \mathrm{eval}(a)\, \phi^a + O(\varepsilon), \tag{4.3}$$

where the operator $\mathrm{eval}(a)$ is defined in section 3.3.

. . . . 14. In the original fast multipole method, the function of the vector ψ^a was performed by a vector of multipole coefficients representing q^a, and the function of ϕ^a was performed by a vector of expansion coefficients for u^a_{far} in a basis of harmonic polynomials.

4.4. Hierarchical construction of representations. Both the incoming and the outgoing representations can be computed recursively. Specifically, if a is any nonleaf box a, its outgoing representation can be computed from the outgoing representations of its children via the formula

$$\psi^a = \sum_{b \in \mathrm{L}_{\mathrm{children}}(a)} \mathrm{oo}(a, b)\, \psi^b, \tag{4.4}$$

where the matrices $\mathrm{oo}(a, b)$ are defined in section 3.5. Similarly, if a is any box other than the root box, the incoming representation ϕ^a (representing the potential u^a_{far} caused by charges in all boxes that are well-separated from a) can be constructed by combining the incoming potential of its parent b with the outgoing potentials of all boxes in the interaction list of a via the formula

$$\phi^a = \mathrm{ii}(a, b)\, \phi^b + \sum_{c \in \mathrm{L}_2(a)} \mathrm{oi}(a, c)\, \psi^c, \tag{4.5}$$

where the list $L_2(a)$ is defined in section 4.2, the matrix $\mathrm{ii}(a, b)$ is defined in section 3.5, and the matrices $\mathrm{oi}(a, c)$ are defined in section 3.4.

4.5. Precomputation. Given a set of locations $(x_n)_{n=1}^N$ and a kernel function $K(x,y)$, the precomputation stage consists of the following steps:

1. Divide the computational domain into a tree structure, as described in section 4.2.
2. Construct the lists described in section 4.2.
3. Loop over all leaf boxes. For each box a, construct the regular outgoing and incoming skeletons. Simultaneously, determine the matrices $\mathrm{proj}(a)$ and $\mathrm{eval}(a)$, described in section 3.3.
4. Loop over all nonleaf boxes, going from finer to coarser levels. For each box a, construct the regular outgoing and incoming skeletons by merging the skeletons of its children b_i. Simultaneously, determine the matrices $\mathrm{oo}(a,b_i)$ and $\mathrm{ii}(b_i,a)$, as described in section 3.5.
5. Loop over all boxes a, then over all elements b in the list $\mathrm{L}_2(a)$. For each such pair (a,b), construct the translation operator $\mathrm{oi}(a,b)$, as described in section 5.3.

The total cost of the steps described above depends on the kernel and on the charge distribution. If the kernel satisfies Assumption I in section 3.6, then the computational cost is typically either $O(N)$ or $O(N\log N)$, and the amount of storage required is typically $O(N)$; cf. Remark 15.

4.6. A general fast multipole method. We have now assembled the tools for computing the sum (4.1) through two passes through the hierarchical tree: one upwards and one downwards.

1. Sweep over all leaf boxes a. For each box, construct its outgoing representation from the values of the charges inside it; cf. (4.2):

$$\psi^a = \mathrm{proj}(a)\, q^a.$$

2. Sweep over all nonleaf boxes a, going from finer to coarser levels. For each box a, construct its outgoing representation by merging the outgoing representations of its children; cf. (4.4):

$$\psi^a = \sum_{b\in \mathrm{L}_{\mathrm{children}}(a)} \mathrm{oo}(a,b)\,\psi^b.$$

3. Set $\phi^r = 0$ for the root box r. Then loop over all boxes (including the root box), going from coarser to finer levels. For each box a, form its incoming representation by combining the incoming representation of its parent, box b, with the contributions from the boxes in $\mathrm{L}_2(a)$ (its "interaction list"); cf. (4.5):

$$\phi^a = \mathrm{ii}(a,b)\,\phi^b + \sum_{c\in \mathrm{L}_2(a)} \mathrm{oi}(a,c)\,\psi^c.$$

4. Sweep over all leaf nodes a. For each node, form the potential u^a by evaluating the incoming representation and directly adding the contributions from the charges inside a and in all boxes that are not well-separated from a; cf. (4.3):

$$u^a = \mathrm{eval}(a)\,\phi^a + \mathrm{direct}(a,a)\,q^a + \sum_{c\in \mathrm{L}_{\mathrm{close}}(a)} \mathrm{direct}(a,c)\,q^c.$$

For kernels satisfying Assumption I, the computational cost of the steps described in this subsection is typically $O(N)$.

15. It is possible to construct (highly nonuniform) charge distributions for which the computational cost of the steps described in sections 4.5 and 4.6 exceed $O(N)$. A discussion of this phenomenon, and a modification to the scheme that makes it always retain $O(N)$ complexity, can be found in [21].

5. An accelerated kernel-independent FMM.

5.1. Outline. In this section, we again consider the problem of rapidly evaluating the sum (4.1). We describe a technique for doing so that is similar to the technique described in section 4, but with the difference that even interactions between adjacent boxes are compressed. To enable this additional compression, we keep track of four representations for each leaf box: the regular outgoing and incoming ones described in section 3.3, and also the rich outgoing and incoming representations described in section 3.7. The rich representations are used exclusively for the purpose of evaluating interactions involving at least one leaf box.

The asymptotic cost for the algorithm described in this section scales in the same way with N as the cost for the algorithm described in section 4.6 (typically, $O(N \log N)$ for precomputation and $O(N)$ for evaluation). However, the constants involved are smaller.

5.2. Tree structure. The accelerated algorithm uses the same tree structure that was described in section 4.2. In addition to the lists described in that section, the accelerated algorithm also uses the following three lists:

$\mathrm{L}_1(a)$: For a leaf box a, this is a list of the leaf boxes that directly border a. For a nonleaf box, $\mathrm{L}_1(a)$ is empty.

$\mathrm{L}_3(a)$: For a leaf box a, this is a list of all boxes on finer levels than a that are separated from a but whose parents are not separated from the parent of a. For a nonleaf box a, $\mathrm{L}_3(a)$ is empty.

$\mathrm{L}_4(a)$: The dual of L_3. In other words, $b \in \mathrm{L}_4(a)$ if and only if $a \in \mathrm{L}_3(b)$.

5.3. Translation operators. The algorithm described in section 4.6 utilizes a single type of outgoing-to-incoming translation operator. This operator maps a regular outgoing representation ψ^a to a regular incoming representation ϕ^b for all pairs (a, b) such that $a \in \mathrm{L}_2(b)$. This translation operator is also used in the accelerated algorithm described in this section, we label it $\mathrm{oi}_2(a, b)$. The accelerated algorithm requires three additional outgoing-to-incoming translation operators: For each pair (a, b) such that $b \in \mathrm{L}_1(a)$, the operator $\mathrm{oi}_1(a, b)$ maps a rich representation to a rich representation. For each pair (a, b) such that $b \in \mathrm{L}_3(a)$, the operator $\mathrm{oi}_3(a, b)$ maps a regular representation to a rich representation. For each pair (a, b) such that $b \in \mathrm{L}_4(a)$, the operator $\mathrm{oi}_4(a, b)$ maps a rich representation to a regular representation. Each matrix $\mathrm{oi}_j(a, b)$ is the restriction of the original matrix of interaction $\mathrm{direct}(a, b)$ to the relevant outgoing and incoming skeletons for a and b, respectively.

5.4. Precomputation. The precomputation for the accelerated algorithm is very similar to the precomputation described in section 4.5. The difference is that for all leaf nodes, we compute both the regular and the rich skeletons (and the corresponding operators $\overline{\mathrm{eval}}$ and $\overline{\mathrm{proj}}$). We also determine the additional lists L_1, L_3, and L_4, as well as the corresponding translation operators oi_1, oi_3, and oi_4.

5.5. Potential evaluations. After the particle locations $(x_n)_{n=1}^N$ and the kernel $K(x, y)$ have been fixed, and the precomputation described in section 5.4 has been

completed, the following steps will compute the vector of potentials $(u_n)_{n=1}^N$ from a vector of charges $(q_n)_{n=1}^N$ cf. (4.1)) to within a precision of computations ε:

1. Loop over all leaf boxes. For each box a, compute Ψ^a and then ψ^a:

$$\Psi^a = \mathrm{Proj}(a)\, q^a \quad \text{and then} \quad \psi^a = \overline{\mathrm{proj}(a)}\, \Psi^a.$$

2. Loop over all nonleaf boxes, going from finer levels to coarser. For each box a, compute ψ^a by combining the outgoing representations of its children,

$$\psi^a = \sum_{b \in \mathrm{L}_{\mathrm{children}}(a)} \mathrm{oo}(a,b)\, \psi^b.$$

3. Loop over all leaf boxes. For each box a, add up the contributions from the boxes in $\mathrm{L}_1(a)$,

$$\Phi_1^a = \sum_{b \in \mathrm{L}_1(a)} \mathrm{oi}_1(a,b) \Psi^b.$$

4. Loop over all boxes. For each box a, add up the contributions from the boxes in $\mathrm{L}_2(a)$,

$$\phi_2^a = \sum_{b \in \mathrm{L}_2(a)} \mathrm{oi}_2(a,b) \psi^b.$$

5. Loop over all leaf boxes. For each box a, add up the contributions from the boxes in $\mathrm{L}_3(a)$,

$$\Phi_3^a = \sum_{b \in \mathrm{L}_3(a)} \mathrm{oi}_3(a,b) \psi^b.$$

6. Loop over all boxes. For each box a, add up the contributions from the boxes in $\mathrm{L}_4(a)$,

$$\phi_4^a = \sum_{b \in \mathrm{L}_4(a)} \mathrm{oi}_4(a,b) \Psi^b.$$

7. Set $\phi^r = 0$ for the root box r. Then loop over all nonleaf boxes (including the root box) from coarser levels to finer. For each box a, and for each child b of a, construct the incoming regular representations for b:

$$\phi^b = \phi_2^b + \phi_4^b + \mathrm{ii}(b,a)\, \phi^a.$$

8. Loop over all leaf boxes. For each box a, construct its rich incoming representation by adding the various contributions:

$$\Phi^a = \Phi_1^a + \Phi_3^a + \overline{\mathrm{eval}(a)}\, \phi^a.$$

9. Loop over all leaf boxes. For each box a, construct u^a by interpolating Φ^a and adding the contributions from q^a:

$$u^a = \mathrm{Eval}(a)\, \Phi^a + \mathrm{direct}(a,a) q^a.$$

16. The vectors ϕ_2^a, ϕ_4^a, Φ_1^a, and Φ_3^a are never stored; they are added directly to either ϕ^a or Φ^a as they are computed.

6. Numerical examples. The numerical algorithm described in section 5 has been implemented in FORTRAN 77 and tested on several model problems. In this section, we summarize the results from tests involving three different kernels.

1. This example involves the evaluation of the sum

$$u_m = \sum_{\substack{n=1 \\ n \neq m}}^{N} \left(\log |x_m - x_n| \right) q_n \qquad \text{for } m = 1, \ldots, N, \tag{6.1}$$

where the points $(x_n)_{n=1}^N$ were drawn from a uniform random distribution on the interval [0, 1].

2. This example involves the evaluation of the sum

$$u_m = \sum_{n=1}^{N} \frac{p_{k+1}(x_m)p_k(x_n) - p_k(x_m)p_{k+1}(x_n)}{x_m - x_n} q_n \quad \text{for } m = 1, \ldots, N, \tag{6.2}$$

where p_k is the kth Legendre polynomial and the points $(x_n)_{n=1}^N$ are the Gaussian nodes on the interval $[-1, 1]$. The sum (6.2) arises in evaluating orthogonal projections onto the space of order k polynomials in $L^2([-1, 1])$; see [14]. Similar sums are encountered in the construction of fast algorithms for the harmonic expansions on the sphere, in the FMM for the Helmholtz and Maxwell equations, etc.; see [4, 22]. In the numerical examples reported, k was one third of N (rounded to the nearest integer).

3. This example involves the evaluation of the sum

$$u_m = \sum_{n=1}^{N} \frac{\sin(a(x_m - x_n))}{x_m - x_n} q_n \qquad \text{for } m = 1, \ldots, N, \tag{6.3}$$

where $(x_n)_{n=1}^N$ are equispaced nodes in the interval $[-1, 1]$. Such sums occur frequently in signal processing and many other areas; see [3, 7, 23]. The parameter a was chosen so that there were 5 nodes per wavelength, i.e., $a = \pi N/5$.

The CPU times required for the accelerated matrix-vector multiplication in the three examples are given in Figure 2. The experiments were carried out on a 3.2GHz Pentium IV desktop with 2Gb of RAM. All calculations shown were carried out with a requested accuracy of $\varepsilon = 10^{-10}$. Detailed CPU time requirements are given in Tables 2, 3, and 4 in the appendix. These tables also report the memory requirements of the algorithm, as well as the time required for the precomputing step.

For comparison, Figure 2 also reports the CPU times required for uncompressed matrix-vector multiplies, both for the case where the matrix elements are precomputed and stored and the case where the matrix elements are computed on the fly. The break-even points obtained by extrapolating the lines in Figure 2 are given in Table 1. Finally, Figure 2 reports the CPU time requirement for a Fast Fourier Transform (FFT) of length N (with equispaced nodes). We note that for large problems, the matrix-vector multiply reported here is about one order of magnitude slower than an FFT.

In order to investigate the dependence of the CPU time requirement on the requested accuracy, the matrix-vector multiplication in Example 1 was carried out for $\varepsilon = 10^{-3.5}$, 10^{-7}, and 10^{-14}. The resulting CPU times are reported in Figure 3. This table also specifies the CPU times required for the precomputational stage. We remark that no attempt whatsoever was made to optimize this part of the code, and its CPU time requirements can be reduced dramatically.

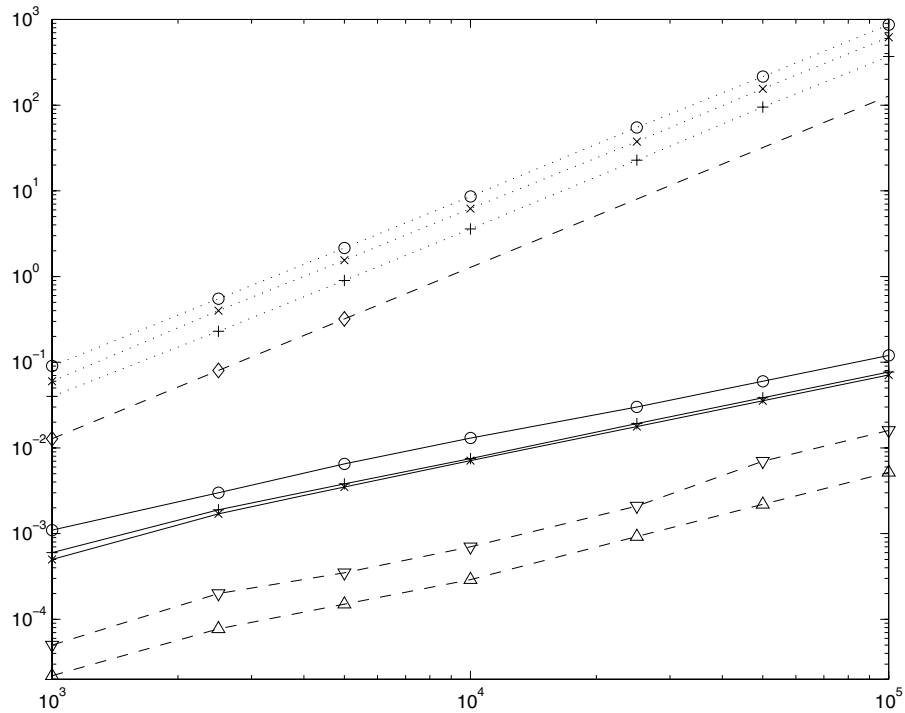


FIG. 2. *The CPU times (in seconds) required for evaluating the sums* (6.1), (6.2), *and* (6.3) *versus problem size* N. *The markers "×," "+," and "∘" label the three cases. The dotted lines mark the times required for uncompressed matrix-vector multiplies, the solid lines mark the times required for the algorithm of section* 5. *The times for a matrix-vector multiply with a stored matrix are marked with "◇" and the times for the FFT are marked "▽" for an FFTPACK implementation and "△" for an FFTW implementation.*

TABLE 1
Break-even points extrapolated from Figure 2.

	Matrix elements computed on the fly	Matrix elements precomputed
Example 1	11	52
Example 2	20	60
Example 3	14	94

7. Conclusions. This paper describes an FFM for the rapid evaluation of sums of the form (1.1). In one and two dimensions, the computational complexity of this method is $O(N)$.

The method does not use any analytic expansions of the kernel; instead, it numerically compresses the kernel in a preprocessing stage whose computational cost is $O(N \log N)$. The combined cost of the precomputation, and a single potential evaluation using the present scheme, is larger than the cost of a single evaluation using some existing precomputation-free fast summation techniques. However, if the evaluation is to be performed for a sequence of different charge distributions (on a fixed set of charge locations), then the current method outperforms most existing methods, with the important exception of convolutional sums that can be evaluated using the FFT. Moreover, since the current scheme does not explicitly rely on analytical properties

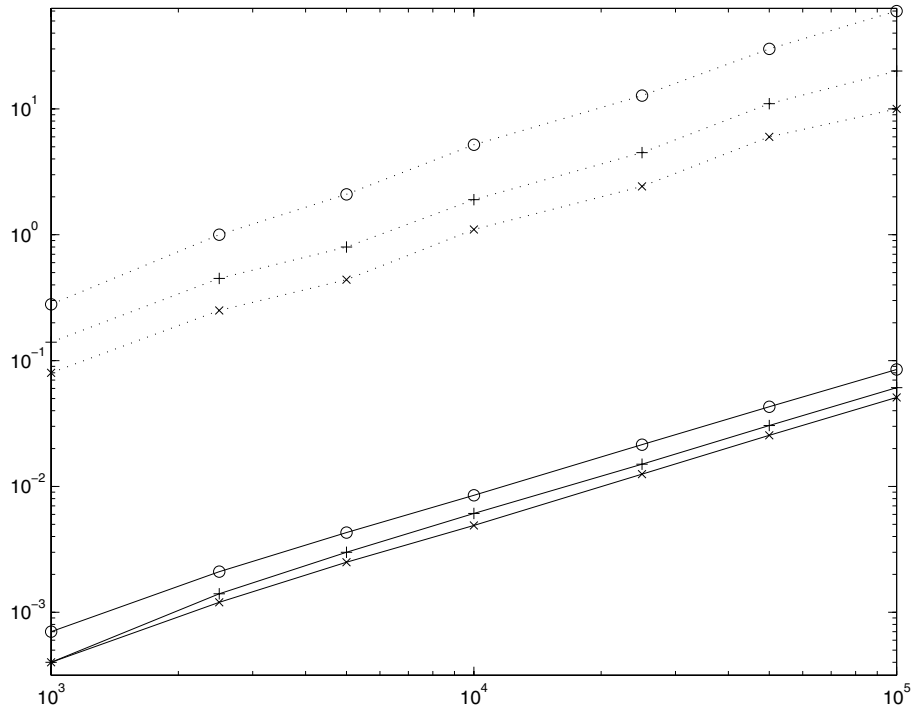


FIG. 3. *The CPU time required for evaluating the sum in* (6.1) *with* $\varepsilon = 10^{-3.5}$, 10^{-7}, *and* 10^{-14}, *using the algorithm of section* 5, *plotted against problem size* N. *The solid lines give the time for a matrix-vector multiply, while the dotted lines give the time required for precomputation.*

of the kernel, it is applicable in many environments in which precomputation-free methods are not available.

One application that seems particularly well suited for the fast summation technique of this paper concerns the rapid computation of the SVD of a matrix. A very fast algorithm for this task based on a divide-and-conquer technique is described in [12]. A core observation of [12] is that the seemingly expensive task of updating a unitary matrix can, surprisingly, be accelerated using the FMM. However, the break-even point of previous versions of the FMM made the algorithm of [12] competitive only for very large matrices. Research into combining the fast summation technique of this paper with the algorithm of [12] is currently under way.

Appendix. Computational results. This appendix contains detailed statistics for the computational experiments summarized in section 6. The numbers upon which Figure 2 is based can be found in Tables 2, 3, and 4. The numbers upon which Figure 3 is based can be found in Table 5. Table 6 provides the CPU times required for an uncompressed matrix-vector multiply and a length-N FFT using FFTPACK and FFTW.

The following numbers are given for each experiment:

N	Problem size.
ε	Requested accuracy.
$E_{\max}$	Maximum error (see Remark 17).
E_{rms}	Root mean square error (see Remark 17).
t_{pre}	CPU time required for precomputation.

TABLE 2
Computational results for Example 1, *with* $\varepsilon = 10^{-10}$.

N	t_{pre}/N	$E_{\max}$	E_{rms}	t_{eval}/N	M_{keep}/N	M_{pre}/N	t_{uncomp}/N
1000	1.8E-04	2.3E-10	3.0E-11	5.0E-07	9.9E+01	1.4E+03	7.0E-05
2500	2.6E-04	2.5E-10	2.6E-11	6.8E-07	1.0E+02	7.5E+02	1.6E-04
5000	2.5E-04	3.9E-10	3.1E-11	7.0E-07	1.0E+02	4.9E+02	3.1E-04
10000	3.2E-04	2.4E-10	2.9E-11	7.1E-07	1.0E+02	3.3E+02	6.2E-04
25000	3.0E-04	2.4E-10	2.7E-11	7.1E-07	1.1E+02	2.1E+02	1.5E-03
50000	3.6E-04	3.0E-10	2.5E-11	7.1E-07	1.1E+02	1.7E+02	3.1E-03
100000	3.4E-04	2.0E-10	2.4E-11	7.1E-07	1.1E+02	1.4E+02	6.2E-03

TABLE 3
Computational results for Example 2, *with* $\varepsilon = 10^{-10}$.

N	t_{pre}/N	$E_{\max}$	E_{rms}	t_{eval}/N	M_{keep}/N	M_{pre}/N	t_{uncomp}/N
1000	2.2E-04	1.5E-10	8.9E-13	6.0E-07	1.1E+02	1.3E+03	4.0E-05
2500	2.9E-04	1.6E-09	3.6E-12	7.6E-07	1.1E+02	8.0E+02	9.2E-05
5000	2.6E-04	1.5E-09	1.6E-12	7.6E-07	1.2E+02	5.4E+02	1.8E-04
10000	2.9E-04	2.0E-09	1.0E-12	7.5E-07	1.2E+02	3.7E+02	3.6E-04
25000	3.9E-04	4.6E-09	1.4E-12	7.7E-07	1.1E+02	2.4E+02	9.1E-04
50000	4.3E-04	2.4E-09	5.5E-13	7.7E-07	1.1E+02	1.8E+02	1.9E-03
100000	5.5E-04	6.4E-08	2.5E-12	7.7E-07	1.1E+02	1.5E+02	3.7E-03

t_{eval} CPU time required for an accelerated matrix-vector multiply.
M_{keep} Amount of memory required to store the compressed operator.
M_{pre} Amount of memory required for precomputation.
t_{uncomp} CPU time required for an uncompressed matrix-vector multiply.
t_{matvec} CPU time required for multiplying a stored matrix by a vector.
t_{fft} CPU time required for a length-N (equispaced) FFT using FFTPACK.
t_{fftw} CPU time required for a length-N (equispaced) FFT using FFTW.

All CPU times are given in seconds, and all memory requirements are given in terms of storage for double precision reals. The memory required for applying the operator to a vector is not reported since it is far smaller than M_{keep} (asymptotically, it is a couple of reals per node).

17. We report both the maximum error $E_{\max}$ and the root mean square error E_{rms}. Letting $(u_n)_{n=1}^N$ denote the result of an uncompressed matrix-vector multiply, and letting $(u'_n)_{n=1}^N$ denote the result of an accelerated matrix-vector multiply, we have

$$E_{\max} = \frac{\max_{1\le n\le N} |u_n - u'_n|}{(1/N)\sum_{n=1}^N |u_n|} \quad \text{and} \quad E_{\text{rms}} = \sqrt{\frac{\sum_{n=1}^N (u_n - u'_n)^2}{\sum_{n=1}^N |u_n|^2}}. \tag{A.1}$$

In each of the experiments reported, the charges q_n were drawn from a uniform random distribution on $[-1, 1]$.

TABLE 4
Computational results for Example 3, *with* $\varepsilon = 10^{-10}$.

N	t_{pre}/N	$E_{\max}$	E_{rms}	t_{eval}/N	M_{keep}/N	M_{pre}/N	t_{uncomp}/N
1000	7.0E-04	9.1E-10	1.6E-10	1.1E-06	1.9E+02	1.4E+03	9.0E-05
2500	8.3E-04	2.6E-09	1.9E-10	1.2E-06	1.9E+02	8.0E+02	2.2E-04
5000	1.1E-03	3.6E-09	3.7E-10	1.3E-06	2.0E+02	5.7E+02	4.3E-04
10000	1.2E-03	4.6E-09	2.0E-10	1.3E-06	2.0E+02	4.2E+02	8.6E-04
25000	1.4E-03	5.8E-09	4.6E-10	1.2E-06	2.0E+02	3.0E+02	2.2E-03
50000	1.4E-03	4.7E-09	1.2E-10	1.2E-06	2.0E+02	2.6E+02	4.3E-03
100000	1.7E-03	2.2E-09	6.8E-11	1.2E-06	2.0E+02	2.3E+02	8.7E-03

TABLE 5
Computational results for Example 1, *with three different values of* ε.

	$\varepsilon = 10^{-3.5}$		$\varepsilon = 10^{-7}$		$\varepsilon = 10^{-14}$	
N	E_{rms}	t_{eval}/N	E_{rms}	t_{eval}/N	E_{rms}	t_{eval}/N
1000	2.1E-04	4.0E-07	5.7E-08	4.0E-07	5.1E-15	7.0E-07
2500	2.4E-04	4.8E-07	6.2E-08	5.6E-07	3.1E-15	8.4E-07
5000	2.3E-04	5.0E-07	8.1E-08	6.0E-07	3.8E-15	8.6E-07
10000	2.2E-04	4.9E-07	1.0E-07	6.1E-07	4.5E-15	8.5E-07
25000	2,2E-04	5.0E-07	1.1E-07	6.0E-07	7.0E-15	8.6E-07
50000	2.1E-04	5.1E-07	1.2E-07	6.1E-07	9.0E-15	8.6E-07
100000	2.0E-04	5.1E-07	1.3E-07	6.1E-07	1.1E-14	8.5E-07

TABLE 6
CPU times for the FFT and an uncompressed matrix-vector multiply with a stored matrix.

N	1000	2500	5000	10000	25000	50000	100000
t_{fft}/N	5.0E-8	8.0E-8	6.0E-8	6.5E-8	8.4E-8	1.4E-7	1.6E-7
t_{fftw}/N	2.2E-8	3.1E-8	3.0E-8	2.9E-8	3.7E-8	4.4E-8	5.2E-8
t_{matvec}/N	1.2E-5	3.2E-5	6.4E-5	—	—	—	—

Acknowledgment. The authors wish to thank M. Tygert for constructive discussions.

REFERENCES

[1] C. R. Anderson, *An implementation of the fast multipole method without multipoles*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 923–947.
[2] J. Barnes and P. Hut, *A hierarchical $o(n \log n)$ force calculation algorithm*, Nature, 324 (1986), pp. 446–449.
[3] J. P. Boyd, *A fast algorithm for Chebyshev, Fourier, and sinc interpolation onto an irregular grid*, J. Comput. Phys., 103 (1992), pp. 243–257.
[4] H. Cheng, W. Crutchfield, Z. Gimbutas, L. Greengard, F. Ethridge, V. Rokhlin, N. Yarvin, and J. Zhao, *A Wideband Fast Multipole Method for the Helmholtz Equation in Three Dimensions*, Technical report 514, MadMax Optics, Inc., Hamden, CT, 2005.
[5] H. Cheng, Z. Gimbutas, P. G. Martinsson, and V. Rokhlin, *On the compression of low rank matrices*, SIAM J. Sci. Comput., 26 (2005), pp. 1389–1404.
[6] W. C. Chew, J. M. Jin, E. Michielsen, and J. Song, *Fast and Efficient Algorithms in Computational Electromagnetics*, Artech House, Norwood, MA, 2001.
[7] L. Debnath, *Integral Transforms and their Applications*, CRC Press, Boca Raton, FL, 1995.
[8] Z. Gimbutas and V. Rokhlin, *A generalized fast multipole method for nonoscillatory kernels*, SIAM J. Sci. Comput., 24 (2002), pp. 796–817.
[9] L. Greengard, *The Rapid Evaluation of Potential Fields in Particle Systems*, MIT Press, Cambridge, MA, 1988.
[10] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comput. Phys., 73 (1987), pp. 325–348.
[11] L. Greengard and V. Rokhlin, *A new version of the fast multipole method for the Laplace equation in three dimensions*, Acta Numer., 6 (1997), pp. 229–269.
[12] M. Gu and S. C. Eisenstat, *A divide-and-conquer algorithm for the bidiagonal SVD*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 79–92.
[13] M. Gu and S. C. Eisenstat, *Efficient algorithms for computing a strong rank-revealing QR factorization*, SIAM J. Sci. Comput., 17 (1996), pp. 848–869.
[14] R. Jakob-Chien and B. Alpert, *A fast spherical filter with uniform resolution*, J. Comput. Phys., 136 (1997), pp. 580–584.
[15] S. Kapur and D. Long, *IES^3: Efficient electrostatic and electromagnetic simulation*, IEEE Comput. Sci. Eng., 5 (1998), pp. 60–67.
[16] J. Makino, *Yet another fast multipole method without multipoles—pseudoparticle multipole method*, J. Comput. Phys., 151 (1999), pp. 910–920.
[17] P. G. Martinsson, V. Rokhlin, and M. Tygert, *On Interpolation and Integration in Finite-dimensional Spaces of Bounded Functions*, Technical report YALEU/DCS/RR-1317, Department of Computer Science, Yale, New Haven, CT, 2005.
[18] P. G. Martinsson and V. Rokhlin, *A fast direct solver for boundary integral equations in two dimensions*, J. Comput. Phys., 205 (2005), pp. 1–23.
[19] E. Michielssen and A. Boag, *A multilevel matrix decomposition algorithm for analyzing scattering from large structures*, IEEE Trans. Antennas Propagat., 44 (1996), pp. 1086–1093.
[20] E. Michielssen, A. Boag, and W. C. Chew, *Scattering from elongated objects: direct solution in $O(N \log^2 N)$ operations*, IEEE Proc. Microwaves Antennas Propagat., 143 (1996), pp. 277–283.
[21] K. Nabors, F. T. Korsmeyer, F. T. Leighton, and J. White, *Preconditioned, adaptive, multipole-accelerated iterative methods for three-dimensional first-kind integral equations of potential theory*, Iterative Methods in Numerical Linear Algebra (Copper Mountain Resort, CO, 1992), SIAM J. Sci. Comput., 15 (1994), pp. 713–735.
[22] V. Rokhlin and M. Tygert, *Fast algorithms for spherical harmonic expansions*, SIAM J. Sci. Comput., 27 (2006), pp. 1903–1928.
[23] D. Slepian and H. O. Pollak, *Prolate spheroidal wave functions, Fourier analysis, and uncertainty*, I, Bell System Tech. J., (1961), pp. 43–63.
[24] R. Suda and S. Kuriyama, *Another preprocessing algorithm for generalized one-dimensional fast multipole method*, J. Comput. Phys., 195 (2004), pp. 790–803.
[25] N. Yarvin and V. Rokhlin, *An improved fast multipole algorithm for potential fields on the line*, SIAM J. Numer. Anal., 36 (1999), pp. 629–666.
[26] L. Ying, G. Biros, and D. Zorin, *A kernel-independent adaptive fast multipole algorithm in two and three dimensions*, J. Comput. Phys., 196 (2004), pp. 591–626.

SIAM J. SCI. COMPUT.
Vol. 29, No. 3, pp. 1179–1206
© 2007 Society for Industrial and Applied Mathematics

NONOSCILLATORY INTERPOLATION METHODS APPLIED TO VLASOV-BASED MODELS*

J. A. CARRILLO† AND F. VECIL†

Abstract. We demonstrate the ability of nonoscillatory interpolation strategies for solving efficiently the transport phase in kinetic systems with applications to charged particle transport in plasmas and semiconductors. Pointwise weighted essentially nonoscillatory (PWENO) interpolation is applied to obtain semi-Lagrangian and flux balance methods that together with splitting techniques form the building blocks of our numerical approach. These methods do not present the restrictive CFL condition typical of finite-difference methods with explicit time-solvers, and, moreover, they provide reliable results controlling parasite oscillations from classical polynomial interpolation while giving highly accurate approximations of smooth parts of the solutions. We perform and compare these methods in different benchmark problems for Vlasov or collisional models for charged particle transport.

Key words. kinetic equations, Vlasov equations, semi-Lagrangian method, flux balance method, Strang's splittings, WENO

AMS subject classifications. 35L60, 35L45

DOI. 10.1137/050644549

1. Introduction.

The Vlasov equation

$$\frac{\partial f}{\partial t} + v \cdot \nabla_x f + \frac{F}{m} \cdot \nabla_v f = 0$$

is the basic kinetic model for the description of the motion of charged particles in electronic devices and plasmas under the effect of a force field $F(t,x)$. Here, $f(t,x,v)$ represents the particle number density in phase space (x,v) at time $t > 0$. In order to compute the force field $F = -\nabla_x \Phi$ it is coupled with the Poisson equation

$$\epsilon_0 \Delta_x \Phi = e\,(\rho[f] - C) = e\left(\int f dv - C\right)$$

for the computation of the electric potential $\Phi(t,x)$ due to both the self-consistent electric potential and an external density $C(x)$, like the doping profile in semiconductors or a background ion density in plasmas. Moreover, when collisional effects have to be taken into account, a Boltzmann operator appears as the right-hand side of the Vlasov equation

$$\frac{\partial f}{\partial t} + v \cdot \nabla_x f + \frac{F}{m} \cdot \nabla_v f = \mathcal{Q}[f].$$

One of the main problems while simulating Vlasov-like systems is the presence of violent gradients in the distribution function, which, if not properly controlled by the numerical method, may drive to a false physical description of the phenomena due to

*Received by the editors November 7, 2005; accepted for publication (in revised form) December 22, 2006; published electronically May 29, 2007. The authors acknowledge support from the Spanish DGI-MEC (FEDER) project MTM2005-08024.
http://www.siam.org/journals/sisc/29-3/64454.html
†ICREA - Departament de Matemàtiques, Universitat Autònoma de Barcelona, 08193 Bellaterra, Spain (carrillo@mat.uab.es, fvecil@mat.uab.es).

a massive formation of spurious oscillations. Several approaches to solving Vlasov-based equations have been proposed in the literature; we refer the reader to [13] for a good review and comparison of different Eulerian solvers. These oscillations appear naturally due to the energy conservation of the system and its Hamiltonian character, the phenomena of the Landau damping being an example of this behavior [30, 13, 14]. When collisions are considered, oscillations are damped, but the control of spurious oscillations continues to be important since high gradients on the moments of the solution (density, mean velocity, and temperature) are typical in the simulations; see [7, 5, 6], for instance, in the semiconductor case.

One of the approaches to attacking the numerical simulation of these models is based on a classical time splitting method, whose theoretical bases were given by Strang [29] and whose application to the Vlasov system goes back to Cheng and Knorr [9]. Time splitting algorithms are used to subdivide a problem into simpler tasks—in our case the transport and the collision steps. In semiconductor devices and plasma simulations, the transport (Vlasov) step in (x, v)- or (x, p)-phase space can be reduced by dimensional splitting to performing advection steps in either dimension. We will elaborate on this later, but let us remark that there are situations in which this dimensional splitting cannot be performed; see [18] for fully relativistic models. Semi-Lagrangian methods were first introduced for meteorology problems [2] and for turbulence study [1]. The application to the Vlasov equation was started by Sonnendrücker et al. in [27]. This method is based on following backward the characteristics, and makes use of some interpolation method: Lagrange polynomials, spline [27], Hermite, or essentially nonoscillatory (ENO) [26]; see [13] for an overview. Flux balance methods were proposed in [14], the third order positive flux conservative method (PFC-3) being a particular case. This method preserves mass and positivity of the solution apart from controlling the total variation of the solution, although it is only third order accurate. This method has been modified in [10] in order to conserve the total energy, too. Several improvements concerning unstructured and adaptive grids and different interpolation techniques have recently been developed in [15, 3].

Another successful approach in the case of collisional semiconductor and plasma models is based on the method of lines with high order nonoscillatory finite-difference reconstructions of the derivatives of the distribution function $f(t, x, v)$ in phase space while performing a total-variation-diminishing (TVD) third order explicit Runge–Kutta method for the time discretization. In this case, spurious oscillations are controlled by ENO [26] or weighted ENO (WENO) [19, 25] reconstructions of the fluxes and thus of the derivatives.

We propose in this paper a semi-Lagrangian (SL) method and a flux balance method (FBM) for Vlasov's systems based on a WENO interpolation for the computation of point values (PWENO); see [24]. We will review this numerical technique for point values reconstruction in section 2.

In section 3, the SL method and the FBM are described. Both methods are based on integrating backward in the characteristics, the second one being mass-conservative and the first one being better as for the control of the total variation. The main advantage with respect to previous semi-Lagrangian methods is the oscillation control while keeping a high order approximation of the solution. Compared to WENO finite-difference methods, our approach avoids the restrictive CFL condition on the time stepping [21] for plasmas and [7, 5, 6] for semiconductors. The proposed numerical method allows the time step Δt to be larger and constant, and the method requires far fewer steps (of an equivalent computational cost) than WENO finite differences. Finally, these numerical techniques could be extended for multidomain or multigrid

computations as done in [24].

In section 4, we show the numerical simulations of five experiments: two pure transport problems and three coupled with collisions. We will elaborate on the advantages and drawbacks of this method in both situations. The exact problems are as follows:

1. A one-dimensional (1D) nonlinear Landau damping [14], where we can observe the filamentation of the phase space during the first phase (strong oscillations in the (x, v)-space) and check whether the method is able to not add noise.
2. A 1D symmetric two stream instability, where we want to observe a vortex appearing after initial time, rotate, and form a typical hole structure.
3. A 1D Vlasov–Boltzmann equation with given confining potential and a linear relaxation operator. Its asymptotic behavior was described in [4, 16].
4. A 1D Vlasov–Fokker–Planck equation with given confining potential. Its asymptotic behavior was described in [11, 17] and it is useful for checking the behavior when variable advection terms appear.
5. A Silicon $n^+ - n - n^+$-structured semiconductor [8, 7], where collisions are modeled through a linear Boltzmann operator: we compare results given by time splitting methods and finite-difference methods with explicit and implicit-explicit time integrators.

The simulation results demonstrate overall the main features of this numerical approach: no CFL condition, good control of numerical oscillations, and high accuracy of the approximations for transport phases.

2. PWENO interpolations. Pointwise weighted essentially nonoscillatory interpolation methods have been developed in order to compute a direct reconstruction which avoids Lagrange spurious oscillations produced where high derivatives appear.

2.1. Description of the method. We have an interval $[x_L, x_R]$ divided into a grid of N points $x_L = x_0 < x_1 < x_2 < \cdots < x_{N-1} = x_R$, and the values of a function f in these points $\{f_i = f(x_i)\}_{i=0,\ldots,N-1}$. We choose a stencil of *ntot* consecutive points from this grid

$$\mathcal{S} = \{x_{first}, \ldots, x_{last}\} = \{x_{first}, \ldots, x_{first+ntot-1}\}$$

and *nlp* substencils

$$\mathcal{S}_r = \{x_{last-lpo+1-r}, \ldots, x_{last-r}\} = \{x_{i-r}, \ldots, x_{last-r}\}$$

for $r = 0, \ldots, nlp - 1$, where $lpo := ntot - nlp + 1$ and $i := last - lpo + 1$. Refer to Figure 2.1 for a summary of all the parameters. Let $p_r(x)$ be the Lagrange polynomials of order *lpo* interpolating f in the stencil $\mathcal{S}_r$:

$$p_r(x) = p_r^{lpo}(x) = \sum_{j=0}^{lpo-1} f_{i-r+j} \prod_{l=0,\, l\neq j}^{lpo-1} \frac{x - x_{i-r+l}}{x_{i-r+j} - x_{i-r+l}} = \sum_{j=0}^{lpo-1} f_{i-r+j} c_{r,j}(x).$$

The reconstructed value at point x will be a convex combination of the reconstructions at point x given by the *nlp* Lagrange polynomials in the substencils $\mathcal{S}_r$:

$$p^W(x) = \sum_{r=0}^{nlp-1} \omega_r(x) p_r(x).$$

The weights $\omega_r(x)$ shall give more or less relevance to the stencils depending on how smooth the $p_r(x)$ are, in a way such that the reconstruction is nonoscillatory.

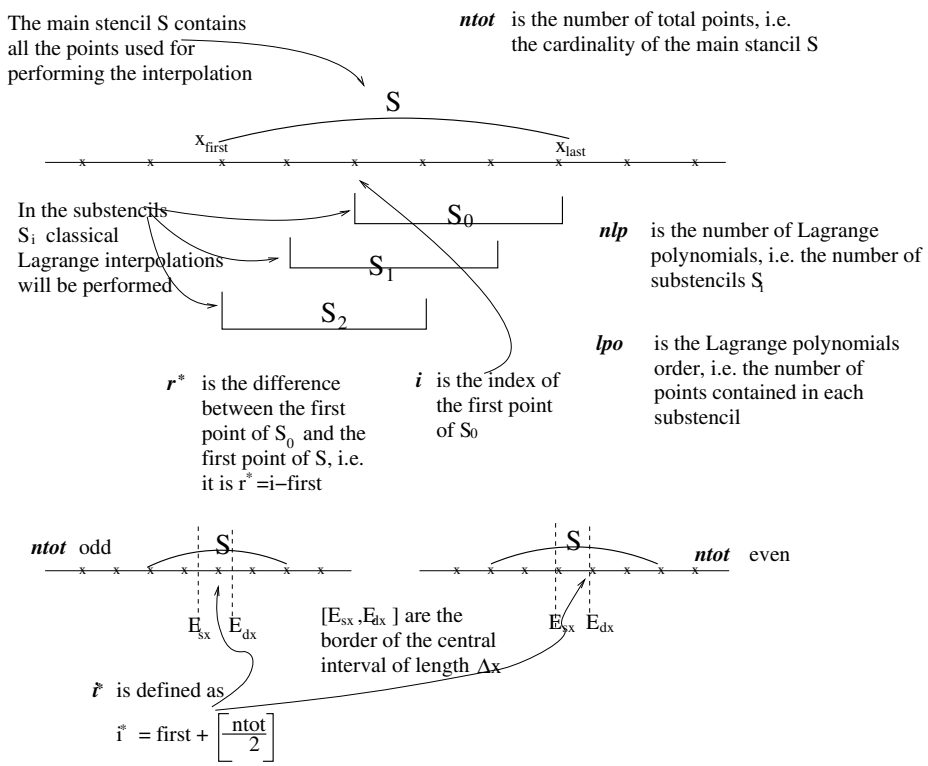


FIG. 2.1. *The notation for all the parameters used in the description of the method.*

2.2. The smoothness indicators β_r. We shall call smoothness of the polynomial $p_r(x)$ a measure of its derivatives in the interval

$$[\mathcal{E}_L, \mathcal{E}_R] = \begin{cases} \left[x_{i^*-\frac{1}{2}}, x_{i^*+\frac{1}{2}}\right] & \text{if } ntot \text{ is odd,} \\ [x_{i^*-1}, x_{i^*}] & \text{if } ntot \text{ is even,} \end{cases}$$

where the dependencies of $\mathcal{E}_L$ and $\mathcal{E}_R$ will be omitted and we have used the notation

$$i^* = first + \left[\frac{ntot}{2}\right],$$

in which $[x]$ means the integer part of x. Figure 2.1 graphically shows the meaning of the last notation. If the derivatives are large, the smoothness indicator should be large, and vice versa. The following measurement is proposed by Jiang and Shu in [19]:

$$\beta_r = \sum_{l=1}^{lpo-1} \Delta x^{2l-1} \left\| \frac{d^l}{dx^l} p_r \right\|^2_{L^2(\mathcal{E}_L, \mathcal{E}_R)}.$$

This a sort of Sobolev norm, where the weights Δx^{2l-1} are needed to make all the terms of the sum independent of Δx, i.e., to make them all be of the same order. Other measurements would be possible, but we shall omit discussing this point.

2.3. The weights $d_r(x)$. In order to achieve a high order reconstruction for regular functions, we define coefficients $d_r(x)$ as

$$p[f](x) = \sum_{r=0}^{nlp-1} d_r(x) p_r(x), \tag{2.1}$$

where $p[f](x)$ is the *ntot* order Lagrange polynomial interpolating the values of f in the whole stencil $\mathcal{S}$

$$p[f](x) = \sum_{j=0}^{ntot-1} f_{first+j} \prod_{l=0,\, l\neq j}^{ntot-1} \frac{x - x_{first+j}}{x_{first+j} - x_{first+l}}.$$

Lagrange interpolation gives an *ntot* order method; by means of these coefficients we want PWENO-*ntot*,*lpo* interpolation to approach an *ntot* order method nonoscillatory for homogeneously regular functions, i.e., whenever all the weights β_r have the same order of magnitude.

PROPOSITION 2.1 (existence and uniqueness of the weights $d_r(x)$). [illegible] $\mathcal{I} = [x_L, x_R] \subset \mathbb{R}$ [illegible] $x_L = x_0 < x_1 < \cdots < x_{N-1} = x_R$ [illegible] $d_r(x)$ [illegible] (2.1) [illegible]. The *ntot* order Lagrange interpolation is exact on $\mathbb{P}^{ntot-1}$, i.e.,

$$f \in \mathbb{P}^{ntot-1} \mapsto p[f](x) = f(x),$$

since $p[f]$ is the Lagrange polynomial which interpolates f in *ntot* points. If $\mathcal{B} = \{b_i\}_{i=0}^{ntot-1}$ is a basis of $\mathbb{P}^{ntot-1}$, $f(x) = \sum_{i=0}^{ntot-1} f^i b_i(x)$ and

$$c_{r,j}(x) = \prod_{l=0,\, l\neq j}^{lpo-1} \frac{x - x_{i-r+l}}{x_{i-r+j} - x_{i-r+l}};$$

$$p\left[f\right](x) = p\left[\sum_i f^i b_i\right](x) = \sum_j \sum_i f^i b_i(x_j) c_{r,j}(x) \tag{2.2}$$

$$= \sum_i f^i \sum_j b_i(x_j) c_{r,j}(x) = \sum_i f^i p\left[b_i\right](x), \tag{2.3}$$

so we need only to impose condition (2.1) on the elements of $\mathcal{B}$. Take as basis

$$\mathcal{B} = \{b_0, b_1, \ldots, b_{ntot-1}\} = \left\{1, \prod_{l=0}^{m} (x - x_{first+l}),\, m \in \{0, \ldots, ntot-2\}\right\}.$$

For $deg \leq lpo - 1$, condition (2.1) gives $b_{deg}(x) = \sum_{r=0}^{nlp-1} d_r(x) b_{deg}(x)$, which means

$$\sum_{r=0}^{nlp-1} d_r(x) = 1.$$

For $lpo \leq deg \leq ntot - 1$, condition (2.1) gives

$$b_{deg}(x) = \sum_{r=0}^{nlp-1} d_r(x) p_{r,deg}(x),$$

where $p_{r,deg}(x)$ is the Lagrange polynomial interpolating polynomial $b_{deg}(x)$ at points $\{x_{i-r},\dots,x_{i-r+lpo-1}\}$, i.e.,

$$p_{r,deg}(x)=\sum_{j=0}^{lpo-1} b_{deg}(x_{i-r+j})c_{r,j}(x).$$

We get the following linear system:

$$L=\left(\begin{array}{cccc|c} 1 & 1 & \dots & 1 & 1\\ p_{0,lpo}(x) & p_{1,lpo}(x) & \dots & p_{nlp-1,lpo}(x) & b_{lpo}(x)\\ p_{0,lpo+1}(x) & p_{1,lpo+1}(x) & \dots & p_{nlp-1,lpo+1}(x) & b_{lpo+1}(x)\\ \vdots & \vdots & \vdots & \vdots & \vdots\\ p_{0,ntot-1}(x) & p_{1,ntot-1}(x) & \dots & p_{nlp-1,ntot-1}(x) & b_{ntot-1}(x) \end{array}\right).$$

Note that $b_{deg}(x_{i-r+j})=0$ for $i-r+j\le deg-1$, which implies that the matrix has the following appearance:

$$L=\left(\begin{array}{ccccc|c} 1 & 1 & 1 & \dots & 1 & 1\\ p_{0,lpo}(x) & p_{1,lpo}(x) & \dots & p_{nlp-2,lpo}(x) & 0 & b_{lpo}(x)\\ p_{0,lpo+1}(x) & p_{1,lpo+1}(x) & \dots & 0 & 0 & b_{lpo+1}(x)\\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ p_{0,ntot-1}(x) & 0 & 0 & \dots & 0 & b_{ntot-1}(x) \end{array}\right). \tag{2.4}$$

Call $L=(L_1|L_2)$. The computation of the determinant of L_1 is straightforward:

$$\det(L_1)=-\prod_{s=0}^{nlp-2} p_{s,(ntot-1)-s}(x)\neq 0,$$

which is not zero because none of the terms can be zero. Since

$$p_{s,(ntot-1)-s}(x)=\sum_{j=0}^{lpo-1} b_{(ntot-1)-s}(x_{i-s+j})c_{s,j}(x)$$

and $b_{(ntot-1)-s}(x_{i-s+j})=0$ for $j=0,\dots,lpo-2$, then

$$p_{s,(ntot-1)-s}(x)=b_{(ntot-1)-s}(x_{i-s+lpo-1})c_{s,lpo-1}(x)\neq 0$$

because $b_{(ntot-1)-s}(x_{i-s+lpo-1})\neq 0$; if not, we should get a contradiction: a nonzero polynomial of degree $(ntot-1)-s$ would have $ntot-1$ zeros. $c_{s,lpo-1}(x)\neq 0$ because x does not belong to the points of the grid. In this way, the existence and uniqueness of the $d_r(x)$ have been proven. □

The proof of this proposition, which is quite straightforward, has been reported in full detail since the general code computing any PWENO interpolation, i.e., with any *ntot* and *lpo*, uses the linear system (2.4) to compute the weights $d_r(x)$.

2.3.1. Uniqueness of the weights $d_r(x)$ as polynomials. We are able to get an explicit formula for the polynomials $d_r(x)$ by an iterative method. Assume the interpolation points are $\{x_0,x_1,\dots,x_n\}$. Suppose we have two polynomials: $p(x)$ interpolates a function f at the points $x_0,x_1,\dots,x_{n-1}$, and $q(x)$ interpolates at the points $x_1,x_2,\dots,x_n$. A simple exactness argument (Aitken–Neville method) allows

us to check that the polynomial interpolating f at the points $x_0, x_1, \ldots, x_n$ is given by

$$r(x) = \frac{p(x)(x - x_n) - q(x)(x - x_0)}{x_0 - x_n}.$$

An explicit recursion procedure based on the previous formula gives us the explicit value of $\tilde{d}_r(x)$. In the case of three points ($n = 2$) we get

$$\begin{cases} \tilde{d}_0(x) = \frac{(x-x_{i-1})(x-x_{i-2})}{(x_{i-1}-x_{i+2})(x_{i-2}-x_{i+2})}, \\ \tilde{d}_1(x) = -\left[\frac{(x-x_{i+2})(x-x_{i-2})}{(x_{i-1}-x_{i+2})(x_{i-2}-x_{i+2})} + \frac{(x-x_{i-2})(x-x_{i+2})}{(x_{i-2}-x_{i+1})(x_{i-2}-x_{i+2})}\right], \\ \tilde{d}_2(x) = \frac{(x-x_{i+1})(x-x_{i+2})}{(x_{i-2}-x_{i+1})(x_{i-2}-x_{i+2})}. \end{cases}$$

PROPOSITION 2.2 (uniqueness of the weights $d_r(x)$). [illegible] $\mathcal{I} = [x_L, x_R] \subset \mathbb{R}$ [illegible] $x_L = x_0 < x_1 < \cdots < x_{N-1} = x_R$ [illegible] $d_r(x)$ [illegible] $\mathbb{P}^{ntot-lpo}$ [illegible]. Polynomials $\tilde{d}_r(x)$ are explicitly constructed by a standard recursive method based on finite differences, so they exist (no denominator can be zero because $i \neq j \Rightarrow x_i \neq x_j$) and their degree is $(ntot - 1) - (lpo - 1) = ntot - lpo$. Moreover,

$$\forall x \in \mathbb{R} \setminus \{x_i\}_{i=0,\ldots,N-1}, \qquad \tilde{d}_r(x) = d_r(x)$$

by construction and Proposition 2.1; then the weights $d_r(x)$ are unique as polynomials in $\mathbb{P}^{ntot-lpo}$. □

2.4. The coefficients $\omega_r(x)$. Once we have computed the smoothness indicators, we define

$$\tilde{\omega}_r(x) = \frac{d_r(x)}{(\epsilon + \beta_r)^2}, \tag{2.5}$$

where ϵ is just a constant which keeps the denominator from being zero (in the code $\epsilon = 10^{-6}$ is used). To get the weights $\omega_r(x)$ we just have to normalize the $\tilde{\omega}_r(x)$ given by (2.5):

$$\omega_r(x) = \frac{\tilde{\omega}_r(x)}{\sum_{j=0}^{nlp-1} \tilde{\omega}_j(x)}.$$

On one hand, if the function is smooth enough in the stencil $\mathcal{S}$ (all the smoothness indicators are of the same order), then the main role in the coefficients $\omega_r(x)$ is played by the weights $d_r(x)$ making the PWENO reconstruction approach a Lagrange reconstruction, which achieves the best interpolation order for smooth functions. For instance, if all the β_r were equal, then PWENO interpolation would coincide with Lagrange interpolation. On the other hand, if the function has some "irregularity" in the stencil $\mathcal{S}$ (the smoothness indicators are not all of the same order), then the main role in the coefficients $\omega_r(x)$ is played by the smoothness indicators β_r giving more or less weight to the substencils $\mathcal{S}_r$ following the regularity of $p_r(x)$.

2.5. The order of the method. First we recall a standard result about the error committed by Lagrange interpolation (see, for instance, [28]).

PROPOSITION 2.3 (order of Lagrange reconstruction). $\mathcal{I} = [x_L, x_R] \subset \mathbb{R}$ $f \in \mathcal{C}^{n+1}[x_L, x_R]$ $p \in \mathbb{P}^n$ f $n+1$ $\{x_0, x_1, \dots, x_n\}$ $[x_L, x_R]$ $\forall x \in [x_L, x_R]$ $\xi_x \in]x_L, x_R[$

$$f(x) - p(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi_x) \prod_{i=0}^{n} (x - x_i).$$

In the case of a regular grid this means that

$$p(x) = f(x) + O(\Delta x^{n+1}).$$

LEMMA 2.4.

$$\beta_r = A(1 + O(\Delta x^n)), \tag{2.6}$$

A r

$$\omega_r(x) - d_r(x) = O(\Delta x^n). \tag{2.7}$$

. Straightforward calculations using Taylor's theorem imply that

$$\frac{1}{(\epsilon + \beta_r)^2} = \frac{1}{(\epsilon + A)^2} + O(\Delta x^n)$$

and

$$\tilde{\omega}_r = \frac{d_r}{(\epsilon + \beta_r)^2} = d_r \left[\frac{1}{(\epsilon + A)^2} + O(\Delta x^n) \right] = \frac{1}{(\epsilon + A)^2} d_r + O(\Delta x^n)$$

with

$$\sum_{j=0}^{nlp-1} \tilde{\omega}_j(x) = \frac{1}{(\epsilon + A)^2} d_r + O(\Delta x^n).$$

Finally, putting together previous estimates, we get

$$\omega_r = \frac{\tilde{\omega}_r}{\sum_{j=0}^{nlp-1} \tilde{\omega}_j(x)} = \frac{\frac{1}{(\epsilon+A)^2} d_r + O(\Delta x^n)}{\frac{1}{(\epsilon+A)^2} + O(\Delta x^n)} = d_r + O(\Delta x^n),$$

giving the desired accuracy result. □

The general error for PWENO interpolation is obtained now.

PROPOSITION 2.5 (order of PWENO reconstruction). $d_r(x)$

$$\sum_{r=0}^{nlp-1} d_r(x) = 1 \tag{2.8}$$

$\beta_r = A(1 + O(\Delta x^n))$ A r
$ntot,lpo$ $(lpo + n)$

. First, by Lemma 2.4 we know that (2.7) holds. Since

$$\begin{aligned} p^W(x) - f(x) &= p^W(x) - p(x) + p(x) - f(x) \\ &= \sum_{r=0}^{nlp-1} \omega_r(x) p_r(x) - \sum_{r=0}^{nlp-1} d_r(x) p_r(x) + \sum_{r=0}^{nlp-1} d_r(x) p_r(x) - f(x) \\ &= \sum_{r=0}^{nlp-1} \omega_r(x) p_r(x) - \sum_{r=0}^{nlp-1} d_r(x) p_r(x) + O(\Delta x^{ntot}), \end{aligned}$$

we already know that the method cannot be more than *ntot* order, and we need to calculate

$$\mathcal{I} = \sum_{r=0}^{nlp-1} \omega_r(x) p_r(x) - \sum_{r=0}^{nlp-1} d_r(x) p_r(x)$$

to check which is the order. By simple manipulations,

$$\begin{aligned} \mathcal{I} &= \sum_{r=0}^{nlp-1} \omega_r(x) p_r(x) - \sum_{r=0}^{nlp-1} d_r(x) p_r(x) = \sum_{r=0}^{nlp-1} \left[\omega_r(x) - d_r(x)\right] p_r(x) \\ &= \sum_{r=0}^{nlp-1} \left[\omega_r(x) - d_r(x)\right] \left[p_r(x) - f(x)\right], \end{aligned}$$

due to (2.8).

$$\mathcal{I} = \sum_{r=0}^{nlp-1} \left[\omega_r(x) - d_r(x)\right] \left[p_r(x) - f(x)\right] = \sum_{r=0}^{nlp-1} \left[\omega_r(x) - d_r(x)\right] O(\Delta x^{lpo})$$

because of Proposition 2.3. We finally deduce

$$\mathcal{I} = \sum_{r=0}^{nlp-1} \left[\omega_r(x) - d_r(x)\right] O(\Delta x^{lpo}) = \sum_{r=0}^{nlp-1} O(\Delta x^n) O(\Delta x^{lpo}),$$

due to (2.7). □

Our problem is now translated to the Taylor expansion of the smoothness indicators β_r. Once we have been able to compute n in (2.6), then the method will be order $lpo + n$. By simple manipulations, the smoothness indicators can be expressed as

$$\begin{aligned} \beta_r &= \sum_{l=1}^{lpo-1} \int_{\mathcal{E}} \Delta x^{2l-1} \left[D^l p_r(x)\right]^2 dx \\ &= \sum_{j,k} f_{i-r+j} f_{i-r+k} K^r_{j,k}, \end{aligned}$$

where K^r is the symmetric matrix

$$K^r_{j,k} = \sum_{l=1}^{lpo-1} \int_{\mathcal{E}} \Delta x^{2l-1} D^l c_{r,j} D^l c_{r,k} dx.$$

We can now expand $f(x)$ around the point x_i:

$$f(x) = \sum_n \frac{1}{n!} f^{(n)}(x_i)(x - x_i)^n$$

so that

$$f_{i-r+j} = \sum_n \frac{1}{n!} f^{(n)}(x_i)(\Delta x(i - r + j - i))^n = \sum_n \frac{1}{n!} f^{(n)}(x_i)(j - r)^n \Delta x^n.$$

Therefore,

$$\beta_r = \sum_{n,m} \Delta x^{n+m} f^{(n)} f^{(m)} \mathcal{D}^r_{n,m},$$

where $\mathcal{D}^r$ is the symmetric matrix

$$\mathcal{D}^r_{n,m} = \frac{1}{n!m!} \sum_{j,k=0}^{lpo-1} K^r_{j,k}(j - r)^n (k - r)^m.$$

Computation of these coefficients leads directly to the general order of PWENO methods; for instance, we have the following:

- Numerical computation of $\mathcal{D}^r$ for method WENO-6,4 shows that

  $$0 \le r \le 2, \qquad \mathcal{D}^r_{1,1} = 1,\ \mathcal{D}^r_{2,2} = \frac{4}{3},$$

  and therefore, if $f' \neq 0$, then

  $$\beta_r = (f'(x_i)\Delta x)^2(1 + O(\Delta x))$$

  (where $A = (f'(x_i)\Delta x)^2$, $n = 1$ in (2.6)), and if $f'(x_i) = 0$, then

  $$\beta_r = \frac{4}{3}(f''(x_i)\Delta x^2)^2(1 + O(\Delta x))$$

  ($A = \frac{4}{3}(f''(x_i)\Delta x^2)^2$, $n = 1$ in (2.6)), i.e., that the method is fifth order.
- Computations for WENO-5,3 imply that if $f'(x_i) \neq 0$, then

  $$\beta_r = (f'(x_i)\Delta x)^2(1 + O(\Delta x^2))$$

  (where $A = (f'(x_i)\Delta x)^2$, $n = 2$ in (2.6)). This implies that WENO-5,3 has order 5.

3. Time splittings and linear advection. For integrating both the Vlasov–Boltzmann coupling

$$\frac{\partial f}{\partial t} + \underbrace{v \cdot \nabla_x f + F \cdot \nabla_v f}_{\text{Vlasov}} = \underbrace{\mathcal{Q}[f]}_{\text{Boltzmann}}$$

and the Vlasov equation

$$\frac{\partial f}{\partial t} + \underbrace{v \cdot \nabla_x f}_{\text{adv. in } x} + \underbrace{F \cdot \nabla_v f}_{\text{adv. in } v} = 0$$

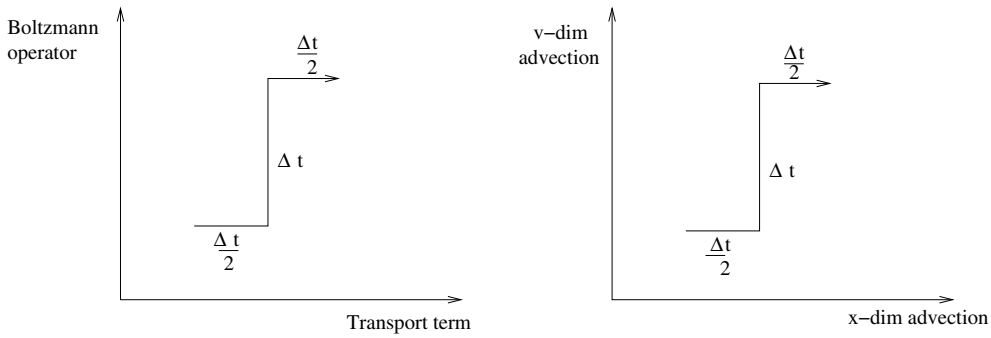


FIG. 3.1. *Time splitting schemes.*

we use Strang [29] and Cheng–Knorr [9] splitting schemes shown in Figure 3.1. In our simulations all reduce to solving the 1D linear advection equation

$$\begin{cases} \dfrac{\partial f}{\partial t} + v\dfrac{\partial f}{\partial x} = 0, \\ f(t=0,x) = f_0(x), \end{cases}$$

whose solution is the translation with velocity v of the initial function

$$f(t,x) = f_0(x - vt)$$

and is obviously mass-conservative.

Two methods are proposed: the semi-Lagrangian (SL) method and the flux balance method (FBM), the second one being mass-conservative and the first one being better as for controlling oscillations.

3.1. Semi-Lagrangian method. This method is based in directly following backward the characteristics of the system; i.e., knowing

$$f_i^n = f(t^n, x_i),$$

we perform the step in time this way:

$$f_i^{n+1} = f(t^{n+1}, x_i) = f(t^n, x_i - v\Delta t) \simeq p^n(x_i - v\Delta t),$$

where p^n is any chosen interpolation from point values of $f(t^n, \cdot)$. Here, we will choose PWENO as the interpolation method. This method is not mass-conservative.

3.2. Flux balance method. The flux balance method (FBM) is used in [14] to construct a conservative method. We already know that

$$f(t + \Delta t, x) = f(t, x - v\Delta t).$$

Now, let us integrate over an interval $[b_1, b_2]$ to get

$$\begin{aligned} \int_{b_1}^{b_2} f(t+\Delta t,\xi)d\xi &= \int_{b_1}^{b_2} f(t,\xi - v\Delta t)d\xi = \int_{b_1 - v\Delta t}^{b_2 - v\Delta t} f(t,\xi)d\xi \\ &= \int_{b_1 - v\Delta t}^{b_1} f(t,\xi)d\xi + \int_{b_1}^{b_2} f(t,\xi)d\xi - \int_{b_2 - v\Delta t}^{b_2} f(t,\xi)d\xi. \end{aligned}$$

If we use as notation

$$\Psi(t,x) = \int_{x-v\Delta t}^{x} f(t,\xi)d\xi,$$

we get

$$\int_{b_1}^{b_2} f(t+\Delta t,\xi)d\xi = \int_{b_1}^{b_2} f(t,\xi)d\xi + \Psi(t,b_1) - \Psi(t,b_2),$$

and dividing by $\Delta = b_2 - b_1$,

$$\frac{\int_{b_1}^{b_2} f(t+\Delta t,\xi)d\xi}{\Delta} = \frac{\int_{b_1}^{b_2} f(t,\xi)d\xi}{\Delta} + \frac{\Psi(t,b_1)-\Psi(t,b_2)}{\Delta},$$

which means

$$\bar{f}_{(b_1,b_2)}(t+\Delta t) = \bar{f}_{(b_1,b_2)}(t) + \frac{\Psi(t,b_1)-\Psi(t,b_2)}{\Delta},$$

i.e., the local description of mass-conservation. By denoting by $F(t,\cdot)$ the primitive of $f(t,\cdot)$, the numerical method we get applying the previous argument on the interval $[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}]$ is

$$f_i^{n+1} = f_i^n + \frac{\Psi^n(x_{i-\frac{1}{2}}) - \Psi^n(x_{i+\frac{1}{2}})}{\Delta x},$$

where we have approximated the mean value of the function over the interval by its value at the center and

$$\Psi^n(x_{i-\frac{1}{2}}) = \int_{x_{i-\frac{1}{2}}-v\Delta t}^{x_{i-\frac{1}{2}}} f(t^n,\xi)d\xi = F(x_{i-\frac{1}{2}}) - F(x_{i-\frac{1}{2}} - v\Delta t).$$

Finally, $F(x_{i-\frac{1}{2}} - v\Delta t)$ will be approximated by a chosen interpolation from the known point values of F at $x_{i+\frac{1}{2}}$. Here, we will choose PWENO as the interpolation method. A method belonging to this family is the positive flux conservative method (PFC-3) introduced in [14].

3.3. Total variation control. The discrete total variation (DTV) is defined as

$$\sum_{i=0}^{N-2} \|f_i^n - f_{i+1}^n\|.$$

The exact solution of the linear advection equation obviously conserves the total variation. If the numerical method is able to respect this property, then it is not adding spurious oscillations due to the interpolation to the shape of the function. Take as initial datum

$$f^{step}(x) = \begin{cases} 0 & \text{if } x < 0, \\ 1 & \text{if } x \geq 0 \end{cases}$$

for the linear advection equation with periodic boundary conditions on $x \in [-1,1]$. As we see in Figure 3.2, Lagrange interpolation has the worst behavior both in the SL method and in the FBM, while PWENO interpolation has a very good conservation of the DTV in the SL method. In the FBM PWENO behaves a little bit worse, but in exchange this second method conserves the mass.

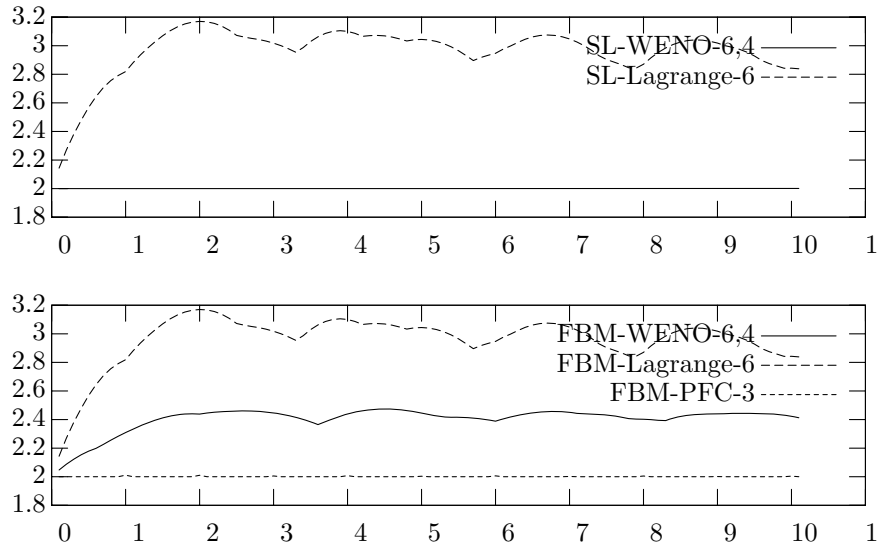


FIG. 3.2. *The evolution of DTV against time for the step as initial function.*

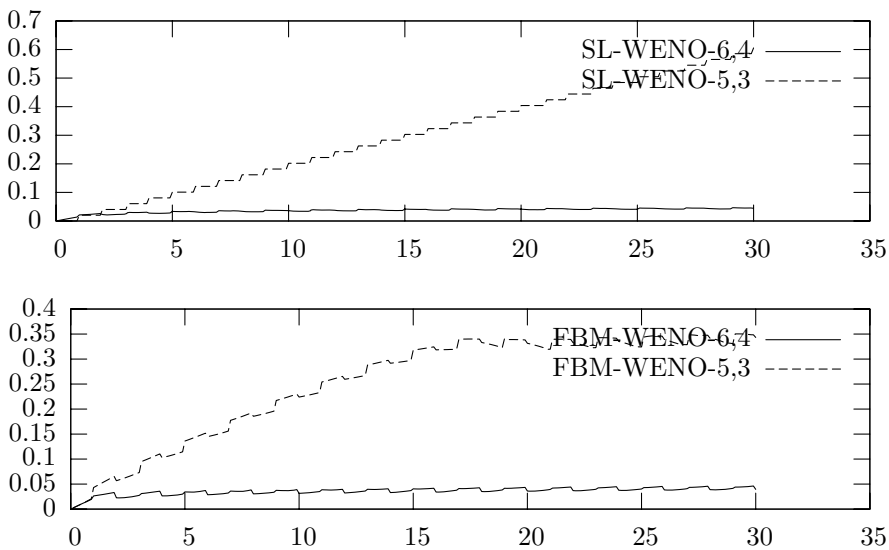


FIG. 3.3. *The evolution of L^1-norm error against the time for $N = 100$, $x \in [-\pi, \pi]$, $\Delta t = 0.1$, $t_{max} = 30$, $f_0(x) = f^{step}(x)$.*

3.4. Disphasement errors. Now we want to focus our attention on how important the choices of the substencils, and thus, of the parameters *ntot* and *lpo* in the PWENO-*ntot*,*lpo* interpolation, are. In Figure 3.3, we see that WENO-6,4 is more accurate than WENO-5,3 both in the semi-Lagrangian method and in the flux balance methods. This difference is due to the fact that in WENO-5,3 not all the substencils "feel" that there is a jump point. Suppose we are interpolating close to x_i, and suppose the jump point is situated between x_i and x_{i+1}. If there is a substencil

which does not contain the irregularity, the WENO method will give weights 0 to all of them but this one. This means that the jump runs with the wrong speed $\frac{\Delta x}{\Delta t}$ with an error after time T with respect to the real speed of $\frac{\Delta x}{\Delta t}\,[\alpha - 1] \times T$ with $\alpha = v\frac{\Delta t}{\Delta x}$. Then, it is essential to use a WENO method such that every substencil contains any discontinuity which may appear, like WENO-6,4. In general, it must be

$$lpo \geq \left[\frac{ntot+1}{2}\right] + 1.$$

3.5. Variable advection equation. Previous methods were written for constant advection equations since most of the time they are the ones appearing in Vlasov-like models. However, both the semi-Lagrangian method and the FBM can be applied for transport equations of the type

$$\begin{cases} \dfrac{\partial f}{\partial t} + \dfrac{\partial}{\partial x}[a(t,x)f] = 0, \\ f(t=0,x) = f_0(x), \end{cases}$$

whose solution can be written in terms of the characteristics given by the solution $r(s;t,x)$ of $r'(s) = a(s, r(s)))$ with $r(t) = x$ as

$$f(t,x) = f_0(r(0;t,x))\frac{\partial r}{\partial x}(0;t,x);$$

see also [13]. The semi-Lagrangian approximation is obtained as

$$f_i^{n+1} = f(t^{n+1}, x_i) = f(t^n, r(t^n; t^{n+1}, x_i))\,\frac{\partial r}{\partial x}(t^n; t^{n+1}, x_i),$$

where $f(t^n, \cdot)$ is approximated by interpolation if the Jacobian of the flow map is explicitly known. The flux balance method can also be applied in this case without any change since the solution to our transport equation satisfies

$$\int_{b_1}^{b_2} f(t+\Delta t, \xi)\, d\xi = \int_{r(t;t+\Delta t,b_1)}^{r(t;t+\Delta t,b_2)} f(t,\xi)\, d\xi$$

for any interval $[b_1, b_2]$ from which the same procedure as in subsection 3.2 applies to obtain the FBM; see also [13].

3.6. Splitting and collisions. Semi-Lagrangian and flux balance methods do not have any CFL limitation as already stressed in the introduction. This is a clear advantage especially when dealing with kinetic equations that possess very large velocities. However, there is an important drawback of using SL methods or FBMs. Due to their "characteristic nature" it is hard to couple them efficiently with a collisional/relaxation right-hand side when this relaxation part is stiff with respect to transport terms. This situation happens in diffusive scalings of kinetic equations, for instance [20]. This is due to the fact that the choice of time integrators is limited to splitting strategies which have a limited accuracy of second order in time away from stiff regions of the right-hand side. The reader should be aware of this potential loss of accuracy induced by the coupling with collisions through splitting.

Finally, let us remark that dimensional splitting for the computation of the transport phases for collisionless Vlasov equations is not always a good choice. Actually, for the Vlasov–Maxwell system for laser-plasma interaction, the instability of a particular

dimensional splitting scheme has been reported in [18]. This system has the feature of having variable advection in both position and momentum and it has to be written in divergence form in phase space leading to a two-dimensional (2D) numerical scheme; we refer the reader to [18] for full details.

4. Numerical simulations. We expose and compare results concerning the simulation of five models:

- A noncollisional coupled Vlasov–Poisson system, by which we study the nonlinear Landau damping.
- A noncollisional coupled Vlasov–Poisson system, by which we study the symmetric two stream instability problem.
- A 1D Vlasov system with a linear Boltzmann operator and a given confining potential.
- A 1D Vlasov system with a linear Fokker–Planck operator and a given confining potential.
- A collisional Vlasov–Poisson system, by which a silicon n^+-n-n^+-structured semiconductor is simulated.

4.1. 1D nonlinear Landau damping. The (normalized) model is

$$\begin{cases} \dfrac{\partial f}{\partial t} + v\dfrac{\partial f}{\partial x} - \dfrac{\partial \Phi}{\partial x}\dfrac{\partial f}{\partial v} = 0, \\ \dfrac{\partial^2 \Phi}{\partial x^2} = 1 - \displaystyle\int_{\mathbb{R}} f\,dv, \\ f_0(x,v) = f_0(x,v) = \dfrac{1}{\sqrt{2\pi}} e^{-\frac{v^2}{2}} \left[1 + 0.5\cos\left(\dfrac{x}{2}\right)\right]. \end{cases}$$

Landau theory has been developed in order to study the propagation of small amplitude waves in a uniform plasma (with no magnetic field and no collisions). It conjectures an interchange of energy between the electric field (potential energy) and resonant particles (kinetic energy) driven by the wave, which produces an oscillating evolution of the electric energy

$$\int_0^L |E(t,x)|^2 dx.$$

Its decay is expected to be exponential in the first phase of the evolution, then it should start oscillating around some equilibrium value.

The numerical method must properly take into account two aspects: the filamentation of the phase space and the conservation properties.

The best results are given by FBM-PWENO-5,3: this method is able to describe the filamentation of the phase space; it controls the strong oscillations (part of the physical phenomenon), which, after some time, should start to disappear. It has a good conservation of the total energy (oscillations about 0.03%, 0.015% for Lagrange and 0.05% for PFC-3, as we see in Figure 4.1).

As for FBM-Lagrange-5 and FBM-PFC-3, both give physically unreliable results, the first one because of the spurious growth of the oscillations, and the second one because of the almost total repression of them, as we can see in Figure 4.2. From Figure 4.3 we can see that the reconstruction given by PFC-3 is, in effect, less definite than the other ones, due to the low order. Lagrange reconstruction is more irregular, due to the parasite oscillations.

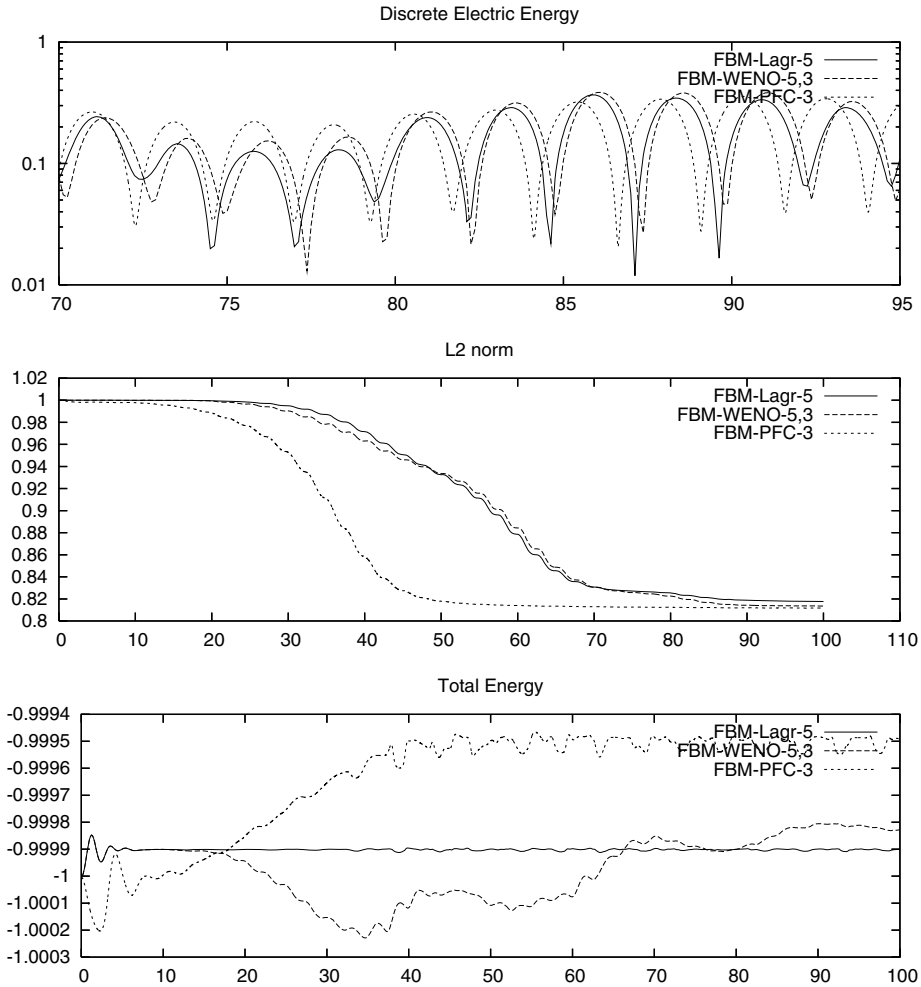


FIG. 4.1. 1D *nonlinear Landau damping. Evolution of the discrete electric energy, the* L^2*-norm, and the total energy. Test performed with* 256×256 *points,* $\Delta t = 0.125$ *for WENO,* $\Delta t = 0.01$ *for PFC-3,* $x \in [0, 4\pi]$, $v \in [-6, 6]$.

4.2. Two stream instability. The (normalized) model is

$$\begin{cases} \dfrac{\partial f}{\partial t} + v\dfrac{\partial f}{\partial x} - \dfrac{\partial \Phi}{\partial x}\dfrac{\partial f}{\partial v} = 0, \\ \dfrac{\partial^2 \Phi}{\partial x^2} = 1 - \displaystyle\int_{\mathbb{R}} f dv, \\ f_0(x,v) = Z\dfrac{2}{7\sqrt{2\pi}}\left(1+5v^2\right)\left[1+\alpha\left(\dfrac{\cos(2kx)+\cos(3kx)}{1.2}+\cos(kx)\right)\right]e^{-\frac{v^2}{2}} \end{cases}$$

with $\alpha = 0.01$, $k = 0.5$, and Z a normalizing factor to impose periodicity boundary conditions on the electric field. This test has been studied in [13]; there, Filbet and Sonnendrücker compare several methods for a coarse points grid and observe which ones have good conservation properties and which ones are able to follow thin details

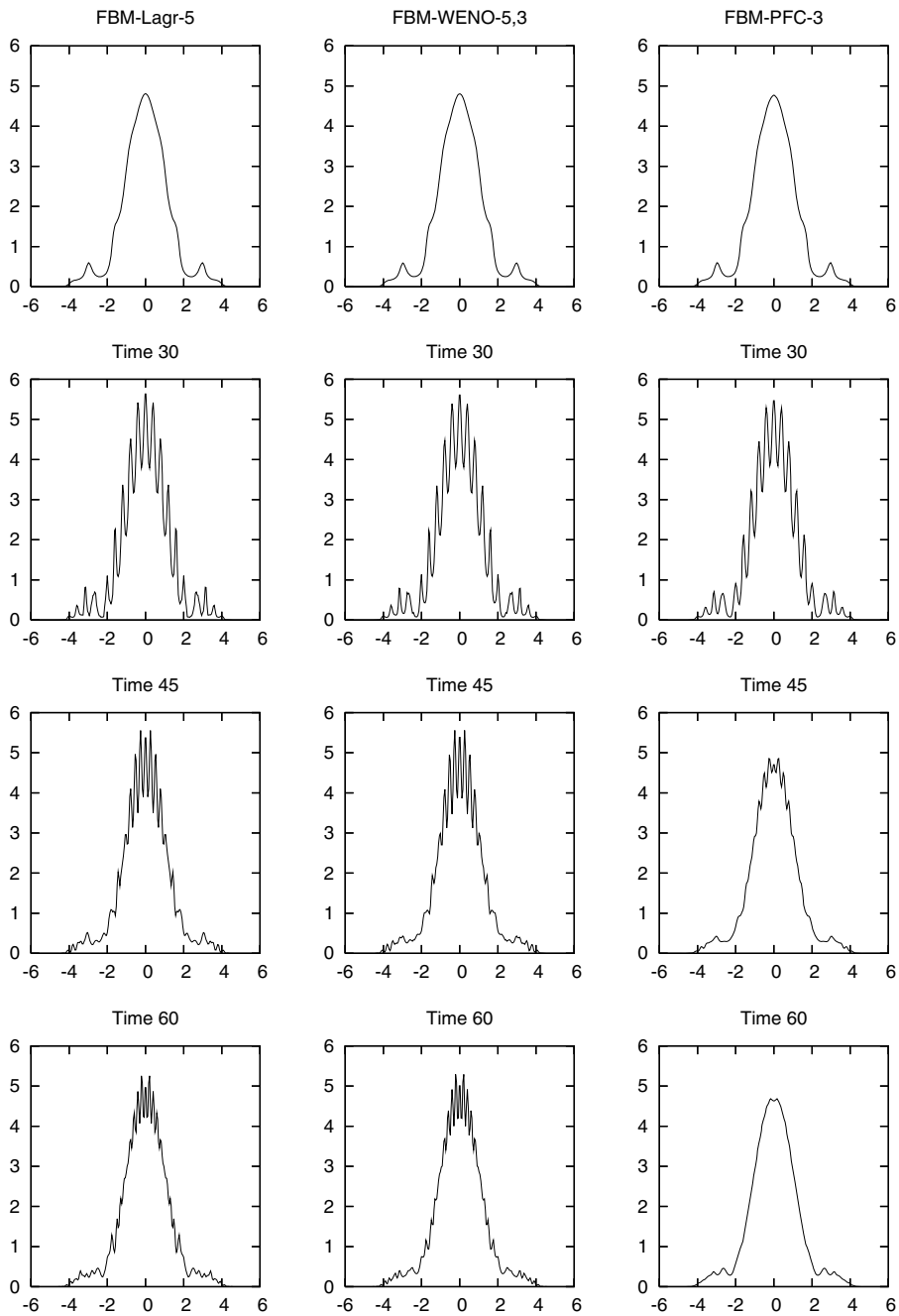


FIG. 4.2. 1D *nonlinear Landau damping. Evolution of the x-integrated distribution function for several methods. Test performed with* 256×256 *points,* $\Delta t = 0.125$ *for WENO,* $\Delta t = 0.01$ *for PFC-*3, $x \in [0, 4\pi]$, $v \in [-6, 6]$.

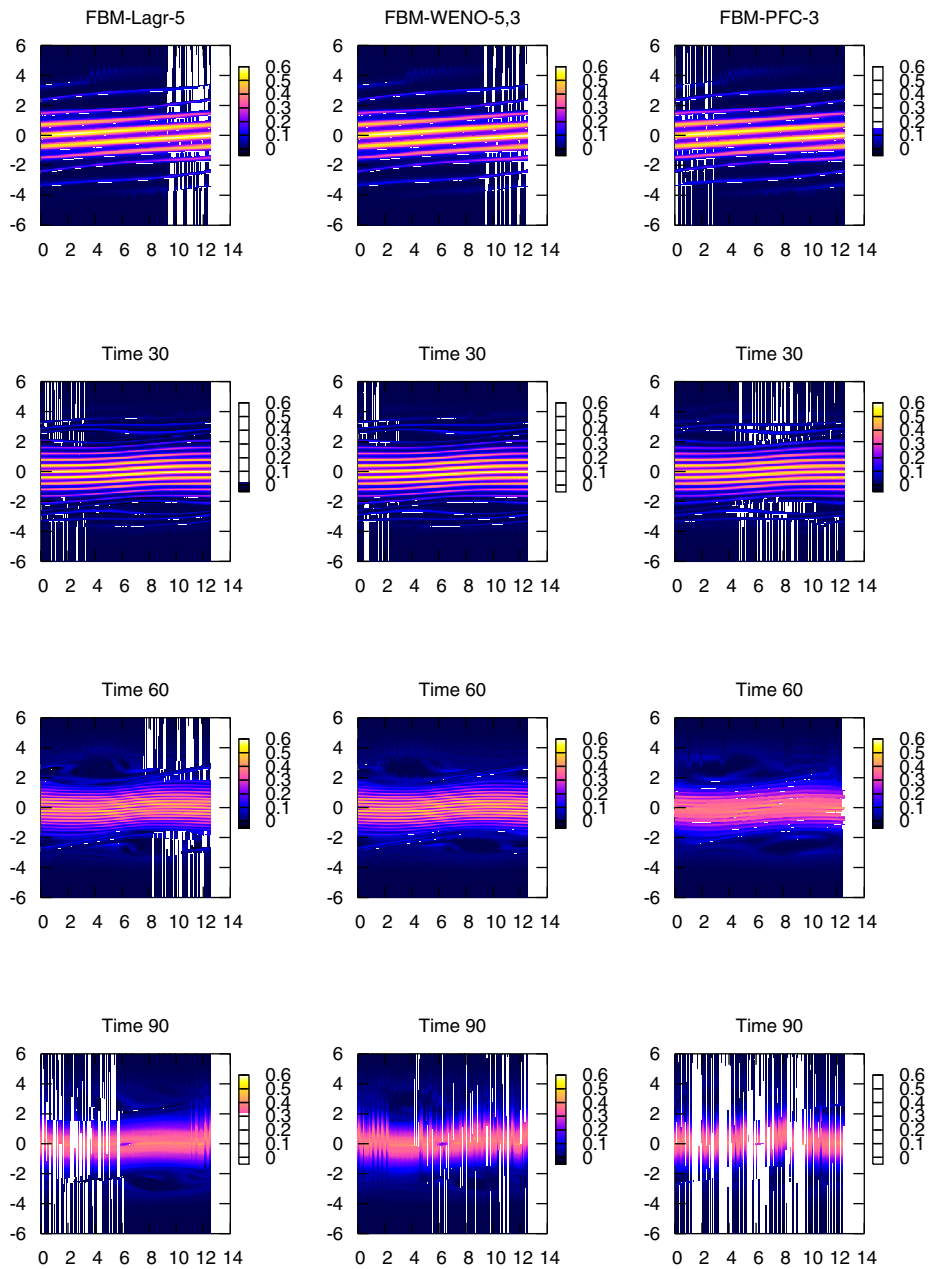


FIG. 4.3. 1D *nonlinear Landau damping. Evolution of the level curves of the distribution function for several methods. Test performed with* 256×256 *points,* $\Delta t = 0.125$ *for WENO,* $\Delta t = 0.01$ *for PFC-3,* $x \in [0, 4\pi]$, $v \in [-6, 6]$.

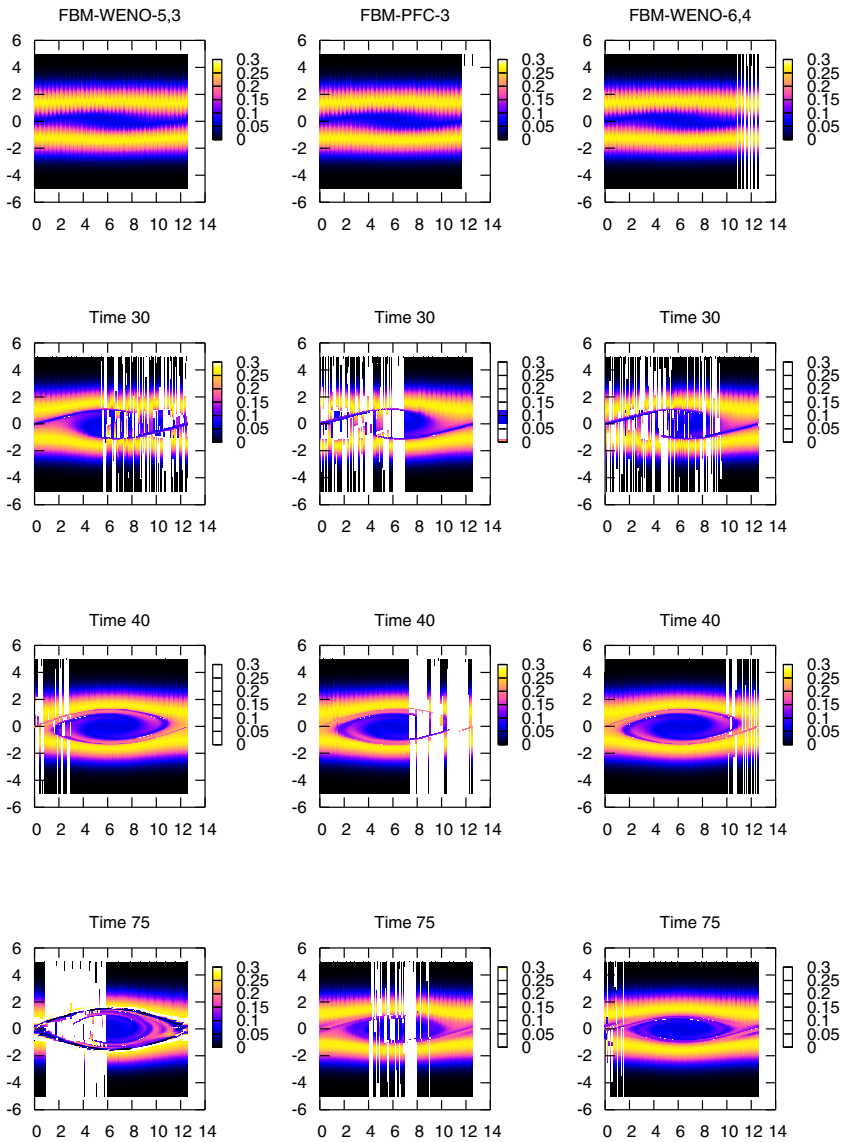


FIG. 4.4. *Two stream instability. Evolution of the level curves. Simulation performed with* 256×256 *points,* $\Delta t = 0.125$ *for WENO methods,* $\Delta t = 0.01$ *for PFC-3,* $x \in [0, 4\pi]$, $v \in [-5, 5]$.

of the distribution function: among their cases, a semi-Lagrangian method with a cubic spline interpolation has given the most reliable results.

Our simulations show that even if both FBM-WENO-5,3 and FBM-WENO-6,4 are fifth order, there may be a substantial difference in their reconstructions, due to the problem the first method has in following shock-like situations, which is the case of this test. Dispersion errors become too important, and in the phase space violent oscillations are produced (look at the level curves in Figure 4.4) which desta-

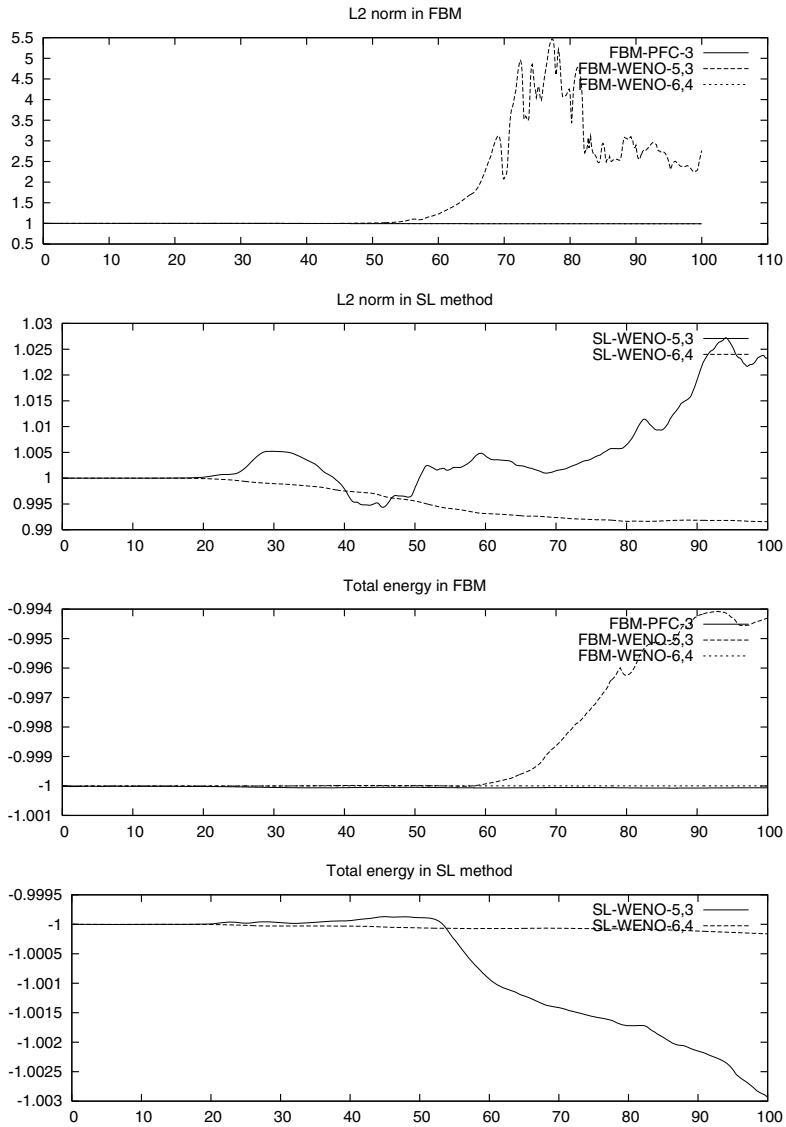


FIG. 4.5. *Two stream instability. Evolution of the L^2-norm and the total energy. Simulation performed with 256×256 points, $\Delta t = 0.125$ for WENO methods, $\Delta t = 0.01$ for PFC-3, $x \in [0, 4\pi]$, $v \in [-5, 5]$.*

bilize the simulation (the variation of the L^2-norm is about 400%, as we can remark from Figure 4.5). As for the PFC-3 method, it has good conservation properties (Figure 4.5), but, compared with FBM-WENO-6,4, observing the level curves in Figure 4.4, it is evident that it is lower order and the reconstruction is less detailed.

Let us finally remark that the semi-Lagrangian schemes behave similarly: SL-WENO-5,3 is unstable, too, even if it behaves better than FBM-WENO-5,3; SL-WENO-6,4 as well as FBM-WENO-6,4 gives a fairly detailed reconstruction almost indistinguishable; and the L^2-norm and the total energy are worse-conserved in the SL method.

4.3. Vlasov–Boltzmann with confining potential. We solve the 1D Vlasov–Boltzmann equation with a confining potential $\Phi_0(x)$ and the simplest linear collision operator:

$$\begin{cases} \dfrac{\partial f}{\partial t} + v\dfrac{\partial f}{\partial x} - \dfrac{\partial \Phi_0}{\partial x}\dfrac{\partial f}{\partial v} = \dfrac{1}{\tau}\left[\rho M_1 - f\right], \\ f(0,x) = f_0(x). \end{cases}$$

From [4] we know that the solution tends to a global equilibrium given by

$$f_s = M\left(\int_{\mathbb{R}} \exp\left(-\Phi_0(x)\right) dx\right)^{-1} \exp\left(-\Phi_0(x)\right) M_1(v)$$

in the L^1-norm, whenever the external potential verifies the confinement conditions

$$\begin{cases} \bullet\ \Phi_0 \geq 0,\ \Phi_0 \in C^\infty(\mathbb{R}), \\ \bullet\ \exp(-\Phi_0(x)) \in L^1(\mathbb{R}), \\ \bullet\ \Phi_0 \text{ is a bounded perturbation of a uniformly convex potential on } \mathbb{R}: \\ \Phi_0 = \Phi_0^{uc} + \Phi_0^{bp} \text{ such that} \\ \qquad \text{there exists } \lambda_1 > 0 \text{ such that } \frac{\partial^2}{\partial^2 x}\Phi_0^{uc}(x) \geq \lambda_1\ \forall x \in \mathbb{R}, \\ \text{and} \\ \qquad \text{there exist } a \text{ and } b \text{ such that } 0 < a \leq \Phi_0^{bp}(x) \leq b < \infty\ \forall x \in \mathbb{R}. \end{cases}$$

The decay rate was proved to be "almost exponential" [4], i.e.,

$$\|f - f_s\|_{L^1}^2 \leq H[f; f_s] \leq C(\epsilon, f_0) t^{-\frac{1}{\epsilon}} \tag{4.1}$$

$\forall \epsilon > 0$. Global and local relative entropies are measures of how far f is from the global equilibrium f_s and the local equilibrium $\rho(t,x)M_1$:

$$\begin{cases} H[f; f_s] = \displaystyle\int_{\mathbb{R}}\int_{\mathbb{R}} \frac{|f - f_s|^2}{f_s}\, dvdx, \\ \tilde{H}[f; \rho M_1] = \displaystyle\int_{\mathbb{R}}\int_{\mathbb{R}} \frac{|f - \rho M_1|^2}{f_s}\, dvdx. \end{cases} \tag{4.2}$$

Global and local relative entropies satisfy the ODE inequalities system [4]

$$\begin{cases} -\dfrac{d}{dt} H[f; f_s] \geq K\tilde{H}[f; \rho M_{\theta_0}], \\ \dfrac{d^2}{dt^2}\tilde{H}[f; \rho M_{\theta_0}] \geq K' H[f; f_s] - C(f, \epsilon)\tilde{H}[f; \rho M_{\theta_0}]. \end{cases} \tag{4.3}$$

We show numerical results in the particular case of $\Phi_0(x) = \frac{x^2}{2}$. In this case, the Vlasov part gives a rotation of the initial function $f_0(x,v)$ while the collision part thermalizes the velocity distribution toward $M_1(v)$. Moreover, an analysis of the spectrum of operator

$$f \longmapsto -v\frac{\partial f}{\partial x} + x\frac{\partial f}{\partial v} + \frac{1}{\tau}\left[\rho M_1 - f\right]$$

has recently been done in [16] showing the existence of a spectral gap in a suitable L^2-weighted space and, thus, of the exponential convergence in that space and as a consequence in L^1 toward f_s.

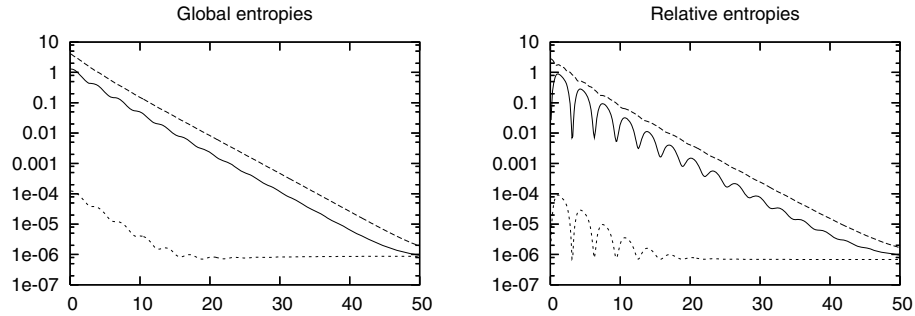


FIG. 4.6. 1D *Vlasov–Boltzmann with confining potential. Tests performed with* 400×400 *points,* $\Delta t = 0.1$*, SL method with PWENO-6,4,* $\tau = 3.5$*. Left figure: H-entropy; right figure:* $\tilde{H}$*-entropy. For the three cases: solid line* ($f_0^{(1)}$)*, dashed line* ($f_0^{(2)}$)*, and small dashed line* ($f_0^{(3)}$)*.*

The system of ODE inequalities (4.3) suggests the appearance of oscillations in the trend of solutions toward global equilibrium. The ODE inequalities (4.3) show that the trend of convergence toward local equilibria is compensated by the transport term that should push the solution out of the local equilibria manifold whenever the solution approaches a local equilibrium which is not the global one. This fact may induce an oscillation both in the local and global relative entropies, and thus, on the trend of convergence toward equilibrium.

In Figure 4.6, we compare the decay of global and relative entropies for three different initial functions, two of them far from equilibrium,

$$f_0^{(1)}(x) = Z_1 \sin^2\left(\frac{x^2}{2}\right) e^{-\frac{x^2+v^2}{2}},$$

$$f_0^{(2)}(x) = Z_2 \sin^2\left(\frac{x^2}{2}\right) \sin^2\left(\frac{v^2}{2}\right) e^{-\frac{x^2+v^2}{2}},$$

and a close-to-equilibrium initial data

$$f_0^{(3)}(x) = Z_3 \left[1 + 0.05 \sin^2\left(\frac{x^2}{2}\right)\right] e^{-\frac{x^2+v^2}{2}}$$

with suitable normalizing factors Z_i, $i = 1, \ldots, 3$.

From these log-plot results, we first observe the exponential L^1-convergence of f toward f_s, as shown in [16], led by the exponential decay of $H[f; f_s]$ due to the first inequality in (4.1). Oscillations of both the global and the local entropies appear. Even if the amplitude of the oscillation is different, stronger for local entropies, it seems evident that the oscillation rate and the decay slope coincide. These oscillations are damped gradually in time; this behavior is observed for the two initial data far from equilibrium, by computing for longer times, while for the close-to-equilibrium initial data this is observed quickly before achieving a steady value due to the accuracy. In [12] oscillations have been reported in the case of the nonlinear Boltzmann equation for rarefied gases in a box with periodic boundary conditions.

4.4. 1D Vlasov–Fokker–Planck with confining potential. We solve the 1D Vlasov equation with a confining potential $\Phi_0(x) = \frac{x^2}{2}$ and a Fokker–Planck operator

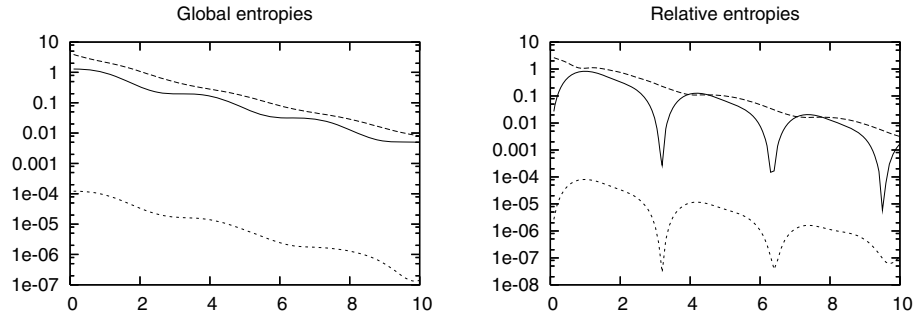


FIG. 4.7. 1D *Vlasov–Fokker–Planck. Test performed with* 400×400 *points,* $\Delta t = 0.01$, *SL method with PWENO-6,4,* $\tau = 3.5$, $\Theta = 1$. *Left figure: H-entropy; right figure:* $\tilde{H}$*-entropy. For the three cases: solid line* ($f_0^{(1)}$), *dashed line* ($f_0^{(2)}$), *and small dashed line* ($f_0^{(3)}$).

as collision operator:

$$\begin{cases} \dfrac{\partial f}{\partial t} + v\dfrac{\partial f}{\partial x} - x\dfrac{\partial f}{\partial v} = \dfrac{1}{\tau}\dfrac{\partial}{\partial v}\left[vf + \Theta\dfrac{\partial f}{\partial v}\right], \\ f(0,x) = f_0(x). \end{cases} \tag{4.4}$$

As for the Vlasov–Boltzmann equation, the solution of the above equation tends exponentially fast to a global equilibrium given by

$$f_s = M\left(\int_{\mathbb{R}} \exp\left(-\frac{x^2}{2}\right) dx\right)^{-1} \exp\left(-\frac{x^2}{2}\right) M_\Theta(v)$$

in the L^1-norm. Actually, an almost exponential rate of convergence was obtained in [11] based on a different system of global and local relative logarithmic entropies. In [17] the exponential convergence was shown by spectral theory. In Figure 4.7 we show how far f is from the global equilibrium f_s and the local equilibrium ρM_Θ by means of the global and local entropies, respectively, as in the previous test.

Equation (4.4) becomes, after an obvious manipulation,

$$\frac{\partial f}{\partial t} + v\frac{\partial f}{\partial x} - \frac{\partial f}{\partial v}\left[-\left(x + \frac{v}{\tau}\right) f\right] = \frac{\Theta}{\tau}\frac{\partial^2 f}{\partial v^2}.$$

In this case the v-advection velocity is not constant: the term $\left[-\left(x + \frac{v}{\tau}\right) f\right]$ depends on v. Therefore, in order to follow backward the characteristics, we need to solve the ODE $\frac{dv}{dt} = -\frac{v}{\tau} - x$. Applying the procedure of subsection 3.5, the v-advection step with a Δt time step is

$$f(t + \Delta t, x, v) = f\left[t, x, v\, e^{\Delta t/\tau} - \tau x\left(e^{\Delta t/\tau} - 1\right)\right] e^{\Delta t/\tau}.$$

As for the collision step,

$$\frac{\partial f}{\partial t} = \frac{\Theta}{\tau}\frac{\partial^2 f}{\partial v^2}$$

is solved through an explicit Euler step, the second derivative being approximated by centered second order finite difference keeping the standard parabolic CFL condition inside the collision step.

4.5. Simple model of transport in semiconductors. The model is

$$\begin{cases} \dfrac{\partial f}{\partial t} + v\dfrac{\partial f}{\partial x} - \dfrac{e}{m}E\dfrac{\partial f}{\partial v} = \dfrac{1}{\tau}\left[M_{\Theta_0}\rho[f] - f\right], \\ \epsilon_{Si}\dfrac{\partial^2 \Phi}{\partial x^2} = -e\left[\rho[f] - C\right], \qquad E = -\dfrac{\partial \Phi}{\partial x}, \\ f_0(x,v) = M_{\Theta_0}(v)C(x), \\ \tau(t,x) = \dfrac{m}{e}\dfrac{2\mu_0}{1+\sqrt{1+4((\mu_0/v_0)E(t,x))^2}}, \end{cases}$$

which describes the electron transport in an $n^+ - n - n^+$ diode where the interaction with the semiconductor crystal is taken into account by an effective relaxation time operator. The lattice temperature is set $T_0 = 300K$ and is related to the thermal energy of the lattice by $\Theta_0 = \frac{k_B}{m}T_0$. The relaxation time depends nonlinearly on the electric field producing a saturation on the drift speed; see [7, 8] for more details. We consider a diode of channel length $L = 0.4\mu m$ and doping profile

$$C(x) = \begin{cases} 5\times 10^5 \frac{1}{\mu m^3}, & 0 \le x \le 0.1, \\ 2\times 10^3 \frac{1}{\mu m^3}, & 0.1 < x < 0.5, \\ 5\times 10^5 \frac{1}{\mu m^3}, & 0.5 \le x \le 0.6. \end{cases}$$

We apply a potential drop at the drain, $\Phi(t,0) = 0$ V, and $\Phi(t,L) = V_{bias}$ V. Concerning boundary conditions at $x = 0$ and $x = L$, we use inflow boundary conditions fixed by Maxwellians with the lattice temperature for the ghost points outside the physical grid needed for computing PWENO interpolation.

In Figure 4.8, density, mean velocity, electric field, potential, energy, and the distribution function in v at a point near the end of the channel at the stationary state are plotted. We are comparing the results of our time splitting method based on FBM-WENO-6,4 with time step $\Delta t = 0.01$, finite-difference WENO5 coupled with an explicit adaptive in time third order Runge–Kutta (RK3) scheme for time discretization [7, 8] and finite-difference WENO5 coupled with an implicit (collisions)-explicit (transport) Runge–Kutta (IMEX-RK) LRR$(3,2,2)$ scheme for time discretization [22, 23] with time step $\Delta t = 0.001$. Results of three simulations are almost indistinguishable; the time step of the explicit RK3 is $\Delta t \simeq 0.0005$ chosen by the CFL due to the explicit discretization of the transport terms.

In Figure 4.9, we compare the results given by the explicit RK3 finite-difference WENO5 method and by FBM-WENO-6,4 for several time steps. One of the main advantages of SL or FBM-WENO schemes is that they do not have an a priori restriction on the time step as the CFL condition for finite-difference WENO5. Therefore, our scheme is much less costly from the computational point of view. Actually, 200 time steps corresponding to $\Delta t = 0.05$ of FBM-WENO-6,4 are enough to give results with less than 4% L^∞-difference in density to 9000 time steps corresponding to $\Delta t \simeq 0.00075$ of the WENO5 method with explicit RK3 time integrator with comparable computational cost for each time step between the methods. On the other hand, WENO5 with IMEX-RK time integrator with $\Delta t \simeq 0.003$, 2250 time steps, gives comparable results, less than 4% L^∞-difference in density, while the cost is higher since at each time step one linear system of the order of the velocity grid per position grid point has to be solved.

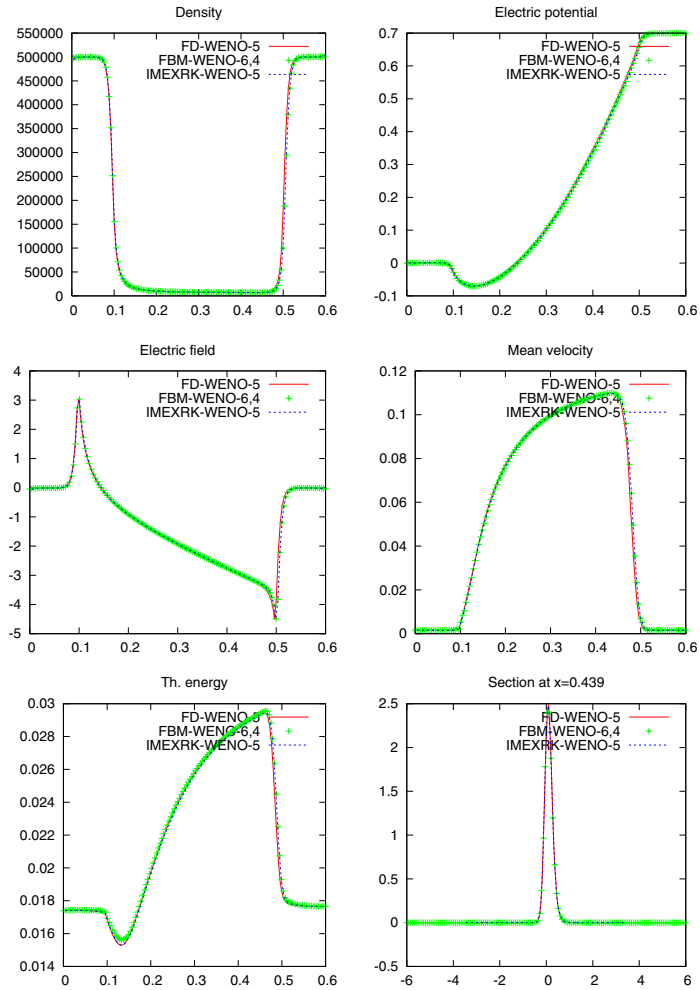


FIG. 4.8. 1D $n^+ - n - n^+$ *Silicon semiconductor. Evolution of several macroscopic magnitudes, for explicit Runge–Kutta-3 WENO-5 finite-difference method (CFL is set to* 0.3*), WENO-6,4 flux balance method with* $\Delta t = 0.01$*, and an IMEX-LRR*$(3, 2, 2)$ *WENO-5 finite-difference method with* $\Delta t = 0.001$*. Grids are set* 150×150 *points.*

Finally, we show in the next table the maximal time step needed to get such 4% L^∞-relative difference between density for different values of the temperature:

Temperature	Maximal Δt for densities
300	0.05
273	0.05
177	0.04
77	0.01

This table together with the behavior of the L^1-relative difference for different temperatures observed on the right in Figure 4.9 shows the problem gets stiffer as the temperature goes down and the mesh has to be taken finer. We summarize the phys-

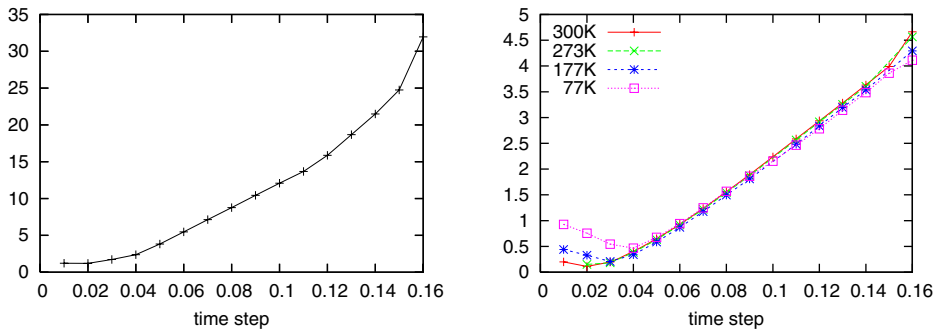


FIG. 4.9. *Left: L^∞-relative difference in % between densities given by the explicit RK*3 *finite-difference WENO*5 *method and by the FBM-WENO*-6,4 *for several time steps. Right: L^1-relative difference in % between densities given by the explicit RK*3 *finite-difference WENO*5 *method and by the FBM-WENO*-6,4 *for several time steps and temperatures. All runs are with a mesh* 150×150.

ical constants used for the simulation:

Constant	Physical meaning	Magnitude
m	effective electron mass	$0.26 \times 0.9109(10^{-30}Kg)$
e	elementary electric charge	$0.1602(10^{-18}C)$
k_b	Boltzmann's constant	$0.138046 \times 10^{-4}(10^{-18}\frac{J}{K})$
ϵ_{Si}	Silicon dielectric permittivity	$11.7 \times 8.85418(10^{-18}\frac{F}{\mu m})$
μ_0	bulk mobility	0.1323
v_0	saturation velocity	0.13.

5. Conclusions. We have proposed a general family of methods based on essentially nonoscillatory interpolations (PWENO) for solving transport phases of Vlasov-like systems. Dimensional splitting has been used to reduce transport phases to 1D advection. The 1D advection is solved by semi-Lagrangian or flux balance methods coupled with the PWENO interpolation. Interpolation methods that feel all candidate stencils perform better in shock-like situations, and due to mass conservation, we take FBM-WENO-6,4 as the preferred method for transport phases. The performances of these methods have been shown in the detailed list of numerical experiments both for collisionless systems and nonstiff collisional systems where coupling has been done through time-splitting. Since the coupling of collisions and transport is reduced to splitting techniques of low order in time due to the characteristic nature of the transport schemes, the reader should be aware of the potential loss of accuracy when coupling this kind of method with stiff collisional terms.

Acknowledgments. We thank Simon Labrunie for fruitful comments and discussions and José M. Mantas for his help with the IMEX-RK method and general purpose codes applied to the last test.

REFERENCES

[1] R. ABGRALL AND C. BASDEVANT, *A semi-Lagrangian numerical scheme for two-dimensional turbulence*, C. R. Acad. Sci. Paris Sér. I Math., 305 (1987), pp. 315–318.

[2] J. R. BATES, *Semi-Lagrangian advective schemes and their use in meteorological modeling*, in Large-Scale Computations in Fluid Mechanics, Part 1 (La Jolla, CA, 1983), Lectures in Appl. Math. 22-1, AMS, Providence, RI, 1985, pp. 1–29.

[3] N. Besse and E. Sonnendrücker, *Semi-Lagrangian schemes for the Vlasov equation on an unstructured mesh of phase space*, J. Comput. Phys., 191 (2003), pp. 341–376.

[4] J. A. Carrillo, M. J. Cáceres, and T. Goudon, *Equilibration rate for the linear inhomogeneous relaxation-time Boltzmann equation for charged particles*, Comm. Partial Differential Equations, 28 (2003), pp. 969–989.

[5] J. A. Carrillo, I. M. Gamba, A. Majorana, and C. W. Shu, *A WENO-solver for the transients of Boltzmann–Poisson system for semiconductor devices. Performance and comparisons with Monte Carlo methods*, J. Comput. Phys., 184 (2003), pp. 498–525.

[6] J. A. Carrillo, I. M. Gamba, A. Majorana, and C. W. Shu, *2D semiconductor device simulations by WENO–Boltzmann schemes: Efficiency, boundary conditions, and comparison to Monte Carlo methods*, J. Comput. Phys., 214 (2006), pp. 55–80.

[7] J. A. Carrillo, I. Gamba, and C. W. Shu, *Computational macroscopic approximations to the one-dimensional relaxation-time kinetic system for semiconductors*, Phys. D, 146 (2000), pp. 289–306.

[8] C. Cercignani, I. Gamba, J. Jerome, and C. W. Shu, *Device benchmark comparisons via kinetic, hydrodynamic, and high-field models*, Comput. Methods Appl. Mech. Engrg., 181 (2000), pp. 381–392.

[9] C. Z. Cheng and G. Knorr, *The integration of the Vlasov equation in configuration space*, J. Comput. Phys., 22 (1976), pp. 330–348.

[10] N. Crouseilles and F. Filbet, *Numerical approximation of collisional plasmas by high order methods*, J. Comput. Phys., 201 (2004), pp. 546–572.

[11] L. Desvillettes and C. Villani, *On the trend to global equilibrium in spatially inhomogeneous entropy-dissipating systems: The linear Fokker–Planck equation*, Comm. Pure Appl. Math., 54 (2001), pp. 1–42.

[12] F. Filbet, C. Mouhot, and L. Pareschi, *Solving the Boltzmann equation in $N \log_2 N$*, SIAM J. Sci. Comput., 28 (2006), pp. 1029–1053.

[13] F. Filbet and E. Sonnendrücker, *Comparison of Eulerian Vlasov solvers*, Comput. Phys. Comm., 150 (2003), pp. 247–266.

[14] F. Filbet, E. Sonnendrücker, and P. Bertrand, *Conservative numerical schemes for the Vlasov equation*, J. Comput. Phys., 172 (2001), pp. 166–187.

[15] M. Gutnic, M. Haefele, I. Paun, and E. Sonnendrücker, *Vlasov simulations on an adaptive phase-space grid*, Comput. Phys. Comm., 164 (2004), pp. 214–219.

[16] F. Hérau, *Hypercoercivity and exponential time decay for the linear inhomogeneous relaxation Boltzmann equation*, Asymptot. Anal., 46 (2006), pp. 349–359.

[17] F. Hérau and F. Nier, *Isotropic hypoellipticity and trend to equilibrium for the Fokker–Planck equation with high degree potential*, Arch. Ration. Mech. Anal., 171 (2004), pp. 151–218.

[18] F. Huot, A. Ghizzo, P. Bertrand, E. Sonnendrücker, and O. Coulaud, *Instability of the time splitting scheme for the one-dimensional and relativistic Vlasov–Maxwell system*, J. Comput. Phys., 185 (2003), pp. 512–531.

[19] G. S. Jiang and C. W. Shu, *Efficient implementation of weighted ENO schemes*, J. Comput. Phys., 126 (1996), pp. 202–228.

[20] S. Jin, L. Pareschi, and G. Toscani, *Uniformly accurate diffusive relaxation schemes for multiscale transport equations*, SIAM J. Numer. Anal., 38 (2000), pp. 913–936.

[21] S. Labrunie, J. A. Carrillo, and P. Bertrand, *Numerical study on hydrodynamic and quasi-neutral approximations for collisionless two-species plasmas*, J. Comput. Phys., 200 (2004), pp. 267–298.

[22] L. Pareschi and G. Russo, *Implicit-explicit Runge-Kutta schemes for stiff systems of differential equations*, in Recent Trends in Numerical Analysis, L. Brugnano and D. Trigiante, eds., Adv. Theory Comput. Math. 3, Nova Science Publishers, Commack, NY, 2000, pp. 269–288.

[23] L. Pareschi and G. Russo, *Implicit-explicit Runge-Kutta methods and applications to hyperbolic systems with relaxation*, J. Sci. Comput., 25 (2005), pp. 129–155.

[24] K. Sebastian and C. W. Shu, *Multidomain WENO finite difference method with interpolation at subdomain interfaces*, J. Sci. Comput., 19 (2003), pp. 405–438.

[25] C.-W. Shu, *Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws*, in Advanced Numerical Approximation of Nonlinear Hyperbolic Equations, Lecture Notes in Math. 1697, Springer-Verlag, Berlin, 1998, pp. 325–432.

[26] C. W. Shu and S. Osher, *Efficient implementation of essentially non oscillatory shock capturing schemes* J. Comput. Phys., 77 (1988), pp. 439–471.

[27] E. Sonnendrücker, J. Roche, P. Bertrand, and A. Ghizzo, *The semi-Lagrangian method for the numerical resolution of the Vlasov equation*, J. Comput. Phys., 149 (1999), pp. 201–220.

[28] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, Springer-Verlag, New York, 1993.
[29] G. Strang, *On the construction and comparison of difference schemes*, SIAM J. Numer. Anal., 5 (1968), pp. 506–517.
[30] T. Zhou, Y. Guo, and C. W. Shu, *Numerical study on Landau damping*, Phys. D, 157 (2001), pp. 322–333.

SIAM J. SCI. COMPUT.
Vol. 29, No. 3, pp. 1207–1216
© 2007 Society for Industrial and Applied Mathematics

NUMERICAL QUADRATURES FOR SINGULAR AND HYPERSINGULAR INTEGRALS IN BOUNDARY ELEMENT METHODS*

MICHAEL CARLEY†

Abstract. A method is developed for the computation of the weights and nodes of a numerical quadrature which integrates functions containing singularities up to order $1/x^2$, without the requirement to know the coefficients of the singularities exactly. The work is motivated by the need to evaluate such integrals on boundary elements in potential problems and is a simplification of a previously published method, but with the advantage of handling singularities at the endpoints of the integral. The numerical performance of the method is demonstrated by application to an integral containing logarithmic, first, and second order singularities, characteristic of the problems encountered in integrating a Green's function in boundary element problems. It is found that the quadrature is accurate to 11–12 decimal places when computed in double precision.

Key words. numerical quadrature, hypersingular integrals, Hilbert transform, integral equations, boundary element methods, Legendre polynomials

AMS subject classifications. 65D32, 74S15

DOI. 10.1137/060666093

1. Introduction.

Boundary integral methods employing hypersingular integrals have become increasingly popular over recent decades with applications in potential problems such as acoustics [6] and fracture mechanics [13]. Hypersingular integrals arise naturally when it is required to compute the field quantities in such problems, for example, the potential and its gradients, and when specialized techniques for the avoidance of "interior resonance" are used, such as that of Burton and Miller [5]. When such integrals arise, they require special numerical treatment and cannot be evaluated using the standard tools of Gaussian quadrature. Instead, approaches tailored to the problem must be used, which can add considerable complexity to the code. This paper introduces a method for the design of numerical quadrature rules which evaluate hypersingular integrals without requiring a detailed analysis of the integrand. The method is based on that of Kolm and Rokhlin [12], who developed a procedure for the design of such rules, but is considerably simplified by the use of Brandão's approach to finite part integrals [4] and is extended to the case of integrals with endpoint singularities—essential if such a scheme is to be used with boundary elements.

To fix ideas, we assume that we are dealing with a two-dimensional potential problem, such as the Laplace or Helmholtz equation:

$$\nabla^2\phi = 0,$$
$$\nabla^2\phi + k^2\phi = 0.$$

In the case of the Helmholtz equation, the singular behavior of the Green's function will be related to the Green's function of the corresponding Laplace equation [8, for example]. The Green's functions for the Laplace equation will have a logarithmic

*Received by the editors July 26, 2006; accepted for publication (in revised form) December 27, 2006; published electronically May 29, 2007.
http://www.siam.org/journals/sisc/29-3/66609.html
†Department of Mechanical Engineering, University of Bath, Bath BA2 7AY, UK (m.j.carley@bath.ac.uk).

singularity in the planar and axisymmetric case (where the Green's function is proportional to an elliptic integral) and also in the case of an asymmetric problem in an axisymmetric domain [3, 6]. In each case, discretization of the boundary of the domain and use of a collocation method give rise to integrals of the form

$$I(x) = \int_{-1}^{1} (\log|x-t| + f(x,t))L(t)\,\mathrm{d}t, \tag{1.1}$$

where t, $-1 \le t \le 1$, is the local coordinate on an element and x is the local coordinate of the collocation point, which may be on the element, or on some adjacent element. The shape function $L(t)$ is typically a polynomial of order up to three.

Obviously, numerical integration of the $\log|x-t|$ term will require that the singular behavior be handled correctly, but since the logarithmic singularity is integrable, this does not present great difficulties; indeed, quadratures for such a problem exist [18]. If, however, it is required to evaluate first, or higher, derivatives of I, the integral is improper and more advanced approaches are required. Such integrals arise in the calculation of acoustic scattering by thin bodies [16], due to the "thin-shape breakdown" of the Helmholtz equation, when the Burton and Miller approach is used to avoid spurious resonances [5] and when gradients of the potential are required as, for example, in computing velocities in potential problems in fluid dynamics. It should be noted that in many problems, such as those in axisymmetric domains, $f(x,t)$ will contain logarithmic terms of the form $(x-t)\log|x-t|$ which are exposed by differentiation so that the integrand will contain singularities of more than one order.

The resulting integrals are of the form

$$\frac{\mathrm{d}I}{\mathrm{d}x} = \int_{-1}^{1} \frac{L(t)}{(x-t)} + L(t)g(x,t)\,\mathrm{d}t \tag{1.2}$$

and

$$\frac{\mathrm{d}^2 I}{\mathrm{d}x^2} = \int_{-1}^{1} \frac{L(t)}{(x-t)^2} + L(t)h(x,t)\,\mathrm{d}t \tag{1.3}$$

and must be interpreted as a Cauchy principal value [14, p. 37] and as a Hadamard finite part, respectively [14, p. 31], denoted by $\fint$ and $\fint\!\!\!\fint$, when $-1 \le x \le 1$. Even when the collocation point x is not on the element but is close to an endpoint, numerical problems arise due to the "offstage singularity" [1], meaning that the integrand is not well approximated by a polynomial.

A number of approaches have been used to handle the strong singularities in the singular and hypersingular integrals. The most obvious is to analyze the integral kernel to extract the singular parts explicitly so that the finite part integral can be evaluated directly [9]. This has the advantage of reliability as the integrand is broken into a singular part or parts and a regular part. The regular part is integrated using Gaussian quadrature and the integrals of the singular part(s) can be interpreted as principal value or finite part integrals using published results [11, for example]. Such an approach is limited, however, in that it requires an analysis of the integrand, i.e., of the kernel of the integral equation, which may be a complicated function not amenable to detailed study.

An alternative approach is to use a quadrature rule which evaluates the integral directly without requiring that it be broken into its singular part(s) and a regular remainder. Such quadratures have been reviewed and developed by Monegato [17]

among others and allow the integration of functions with singularities of order one or two. While this is useful in some applications, it is preferable to be able to integrate functions containing singularities of all orders up to and including two without having to extract them explicitly. An algorithm for the design of such quadrature rules has been presented by Kolm and Rokhklin [12] and allows finite part integrals to be evaluated using only the value of the integrand at each node and without having to analyze the function. This flexibility, however, comes at the price of losing the property of Gaussian quadratures that they integrate exactly all polynomials up to order $2N-1$, where N is the number of points in the quadrature rule. In practice, this is not a serious drawback as accuracy indistinguishable from machine precision is rapidly attained. In the case of offstage singularities, they use the algorithm of Ma, Rokhlin, and Wandzura [15] to derive a generalized Gaussian quadrature which can be used to integrate singular functions exactly. The design of such quadratures requires the use of Newton's method to solve a nonlinear system of equations. The effectiveness and accuracy of the method were demonstrated by application to integrals containing singularities of various orders up to and including two over a range of x, $-2.002 \leq x \leq 2.002$, $x \neq \pm 1$.

The method presented in this paper is similar to that of Kolm and Rokhlin [12] but is simpler to implement and includes the important case of a singularity at an endpoint, a case not considered by Kolm and Rokhlin. The simplification comes about through the use of Brandão's approach to finite part integrals [4]. In the original method, the finite part integrals are computed using special quadratures which must be found as part of the calculation; in this paper, standard Gauss–Legendre quadratures are used, simplifying implementation. It should also be noted that the use of the simpler approach to finite part integrals allows the extension of the method to higher order singularities with no modification to the theory—it is not clear how the special quadratures in the Kolm and Rokhlin approach can be generalized to include singularities of order three and higher or, indeed, singularities of noninteger order.

The extension to include endpoint singularities, easily implemented via the alternative approach to finite part integrals, is important because in boundary element calculations there will always be a need to evaluate integrals at the endpoints of elements and the Kolm and Rokhlin method as published is not applicable to these cases.

2. Quadratures. This section introduces a method for the calculation of the weights of quadrature rules which integrate functions with singularities up to second order, evaluating the integrals in the finite part or principal value sense. The technique is based on that of Kolm and Rokhlin [12], who base their rules on the nodes of an N-point Gauss–Legendre quadrature and compute the weights by solving a system of equations for the integrals of Legendre polynomials weighted by each of the singularities which must be handled (equation (2.7) in this paper). The integrals of the weighted polynomials are found numerically, using specially computed quadratures, whose weights must be found as part of the algorithm. In this paper, the procedure is simplified by using Brandão's approach to finite part integrals [4], outlined in section 2.1, which allows the finite part integrals to be computed using standard quadratures. This also allows the method to handle functions with endpoint singularities; this is important when boundary elements with collocation points at their ends are used.

2.1. Finite part integrals. The most convenient approach to the evaluation of finite part integrals seems to be that of Brandão [4], who gives a simple method for computing a finite part integral by breaking the integrand into a singular part and a

regular part which can be integrated exactly using standard Gaussian quadratures:

$$\text{(2.1a)}\qquad \fint_{-1}^{1} \frac{f(t)}{x-t}\,\mathrm{d}t = \int_{-1}^{1} \frac{f(t)-f(x)}{x-t}\,\mathrm{d}t + f(x)\fint_{-1}^{1} \frac{1}{x-t}\,\mathrm{d}t,$$

$$\fint_{-1}^{1} \frac{f(t)}{(x-t)^2}\,\mathrm{d}t = \int_{-1}^{1} \frac{f(t)-f(x)+f'(x)(x-t)}{(x-t)^2}\,\mathrm{d}t$$

$$\text{(2.1b)}\qquad + f(x)\fint_{-1}^{1} \frac{1}{(x-t)^2}\,\mathrm{d}t - f'(x)\fint_{-1}^{1} \frac{1}{x-t}\,\mathrm{d}t.$$

The finite part integrals are (see also Monegato [17])

$$\text{(2.2a)}\qquad \fint_{-1}^{1} \frac{1}{x-t}\,\mathrm{d}t = \begin{cases} \log|(x+1)/(1-x)|, & |x|\neq 1,\\ \pm\log 2, & x=\pm 1,\end{cases}$$

$$\text{(2.2b)}\qquad \fint_{-1}^{1} \frac{1}{(x-t)^2}\,\mathrm{d}t = \begin{cases} 2/(x^2-1), & |x|\neq 1,\\ -1/2, & x=\pm 1.\end{cases}$$

The formulae given are also valid for x outside the range of integration so that the same approach can also be used for "near-singular" integrals.

2.2. Integrals of Legendre polynomials. A basic tool in the calculation of the quadrature weights in section 2.3 is the evaluation of finite part and principal value integrals of Legendre polynomials, which are the polynomials orthogonal on the range $-1 \le t \le 1$ with a unit weight function. The required integrals are listed below. In most cases, these can be found in standard tables [10] or have been published elsewhere [11]. For $x = \pm 1$, the values are derived in the appendix.

$$\text{(2.3a)}\qquad \int_{-1}^{1} P_n(t)\,\mathrm{d}t = \begin{cases} 2, & n=0,\\ 0, & n\neq 0,\end{cases}$$

$$\text{(2.3b)}\qquad \int_{-1}^{1} P_n(t)\log|x-t|\,\mathrm{d}t = 2(Q_{n+1}(x)-Q_{n-1}(x))/(2n+1), \quad |x|\neq 1,$$

$$\text{(2.3c)}\qquad \int_{-1}^{1} P_{2m}(t)\log|\pm 1-t|\,\mathrm{d}t = \frac{2}{4^m}\sum_{k=0}^{m} \frac{(-1)^k(4m-2k)!}{k!(2m-k)!(2m-2k+1)!} \times \left(\log 2 - \sum_{q=0}^{m-k} \frac{1}{2q+1}\right),$$

$$\text{(2.3d)}\qquad \int_{-1}^{1} P_{2m+1}(t)\log|\pm 1-t|\,\mathrm{d}t = \pm\frac{1}{4^m}\sum_{k=0}^{m} \frac{(-1)^k(4m-2k+2)!}{k!(2m-k+1)!(2m-2k+2)!} \times \sum_{q=0}^{m-k} \frac{1}{2q+1},$$

$$\text{(2.3e)}\qquad \fint_{-1}^{1} \frac{P_n(t)}{x-t}\,\mathrm{d}t = P_n(x)\fint_{-1}^{1} \frac{1}{x-t}\,\mathrm{d}t + \sum_{j=0}^{M-1} w_j \frac{P_n(t_j)-P_n(x)}{x-t_j},$$

$$\text{(2.3f)}\qquad \fint_{-1}^{1} \frac{P_n(t)}{(x-t)^2}\,\mathrm{d}t = P_n(x)\fint_{-1}^{1} \frac{1}{(x-t)^2}\,\mathrm{d}t - P_n'(x)\fint_{-1}^{1} \frac{1}{x-t}\,\mathrm{d}t + \sum_{j=0}^{M-1} w_j \frac{P_n(t_j)-P_n(x)+P_n'(x)(x-t_j)}{x-t_j},$$

where $Q_n(x)$ is the Legendre function of the second kind, and (2.3e) and (2.3f) are Brandão's method for finite part integrals using an M-point Gauss–Legendre quadrature rule (t_j, w_j).

The derivatives of the Legendre polynomials, required in evaluating the finite part integral, (2.3f), can be found in standard tables [10] for $x \neq 1$ and are derived in the appendix for $x = \pm 1$:

$$P_n'(x) = \begin{cases} n(xP_n(x) - P_{n-1}(x))/(x^2-1), & |x| \neq 1, \\ n(n+1)/2, & x = 1, \\ -(-1)^n n(n+1)/2, & x = -1. \end{cases} \tag{2.4}$$

2.3. Quadrature rules. The paper of Kolm and Rokhlin [12] presents an algorithm for the design of quadrature rules for integrals of the form

$$f(x,t) = A(x,t) + B(t)\log|x-t| + \frac{C(t)}{x-t} + \frac{D(t)}{(x-t)^2}. \tag{2.5}$$

The N-point rule approximates the integral of f,

$$⨍_{-1}^{1} f(x,t)\,\mathrm{d}t \approx \sum_{j=0}^{N-1} w_j f(x,t_j), \tag{2.6}$$

with the weights w_i being functions of x. The advantage of such a quadrature is that the functions A, B, C, and D of (2.5) need not be known explicitly; it is assumed only that they are polynomials of order up to $M-1$. Then they can be represented as sums of Legendre polynomials P_n, and the requirement for the quadrature rule is that it integrate exactly $P_i(t)$, $P_i(t)\log|x-t|$, $P_i(t)/(x-t)$, and $P_i(t)/(x-t)^2$ for $i = 0, 1, \ldots, M-1$.

The nodes of the rule are chosen to be the nodes of an N-point Gauss–Legendre quadrature which can be computed using the method of Davis and Rabinowitz [7, p. 369]. Then the weights are the solution of the system of equations

$$[\psi_i(x,t_j)]\,[w_j] = [m_i], \tag{2.7}$$

where

$$\psi_i(x,t) = \begin{cases} P_i(t), & i = 0, \ldots, M-1, \\ P_i(t)\log|x-t|, & i = M, \ldots, 2M-1, \\ P_i(t)/(x-t), & i = 2M, \ldots, 3M-1, \\ P_i(t)/(x-t)^2, & i = 3M, \ldots, 4M-1, \end{cases} \tag{2.8}$$

and

$$m_i(x) = ⨍_{-1}^{1} \psi_i(x,t)\,\mathrm{d}t. \tag{2.9}$$

This is essentially the method of Kolm and Rokhlin [12] simplified by the use of Brandão's approach to finite part integrals. Equations (2.3) allow m_i to be calculated using Gauss–Legendre quadrature instead of special quadratures for different orders of singularity [12, equations 71–73]. Using this approach to finite part integrals also opens the way to handling higher order singularities, so-called "supersingular" integrals, and to incorporating singularities of noninteger order. Brandão also gives formulae for finite part integrals with endpoint singularities, a case not considered by Kolm and Rokhlin, and these formulae make the inclusion of endpoint singularities no more difficult than the handling of internal or external singularities.

2.4. Algorithm. Combining the results of sections 2.1–2.3, an algorithm for the computation of an N-point quadrature rule for functions containing singularities up to and including second order and polynomials up to order $M-1$ is made up of the following steps:

1. Find the nodes of the N-point Gauss–Legendre quadrature, using, for example, the algorithm of Davis and Rabinowitz [7].
2. At each node, compute the functions $\psi_i(x,t_j)$ (see (2.8)).
3. Compute the moments m_i (see (2.9)), using (2.3) and an M-point Gauss–Legendre quadrature.
4. Solve (2.7) for the weights w_j.

The $N\times 4M$ system of equations (2.7) may be well-posed or under- or overdetermined. It is solved in the least-squares sense when overdetermined and in the minimum norm sense when underdetermined, using the appropriate LAPACK solver [2].

3. Numerical performance. As a check on the accuracy of the method, a reference integral

$$I_n(x) = \int_{-1}^{1} t^n \log|x-t|\,\mathrm{d}t, \tag{3.1}$$

$$I_{2m}(x) = \frac{1}{2m+1}\left[(1-x^{n+1})\log|1-x| + ((-1)^n + x^{n+1})\log|1+x|\right] - \frac{2}{2m+1}\sum_{k=0}^{m}\frac{x^{2k}}{2m-2k+1}, \tag{3.2}$$

$$I_{2m+1}(x) = \frac{1}{2m+2}\left[(1-x^{n+1})\log|1-x| + ((-1)^n + x^{n+1})\log|1+x|\right] - \frac{2}{2m+2}\sum_{k=0}^{m}\frac{x^{2k+1}}{2m-2k+1} \tag{3.3}$$

was computed, using quadrature rules of the type derived above. This integral was chosen because it has an integrand of the type which is common in boundary element calculations and because it has an analytical solution which can be compared to the numerical results. The analytical solution can also be differentiated to check the quadrature rule when used on higher order singularities. In each case, the quadrature rule was computed in double precision arithmetic. Kolm and Rokhlin [12] note that constructing the quadratures in double precision gives "11 or 12 correct digits" which is sufficient accuracy for most applications.

Figure 3.1 shows the relative error ϵ for $M=4$ and $n=0,3,5$ as a function of the number of quadrature points N, with

$$\epsilon = \frac{\left(\sum_i |K_n(x_i) - J_n(x_i)|^2\right)^{1/2}}{\left(\sum_i K_n^2(x_i)\right)^{1/2}}, \tag{3.4}$$

where $K_n(x)$ is the exact value of the integral,

$$K_n(x) = I_n(x) + \frac{\mathrm{d}I_n}{\mathrm{d}x} + \frac{\mathrm{d}^2 I_n}{\mathrm{d}x^2}, \tag{3.5}$$

and $J_n(x)$ is the value computed using the quadrature rule. The integrals were evaluated for $x_i = -2 + 0.0125i$, $i=0,1,\ldots,320$, which includes $x_i=\pm 1$. The values of

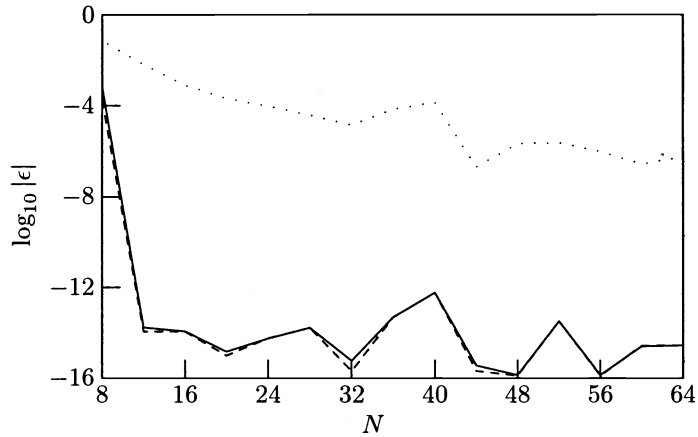


FIG. 3.1. *Relative error* ϵ *for* $M = 4$, $N = 12, 16, \ldots, 64$, *and* $n = 0$, 3, *and* 5 *(solid, dashed, and dotted, respectively).*

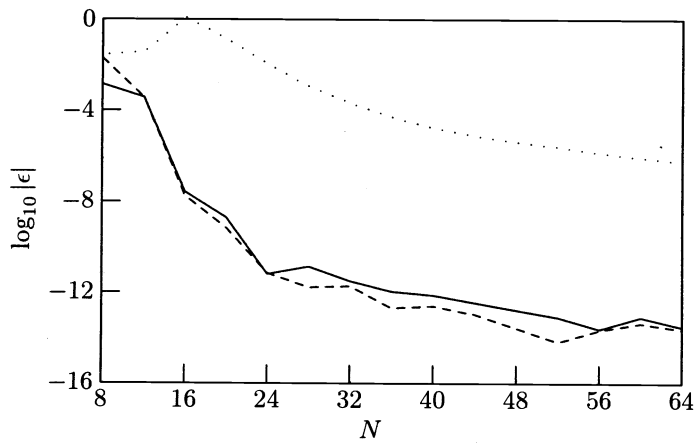


FIG. 3.2. *Relative error* ϵ *for* $M = 8$, $N = 12, 16, \ldots, 64$, *and* $n = 0$, 7, *and* 11 *(solid, dashed, and dotted, respectively).*

n were selected as a check on the performance of the quadrature rule for values of n within the selected range of order, i.e., $n \leq M - 1$, and to examine the performance of the algorithm for n outside that range. As can be seen, for $n = 0, 3$ the relative error is quite large (10^{-4}) for the eight point quadrature but quickly drops to, and remains, less than 10^{-12} for $N \geq 12$. The "out of range" function with $n = 5$ is never accurately integrated, as would be expected from the definition of the quadrature, showing that care would have to be taken in applications where the Green's function contains terms of higher order such as, for example, in the case of the elliptic integral in axisymmetric potential problems. On the other hand, it is useful to know that the full accuracy of the quadrature is attained at about $N = 16$ so that the computational burden of using it is not too great.

Figure 3.2 plots similar data for $M = 8$ and $n = 0, 7, 11$. The behavior is similar to that in the $M = 4$ case, but an error of $\epsilon \leq 10^{-12}$ is not attained until about $N = 24$. Likewise, the error in the $n > M$ case is quite large ($\epsilon \approx 1$) and does not

drop below about 10^{-4}. The most obvious conclusion to be drawn is that the best accuracy of the rule is achieved at about $N = 3M$ and that there is little be gained by increasing N further, unless M is also increased. The results also confirm the conclusion of Kolm and Rokhlin that 11–12 places of accuracy can be achieved when the quadrature weights are computed in double precision.

4. Conclusions. A method has been developed for the derivation of quadrature rules suitable for use in boundary element methods where the integrand has strong singularities up to order two. The technique is a simplification and extension of that of Kolm and Rokhlin [12], allowing for singularities inside and outside the range of integration and at the endpoints, which is an important feature for boundary element methods. The approach has been tested on a reference integral of the type used in boundary element calculations and has been found to be accurate and robust, being accurate to 11–12 places when the quadrature weights are computed in double precision. The approach which has been used is quite general and can be extended to singularities of order higher than two. This will also make the method useful in developing quadratures for three-dimensional boundary element methods.

Appendix. Formulae.

Most of the formulae used in developing the quadrature technique can be found in standard references [10], but some need to be derived, in particular for the behavior of the Legendre functions at $x = \pm 1$.

The derivative of $P_n(x)$ at $x = \pm 1$ can be most readily evaluated from the definition [10, 8.910.2],

$$P_n(x) = \frac{1}{2^n n!} \frac{\mathrm{d}^n}{\mathrm{d}x^n}(x^2 - 1)^n, \tag{A.1}$$

and the derivative is

$$P_n'(x) = \frac{1}{2^n n!} \frac{\mathrm{d}^{n+1}}{\mathrm{d}x^{n+1}}(x^2 - 1)^n. \tag{A.2}$$

Application of Leibnitz's rule [10, 0.42] yields

$$P_n'(x) = \frac{1}{2^n n!} \sum_{q=0}^{n+1} \binom{n+1}{q} \frac{\mathrm{d}^{n+1-q}}{\mathrm{d}x^{n+1-q}}(x+1)^n \frac{\mathrm{d}^q}{\mathrm{d}x^q}(x-1)^n, \tag{A.3}$$

and, since only the term with $q = 1$ survives when $x = -1$,

$$P_n'(-1) = -(-1)^n \frac{n(n+1)}{2}. \tag{A.4}$$

Similarly

$$P_n'(1) = \frac{n(n+1)}{2}. \tag{A.5}$$

The second formula required is the integral

$$\int_{-1}^{1} P_n(t) \log |x - t| \, \mathrm{d}t \tag{A.6}$$

for $x = \pm 1$. Standard tables [10] give

$$\int_{-1}^{1} t^{2m} \log|\pm 1 - t| \, \mathrm{d}t = \frac{2}{2m+1} \left(\log 2 - \sum_{q=0}^{m} \frac{1}{2q+1} \right), \tag{A.7a}$$

$$\int_{-1}^{1} t^{2m+1} \log|\pm 1 - t| \, \mathrm{d}t = \frac{\pm 1}{m+1} \sum_{q=0}^{m} \frac{1}{2q+1}, \tag{A.7b}$$

which, upon insertion into the expansion for the Legendre polynomials [10, 8.911.1], yields

$$\int_{-1}^{1} P_{2m}(t) \log|\pm 1 - t| \, \mathrm{d}t = \frac{2}{4^m} \sum_{k=0}^{m} \frac{(-1)^k (4m-2k)!}{k!(2m-k)!(2m-2k+1)!} \times \left(\log 2 - \sum_{q=0}^{m-k} \frac{1}{2q+1} \right), \tag{A.8a}$$

$$\int_{-1}^{1} P_{2m+1}(t) \log|\pm 1 - t| \, \mathrm{d}t = \frac{\pm 1}{4^m} \sum_{k=0}^{m} \frac{(-1)^k (4m-2k+2)!}{k!(2m-k+1)!(2m-2k+2)!} \times \sum_{q=0}^{m-k} \frac{1}{2q+1}. \tag{A.8b}$$

REFERENCES

[1] F. S. Acton, *Numerical Methods that Work*, Mathematical Association of America, Washington, DC, 1990.

[2] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, 3rd ed., Software Environ. Tools 9, SIAM, Philadelphia, 1999.

[3] J. Björkberg and G. Kristensson, *Electromagnetic scattering by a perfectly conducting elliptic disk*, Canad. J. Phys., 65 (1987), pp. 723–734.

[4] M. P. Brandão, *Improper integrals in theoretical aerodynamics: The problem revisited*, AIAA J., 25 (1987), pp. 1258–1260.

[5] A. J. Burton and G. F. Miller, *The application of integral equation methods to the numerical solution of some exterior boundary-value problems*, Proc. Roy. Soc. London Ser. A., 323 (1971), pp. 201–210.

[6] M. Carley, *Scattering by quasi-symmetric pipes*, J. Acoustical Society of America, 119 (2006), pp. 817–823.

[7] P. J. Davis and P. Rabinowitz, *Methods of Numerical Integration*, Academic Press, New York, 1975.

[8] T. W. Dawson, *On the singularity of the axially symmetric Helmholtz Green's function, with application to BEM*, Appl. Math. Model., 19 (1995), pp. 590–599.

[9] M. H. Dunn, J. Tweed, and F. Farassat, *The application of a boundary integral equation method to the prediction of ducted fan engine noise*, J. Sound Vibration, 227 (1999), pp. 1019–1048.

[10] I. Gradshteyn and I. M. Ryzhik, *Table of Integrals, Series, and Products*, 5th ed., Academic Press, London, 1980.

[11] A. C. Kaya and F. Erdogan, *On the solution of integral equations with strongly singular kernels*, Quart. Appl. Math., 45 (1987), pp. 105–122.

[12] P. Kolm and V. Rokhlin, *Numerical quadratures for singular and hypersingular integrals*, Comput. Math. Appl., 41 (2001), pp. 327–352.

[13] A. M. Korsunsky, *On the use of interpolative quadratures for hypersingular integrals in fracture mechanics*, Proc. Roy. Soc. London Ser. A., 458 (2002), pp. 2721–2733.

[14] M. J. Lighthill, *An Introduction to Fourier Analysis and Generalized Functions*, Cambridge University Press, Cambridge, UK, 1958.

[15] J. Ma, V. Rokhlin, and S. Wandzura, *Generalized Gaussian quadrature rules for systems of arbitrary functions*, SIAM J. Numer. Anal., 33 (1996), pp. 971–996.
[16] R. Martinez, *The thin-shape breakdown (TSB) of the Helmholtz integral equation*, J. Acoustical Society of America, 90 (1991), pp. 2728–2738.
[17] G. Monegato, *Numerical evaluation of hypersingular integrals*, J. Comput. Appl. Math., 50 (1994), pp. 9–31.
[18] R. N. L. Smith, *Direct Gauss quadrature formulae for logarithmic singularities on isoparametric elements*, Engineering Analysis with Boundary Elements, 24 (2000), pp. 161–167.

SIAM J. SCI. COMPUT.
Vol. 29, No. 3, pp. 1217–1243
© 2007 Society for Industrial and Applied Mathematics

A HYBRID HIGH-ORDER ALGORITHM FOR RADAR CROSS SECTION COMPUTATIONS*

M. GANESH† AND S. C. HAWKINS‡

Abstract. We describe a high-order method for computing the monostatic and bistatic radar cross section (RCS) of a class of three-dimensional targets. Our method is based on an electric field surface integral equation reformulation of the Maxwell equations. The hybrid nature of the scheme is due to approximations based on a combination of tangential and nontangential basis functions on a parametric reference spherical surface. A principal feature of the high-order algorithm is that it requires solutions of linear systems with substantially fewer unknowns than existing methods. We demonstrate that very accurate RCS values for medium (electromagnetic-) sized scatterers can be computed using a few tens of thousands of unknowns. Thus linear systems arising in the high-order method for low to medium frequency scattering can be solved using direct solves. This is extremely advantageous in monostatic RCS computations, for which transmitters and receivers are co-located and hence the discretized electromagnetic linear system must be solved for hundreds of right-hand sides corresponding to receiver locations. We demonstrate the high-order convergence of our method for several three-dimensional targets. We prove the high-order spectral accuracy of our approximations to the RCS for a class of perfect conductors described globally in spherical coordinates.

Key words. radar cross section, monostatic, bistatic, electromagnetic scattering, Maxwell's equations, surface integral, tangential basis

AMS subject classifications. 65R20, 65N35

DOI. 10.1137/060664859

1. Introduction.

Computer simulations of the radar cross section (RCS) of three-dimensional model targets through electromagnetic scattering are a cost effective tool for designing stealth objects. The RCS values (measured using simulated transmitters and receivers) reflect the stealth property of an obstacle [20]. Hence a prerequisite for manufacturing vehicles such as stealth aircraft is to simulate the monostatic and bistatic RCS of various geometries [30] at several frequencies to high-order accuracy. Such simulations are also useful in biological applications. For example, it is common to use light scattering measurements in laboratories to count erythrocytes (red bloods cells). Of recent interest [19, 31] is the simulation of the RCS of a celebrated model of an erythrocyte, which is biconcave, for an incident plane wave whose wavelength is about one tenth the diameter of the erythrocyte. Computational electromagnetic scattering algorithms with high-order accuracy facilitate efficient simulations of the RCS values.

In this work we develop, implement, and analyze a high-order algorithm to compute the monostatic and bistatic RCS of a perfectly conducting obstacle $D \subseteq \mathbb{R}^3$, with surface ∂D, situated in a homogeneous medium with vanishing conductivity, the free space permittivity $\epsilon_0 = 10^7/(4\pi c^2)\mathrm{F/m}$, and permeability $\mu_0 = 4\pi \times 10^{-7}\mathrm{H/m}$,

*Received by the editors July 12, 2006; accepted for publication (in revised form) January 5, 2007; published electronically May 29, 2007. Support of the Australian Research Council is gratefully acknowledged.

http://www.siam.org/journals/sisc/29-3/66485.html

†Department of Mathematical and Computer Sciences, Colorado School of Mines, Golden, CO 80401 (mganesh@mines.edu).

‡School of Mathematics and Statistics, University of New South Wales, Sydney, NSW 2052, Australia (stuart@maths.unsw.edu.au).

where $c = 299\ 792\ 458$m/s is the speed of light. Our simulation geometries include benchmark radar targets [30] and erythrocytes [19, 31].

The RCS of the obstacle, induced by a transmitter located in the direction $-\widehat{\boldsymbol{d}}_0$ and measured in decibels by a receiver located in the direction

$$\widehat{\boldsymbol{x}} = \boldsymbol{p}(\theta,\phi) = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)^T, \qquad \theta,\phi \in \mathbb{R}, \tag{1.1}$$

is a functional on the unit sphere (denoted by ∂B) and is defined as

$$\sigma^{\mathrm{dB}}(\widehat{\boldsymbol{x}};\widehat{\boldsymbol{d}}_0) = 10\log_{10}\left(4\pi\left|\boldsymbol{E}_\infty(\widehat{\boldsymbol{x}})\right|^2/k^2\right), \qquad \widehat{\boldsymbol{x}},\widehat{\boldsymbol{d}}_0 \in \partial B, \tag{1.2}$$

where

$$\boldsymbol{E}_\infty(\widehat{\boldsymbol{x}}) = \lim_{|\boldsymbol{x}|\to\infty} \boldsymbol{E}^{\mathrm{s}}(\boldsymbol{x}) e^{-ik|\boldsymbol{x}|}|\boldsymbol{x}|, \qquad \widehat{\boldsymbol{x}} = \boldsymbol{x}/|\boldsymbol{x}| \in \partial B, \tag{1.3}$$

is the far field pattern of the scattered field, which has frequency ω and wavenumber $k = 2\pi\omega/c$. The scattered electric field $\boldsymbol{E}^{\mathrm{s}}$ and magnetic field $\boldsymbol{H}^{\mathrm{s}}$ satisfy the time harmonic Maxwell equations

$$\mathbf{curl}\,\boldsymbol{E}(\boldsymbol{x}) - ik\boldsymbol{H}(\boldsymbol{x}) = \boldsymbol{0}, \qquad \mathbf{curl}\,\boldsymbol{H}(\boldsymbol{x}) + ik\boldsymbol{E}(\boldsymbol{x}) = \boldsymbol{0}, \quad \boldsymbol{x} \in \mathbb{R}^3 \setminus \overline{D}, \tag{1.4}$$

and the Silver–Müller radiation condition

$$\lim_{|\boldsymbol{x}|\to\infty} [\boldsymbol{H}^{\mathrm{s}}(\boldsymbol{x}) \times \boldsymbol{x} - |\boldsymbol{x}|\boldsymbol{E}^{\mathrm{s}}(\boldsymbol{x})] = \boldsymbol{0},$$

subject to the perfect conductor boundary condition

$$\boldsymbol{n}(\boldsymbol{x}) \times \boldsymbol{E}^{\mathrm{s}}(\boldsymbol{x}) = -\boldsymbol{n}(\boldsymbol{x}) \times \boldsymbol{E}^{\mathrm{i}}(\boldsymbol{x}) =: \boldsymbol{f}(\boldsymbol{x}), \quad \boldsymbol{x} \in \partial D, \tag{1.5}$$

where $\boldsymbol{n}(\boldsymbol{x})$ denotes the unit outward normal at $\boldsymbol{x} \in \partial D$. The boundary condition in (1.5) (and hence the RCS in (1.2)) is induced by the incident plane wave $\left[\boldsymbol{E}^{\mathrm{i}}, \boldsymbol{H}^{\mathrm{i}}\right]$, with direction $\widehat{\boldsymbol{d}}_0$ and polarization $\widehat{\boldsymbol{p}}_0$ (perpendicular to $\widehat{\boldsymbol{d}}_0$), which originates from the transmitter and is defined as

$$\boldsymbol{E}^{\mathrm{i}}(\widehat{\boldsymbol{x}}) = ik\widehat{\boldsymbol{p}}_0 e^{ik\boldsymbol{x}\cdot\widehat{\boldsymbol{d}}_0}, \quad \boldsymbol{H}^{\mathrm{i}}(\widehat{\boldsymbol{x}}) = ik(\widehat{\boldsymbol{d}}_0 \times \widehat{\boldsymbol{p}}_0) e^{ik\boldsymbol{x}\cdot\widehat{\boldsymbol{d}}_0}. \tag{1.6}$$

When the transmitter and receiver are co-located, $\widehat{\boldsymbol{d}}_0 = -\widehat{\boldsymbol{x}}$, and σ^{dB} in (1.2) is called the monostatic RCS. When the transmitter and receiver are not co-located and $\widehat{\boldsymbol{d}}_0$ is fixed, σ^{dB} in (1.2) is called the bistatic RCS. The RCS with receiver polarization $\widehat{\boldsymbol{p}}_1$ is defined by

$$\sigma^{\mathrm{dB}}(\widehat{\boldsymbol{x}},\widehat{\boldsymbol{p}}_1;\widehat{\boldsymbol{d}}_0) = 10\log_{10}\left(4\pi\left|\boldsymbol{E}_\infty(\widehat{\boldsymbol{x}})\cdot\widehat{\boldsymbol{p}}_1\right|^2/k^2\right), \qquad \widehat{\boldsymbol{x}},\widehat{\boldsymbol{p}}_1,\widehat{\boldsymbol{d}}_0 \in \partial B. \tag{1.7}$$

The industrial standard tool for simulation of electromagnetic scattering from a perfect conductor is the Fast Illinois Solver Code (FISC); see [25, 6] and references therein for related boundary integral algorithms. FISC is based on a surface integral reformulation of the Maxwell equations and is solved using the method of moments. For medium frequency scattering problems, implementation of FISC requires solution of a dense complex linear system with hundreds of thousands of unknowns to achieve a couple of digits of accuracy. (For example, in a FISC simulation exterior to a sphere of diameter 12 times the incident wavelength, 602 112 unknowns are required

to compute the RCS with just one decimal place of accuracy [25].) Solution of large dense linear systems in FISC calculations is made possible only by using an iterative solver with matrix vector products implemented using the fast multipole method (FMM), and recently by using modified FMM algorithms based on efficient spherical harmonic expansions. A high-order algorithm for acoustic scattering was given in [4] and applied to electromagnetic scattering in [3, p. 56]. We refer the reader to [2, 3, 6, 9, 10, 19, 21, 23, 31] and references therein for the current state of the art in computational electromagnetics. The existing three-dimensional electromagnetic scattering algorithms with demonstrated numerical experiments do not achieve spectral accuracy. Spectral boundary integral methods for three-dimensional model problems in acoustic, elasticity, and potential theory were considered in [12, 13, 14, 17, 29].

In [15] we considered an electromagnetic counterpart of [12] using vectorial basis functions that are tangential on the reference surface. Such vector spherical harmonics are widely used in many applications involving vector valued quantities on the sphere [11, 26, 28]. However, the approach in [15] does not yield spectrally accurate approximations [16, Remark 2]. In [16] we used a componentwise basis that is not tangential, for which we proved and demonstrated spectral accuracy. However, tangential basis functions are a natural choice for the surface integral formulation of the Maxwell equations. Further, for spherical scatterers, approximation of surface currents using tangential spherical harmonics can give almost a one third reduction in the number of unknowns required in [16]. This work is motivated by the desire to obtain this reduction by using tangential harmonics and yet obtain a spectrally accurate algorithm. We achieve the reduction and spectral accuracy in this paper with hybrid approximations, combining ideas from [15] and [16] and involving a combination of tangential and nontangential harmonics.

A clearly described parametrization of the two-dimensional manifold ∂D (based on a reference domain or surface) is essential for computer implementation of any surface integral equation algorithm. We assume throughout the paper that each coordinate (x, y, z) on the bounded target surface ∂D can be parametrized as

$$(1.8) \qquad (x, y, z)^T = (q_1(\theta, \phi), q_2(\theta, \phi), q_3(\theta, \phi))^T, \qquad (x, y, z)^T \in \partial D, \quad \theta, \phi \in \mathbb{R},$$

for some nonlinear functionals $q_j : \mathbb{R}^2 \to \mathbb{R}$ for $j = 1, 2, 3$ that yield, via $\boldsymbol{p}$ in (1.1), a bijective parametrization map $\boldsymbol{q} : \partial B \to \partial D$. In fact, it is sufficient to know a suitable approximation to the parametrization map $\boldsymbol{q}$, for example, based on the Fourier coefficients of q_j for $j = 1, 2, 3$. Our algorithm has spectral accuracy only when $\boldsymbol{q}$ is smooth.

Such a specific description of the target is required in our mathematical analysis to prove the spectral accuracy of the method. Many benchmark radar targets, such as those described in [30], satisfy the specific description (1.8) after converting the material construction (cylindrical) coordinates [30] to spherical coordinates [12], and biologically interesting models such as the erythrocyte [19, 31] can be written in the parametric form (1.8). Simulation geometries considered in this work and those in [12, 16] demonstrate the wealth of obstacles satisfying (1.8). Application of the algorithm in this paper to complicated geometries is restricted by the difficulty of finding globally smooth parametrizations. Application to such geometries would require partition of the conductor surface into several charts, with appropriate use of local mappings. This approach was used in combination with a partition of unity to evaluate weakly singular integrals to spectral accuracy in [4]. Such an extension to our algorithm is nontrivial and not considered in this paper.

In many cases the obstacle restriction imposed in this paper is not a disadvantage. For example, in forward scattering by complex structures the surface integral equation often arises through introduction of a smooth artificial interface surrounding the scatterer [1, 2, 21]. A partial differential equation form of the Maxwell equations—formulated in the interior of the artificial interface and coupled with the surface integral equation on the interface—is solved by finite elements. In inverse scattering, obtaining general qualitative information about the scatterer is more important than resolving fine details such as corners and edges.

We test our algorithm on a range of perfectly conducting scatterers. We include scatterers with concave surface features, and nonsmooth scatterers. We show that just a few tens of thousands of unknowns are sufficient in our algorithm to obtain satisfactory solutions for medium electromagnetic-sized scatterers. Such relatively small linear systems allow us to use direct solves as well as iterative solves. In the case of smooth scatterers our solutions are very accurate, with several decimal places of accuracy. In the case of nonsmooth scatterers with singularity points in their surface, our algorithm yields visually accurate solutions but does not yield the rapid convergence demonstrated for smooth scatterers. In general, only a few decimal places of accuracy are possible for nonsmooth scatterers.

The possibility of using direct solves is a great advantage in monostatic RCS computations. For the monostatic case, since the transmitter and receiver are co-located and because hundreds of receiver measurements are required for the RCS analysis, the exterior Maxwell equations are to be solved with hundreds of boundary conditions induced by various receiver directions. This leads to the requirement of solving hundreds of linear systems with one electromagnetic discretization matrix and hundreds of different right-hand sides. A direct LU factorization of the electromagnetic matrix is very cost effective in this case. Such factorizations are not possible using surface discretization based algorithms such as FISC, because they lead to dense complex linear systems with hundreds of thousands of unknowns. We demonstrate that we can compute the monostatic RCS of medium (electromagnetic-) sized scatterers at over one thousand points in a few hours of CPU time.

The structure of the paper follows that in [16], and we recall various derivations from [16] that are essential for this paper. We follow [16] in section 2 to describe the electric field surface integral equation reformulation of (1.4)–(1.5) using a standard ansatz for electromagnetic fields exterior to the conductor surface ∂D, and we use (1.8) to appropriately transplant the equation onto the surface ∂B, the set of all unit directional vectors. In section 3 we give the details of our method based on tangential and nontangential bases for computing a spectrally accurate approximation to the surface current on ∂D that solves the surface integral equation. In section 4 we describe an efficient way to use the surface current to compute the RCS of the scatterer. In section 5 we present numerical results showing the high-order convergence of our method.

Throughout the paper we denote by $\underline{C}(\partial D)$ and $\underline{C}^{0,\alpha}(\partial D)$, respectively, the spaces of all continuous and uniformly Hölder continuous vector fields on ∂D, where α is the Hölder exponent. We denote by $\underline{\mathcal{T}}(\partial D)$ and $\underline{\mathcal{T}}^{0,\alpha}(\partial D)$, respectively, the spaces of all continuous and uniformly Hölder continuous tangential vector fields on ∂D.

2. Reformulation of the exterior Maxwell equations. Standard boundary integral formulations for the time harmonic Maxwell equations restrict the domain of the magnetic dipole operator to $\underline{\mathcal{T}}(\partial D)$ [8, p. 167]. The restriction is useful because it reduces the order of the singularity of the operator. In fact, it is sufficient to

project the density in the integral operator to $\underline{T}(\partial D)$ (rather than restricting the domain) to achieve the reduction in the order of the singularity. Any vector field in $\underline{C}(\partial D)$ can be projected onto $\underline{T}(\partial D)$ by applying the 3×3 matrix projection operator $\mathcal{P}(\boldsymbol{y}) = I - \boldsymbol{n}(\boldsymbol{y})\boldsymbol{n}(\boldsymbol{y})^T$ for $\boldsymbol{y} \in \partial D$. For $\boldsymbol{a} \in \underline{T}(\partial D)$, since $\mathcal{P}\boldsymbol{a} = \boldsymbol{a}$, the generalized magnetic dipole operator

$$(\mathcal{M}\boldsymbol{a})(\boldsymbol{x}) = 2\int_{\partial D} \boldsymbol{n}(\boldsymbol{x}) \times \mathbf{curl}_{\boldsymbol{x}} \{\Phi(\boldsymbol{x},\boldsymbol{y})\mathcal{P}(\boldsymbol{y})\boldsymbol{a}(\boldsymbol{y})\}\, ds(\boldsymbol{y}), \quad \boldsymbol{x} \in \partial D, \tag{2.1}$$

is the same as the standard magnetic dipole operator [8, p. 167]. In (2.1),

$$\Phi(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{4\pi}\frac{e^{ik|\boldsymbol{x}-\boldsymbol{y}|}}{|\boldsymbol{x}-\boldsymbol{y}|} \tag{2.2}$$

is the fundamental solution of the Helmholtz equation. Application of [8, Theorem 6.13] shows that the linear operator $\mathcal{M}$ is bounded from $\underline{C}(\partial D)$ into $\underline{T}^{0,\alpha}(\partial D)$ for $0 < \alpha < 1$.

Using (1.5), the boundary data $\boldsymbol{f} \in \underline{T}^{0,\alpha}(\partial D)$, and so if $\boldsymbol{w} \in \underline{C}(\partial D)$ is a solution of the surface integral equation

$$\boldsymbol{w}(\boldsymbol{x}) + (\mathcal{M}\boldsymbol{w})(\boldsymbol{x}) = 2\boldsymbol{f}(\boldsymbol{x}), \qquad \boldsymbol{x} \in \partial D, \tag{2.3}$$

then $\boldsymbol{w} \in \underline{T}^{0,\alpha}(\partial D)$. If the wavenumber k is not an interior Maxwell eigenvalue, then (2.3) has a unique solution [7, Theorem 4.23]. [illegible]
k [illegible]

Using the unique tangential solution $\boldsymbol{w}$ of (2.3), we define

$$\boldsymbol{E}(\boldsymbol{x}) = \mathbf{curl} \int_{\partial D} \Phi(\boldsymbol{x},\boldsymbol{y})\boldsymbol{w}(\boldsymbol{y})\, ds(\boldsymbol{y}), \quad \boldsymbol{H}(\boldsymbol{x}) = \frac{1}{ik}\mathbf{curl}\, \boldsymbol{E}(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^3 \setminus \overline{D}. \tag{2.4}$$

Then $\boldsymbol{E}, \boldsymbol{H}$ comprise a unique radiating solution of the harmonic Maxwell equations that satisfies the boundary condition $\boldsymbol{n}(\boldsymbol{x}) \times \boldsymbol{E}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x})$, $\boldsymbol{x} \in \partial D$ [7, Theorem 4.19]. A representation of the far field pattern required for RCS calculations can be obtained from the following asymptotics of the fundamental solution in (2.2): For $\boldsymbol{a} \in \underline{C}(\partial D)$ and as $|\boldsymbol{x}| \to \infty$ uniformly for all $\boldsymbol{y} \in \partial D$ [8, p. 164],

$$\mathbf{curl}_{\boldsymbol{x}} \{\boldsymbol{a}(\boldsymbol{y})\Phi(\boldsymbol{x},\boldsymbol{y})\} = \frac{ik}{4\pi}\frac{e^{ik|\boldsymbol{x}|}}{|\boldsymbol{x}|}\left\{e^{-ik\widehat{\boldsymbol{x}}\cdot\boldsymbol{y}}\widehat{\boldsymbol{x}} \times \boldsymbol{a}(\boldsymbol{y}) + O\left(\frac{|\boldsymbol{a}(\boldsymbol{y})|}{|\boldsymbol{x}|}\right)\right\}, \tag{2.5}$$

where $\widehat{\boldsymbol{x}} = \boldsymbol{x}/|\boldsymbol{x}| \in \partial B$. Using (2.4) and (2.5) in (1.3), the electric far field pattern of the scattered electric field $\boldsymbol{E}$ in (2.4) can be represented as

$$\boldsymbol{E}_\infty(\widehat{\boldsymbol{x}}) = \frac{ik}{4\pi}\int_{\partial D} e^{-ik\widehat{\boldsymbol{x}}\cdot\boldsymbol{y}}\widehat{\boldsymbol{x}} \times \boldsymbol{w}(\boldsymbol{y})\, ds(\boldsymbol{y}), \quad \widehat{\boldsymbol{x}} \in \partial B. \tag{2.6}$$

Thus high-order approximate solutions of (2.3) facilitate accurate simulation of the electromagnetic waves and measurement of the RCS using (2.6) and (1.2)–(1.7).

Using the following identity for a tangential field $\boldsymbol{b}$ [8, p. 166],

$$\begin{aligned}\boldsymbol{n}(\boldsymbol{x}) \times\ \mathbf{curl}_{\boldsymbol{x}} \{\Phi(\boldsymbol{x},\boldsymbol{y})\boldsymbol{b}(\boldsymbol{y})\} &= \mathbf{grad}_{\boldsymbol{x}}\Phi(\boldsymbol{x},\boldsymbol{y})\,[\boldsymbol{n}(\boldsymbol{x}) - \boldsymbol{n}(\boldsymbol{y})]^T\,\boldsymbol{b}(\boldsymbol{y}) \\ &\quad -\boldsymbol{b}(\boldsymbol{y})\boldsymbol{n}(\boldsymbol{x})^T\mathbf{grad}_{\boldsymbol{x}}\Phi(\boldsymbol{x},\boldsymbol{y}),\end{aligned} \tag{2.7}$$

and (2.1), (2.7), and (2.2), the generalized magnetic dipole operator can be written as

$$\mathcal{M}\boldsymbol{a}(\boldsymbol{x}) = \int_{\partial D} \left[\frac{1}{|\boldsymbol{x}-\boldsymbol{y}|} m_1(\boldsymbol{x},\boldsymbol{y}) + m_2(\boldsymbol{x},\boldsymbol{y})\right] \mathcal{P}(\boldsymbol{y})\boldsymbol{a}(\boldsymbol{y})\, ds(\boldsymbol{y}), \quad \boldsymbol{x} \in \partial D, \tag{2.8}$$

where for $i = 1, 2$,

$$\begin{aligned} m_i(\boldsymbol{x},\boldsymbol{y}) &= m_{i,1}(\boldsymbol{x},\boldsymbol{y}) \frac{1}{|\boldsymbol{x}-\boldsymbol{y}|^2} (\boldsymbol{x}-\boldsymbol{y}) \left[\boldsymbol{n}(\boldsymbol{x}) - \boldsymbol{n}(\boldsymbol{y})\right]^T \\ &\quad + m_{i,2}(\boldsymbol{x},\boldsymbol{y}) \frac{1}{|\boldsymbol{x}-\boldsymbol{y}|^2} \boldsymbol{n}(\boldsymbol{x})^T (\boldsymbol{x}-\boldsymbol{y}) I + m_{i,3}(\boldsymbol{x},\boldsymbol{y}), \end{aligned} \tag{2.9}$$

with I being the 3×3 identity matrix, $m_{2,3} = 0$ the 3×3 zero matrix, and

$$\begin{aligned} &m_{1,1}(\boldsymbol{x},\boldsymbol{y}) = -\frac{1}{2\pi} t(\boldsymbol{x},\boldsymbol{y}), \qquad m_{1,2}(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{2\pi} t(\boldsymbol{x},\boldsymbol{y}), \\ &m_{1,3}(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{2\pi} ks(\boldsymbol{x},\boldsymbol{y}) \left[\boldsymbol{n}(\boldsymbol{x})^T (\boldsymbol{x}-\boldsymbol{y}) I - (\boldsymbol{x}-\boldsymbol{y}) \left(\boldsymbol{n}(\boldsymbol{x}) - \boldsymbol{n}(\boldsymbol{y})\right)^T\right], \\ &m_{2,1}(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{2\pi} \left[ikt(\boldsymbol{x},\boldsymbol{y}) - is(\boldsymbol{x},\boldsymbol{y})\right], \qquad m_{2,2}(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{2\pi} \left[is(\boldsymbol{x},\boldsymbol{y}) - ikt(\boldsymbol{x},\boldsymbol{y})\right], \\ &s(\boldsymbol{x},\boldsymbol{y}) = \begin{cases} \sin(k|\boldsymbol{x}-\boldsymbol{y}|)/|\boldsymbol{x}-\boldsymbol{y}| & \text{if } \boldsymbol{x} \neq \boldsymbol{y}, \\ k & \text{if } \boldsymbol{x} = \boldsymbol{y}, \end{cases} \qquad t(\boldsymbol{x},\boldsymbol{y}) = \cos(k|\boldsymbol{x}-\boldsymbol{y}|). \end{aligned} \tag{2.10}$$

Each $m_{i,j}$ $(i = 1, 2,\ j = 1, 2, 3)$ is infinitely differentiable on $\mathbb{R}^3 \times \mathbb{R}^3$. It is important for our spectrally accurate method to separate the kernel of $\mathcal{M}$ in (2.1) into its weakly singular and smooth parts in this way. Next we use the bijective parametrization $\boldsymbol{q} : \partial B \to \partial D$ to transplant (2.8) onto the reference spherical surface, which facilitates exact treatment of the weakly singular part using a singularity division technique.

For a given density field $\boldsymbol{A} \in \underline{C}(\partial B)$, using the change of variables, we derive the equivalent form of (2.8),

$$\mathcal{M}\boldsymbol{A}(\widehat{\boldsymbol{x}}) = \mathcal{M}_1\boldsymbol{A}(\widehat{\boldsymbol{x}}) + \mathcal{M}_2\boldsymbol{A}(\widehat{\boldsymbol{x}}), \qquad \widehat{\boldsymbol{x}} \in \partial B, \tag{2.11}$$

with

$$\mathcal{M}_1\boldsymbol{A}(\widehat{\boldsymbol{x}}) = \int_{\partial B} \frac{1}{|\widehat{\boldsymbol{x}}-\widehat{\boldsymbol{y}}|} M_1(\widehat{\boldsymbol{x}},\widehat{\boldsymbol{y}})\boldsymbol{A}(\widehat{\boldsymbol{y}})\, ds(\widehat{\boldsymbol{y}}), \tag{2.12}$$

$$\mathcal{M}_2\boldsymbol{A}(\widehat{\boldsymbol{x}}) = \int_{\partial B} M_2(\widehat{\boldsymbol{x}},\widehat{\boldsymbol{y}})\boldsymbol{A}(\widehat{\boldsymbol{y}})\, ds(\widehat{\boldsymbol{y}}), \tag{2.13}$$

where for each $\widehat{\boldsymbol{x}}, \widehat{\boldsymbol{y}} \in \partial B$, $M_1(\widehat{\boldsymbol{x}},\widehat{\boldsymbol{y}})$ and $M_2(\widehat{\boldsymbol{x}},\widehat{\boldsymbol{y}})$ are 3×3 matrices defined by

$$M_1(\widehat{\boldsymbol{x}},\widehat{\boldsymbol{y}}) = R(\widehat{\boldsymbol{x}},\widehat{\boldsymbol{y}}) J(\widehat{\boldsymbol{y}})\, m_1(\boldsymbol{q}(\widehat{\boldsymbol{x}}),\boldsymbol{q}(\widehat{\boldsymbol{y}}))\mathcal{P}(\boldsymbol{q}(\widehat{\boldsymbol{y}})), \tag{2.14}$$

$$M_2(\widehat{\boldsymbol{x}},\widehat{\boldsymbol{y}}) = J(\widehat{\boldsymbol{y}})\, m_2(\boldsymbol{q}(\widehat{\boldsymbol{x}}),\boldsymbol{q}(\widehat{\boldsymbol{y}}))\mathcal{P}(\boldsymbol{q}(\widehat{\boldsymbol{y}})), \tag{2.15}$$

J is the Jacobian of $\boldsymbol{q}$, and

$$R(\widehat{\boldsymbol{x}},\widehat{\boldsymbol{y}}) := \frac{|\widehat{\boldsymbol{x}}-\widehat{\boldsymbol{y}}|}{|\boldsymbol{q}(\widehat{\boldsymbol{x}}) - \boldsymbol{q}(\widehat{\boldsymbol{y}})|}. \tag{2.16}$$

The integral operator $\mathcal{M}_2$ has a smooth kernel that can be approximated to spectral accuracy using spherical polynomials. Spectrally accurate approximation of

the weakly singular operator $\mathcal{M}_1$ can be achieved by introducing a rotated coordinate system on ∂B, under which the weak singularity in the transformed operator occurs only at the north pole. To this end, we first define the 3×3 rotation matrices corresponding to positive rotations by ψ about the z-axis and y-axis:

$$\mathcal{R}_z(\psi) := \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \mathcal{R}_y(\psi) := \begin{pmatrix} \cos\psi & 0 & \sin\psi \\ 0 & 1 & 0 \\ -\sin\psi & 0 & \cos\psi \end{pmatrix}. \tag{2.17}$$

For each $\widehat{\boldsymbol{x}} = \boldsymbol{p}(\theta, \phi)$, the orthogonal matrix

$$T_{\widehat{\boldsymbol{x}}} := \mathcal{R}_z(\phi)\mathcal{R}_y(-\theta)\mathcal{R}_z(-\phi) \tag{2.18}$$

maps $\widehat{\boldsymbol{x}}$ to the north pole $\widehat{\boldsymbol{n}} = (0, 0, 1)^T$. The matrix $T_{\widehat{\boldsymbol{x}}}$ induces a linear transformation

$$\mathcal{T}_{\widehat{\boldsymbol{x}}}\boldsymbol{A}(\widehat{\boldsymbol{z}}) := \boldsymbol{A}(T_{\widehat{\boldsymbol{x}}}^{-1}\widehat{\boldsymbol{z}}), \qquad \widehat{\boldsymbol{z}} \in \partial B, \qquad \boldsymbol{A} \in \underline{C}(\partial B), \tag{2.19}$$

and its bivariate analogue

$$\mathcal{T}_{\widehat{\boldsymbol{x}}}\boldsymbol{A}(\widehat{\boldsymbol{z}}_1, \widehat{\boldsymbol{z}}_2) := \boldsymbol{A}(T_{\widehat{\boldsymbol{x}}}^{-1}\widehat{\boldsymbol{z}}_1, T_{\widehat{\boldsymbol{x}}}^{-1}\widehat{\boldsymbol{z}}_2), \qquad \widehat{\boldsymbol{z}}_1, \widehat{\boldsymbol{z}}_2 \in \partial B, \qquad \boldsymbol{A} \in \underline{C}(\partial B \times \partial B). \tag{2.20}$$

Using the orthogonality of $T_{\widehat{\boldsymbol{x}}}$, we have, for $\widehat{\boldsymbol{x}}, \widehat{\boldsymbol{z}} \in \partial B$ and $\widehat{\boldsymbol{y}} = T_{\widehat{\boldsymbol{x}}}^{-1}\widehat{\boldsymbol{z}}$,

$$|\widehat{\boldsymbol{x}} - \widehat{\boldsymbol{y}}| = |T_{\widehat{\boldsymbol{x}}}^{-1}(\widehat{\boldsymbol{n}} - \widehat{\boldsymbol{z}})| = |\widehat{\boldsymbol{n}} - \widehat{\boldsymbol{z}}|. \tag{2.21}$$

Using (2.12) and the fact that the surface measure on ∂B is invariant,

$$\mathcal{M}_1\boldsymbol{A}(\widehat{\boldsymbol{x}}) = \int_{\partial B} \frac{1}{|\widehat{\boldsymbol{n}} - \widehat{\boldsymbol{z}}|} \mathcal{T}_{\widehat{\boldsymbol{x}}} M_1(\widehat{\boldsymbol{n}}, \widehat{\boldsymbol{z}})\ \mathcal{T}_{\widehat{\boldsymbol{x}}}\boldsymbol{A}(\widehat{\boldsymbol{z}})\ ds(\widehat{\boldsymbol{z}}), \quad \boldsymbol{A} \in \underline{C}(\partial B). \tag{2.22}$$

The transformed coordinate system has two crucial benefits. First, in spherical polar coordinates, with $\widehat{\boldsymbol{z}} = \boldsymbol{p}(\theta', \phi')$, the denominator $|\widehat{\boldsymbol{n}} - \widehat{\boldsymbol{z}}| = 2\sin\theta'/2$ in (2.22) is canceled out by the surface element $\sin\theta'\ d\theta'\ d\phi'$. Second, the operator $(\theta', \phi') \mapsto \mathcal{T}_{\widehat{\boldsymbol{x}}} M_1(\widehat{\boldsymbol{n}}, \boldsymbol{p}(\theta', \phi'))$ is infinitely continuously differentiable, with all derivatives 2π-periodic in each variable, and each partial derivative is uniformly bounded with respect to $\widehat{\boldsymbol{x}} \in \partial B$ (see the proof of [16, Theorem 1]).

Finally, we write $\boldsymbol{W}(\widehat{\boldsymbol{x}}) = \boldsymbol{w}(\boldsymbol{q}(\widehat{\boldsymbol{x}}))$ and $\boldsymbol{F}(\widehat{\boldsymbol{x}}) = \boldsymbol{f}(\boldsymbol{q}(\widehat{\boldsymbol{x}}))$, and we use (2.22) and (2.13) to derive the transplant of (2.3) onto ∂B,

$$\boldsymbol{W}(\widehat{\boldsymbol{x}}) + \mathcal{M}\boldsymbol{W}(\widehat{\boldsymbol{x}}) = 2\boldsymbol{F}(\widehat{\boldsymbol{x}}), \qquad \widehat{\boldsymbol{x}} \in \partial B, \quad \boldsymbol{W} = \boldsymbol{w} \circ \boldsymbol{q}, \qquad \boldsymbol{F} = \boldsymbol{f} \circ \boldsymbol{q}. \tag{2.23}$$

In (2.23) we have an ideal reformulation of (2.3) to obtain high-order approximation of the surface current $\boldsymbol{w}$ through a spectrally accurate evaluation of (2.11), where we approximate both the smooth kernels and the transplanted surface current in a hybrid way from the space of vector spherical polynomials.

3. A computer implementable spectral algorithm. In this section, we describe a high-order fully discrete algorithm that requires no further approximations (such as approximations of Galerkin integrals) to compute spectrally accurate approximate solutions of (2.23). Details required for computer implementation of our algorithm are in section 3.3.

In the special case where the scatterer ∂D is spherical, we seek a spectrally accurate tangential approximation to the solution of (2.23) in the tangential harmonics

$$\underline{\mathbb{T}}_n = \text{span}\left\{ \boldsymbol{Y}_{l,j}^{(\widetilde{k})} \ : \ 1 \le l \le n, \ |j| \le l, \ \widetilde{k} = 1, 2 \right\},$$

where

$$\boldsymbol{Y}_{l,j}^{(1)}(\widehat{\boldsymbol{x}}) = \frac{1}{\sqrt{l(l+1)}} \mathbf{Grad}\, Y_{l,j}(\widehat{\boldsymbol{x}}), \qquad \boldsymbol{Y}_{l,j}^{(2)}(\widehat{\boldsymbol{x}}) = \widehat{\boldsymbol{x}} \times \boldsymbol{Y}_{l,j}^{(1)}(\widehat{\boldsymbol{x}}), \qquad \widehat{\boldsymbol{x}} \in \partial B, \tag{3.1}$$

where **Grad** is the surface gradient [8, p. 167] and using (1.1),

$$Y_{l,j}(\widehat{\boldsymbol{x}}) = (-1)^{(j+|j|)/2} \sqrt{\frac{2l+1}{4\pi} \frac{(l-|j|)!}{(l+|j|)!}} P_l^{|j|}(\cos\theta) e^{ij\phi}, \quad 0 \le l \le n, \ |j| \le l, \tag{3.2}$$

with $P_l^{|j|}$ being the associated Legendre functions. The vector spherical harmonics $\boldsymbol{Y}_{l,j}^{(1)}$ and $\boldsymbol{Y}_{l,j}^{(2)}$ are tangential on ∂B.

In the case of a nonspherical scatterer, since the solution of the transplanted equation (2.23) need not be tangential on ∂B, we seek a spectrally accurate approximation in the augmented space

$$\underline{\mathbb{Q}}_n = \underline{\mathbb{T}}_n \oplus \text{span}\left\{ \boldsymbol{Y}_{l,j}^{(3)} \ : \ 0 \le l \le n, \ |j| \le l \right\}, \tag{3.3}$$

where

$$\boldsymbol{Y}_{l,j}^{(3)}(\widehat{\boldsymbol{x}}) = \widehat{\boldsymbol{x}} Y_{l,j}(\widehat{\boldsymbol{x}}), \quad \widehat{\boldsymbol{x}} \in \partial B. \tag{3.4}$$

Using (1.1), (3.2), (3.1), and (3.4), for $\widetilde{k} = 1, 2, 3$, we have the convenient representation

$$\boldsymbol{Y}_{l,j}^{(\widetilde{k})}(\widehat{\boldsymbol{x}}) = \sum_{t=1}^{3} \alpha_{l,j}^{(\widetilde{k},t)}(\theta) e^{ij\phi} \boldsymbol{v}^{(t)}(\theta, \phi), \qquad \widehat{\boldsymbol{x}} \in \partial B, \tag{3.5}$$

where, denoting the normalizing coefficient in (3.2) by c_l^j,

$$\begin{aligned}
&\boldsymbol{v}^{(1)}(\theta,\phi) = (\cos\theta\cos\phi, \cos\theta\sin\phi, -\sin\theta)^T, \ \boldsymbol{v}^{(2)}(\theta,\phi) = (-\sin\phi, \cos\phi, 0)^T, \\
&\boldsymbol{v}^{(3)}(\theta,\phi) = (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)^T, \\
&\alpha_{l,j}^{(1,1)}(\theta) = \alpha_{l,j}^{(2,2)}(\theta) = c_l^j P_l^{|j|\prime}(\cos\theta)/\sqrt{l(l+1)}, \\
&\alpha_{l,j}^{(1,2)}(\theta) = -\alpha_{l,j}^{(2,1)}(\theta) = c_l^j (ij/\sin\theta) P_l^{|j|}(\cos\theta)/\sqrt{l(l+1)}, \\
&\alpha_{l,j}^{(3,3)}(\theta) = c_l^j P_l^{|j|}(\cos\theta), \\
&\alpha_{l,j}^{(1,3)}(\theta) = \alpha_{l,j}^{(2,3)}(\theta) = \alpha_{l,j}^{(3,1)}(\theta) = \alpha_{l,j}^{(3,2)}(\theta) = 0.
\end{aligned}$$

The standard Galerkin scheme for (2.23) requires evaluation of the L_2 inner product integrals

$$(\boldsymbol{G}, \boldsymbol{H}) = \mathcal{I}(\overline{\boldsymbol{H}}^T \boldsymbol{G}) = \int_{\partial B} \overline{\boldsymbol{H}(\widehat{\boldsymbol{x}})}^T \boldsymbol{G}(\widehat{\boldsymbol{x}}) \, ds(\widehat{\boldsymbol{x}}), \qquad \boldsymbol{G}, \boldsymbol{H} \in \underline{C}(\partial B). \tag{3.6}$$

In practice, a quadrature rule on ∂B is required to evaluate the integrals in (3.6). We approximate the inner product integrals using the Gauss-rectangle quadrature rule

$$\int_{\partial B} G(\widehat{\boldsymbol{x}})\, ds(\widehat{\boldsymbol{x}}) \approx Q_N(G) = \sum_{r=0}^{2N+1} \sum_{s=1}^{N+1} \mu_r^N \nu_s^N G(\widehat{\boldsymbol{z}}_{rs}^N), \quad G \in C(\partial B), \tag{3.7}$$

where, using (1.1), $\widehat{\boldsymbol{z}}_{rs}^N = \boldsymbol{p}(\theta_s^N, \phi_r^N)$, $\theta_s^N = \cos^{-1} z_s$, z_s are the zeros of the Legendre polynomial of degree $N+1$, ν_s^N are the corresponding Gauss–Legendre weights ($s = 1, \ldots, N+1$), and

$$\mu_r^N = \frac{\pi}{N+1}, \qquad \phi_r^N = \frac{r\pi}{N+1}, \qquad r = 0, \ldots, 2N+1. \tag{3.8}$$

Since the quadrature (3.7) is exact for spherical polynomials of degree at most $2N$, it induces the discrete inner product

$$(\boldsymbol{G}, \boldsymbol{H})_N = Q_N(\overline{\boldsymbol{H}}^T \boldsymbol{G}) = \sum_{r=0}^{2N+1} \sum_{s=1}^{N+1} \mu_r^N \nu_s^N \overline{\boldsymbol{H}(\widehat{\boldsymbol{z}}_{rs}^N)}^T \boldsymbol{G}(\widehat{\boldsymbol{z}}_{rs}^N), \quad \boldsymbol{G}, \boldsymbol{H} \in \underline{C}(\partial B), \tag{3.9}$$

on $\underline{\mathbb{P}}_N$, the space of vector spherical polynomials whose Euclidean components are spherical polynomials of degree not greater than N. That is, $(\boldsymbol{G}, \boldsymbol{H}) = (\boldsymbol{G}, \boldsymbol{H})_N$ for $\boldsymbol{G}, \boldsymbol{H} \in \underline{\mathbb{P}}_N$. Since $\boldsymbol{Y}_{n,j}^{(\widetilde{k})}$ for $\widetilde{k} = 1, 2, 3$ in (3.1)–(3.4) is a vector spherical polynomial of degree $n+1$, we have $\underline{\mathbb{Q}}_n \subseteq \underline{\mathbb{P}}_{n+1}$, and hence the vector spherical harmonics in (3.1) and (3.4) are orthonormal with respect to the discrete inner product (3.9) with $N = n+1$.

We approximate the transplanted surface current in (2.23) by a computable projection onto the finite dimensional space $\underline{\mathbb{Q}}_n$. In particular, we approximate the theoretical solution $\boldsymbol{W}$ of (2.23) using a computable representation $\underline{\mathcal{O}}_n \boldsymbol{W}$, where $\underline{\mathcal{O}}_n$ is a fully discrete projection operator defined as

$$\underline{\mathcal{O}}_n \boldsymbol{A} = \sum_{l=0}^{n} \sum_{|j| \le l} \sum_{\widetilde{k}=1}^{3} (\boldsymbol{A}, \boldsymbol{Y}_{l,j}^{(\widetilde{k})})_{n+1} \boldsymbol{Y}_{l,j}^{(\widetilde{k})}, \qquad \boldsymbol{A} \in \underline{C}(\partial B), \tag{3.10}$$

where, for convenience, throughout the paper, we use the notation $\boldsymbol{Y}_{00}^{(1)} = \boldsymbol{Y}_{00}^{(2)} = \boldsymbol{0}$. Using the orthonormality of the vector harmonics with respect to the discrete inner product in (3.10), we get $\underline{\mathcal{O}}_n \boldsymbol{Y}_{l,j}^{(\widetilde{k})} = \boldsymbol{Y}_{l,j}^{(\widetilde{k})}$ for $0 \le l \le n$, $|j| \le l$, $1 \le \widetilde{k} \le 3$. Throughout the paper, when ∂D is a sphere, we replace $\sum_{\widetilde{k}=1}^{3}$ in (3.10) by $\sum_{\widetilde{k}=1}^{2}$, and $\underline{\mathbb{Q}}_n$ by $\underline{\mathbb{T}}_n$.

3.1. Fully discrete approximations of magnetic dipole integrals. For evaluation of inner integrals in the Galerkin scheme, for example, $\mathcal{M}_1 \boldsymbol{A}(\widehat{\boldsymbol{x}})$, we need to approximate the integrand $\mathcal{T}_{\widehat{\boldsymbol{x}}} M_1(\widehat{\boldsymbol{n}}, \cdot)\, \mathcal{T}_{\widehat{\boldsymbol{x}}} \boldsymbol{A}(\cdot)$ in (2.22) for $\boldsymbol{A} \in \underline{C}(\partial B)$, $\widehat{\boldsymbol{x}} \in \partial B$. One approach is to project this integrand using the fully discrete operator $\underline{\mathcal{O}}_n$ in (3.10), as in [15]. However, as described in [16, Remark 2], using a technical analysis, such an approximation, leads to stagnated convergence instead of the spectral accuracy we seek in this work. In this hybrid scheme, to approximate the inner integral, we use the componentwise nontangential harmonics defined using (3.2) as

$$\boldsymbol{Y}_{l,j,\widetilde{k}} = Y_{l,j} \boldsymbol{e}_{\widetilde{k}}, \qquad 0 \le l \le n', \ |j| \le l, \ 1 \le \widetilde{k} \le 3, \tag{3.11}$$

where $\boldsymbol{e}_{\widetilde{k}}$ denotes the $\widetilde{k}$th Euclidean vector. To this end, analogous to (3.10), using the discrete inner product (3.9), we define a fully discrete orthogonal projection operator

$$\underline{\mathcal{L}}_{n'}\boldsymbol{A} = \sum_{l=0}^{n'}\sum_{|j|\le l}\sum_{\widetilde{k}=1}^{3}(\boldsymbol{A},\boldsymbol{Y}_{l,j,\widetilde{k}})_{n'}\boldsymbol{Y}_{l,j,\widetilde{k}}, \qquad \boldsymbol{A}\in\underline{C}(\partial B). \tag{3.12}$$

Clearly $\underline{\mathcal{L}}_{n'}\boldsymbol{Y}_{l,j,\widetilde{k}} = \boldsymbol{Y}_{l,j,\widetilde{k}}$ for $0\le l\le n',\ |j|\le l,\ 1\le\widetilde{k}\le 3$. Here, n' depends on n and will be specified later in this section. We approximate $\mathcal{T}_{\widehat{\boldsymbol{x}}}M_1(\widehat{\boldsymbol{n}},\cdot)\ \mathcal{T}_{\widehat{\boldsymbol{x}}}\boldsymbol{A}(\cdot)$ in (2.22) by $\underline{\mathcal{L}}_{n'}\{\mathcal{T}_{\widehat{\boldsymbol{x}}}M_1(\widehat{\boldsymbol{n}},\cdot)\ \mathcal{T}_{\widehat{\boldsymbol{x}}}\boldsymbol{A}(\cdot)\}$ and follow the details in [16].

More precisely, for $\boldsymbol{A}\in\underline{C}(\partial B)$ and $\widehat{\boldsymbol{x}}\in\partial B$, using (2.22) and (3.12), the approximation to $\mathcal{M}_1\boldsymbol{A}(\widehat{\boldsymbol{x}})$ is

$$\begin{aligned}
\mathcal{M}_{1,n'}\boldsymbol{A}(\widehat{\boldsymbol{x}}) &= \int_{\partial B}\frac{1}{|\widehat{\boldsymbol{n}}-\widehat{\boldsymbol{z}}|}\underline{\mathcal{L}}_{n'}\left\{\mathcal{T}_{\widehat{\boldsymbol{x}}}M_1(\widehat{\boldsymbol{n}},\cdot)\ \mathcal{T}_{\widehat{\boldsymbol{x}}}\boldsymbol{A}(\cdot)\right\}(\widehat{\boldsymbol{z}})\ ds(\widehat{\boldsymbol{z}})\\
&= \int_{\partial B}\frac{1}{|\widehat{\boldsymbol{n}}-\widehat{\boldsymbol{z}}|}\sum_{l=0}^{n'}\sum_{|j|\le l}\sum_{\widetilde{k}=1}^{3}\left(\mathcal{T}_{\widehat{\boldsymbol{x}}}M_1(\widehat{\boldsymbol{n}},\cdot)\ \mathcal{T}_{\widehat{\boldsymbol{x}}}\boldsymbol{A}(\cdot),\boldsymbol{Y}_{l,j,\widetilde{k}}(\cdot)\right)_{n'}\boldsymbol{Y}_{l,j,\widetilde{k}}(\widehat{\boldsymbol{z}})\ ds(\widehat{\boldsymbol{z}})\\
&= \sum_{l=0}^{n'}\sum_{|j|\le l}\sum_{\widetilde{k}=1}^{3}\left(\mathcal{T}_{\widehat{\boldsymbol{x}}}M_1(\widehat{\boldsymbol{n}},\cdot)\ \mathcal{T}_{\widehat{\boldsymbol{x}}}\boldsymbol{A}(\cdot),\boldsymbol{Y}_{l,j,\widetilde{k}}(\cdot)\right)_{n'}\int_{\partial B}\frac{1}{|\widehat{\boldsymbol{n}}-\widehat{\boldsymbol{z}}|}\boldsymbol{Y}_{l,j,\widetilde{k}}(\widehat{\boldsymbol{z}})\ ds(\widehat{\boldsymbol{z}})\\
&= \sum_{l=0}^{n'}\sum_{|j|\le l}\sum_{\widetilde{k}=1}^{3}\left(\mathcal{T}_{\widehat{\boldsymbol{x}}}M_1(\widehat{\boldsymbol{n}},\cdot)\ \mathcal{T}_{\widehat{\boldsymbol{x}}}\boldsymbol{A}(\cdot),\boldsymbol{Y}_{l,j,\widetilde{k}}(\cdot)\right)_{n'}\frac{4\pi}{2l+1}\boldsymbol{Y}_{l,j,\widetilde{k}}(\widehat{\boldsymbol{n}}).
\end{aligned} \tag{3.13}$$

In the last step we have used the standard eigenfunction property of the spherical harmonics [11]. Using (3.9), we expand the discrete inner product and obtain

$$\begin{aligned}
&\mathcal{M}_{1,n'}\boldsymbol{A}(\widehat{\boldsymbol{x}})\\
&= \sum_{l=0}^{n'}\sum_{|j|\le l}\sum_{\widetilde{k}=1}^{3}\sum_{s'=1}^{n'+1}\sum_{r'=0}^{2n'+1}\mu_{r'}^{n'}\nu_{s'}^{n'}\frac{4\pi}{2l+1}\overline{\boldsymbol{Y}_{l,j,\widetilde{k}}(\widehat{\boldsymbol{z}}_{r's'}^{n'})}^{T}\mathcal{T}_{\widehat{\boldsymbol{x}}}M_1(\widehat{\boldsymbol{n}},\widehat{\boldsymbol{z}}_{r's'}^{n'})\ \mathcal{T}_{\widehat{\boldsymbol{x}}}\boldsymbol{A}(\widehat{\boldsymbol{z}}_{r's'}^{n'})\boldsymbol{Y}_{l,j,\widetilde{k}}(\widehat{\boldsymbol{n}})\\
&= \sum_{l=0}^{n'}\sum_{\widetilde{k}=1}^{3}\sum_{s'=1}^{n'+1}\sum_{r'=0}^{2n'+1}\mu_{r'}^{n'}\nu_{s'}^{n'}\frac{4\pi}{2l+1}\boldsymbol{e}_{\widetilde{k}}\boldsymbol{e}_{\widetilde{k}}^{T}\mathcal{T}_{\widehat{\boldsymbol{x}}}M_1(\widehat{\boldsymbol{n}},\widehat{\boldsymbol{z}}_{r's'}^{n'})\ \mathcal{T}_{\widehat{\boldsymbol{x}}}\boldsymbol{A}(\widehat{\boldsymbol{z}}_{r's'}^{n'})\sum_{|j|\le l}\overline{Y_{l,j}(\widehat{\boldsymbol{z}}_{r's'}^{n'})}Y_{l,j}(\widehat{\boldsymbol{n}}).
\end{aligned}$$

Using the addition theorem for the spherical harmonics [8, Theorem 2.8], this simplifies to

$$\begin{aligned}
\mathcal{M}_{1,n'}\boldsymbol{A}(\widehat{\boldsymbol{x}}) &= \sum_{l=0}^{n'}\sum_{\widetilde{k}=1}^{3}\sum_{s'=1}^{n'+1}\sum_{r'=0}^{2n'+1}\mu_{r'}^{n'}\nu_{s'}^{n'}\boldsymbol{e}_{\widetilde{k}}\boldsymbol{e}_{\widetilde{k}}^{T}\mathcal{T}_{\widehat{\boldsymbol{x}}}M_1(\widehat{\boldsymbol{n}},\widehat{\boldsymbol{z}}_{r's'}^{n'})\ \mathcal{T}_{\widehat{\boldsymbol{x}}}\boldsymbol{A}(\widehat{\boldsymbol{z}}_{r's'}^{n'})P_l(\widehat{\boldsymbol{n}}\cdot\widehat{\boldsymbol{z}}_{r's'}^{n'})\\
&= \sum_{s'=1}^{n'+1}\sum_{r'=0}^{2n'+1}\mu_{r'}^{n'}\nu_{s'}^{n'}\alpha_{s'}^{n'}\mathcal{T}_{\widehat{\boldsymbol{x}}}M_1(\widehat{\boldsymbol{n}},\widehat{\boldsymbol{z}}_{r's'}^{n'})\ \mathcal{T}_{\widehat{\boldsymbol{x}}}\boldsymbol{A}(\widehat{\boldsymbol{z}}_{r's'}^{n'}),
\end{aligned} \tag{3.14}$$

where $\alpha_{s'}^{n'}=\sum_{l=0}^{n'}P_l(\cos\theta_{s'}^{n'})$, and we have used $\widehat{\boldsymbol{n}}\cdot\widehat{\boldsymbol{z}}_{r's'}^{n'}=\cos\theta_{s'}^{n'}$. Thus $\mathcal{M}_{1,n'}\boldsymbol{A}$ is a fully discrete computable approximation to the transplanted weakly singular part of the generalized magnetic dipole operator in (2.8).

We use a similar approximation for the smooth part of the operator. We write (2.13) in the rotated coordinate system as

$$\mathcal{M}_2\boldsymbol{A}(\widehat{\boldsymbol{x}}) = \int_{\partial B} \mathcal{T}_{\widehat{\boldsymbol{x}}} M_2(\widehat{\boldsymbol{n}}, \widehat{\boldsymbol{z}})\, \mathcal{T}_{\widehat{\boldsymbol{x}}} \boldsymbol{A}(\widehat{\boldsymbol{z}})\, ds(\widehat{\boldsymbol{z}}). \tag{3.15}$$

Our approximation to $\mathcal{M}_2\boldsymbol{A}$ is then

$$\begin{aligned}
\mathcal{M}_{2,n'}\boldsymbol{A}(\widehat{\boldsymbol{x}}) &= \int_{\partial B} \underline{\mathcal{L}}_{n'} \left\{ \mathcal{T}_{\widehat{\boldsymbol{x}}} M_2(\widehat{\boldsymbol{n}}, \cdot)\, \mathcal{T}_{\widehat{\boldsymbol{x}}} \boldsymbol{A}(\cdot) \right\} (\widehat{\boldsymbol{z}})\, ds(\widehat{\boldsymbol{z}}) \\
&= \int_{\partial B} \sum_{l=0}^{n'} \sum_{|j|\le l} \sum_{\widetilde{k}=1}^{3} \left(\mathcal{T}_{\widehat{\boldsymbol{x}}} M_2(\widehat{\boldsymbol{n}}, \cdot)\, \mathcal{T}_{\widehat{\boldsymbol{x}}} \boldsymbol{A}(\cdot), \boldsymbol{Y}_{l,j,\widetilde{k}}(\cdot) \right)_{n'} \boldsymbol{Y}_{l,j,\widetilde{k}}(\widehat{\boldsymbol{z}})\, ds(\widehat{\boldsymbol{z}}) \\
&= \sum_{l=0}^{n'} \sum_{|j|\le l} \sum_{\widetilde{k}=1}^{3} \left(\mathcal{T}_{\widehat{\boldsymbol{x}}} M_2(\widehat{\boldsymbol{n}}, \cdot)\, \mathcal{T}_{\widehat{\boldsymbol{x}}} \boldsymbol{A}(\cdot), \boldsymbol{Y}_{l,j,\widetilde{k}}(\cdot) \right)_{n'} \int_{\partial B} \boldsymbol{Y}_{l,j,\widetilde{k}}(\widehat{\boldsymbol{z}})\, ds(\widehat{\boldsymbol{z}}).
\end{aligned} \tag{3.16}$$

Noting that $\boldsymbol{Y}_{0,0,\widetilde{k}} = \boldsymbol{e}_{\widetilde{k}}/\sqrt{4\pi}$, the orthogonality of the $\boldsymbol{Y}_{l,j,\widetilde{k}}$ gives

$$\mathcal{M}_{2,n'}\boldsymbol{A}(\widehat{\boldsymbol{x}}) = \sum_{\widetilde{k}=1}^{3} \left(\mathcal{T}_{\widehat{\boldsymbol{x}}} M_2(\widehat{\boldsymbol{n}}, \cdot)\, \mathcal{T}_{\widehat{\boldsymbol{x}}} \boldsymbol{A}(\cdot), \boldsymbol{e}_{\widetilde{k}} \right)_{n'} \boldsymbol{e}_{\widetilde{k}}.$$

Expanding the discrete inner product using (3.9),

$$\begin{aligned}
\mathcal{M}_{2,n'}\boldsymbol{A}(\widehat{\boldsymbol{x}}) &= \sum_{\widetilde{k}=1}^{3} \sum_{s'=1}^{n'+1} \sum_{r'=0}^{2n'+1} \mu_{r'}^{n'} \nu_{s'}^{n'} \boldsymbol{e}_{\widetilde{k}} \boldsymbol{e}_{\widetilde{k}}^{T} \mathcal{T}_{\widehat{\boldsymbol{x}}} M_2(\widehat{\boldsymbol{n}}, \widehat{\boldsymbol{z}}_{r's'}^{n'})\, \mathcal{T}_{\widehat{\boldsymbol{x}}} \boldsymbol{A}(\widehat{\boldsymbol{z}}_{r's'}^{n'}) \\
&= \sum_{s'=1}^{n'+1} \sum_{r'=0}^{2n'+1} \mu_{r'}^{n'} \nu_{s'}^{n'} \mathcal{T}_{\widehat{\boldsymbol{x}}} M_2(\widehat{\boldsymbol{n}}, \widehat{\boldsymbol{z}}_{r's'}^{n'})\, \mathcal{T}_{\widehat{\boldsymbol{x}}} \boldsymbol{A}(\widehat{\boldsymbol{z}}_{r's'}^{n'}).
\end{aligned} \tag{3.17}$$

Combining (3.14) and (3.17), we approximate the reformulated and projected magnetic dipole operator (2.11) by the computable representation

$$\mathcal{M}_{n'}\boldsymbol{A}(\widehat{\boldsymbol{x}}) = \mathcal{M}_{1,n'}\boldsymbol{A}(\widehat{\boldsymbol{x}}) + \mathcal{M}_{2,n'}\boldsymbol{A}(\widehat{\boldsymbol{x}}). \tag{3.18}$$

Finally, we must choose n' to ensure that $\mathcal{M}_{n'}\boldsymbol{A}$ converges to $\mathcal{M}\boldsymbol{A}$ with spectral accuracy. Following the proof of [16, Theorem 1] it can be shown that if $n' = an + 1$ for some fixed constant $a > 1$, and $n' - n \ge 3$, then for , $s \in \mathbb{N}$, there exists $c_s > 0$, independent of n and n', such that

$$\|(\mathcal{M} - \mathcal{M}_{n'})\boldsymbol{P}_n\|_{\infty,\partial B} \le c_s \frac{1}{n^s} \|\boldsymbol{P}_n\|_{\infty,\partial B} \quad \text{for all } \boldsymbol{P}_n \in \underline{\mathbb{P}}_{n+1}. \tag{3.19}$$

As in [16, Theorem 1], if the conductor surface is smooth then (3.19) holds for , number s. We remark that in our computational algorithm, the density function of the fully discrete magnetic dipole operator is a spherical polynomial of degree at most $n + 1$. For computations in section 5 we set $n' = 2n + 1$.

3.2. Complete algorithm. We are now ready to describe the complete algorithm to solve (2.23), using the various spectrally accurate approximations described

above. Our fully discrete scheme for (2.23) is as follows: compute $\boldsymbol{W}_n \in \underline{\mathbb{Q}}_n$ such that

$$\boldsymbol{W}_n + \underline{\mathcal{O}}_n \mathcal{M}_{n'} \boldsymbol{W}_n = 2\underline{\mathcal{O}}_n \boldsymbol{f}. \tag{3.20}$$

Since $\boldsymbol{W}_n \in \underline{\mathbb{Q}}_n$ can be written as

$$\boldsymbol{W}_n(\widehat{\boldsymbol{x}}) = \mathrm{w}_{003}\boldsymbol{Y}_{00}^{(3)}(\widehat{\boldsymbol{x}}) + \sum_{l=1}^{n}\sum_{|j|\le l}\sum_{\widetilde{k}=1}^{3} \mathrm{w}_{lj\widetilde{k}}\boldsymbol{Y}_{l,j}^{(\widetilde{k})}(\widehat{\boldsymbol{x}}), \quad \widehat{\boldsymbol{x}} \in \partial B, \tag{3.21}$$

the operator equation (3.20) is equivalent to the requirement that the coefficients $\mathrm{w}_{lj\widetilde{k}}$ solve the fully discrete Galerkin system

$$(\boldsymbol{W}_n, \boldsymbol{Y}_{l',j'}^{(k')})_{n+1} + (\mathcal{M}_{n'}\boldsymbol{W}_n, \boldsymbol{Y}_{l',j'}^{(k')})_{n+1} = 2(\boldsymbol{F}, \boldsymbol{Y}_{l',j'}^{(k')})_{n+1}, \quad (l', j', k') \in \mathcal{I}, \tag{3.22}$$

where $\mathcal{I} = \{(0,0,3)\} \cup \{(\widehat{l}, \widehat{j}, \widehat{k}) \ : \ 1 \le \widehat{l} \le n,\ |\widehat{j}| \le \widehat{l},\ \widehat{k} = 1,2,3\}$. In matrix form, $\mathrm{w} = (\mathrm{w}_{lj\widetilde{k}})$ satisfies

$$(\mathrm{I} + \mathrm{M})\mathrm{w} = 2\mathrm{f},$$

and for $(l, j, \widetilde{k}), (l', j', k') \in \mathcal{I}$,

$$\begin{aligned} \mathrm{M}_{l'j'k',lj\widetilde{k}} &= (\mathcal{M}_{n'}\boldsymbol{Y}_{l,j}^{(\widetilde{k})}, \boldsymbol{Y}_{l',j'}^{(k')})_{n+1}, \quad \mathrm{I}_{l'j'k',lj\widetilde{k}} = \delta_{ll'}\delta_{jj'}\delta_{\widetilde{k}k'}, \\ \mathrm{f}_{l'j'k'} &= (\boldsymbol{F}, \boldsymbol{Y}_{l',j'}^{(k')})_{n+1}. \end{aligned} \tag{3.23}$$

Our corresponding approximation to the solution $\boldsymbol{w}$ of (2.3) is then

$$\boldsymbol{w}_n(\boldsymbol{x}) = \boldsymbol{W}_n(\boldsymbol{q}^{-1}(\boldsymbol{x})), \quad \boldsymbol{x} \in \partial D. \tag{3.24}$$

Using (3.19) and following the proof of [16, Theorem 3] (with $\underline{\mathcal{L}}_n$ replaced by $\underline{\mathcal{O}}_n$), it can be shown that for smooth conductors, $\boldsymbol{w}_n$ (and $\boldsymbol{W}_n$) are spectrally accurate approximations to the unique solutions $\boldsymbol{w}$ (and $\boldsymbol{W}$) in the maximum norm: $\|\boldsymbol{w} - \boldsymbol{w}_n\|_{\infty,\partial D} = O(n^{-q})$ (and $\|\boldsymbol{W} - \boldsymbol{W}_n\|_{\infty,\partial B} = O(n^{-q})$) for any positive integer q, provided that the Lebesgue constant of $\underline{\mathcal{O}}_n$ is $O(n^{\alpha})$ for some $\alpha < 1$. The proof of [16, Theorem 3] is based on the known result that the Lebesgue constant of $\underline{\mathcal{L}}_n$ is $O(\sqrt{n})$ [16, 22, 24]. In this work, we conjecture that the Lebesgue constant of $\underline{\mathcal{O}}_n$ is the same as that of $\underline{\mathcal{L}}_n$; a complete proof of this result similar to the theoretical analyses in [22, 24] is beyond the scope of this paper.

3.3. Computer implementation details. As a prerequisite for implementation, special care must be taken to avoid overflow in the evaluation of the basis functions—in particular the associated Legendre values in (3.5) when l is large. We compute directly the normalized associated Legendre values

$$\widehat{P}_l^j(x) = \sqrt{\frac{l-j}{l+j}}P_l^j(x), \qquad 0 \le l \le n,\ 0 \le j \le l,$$

using the relation

$$\widehat{P}_l^j(x) = \sigma_l^j Q_l^j(x)$$

and the recurrences

$$(l-j)Q_l^j(x) = (2l-1)xQ_{l-1}^j(x) - (l+j-1)Q_{l-2}^j(x), \qquad l > j+1,$$
$$\sigma_l^j = \sqrt{\frac{l-j}{l+j}}\sigma_{l-1}^j, \qquad l > j+1,$$

with the seeds

$$\sigma_j^j = 1, \quad \sigma_{j+1}^j = \sqrt{\frac{1}{2j+1}},$$
$$Q_j^j(x) = \eta(j)(1-x^2)^{j/2}, \quad Q_{j+1}^j(x) = x(2l+1)Q_j^j(x), \quad \eta(j) = (2j-1)!!/\sqrt{(2j)!}.$$

Overflow in computing $\eta(j)$ is avoided using the stable expression

$$\eta(j) = \prod_{\tilde{j}=1}^{j} \sqrt{\frac{2\tilde{j}-1}{2\tilde{j}}}.$$

As in [16], we describe an efficient way to set up the $N \times N$ electromagnetic scattering matrix M in only $O(N^{2.5})$ operations, where $N = 3(n+1)^2 - 2$. This complexity can be reduced further by utilizing fast transforms. We also describe a parallel implementation.

Using (3.23), (3.18), (3.17), (3.14), and (3.9), we write

$$\mathrm{M}_{l'j'k',lj\tilde{k}} = \sum_{r=0}^{2n+3}\sum_{s=1}^{n+2} \mu_r^{n+1}\nu_s^{n+1} \sum_{r'=0}^{2n'+1}\sum_{s'=1}^{n'+1} \mu_{r'}^{n'}\nu_{s'}^{n'} \overline{\boldsymbol{Y}_{l',j'}^{(k')}(\widehat{\boldsymbol{x}}_{rs})}^T \Big[\alpha_{s'}^{n'} \mathcal{T}_{\widehat{\boldsymbol{x}}_{rs}} M_1(\widehat{\boldsymbol{n}}, \widehat{\boldsymbol{z}}_{r's'})$$
$$+ \mathcal{T}_{\widehat{\boldsymbol{x}}_{rs}} M_2(\widehat{\boldsymbol{n}}, \widehat{\boldsymbol{z}}_{r's'})\Big] \mathcal{T}_{\widehat{\boldsymbol{x}}_{rs}} \boldsymbol{Y}_{l,j}^{(\tilde{k})}(\widehat{\boldsymbol{z}}_{r's'}), \tag{3.25}$$

where $\widehat{\boldsymbol{x}}_{rs} = \boldsymbol{p}(\theta_s^{n+1}, \phi_r^{n+1})$ and $\widehat{\boldsymbol{z}}_{r's'} = \boldsymbol{p}(\theta_{s'}^{n'}, \phi_{r'}^{n'})$.

Direct computation of M using (3.25) has complexity $O(N^4)$ because M has $O(N^2)$ entries, and each entry requires $O(N^2)$ function evaluations to approximate a double integral over ∂B. The efficient assembly algorithm presented below reduces this complexity to $O(N^{2.5})$. Optimal complexity for the matrix would be $O(N^2)$, but the spectrally accurate nature of the algorithm compensates for the slightly higher complexity.

We write $\boldsymbol{y}_{rs}^{r's'} = T_{\widehat{\boldsymbol{x}}_{rs}}^{-1}\widehat{\boldsymbol{z}}_{r's'}$ so that

$$\mathrm{M}_{l'j'k',lj\tilde{k}} = \sum_{r=0}^{2n+3}\sum_{s=1}^{n+2} \mu_r^{n+1}\nu_s^{n+1} \sum_{r'=0}^{2n'+1}\sum_{s'=1}^{n'+1} \mu_{r'}^{n'}\nu_{s'}^{n'} \overline{\boldsymbol{Y}_{l',j'}^{(k')}(\widehat{\boldsymbol{x}}_{rs})}^T \Big[\alpha_{s'}^{n'} M_1(\widehat{\boldsymbol{x}}_{rs}, \widehat{\boldsymbol{y}}_{rs}^{r's'})$$
$$+ M_2(\widehat{\boldsymbol{x}}_{rs}, \widehat{\boldsymbol{y}}_{rs}^{r's'})\Big] \mathcal{T}_{\widehat{\boldsymbol{x}}_{rs}} \boldsymbol{Y}_{l,j}^{(\tilde{k})}(\widehat{\boldsymbol{z}}_{r's'}).$$

From [11, (12.7.4), p. 341] the rotated vector spherical harmonic $\mathcal{T}_{\widehat{\boldsymbol{x}}_{rs}} \boldsymbol{Y}_{l,j}^{(\tilde{k})}$ can be written as a linear combination of vector spherical harmonics of the same degree. As in [12], the complexity of computing M can be reduced by taking advantage of this property. A stable and efficient representation based on [11, (12.7.4), p. 341] using [12,

p. 224] is

$$\mathcal{T}_{\widehat{\boldsymbol{x}}_{rs}} \boldsymbol{Y}^{(k)}_{l,j} = T^{-1}_{\widehat{\boldsymbol{x}}_{rs}} \sum_{|\widetilde{j}|\le l} F_{sl\widetilde{j}j} e^{i(j-\widetilde{j})\phi_r^{n+1}} \boldsymbol{Y}^{(k)}_{l,\widetilde{j}}, \tag{3.26}$$

$$F_{sl\widetilde{j}j} = e^{i(j-\widetilde{j})\pi/2} \sum_{|\widetilde{m}|\le l} d^{(l)}_{\widetilde{j}\widetilde{m}}(\pi/2) d^{(l)}_{j\widetilde{m}}(\pi/2) e^{i\widetilde{m}\theta_s^{n+1}},$$

where

$$d^{(l)}_{\widetilde{j}j}(\pi/2) = 2^{\widetilde{j}} \left[\frac{(l+\widetilde{j})!(l-\widetilde{j})!}{(l+j)!(l-j)!} \right]^{1/2} P^{(j-\widetilde{j},-j-\widetilde{j})}_{l+\widetilde{j}}(0).$$

For given nonnegative integers a, b and $s \ge 0$, $P^{(a,b)}_s(0)$ is the normalized Jacobi polynomial evaluated at zero,

$$P^{(a,b)}_s(0) = 2^{-s} \sum_{t=0}^{s} (-1)^t \begin{pmatrix} s+a \\ s-t \end{pmatrix} \begin{pmatrix} s+b \\ t \end{pmatrix}.$$

When a or b is negative, $d^{(l)}_{\widetilde{j}j}$ can be computed using the symmetry relations

$$d^{(l)}_{\widetilde{j}j}(\alpha) = (-1)^{\widetilde{j}-j} d^{(l)}_{j\widetilde{j}}(\alpha) = d^{(l)}_{-j-\widetilde{j}}(\alpha) = d^{(l)}_{j\widetilde{j}}(-\alpha).$$

Using (3.26), (3.2), and (3.5), we get

$$\begin{aligned}
&\mathrm{M}_{l'j'k',lj\widetilde{k}} \\
&= \sum_{r=0}^{2n+3} \sum_{s=1}^{n+2} \mu_r^{n+1} \nu_s^{n+1} \sum_{r'=0}^{2n'+1} \sum_{s'=1}^{n'+1} \mu_{r'}^{n'} \nu_{s'}^{n'} \sum_{t'=1}^{3} \alpha^{(k',t')}_{l',j'}(\theta_s^{n+1}) e^{-ij'\phi_r^{n+1}} \boldsymbol{v}^{(t')}(\theta_s^{n+1}, \phi_r^{n+1})^T \\
&\quad \left[\alpha^{n'}_{s'} M_1(\widehat{\boldsymbol{x}}_{rs}, \widehat{\boldsymbol{y}}^{r's'}_{rs}) + M_2(\widehat{\boldsymbol{x}}_{rs}, \widehat{\boldsymbol{y}}^{r's'}_{rs}) \right] \sum_{|\widetilde{j}|\le l} F_{sl\widetilde{j}j} e^{i(j-\widetilde{j})\phi_r^{n+1}} \\
&\quad \sum_{t=1}^{3} \alpha^{(\widetilde{k},t)}_{l,\widetilde{j}}(\theta^{n'}_{s'}) e^{i\widetilde{j}\phi^{n'}_{r'}} T^{-1}_{\widehat{\boldsymbol{x}}_{rs}} \boldsymbol{v}^{(t)}(\theta^{n'}_{s'}, \phi^{n'}_{r'}).
\end{aligned}$$

The matrix M, which has $O(N^2) = O(n^4)$ entries, can be computed by successively computing the following arrays. Each array is a sum with $O(n)$ terms and depends on four labels (ignoring $\widetilde{k}$ and k'). Thus M can be computed in $O(n^5) = O(N^{2.5})$ operations.

$$\mathrm{E}_{srs'\widetilde{j}t} = \sum_{r'=0}^{2n'+1} \mu^{n'}_{r'} e^{i\widetilde{j}\phi^{n'}_{r'}} \left[\alpha^{n'}_{s'} M_1(\widehat{\boldsymbol{x}}_{rs}, \widehat{\boldsymbol{y}}^{r's'}_{rs}) + M_2(\widehat{\boldsymbol{x}}_{rs}, \widehat{\boldsymbol{y}}^{r's'}_{rs}) \right] T^{-1}_{\widehat{\boldsymbol{x}}_{rs}} \boldsymbol{v}^{(t)}(\theta^{n'}_{s'}, \phi^{n'}_{r'}),$$

$$\mathrm{D}_{srl\widetilde{j}\widetilde{k}} = \sum_{s'=1}^{n'+1} \sum_{t=1}^{3} \nu^{n'}_{s'} \alpha^{(\widetilde{k},t)}_{l,\widetilde{j}}(\theta^{n'}_{s'}) \mathrm{E}_{srs'\widetilde{j}t}, \quad \mathrm{C}_{srlj\widetilde{k}} = \sum_{|\widetilde{j}|\le l} F_{sl\widetilde{j}j} e^{i(j-\widetilde{j})\phi_r^{n+1}} \mathrm{D}_{srl\widetilde{j}\widetilde{k}},$$

$$\mathrm{B}_{sj't'lj\widetilde{k}} = \sum_{r=0}^{2n+3} \mu_r^{n+1} e^{-ij'\phi_r^{n+1}} \boldsymbol{v}^{(t')}(\theta_s^{n+1}, \phi_s^{n+1})^T \mathrm{C}_{srlj\widetilde{k}},$$

$$\mathrm{M}_{l'j'k',lj\widetilde{k}} = \sum_{s=1}^{n+2} \sum_{t'=1}^{3} \nu_s^{n+1} \alpha^{(k',t')}_{l',j'}(\theta_s^{n+1}) \mathrm{B}_{sj't'lj\widetilde{k}}.$$

Once E has been used to compute D, it is discarded. Similarly, D is discarded after it has been used to compute C, and so on. The fast Fourier transform (FFT) can be used to speed up the computation of the arrays above, and for calculating f and F in (3.23) and (3.26), respectively. As we shall demonstrate in the numerical experiments section, $5 \le n \le 125$ is sufficient for low to medium frequency problems because of the high-order spectral accuracy of our algorithm. Hence we can use direct or iterative solves in a single or multiprocessor environment.

For a cluster computing-type environment (with multiple processors and limited RAM for each processor), we consider a parallel implementation of the above algorithm. For a cluster computer implementation with $n_{\rm proc}$ free CPUs, we partition the s index set $\{1, \ldots, n+2\}$ into $n_{\rm proc}$ disjoint sets S_p for $p = 1, \ldots, n_{\rm proc}$. The array $\mathrm{B}_{sj't'lj\widetilde{k}}$ for $s \in S_p$ is computed and stored on processing node p.

4. High-order RCS and exterior field approximations. In this section, we follow a variant of the approach in [16] to describe a fully discrete high-order approximation to the monostatic and bistatic RCS of the perfect conductor D, and the exterior electromagnetic fields. We use the spectrally accurate numerical solution $\boldsymbol{W}_n$ of (3.21)–(3.23) for this purpose.

4.1. Spectrally accurate RCS computations. The monostatic and bistatic RCS measurements in (1.2) and (1.7) require high-order approximations of the electric far field. Using (2.6), the electric far field corresponding to the unique solution $\boldsymbol{w} = \mathcal{P}\boldsymbol{w}$ of (2.3) can be written as

$$\begin{aligned} \boldsymbol{E}_\infty(\widehat{\boldsymbol{x}}) &= \frac{ik}{4\pi} \int_{\partial D} e^{-ik\widehat{\boldsymbol{x}}\cdot\boldsymbol{y}} \widehat{\boldsymbol{x}} \times \mathcal{P}(\boldsymbol{y}) \boldsymbol{w}(\boldsymbol{y})\, ds(\boldsymbol{y}) \\ &= \int_{\partial B} \widehat{M}_{\widehat{\boldsymbol{x}}}(\widehat{\boldsymbol{y}}) \boldsymbol{W}(\widehat{\boldsymbol{y}})\, ds(\widehat{\boldsymbol{y}}), \quad \widehat{\boldsymbol{x}} \in \partial B, \end{aligned} \tag{4.1}$$

where for each fixed $\widehat{\boldsymbol{x}} \in \partial B$, the smooth function $\widehat{M}_{\widehat{\boldsymbol{x}}}$ is defined by

$$\widehat{M}_{\widehat{\boldsymbol{x}}}(\widehat{\boldsymbol{y}}) \boldsymbol{W}(\widehat{\boldsymbol{y}}) = \frac{ik}{4\pi} J(\widehat{\boldsymbol{y}}) e^{-ik\widehat{\boldsymbol{x}}\cdot\boldsymbol{q}(\widehat{\boldsymbol{y}})} \widehat{\boldsymbol{x}} \times \mathcal{P}(\boldsymbol{q}(\widehat{\boldsymbol{y}})) \boldsymbol{W}(\widehat{\boldsymbol{y}}), \qquad \widehat{\boldsymbol{y}} \in \partial B, \tag{4.2}$$

and $\boldsymbol{W}$ is the solution of (2.23).

Using the approximation $\boldsymbol{W}_n$ to $\boldsymbol{W}$ given by (3.21)–(3.23), our fully discrete approximation $\boldsymbol{E}_{n,\infty}(\widehat{\boldsymbol{x}})$ to $\boldsymbol{E}_\infty(\widehat{\boldsymbol{x}})$ for $\widehat{\boldsymbol{x}} \in \partial B$ is given by

$$\boldsymbol{E}_{n,\infty}(\widehat{\boldsymbol{x}}) = \int_{\partial B} \underline{\mathcal{O}}_n \left\{ \widehat{M}_{\widehat{\boldsymbol{x}}}(\cdot) \boldsymbol{W}_n(\cdot) \right\} (\widehat{\boldsymbol{y}})\, ds(\widehat{\boldsymbol{y}}) \tag{4.3}$$

$$= \sum_{l=0}^{n} \sum_{|j|\le l} \sum_{\widetilde{k}=1}^{3} \mathrm{w}_{lj\widetilde{k}} \int_{\partial B} \underline{\mathcal{O}}_n \left\{ \widehat{M}_{\widehat{\boldsymbol{x}}}(\cdot) \boldsymbol{Y}_{l,j}^{(\widetilde{k})}(\cdot) \right\} (\widehat{\boldsymbol{y}})\, ds(\widehat{\boldsymbol{y}}), \tag{4.4}$$

where, for convenience, we set $\mathrm{w}_{001} = \mathrm{w}_{002} = 0$. Using (3.10) and (3.7),

$$\boldsymbol{E}_{n,\infty}(\widehat{\boldsymbol{x}}) = \sum_{l=0}^{n} \sum_{|j|\le l} \sum_{\widetilde{k}=1}^{3} \mathrm{w}_{lj\widetilde{k}} \sum_{r=0}^{2n+3} \sum_{s=1}^{n+2} \mu_r^{n+1} \nu_s^{n+1} A_{rs} \widehat{M}_{\widehat{\boldsymbol{x}}}(\widehat{\boldsymbol{z}}_{rs}^{n+1}) \boldsymbol{Y}_{l,j}^{(\widetilde{k})}(\widehat{\boldsymbol{z}}_{rs}^{n+1}),$$

where

$$A_{rs} = \sum_{l'=0}^{n} \sum_{|j'|\le l} \sum_{k'=1}^{3} \left(\int_{\partial B} \boldsymbol{Y}_{l',j'}^{(k')}(\widehat{\boldsymbol{y}})\, ds(\widehat{\boldsymbol{y}}) \right) \overline{\boldsymbol{Y}_{l',j'}^{(k')}(\widehat{\boldsymbol{z}}_{rs}^{n+1})}^{T}.$$

Only the integrals $(l', j', k') \in \{(0,0,3), (1,\pm 1,1), (1,\pm 1,2)\}$ are nonzero, and they can be easily computed analytically. Proceeding as in [16], we have

$$\boldsymbol{E}_{n,\infty}(\widehat{\boldsymbol{x}}) = \sum_{l=0}^{n} \sum_{|j|\le l} \sum_{\tilde{k}=1}^{3} \mathrm{w}_{lj\tilde{k}} \widehat{\boldsymbol{M}}_{lj\tilde{k}}^{n+1}(\widehat{\boldsymbol{x}}),$$

$$\widehat{\boldsymbol{M}}_{lj\tilde{k}}^{n+1}(\widehat{\boldsymbol{x}}) = \sum_{r=0}^{2n+3} \sum_{s=1}^{n+2} \mu_r^{n+1} \nu_s^{n+1} A_{rs} \widehat{M}_{\widehat{\boldsymbol{x}}}(\widehat{\boldsymbol{z}}_{rs}^{n+1}) \boldsymbol{Y}_{l,j}^{(\tilde{k})}(\widehat{\boldsymbol{z}}_{rs}^{n+1}).$$

The quantities $\widehat{\boldsymbol{M}}_{lj\tilde{k}}^{n+1}$ do not depend on $\boldsymbol{W}_n$ and can be precomputed independently. As in [16, Theorem 3], assuming the conjecture of section 3.2, we can prove that $\boldsymbol{E}_{n,\infty}$ is a spectrally accurate approximation to $\boldsymbol{E}_\infty$.

The approximation σ_n^{dB} to the RCS σ^{dB} is obtained by replacing $\boldsymbol{E}_\infty$ by $\boldsymbol{E}_{n,\infty}$ in (1.2) and (1.7). That is, we compute the spectrally accurate approximation to the RCS of the perfect conductor $D \subseteq \mathbb{R}^3$ using the fully discrete formulas

$$\sigma_n^{\mathrm{dB}}(\widehat{\boldsymbol{x}}; \widehat{\boldsymbol{d}}_0) = 10 \log_{10} \left(4\pi \left| \boldsymbol{E}_{n,\infty}(\widehat{\boldsymbol{x}}) \right|^2 / k^2 \right),$$

$$\sigma_n^{\mathrm{dB}}(\widehat{\boldsymbol{x}}, \widehat{\boldsymbol{p}}_1; \widehat{\boldsymbol{d}}_0) = 10 \log_{10} \left(4\pi \left| \boldsymbol{E}_{n,\infty}(\widehat{\boldsymbol{x}}) \cdot \widehat{\boldsymbol{p}}_1 \right|^2 / k^2 \right), \quad \widehat{\boldsymbol{x}} \in \partial B. \tag{4.5}$$

For each $\widehat{\boldsymbol{x}} \in \partial B$, direct computation of $\boldsymbol{E}_{n,\infty}(\widehat{\boldsymbol{x}})$ has complexity $O(n^4)$ because $\widehat{\boldsymbol{M}}^{n+1}$ has $O(n^2)$ entries, and each entry requires $O(n^2)$ function evaluations to approximate an integral over ∂B. However, $\boldsymbol{E}_{n,\infty}(\widehat{\boldsymbol{x}})$ can be computed in $O(n^3)$ operations by successively computing the quantities below to obtain $\widehat{\boldsymbol{M}}^{n+1}$:

$$\widehat{\boldsymbol{B}}_{sjt}^{n+1}(\widehat{\boldsymbol{x}}) = \sum_{r=0}^{2n+3} \mu_r^{n+1} A_{rs} \widehat{M}_{\widehat{\boldsymbol{x}}}(\widehat{\boldsymbol{z}}_{rs}^{n+1}) e^{ij\phi_r^{n+1}} \boldsymbol{v}^t(\theta_s^{n+1}, \phi_r^{n+1}),$$

$$\widehat{\boldsymbol{M}}_{lj\tilde{k}}^{n+1}(\widehat{\boldsymbol{x}}) = \sum_{s=1}^{n+2} \sum_{t=1}^{3} \nu_s^{n+1} \alpha_{l,j}^{(\tilde{k},t)}(\theta_s^{n+1}) \widehat{\boldsymbol{B}}_{sjt}^{n+1}(\widehat{\boldsymbol{x}}).$$

The computation of $\widehat{\boldsymbol{B}}^{n+1}$ can be accelerated using the FFT. Once $\widehat{\boldsymbol{M}}^{n+1}(\widehat{\boldsymbol{x}})$ has been evaluated, it can be used for computation of the far field induced by many incident directions in the monostatic RCS approximations.

4.2. Spectrally accurate exterior field computations. We compute a fully discrete approximation to the electric field $\boldsymbol{E}(\boldsymbol{x})$ in the exterior region, given by (2.4), using the approach similar to the far field computation in section 4.1, but we replace $\widehat{M}_{\widehat{\boldsymbol{x}}}$ by $\widetilde{M}_{\boldsymbol{x}}$, where

$$\widetilde{M}_{\boldsymbol{x}}(\widehat{\boldsymbol{y}}) \boldsymbol{W}(\widehat{\boldsymbol{y}}) = J(\widehat{\boldsymbol{y}}) \, \mathbf{grad}_{\boldsymbol{x}} \Phi(\boldsymbol{x}, \boldsymbol{q}(\widehat{\boldsymbol{y}})) \times \mathcal{P}(\boldsymbol{q}(\widehat{\boldsymbol{y}})) \boldsymbol{W}(\widehat{\boldsymbol{y}}), \qquad \widehat{\boldsymbol{y}} \in \partial B. \tag{4.6}$$

We have used (2.4) and the identity $\mathbf{curl}\ \{\phi \boldsymbol{v}\} = \phi\ \mathbf{curl}\ \boldsymbol{v} + \mathbf{grad}\, \phi \times \boldsymbol{v}$, to derive $\widetilde{M}_{\boldsymbol{x}}$. The fully discrete approximation $\boldsymbol{E}_n(\boldsymbol{x})$ to $\boldsymbol{E}(\boldsymbol{x})$ for $\boldsymbol{x} \in \mathbb{R}^3 \setminus \overline{D}$ is then

$$\boldsymbol{E}_n(\boldsymbol{x}) = \sum_{l=0}^{n} \sum_{|j|\le l} \sum_{\tilde{k}=1}^{3} \mathrm{w}_{lj\tilde{k}} \widetilde{\boldsymbol{M}}_{lj\tilde{k}}^{n+1}(\boldsymbol{x}),$$

$$\widetilde{\boldsymbol{M}}_{lj\tilde{k}}^{n+1}(\boldsymbol{x}) = \sum_{r=0}^{2n+3} \sum_{s=1}^{n+2} \mu_r^{n+1} \nu_s^{n+1} A_{rs} \widetilde{M}_{\boldsymbol{x}}(\widehat{\boldsymbol{z}}_{rs}^{n+1}) \boldsymbol{Y}_{l,j}^{(\tilde{k})}(\widehat{\boldsymbol{z}}_{rs}^{n+1}).$$

These computations can be accelerated as in section 4.1.

5. Simulated RCS and exterior electromagnetic fields. We consider the RCS simulation problem (1.1)–(1.7) for various test obstacles including the benchmark radar target cone-sphere [30] and a biologically interesting erythrocyte in standard oblate and in oval of Cassini prolate form [19, 31]. For an incident wave with incident polar and azimuthal angles θ and ϕ, the direction of the plane wave is given by $\widehat{\boldsymbol{d}}_0 = -(\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)^T$. The incident wave is then [illegible] when $\widehat{\boldsymbol{p}}_0 = (\cos\theta\cos\phi, \cos\theta\sin\phi, -\sin\theta)^T$ and [illegible] when $\widehat{\boldsymbol{p}}_0 = (-\sin\phi, \cos\phi, 0)^T$ [27, pp. 8–9]. Throughout this section we denote the RCS $\sigma_n^{\mathrm{dB}}(\boldsymbol{p}(\theta,\phi); \widehat{\boldsymbol{d}}_0)$ in (4.5) by $\sigma_n^{\mathrm{dB}}(\theta,\phi)$.

Given the spatial component of an electric field, denoted by $\boldsymbol{E}$, in $\mathbb{R}^3 \setminus \overline{D}$, the corresponding time harmonic electric field is given by

$$\boldsymbol{\mathcal{E}}(\boldsymbol{x}, t) = \frac{1}{\sqrt{\epsilon_0}} \mathrm{Re}\left\{\boldsymbol{E}(\boldsymbol{x}) e^{-2\pi i \omega t}\right\}, \quad \boldsymbol{x} \in \mathbb{R}^3 \setminus \overline{D}.$$

In our numerical experiments we simulate the monostatic and bistatic RCS and exterior electromagnetic fields induced by smooth convex and nonconvex obstacles, and by the cone-sphere, a benchmark radar target that has conical singularities. The smooth obstacles are a sphere, **sph**([illegible]); a beehive shape, **hive**([illegible]); a Cassini shape in prolate form, **cas**([illegible]); an erythrocyte, **eryth**([illegible]); and a submarine shape, **sub**([illegible]). The nonsmooth benchmark radar target is the perfectly conducting cone-sphere, **cone_sphere**([illegible]). Here [illegible] is the diameter of the obstacle. These obstacles are shown in Figures 1–3. The erythrocyte is constructed according to [31, equation (3)]. As described in [19, 31], the biconcave cross section of the erythrocyte has the shape of the Cassini obstacle in two-dimensional form. The cone-sphere is constructed according to [30]. The hull of the submarine is constructed, as in [18], from an ellipsoid with aspect ratio 10:1:1. The cone-sphere and submarine have higher aspect ratios than the other obstacles considered.

For a fixed positive integer n, the linear system we need to solve to achieve the reported accuracy has $3(n+1)^2 - 2$ unknowns in general. For spherical scattering the linear system has $2(n+1)^2 - 2$ unknowns. The maximum n we used in all the experiments is only 125. Thus we are able to present results obtained with both direct and iterative linear system solvers. In the case of iterative solvers, we use GMRES preconditioned with the inverse of the diagonal part of our matrix. For our experiments, we used a cluster with 2GHz Dual-core Opteron (DcOp) processors. DcOp CPU time mentioned in this paper is for the combined set up, GMRES solve, and error calculation phases of the algorithm.

In the case of plane wave scattering by a sphere, an analytical representation of the scattered field is given by the Mie series. For the sphere case our choice of frequency for the incident wave (and hence of the electromagnetic scattering problem) is motivated by those considered in established electromagnetic scattering algorithms Fastscat [5] and FISC [25]. To compare the efficiency of our algorithm with [5, 25], in the tabulated results we use the RMS error (denoted by ϵ_{dB}) in the bistatic RCS, measured in decibels per square meter. The RMS error is defined as

$$\epsilon_{\mathrm{dB}} = \left\{ \frac{1}{4\pi} \int_{\partial B} \left[\sigma^{\mathrm{dB}}(\widehat{\boldsymbol{x}}) - \sigma_n^{\mathrm{dB}}(\widehat{\boldsymbol{x}})\right]^2 ds(\widehat{\boldsymbol{x}}) \right\}^{1/2},$$

where σ^{dB} and σ_n^{dB} are, respectively, defined by (1.2)–(1.7) and (4.5). We numerically approximate the integral in ϵ_{dB} using over 1300 quadrature points on ∂B.

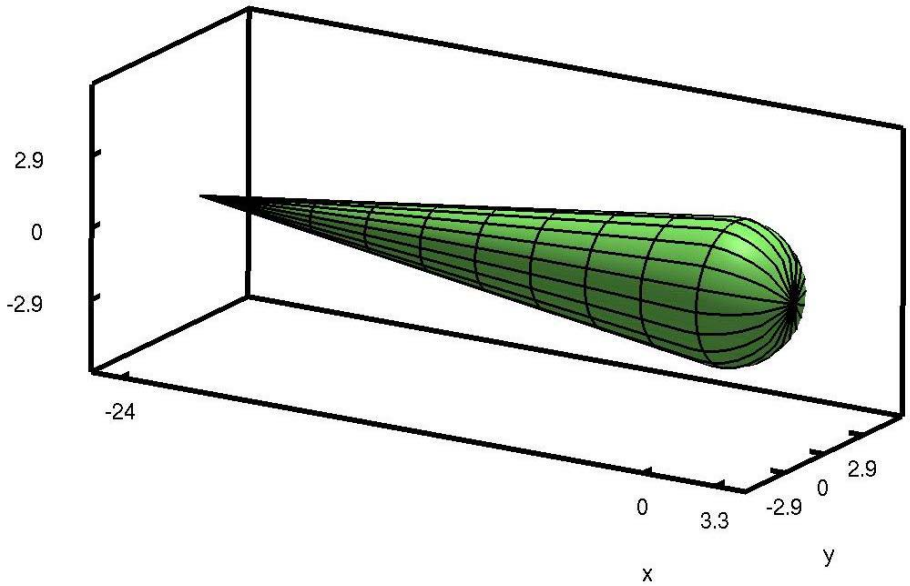


FIG. 1. *Cone-sphere.*

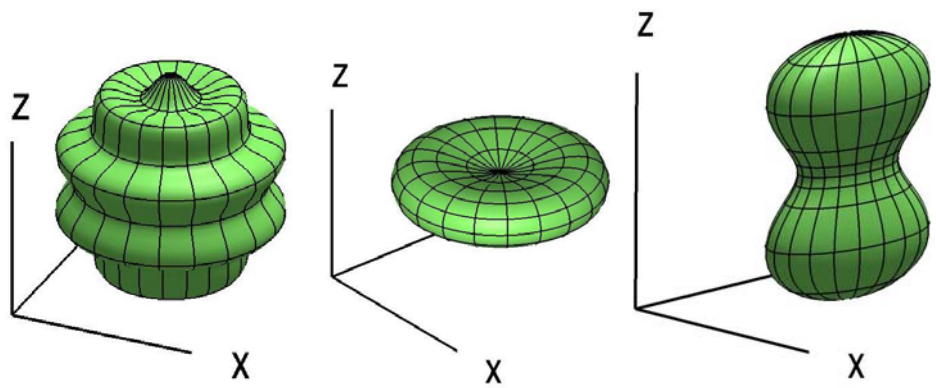


FIG. 2. *Hive, Erythrocyte, and Cassini.*

To demonstrate the spectral convergence of our method for plane wave scattering by nonspherical perfect conductors, we tabulate the monostatic or bistatic RCS at various receiver direction angles. Similar convergence studies for acoustic scattering in two and three dimensions are, respectively, in [8, p. 72] and [12, p. 234].

For the erythrocyte an incident wave with wavelength 632.8 nm (or one tenth the diameter of the erythrocyte) is of considerable recent interest [19, 31]. Hence, for RCS computations of a fixed target, we choose two incident waves: one with wavelength equal to the diameter of the target, and the other with wavelength one tenth the diameter of the target. We observed similar performance for various other targets considered in [16].

The results in Tables 1–15 demonstrate the spectral accuracy of our algorithm.

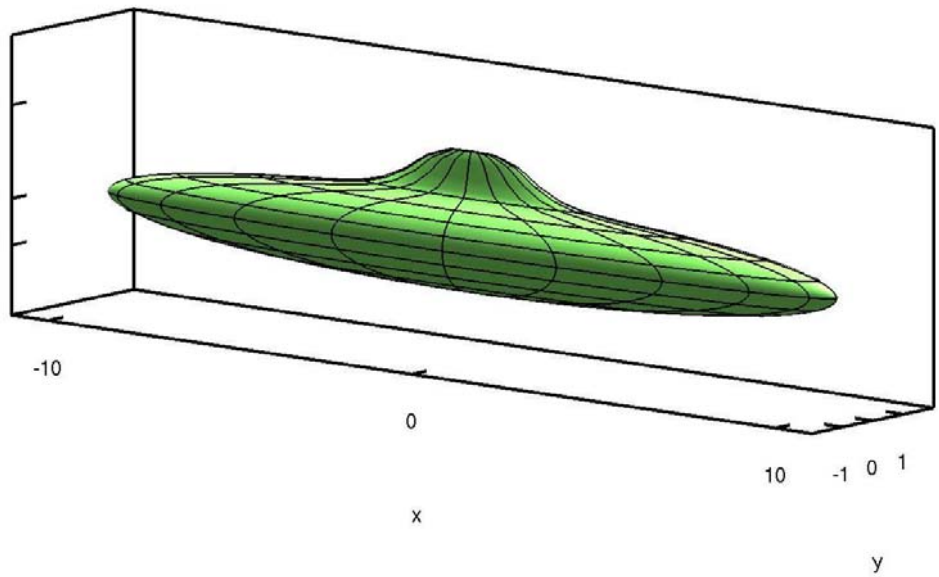


FIG. 3. *Submarine.*

As would be expected, the spherical perfect conductor requires fewer unknowns than the high aspect-ratio and nonconvex obstacles. In general, the number of unknowns depends on the shape and electromagnetic size $\quad /\lambda$ of the conductors, where $\lambda = 2\pi/k = c/\omega$ is the wavelength. The cone-sphere possesses a conical singularity. This nonsmoothness in the object surface precludes spectral accuracy, but our efficient algorithm is able to produce visually accurate results using only a few tens of thousands of unknowns.

We conclude this work by presenting visualizations of RCS values and exterior fields of all the described obstacles simulated using the medium frequency electromagnetic scattering model described in the introduction (see Figures 4–9). The left colorbar values in the figures correspond to the intensity of the surface current. The RCS plots (for all geometries considered in this paper) are given in polar form. For comparison of the experimental and simulated RCS for the cone-sphere benchmark target in [30], we also give the cone-sphere RCS plot in Cartesian form. (All figures in this paper are also at http://www.mines.edu/∼mganesh/ GH_pap2_figures/ GH_RCS_figures.html, to facilitate the viewing of larger pictures.)

TABLE 1
Convergence of bistatic RCS for $\mathbf{sph}(\lambda)$ *at* 299.8MHz *for various observed angles (H-H polarization).*

n	$\sigma_n^{\mathrm{dB}}(\pi/2, 0.0°)$	$\sigma_n^{\mathrm{dB}}(\pi/2, 59.9°)$	$\sigma_n^{\mathrm{dB}}(\pi/2, 120.1°)$	$\sigma_n^{\mathrm{dB}}(\pi/2, 180.0°)$
5	-0.236515555	-2.364344683	-2.364344683	-0.236515555
15	-2.261565541	1.633695466	4.172258451	9.660449813
25	-2.261565541	1.633695466	4.172258451	9.660449813

TABLE 2
Convergence of bistatic RCS for **sph**(10λ) *at* 3.0GHz *for various observed angles (H-H polarization).*

n	$\sigma_n^{\mathrm{dB}}(\pi/2, 0.0^\circ)$	$\sigma_n^{\mathrm{dB}}(\pi/2, 59.9^\circ)$	$\sigma_n^{\mathrm{dB}}(\pi/2, 120.1^\circ)$	$\sigma_n^{\mathrm{dB}}(\pi/2, 180.0^\circ)$
25	14.46493344	6.810093006	6.773474395	23.32394310
35	-1.071614837	-1.100478363	-1.356902718	28.98841095
45	-0.963851816	-1.146497199	-1.226580441	28.98854327
55	-0.963851776	-1.146497197	-1.226580462	28.98854327

TABLE 3
Convergence of bistatic RCS for **hive**(λ) *at* 136.3MHz *for various observed angles (H-H polarization).*

n	$\sigma_n^{\mathrm{dB}}(\pi/2, 0.0^\circ)$	$\sigma_n^{\mathrm{dB}}(\pi/2, 59.9^\circ)$	$\sigma_n^{\mathrm{dB}}(\pi/2, 120.1^\circ)$	$\sigma_n^{\mathrm{dB}}(\pi/2, 180.0^\circ)$
45	2.288774212	7.726946931	11.02191340	15.47284506
55	2.289516548	7.727529734	11.02241885	15.47305847
65	2.289501331	7.727506252	11.02243226	15.47305073
75	2.289525690	7.727521530	11.02244076	15.47305490

TABLE 4
Convergence of bistatic RCS for **hive**(10λ) *at* 1.4GHz *for various observed angles (H-H polarization).*

n	$\sigma_n^{\mathrm{dB}}(\pi/2, 0.0^\circ)$	$\sigma_n^{\mathrm{dB}}(\pi/2, 59.9^\circ)$	$\sigma_n^{\mathrm{dB}}(\pi/2, 120.1^\circ)$	$\sigma_n^{\mathrm{dB}}(\pi/2, 180.0^\circ)$
75	12.21545719	0.995924035	1.630599480	34.71680622
85	12.21598424	0.994537745	1.629710655	34.71678624
95	12.21595582	0.994581511	1.629615318	34.71679053
105	12.21595283	0.994589476	1.629597682	34.71679052

TABLE 5
Convergence of bistatic RCS for **cas**(λ) *at* 299.8MHz *for various observed angles (H-H polarization).*

n	$\sigma_n^{\mathrm{dB}}(\pi/2, 0.0^\circ)$	$\sigma_n^{\mathrm{dB}}(\pi/2, 59.9^\circ)$	$\sigma_n^{\mathrm{dB}}(\pi/2, 120.1^\circ)$	$\sigma_n^{\mathrm{dB}}(\pi/2, 180.0^\circ)$
15	5.943260950	2.430092594	2.390502200	8.643517631
25	5.943542313	2.430600826	2.391709693	8.644488008
35	5.943543764	2.430596856	2.391702443	8.644485832
45	5.943543753	2.430596874	2.391702493	8.644485855

TABLE 6
Convergence of bistatic RCS for **cas**(10λ) *at* 3.0GHz *for various observed angles (H-H polarization).*

n	$\sigma_n^{\mathrm{dB}}(\pi/2, 0.0^\circ)$	$\sigma_n^{\mathrm{dB}}(\pi/2, 59.9^\circ)$	$\sigma_n^{\mathrm{dB}}(\pi/2, 120.1^\circ)$	$\sigma_n^{\mathrm{dB}}(\pi/2, 180.0^\circ)$
75	5.254445300	-2.943734543	-1.852919510	27.01202739
85	5.254455451	-2.946501042	-1.852890342	27.01202909
95	5.254455694	-2.946503971	-1.852892016	27.01202911
105	5.254455695	-2.946503957	-1.852891993	27.01202911

TABLE 7
Convergence of bistatic RCS for **eryth**(λ) *at* 47.6THz *for various observed angles (V-V polarization).*

n	$\sigma_n^{\mathrm{dB}}(0.0^\circ, 0)$	$\sigma_n^{\mathrm{dB}}(59.9^\circ, 0)$	$\sigma_n^{\mathrm{dB}}(120.1^\circ, 0)$	$\sigma_n^{\mathrm{dB}}(180.0^\circ, 0)$
35	-95.26308767	-104.7283471	-104.7901000	-93.60662218
45	-95.26308773	-104.7283471	-104.7900999	-93.60662217
55	-95.26308774	-104.7283471	-104.7900999	-93.60662217

TABLE 8
Convergence of bistatic RCS for **eryth**(10λ) *at* 475.9THz *for various observed angles (V-V polarization).*

n	$\sigma_n^{\mathrm{dB}}(0.0^\circ, 0)$	$\sigma_n^{\mathrm{dB}}(59.9^\circ, 0)$	$\sigma_n^{\mathrm{dB}}(120.1^\circ, 0)$	$\sigma_n^{\mathrm{dB}}(180.0^\circ, 0)$
55	-84.15631909	-107.9716243	-108.5694295	-75.08585121
65	-84.15631929	-107.9716914	-108.5694124	-75.08585126
75	-84.15631929	-107.9716925	-108.5694122	-75.08585127
85	-84.15631929	-107.9716925	-108.5694121	-75.08585127

TABLE 9
Convergence of bistatic RCS for **sub**(λ) *at* 15.0MHz *for various observed angles (H-H polarization).*

n	$\sigma_n^{\mathrm{dB}}(\pi/2, 0.0^\circ)$	$\sigma_n^{\mathrm{dB}}(\pi/2, 59.9^\circ)$	$\sigma_n^{\mathrm{dB}}(\pi/2, 120.1^\circ)$	$\sigma_n^{\mathrm{dB}}(\pi/2, 180.0^\circ)$
65	-9.373283806	-21.40500784	-1.314272969	-21.59519924
75	-9.373250028	-21.40486747	-1.314287529	-21.59504994
85	-9.373233746	-21.40481200	-1.314291518	-21.59501892
95	-9.373225148	-21.40478591	-1.314292190	-21.59502340

TABLE 10
Convergence of bistatic RCS for **sub**(10λ) *at* 149.9MHz *for various observed angles (H-H polarization).*

n	$\sigma_n^{\mathrm{dB}}(\pi/2, 0.0^\circ)$	$\sigma_n^{\mathrm{dB}}(\pi/2, 59.9^\circ)$	$\sigma_n^{\mathrm{dB}}(\pi/2, 120.1^\circ)$	$\sigma_n^{\mathrm{dB}}(\pi/2, 180.0^\circ)$
95	-7.383375719	-5.795755529	3.729626885	11.10711087
105	-7.383132611	-5.795728691	3.729627319	11.10710545
115	-7.383109088	-5.795729699	3.729618894	11.10710286
125	-7.383096050	-5.795728967	3.729613469	11.10710325

TABLE 11
Convergence of monostatic RCS for **cone_sph**(λ) *at* 435.10MHz *for various observed angles (V-V polarization).*

n	$\sigma_n^{\mathrm{dB}}(\pi/2, 0.0^\circ)$	$\sigma_n^{\mathrm{dB}}(\pi/2, 59.9^\circ)$	$\sigma_n^{\mathrm{dB}}(\pi/2, 120.1^\circ)$	$\sigma_n^{\mathrm{dB}}(\pi/2, 180.0^\circ)$
65	-19.88430237	-15.40047977	-15.75751788	-19.98539600
75	-19.80017082	-15.37370602	-15.72647896	-19.87851607
85	-19.73890236	-15.35379602	-15.70245764	-19.79750201
95	-19.68627108	-15.33603751	-15.68130174	-19.72927847

TABLE 12
Convergence of monostatic RCS for **cone_sph**(10λ) *at* 4.35GHz *for various observed angles (V-V polarization).*

n	$\sigma_n^{\mathrm{dB}}(\pi/2, 0.0^\circ)$	$\sigma_n^{\mathrm{dB}}(\pi/2, 59.9^\circ)$	$\sigma_n^{\mathrm{dB}}(\pi/2, 120.1^\circ)$	$\sigma_n^{\mathrm{dB}}(\pi/2, 180.0^\circ)$
95	-19.24152606	-15.95918276	-30.14099402	-29.16753533
105	-19.44876757	-15.98494703	-30.17249637	-29.38451260
115	-19.47098794	-16.03618277	-30.28330884	-28.94057167
125	-19.43245611	-16.08937090	-30.41726902	-28.72183348

TABLE 13
Performance of FastScat, FISC [25, p. 28] *and GH (present). Scattering of a plane wave by a sphere of diameter* 5.4λ.

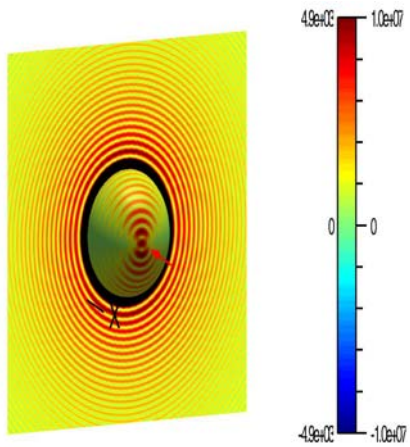
Algorithm	Unknowns	Computer/CPU	CPU time	ϵ_{dB}
FastScat (Nystrom)	5400	Sparc 10	33688 secs	9.7e-02
FastScat (Galerkin)	5400	Sparc 10	74956 secs	2.1e-02
FISC (Galerkin)	5880	Sparc 10	37205 secs	7.2e-01
GH	1350 ($n = 25$)	$1 \times$ DcOp	39 secs	1.1e-06

TABLE 14
Performance of FISC [25, p. 28] *and GH (present). Scattering of a plane wave by a sphere of diameter* 24λ.

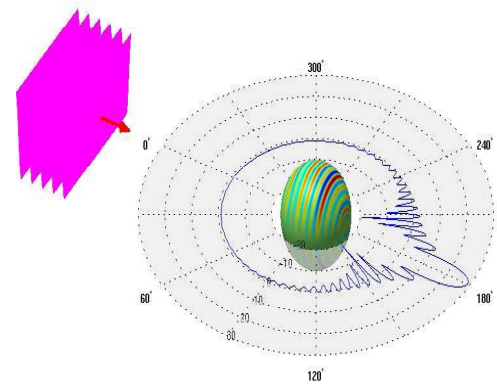
Algorithm	Unknowns	Computer/CPU	CPU time	ϵ_{dB}
FISC	602112	SGI Power Challenge R8000	12 hours	3.0e-01
GH	13120 ($n = 80$)	$2 \times$ DcOp	0.80 hours	5.7e-02
GH	13120 ($n = 80$)	$4 \times$ DcOp	0.48 hours	5.7e-02
GH	15840 ($n = 88$)	$4 \times$ DcOp	0.80 hours	1.5e-05

TABLE 15
CPU time for computing monostatic and bistatic RCS with direct solves for 1202 *sampling directions.*

Geometry	Unknowns	Type	CPU	CPU time
eryth(λ)	6348 ($n = 45$)	Bistatic	$2 \times$ DcOp	8.30 mins
eryth(λ)	6348 ($n = 45$)	Monostatic	$2 \times$ DcOp	12.22 mins
eryth(10λ)	22188 ($n = 85$)	Bistatic	$8 \times$ DcOp	1.13 hours
eryth(10λ)	22188 ($n = 85$)	Monostatic	$8 \times$ DcOp	2.11 hours

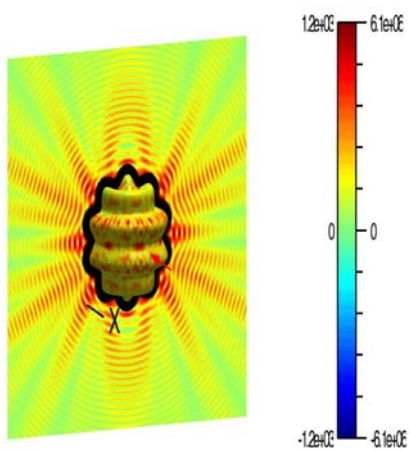


Scattered electric field $|\boldsymbol{\mathcal{E}}(\cdot, t)|$ and surface current $|\boldsymbol{w}(\cdot)|$

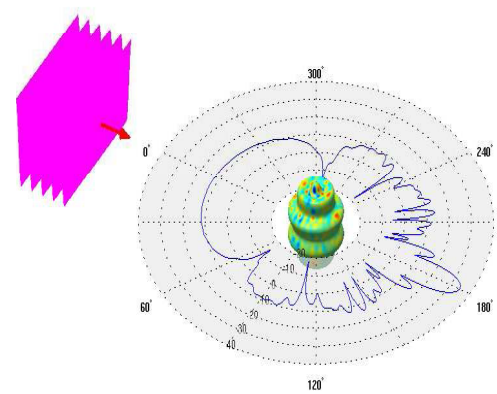


Bistatic RCS

FIG. 4. *Visualizations of the scattered field, surface current, and bistatic RCS induced by a horizontally polarized plane wave impinging on* $\mathbf{sph}(10\lambda)$ *at* 3.0GHz *with* $\widehat{\boldsymbol{d}}_0 = (-1, 0, 0)^T$.

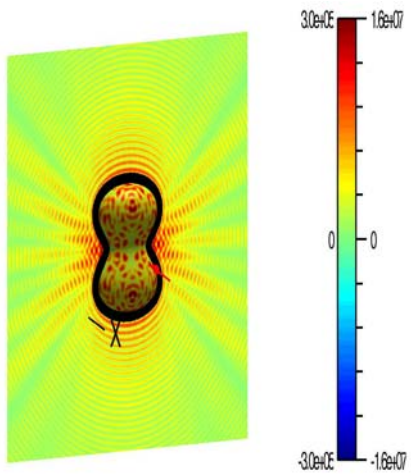


Scattered electric field $|\boldsymbol{\mathcal{E}}(\cdot, t)|$ and surface current $|\boldsymbol{w}(\cdot)|$

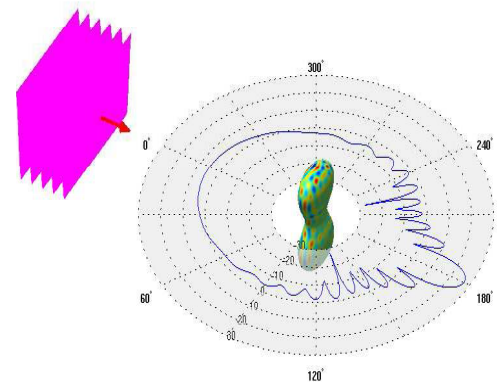


Bistatic RCS

FIG. 5. *Visualizations of the scattered field, surface current, and bistatic RCS induced by a horizontally polarized plane wave impinging on* $\mathbf{hive}(10\lambda)$ *at* 1.4GHz *with* $\widehat{\boldsymbol{d}}_0 = (-1, 0, 0)^T$.

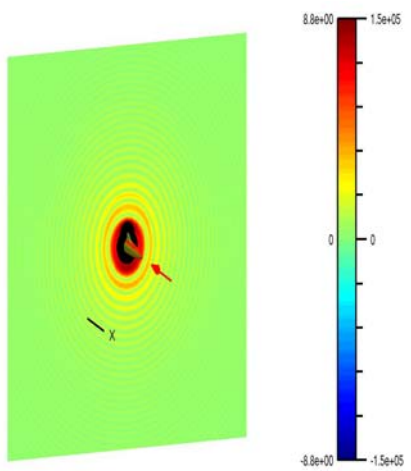


Scattered electric field $|\boldsymbol{\mathcal{E}}(\cdot,t)|$ and surface current $|\boldsymbol{w}(\cdot)|$

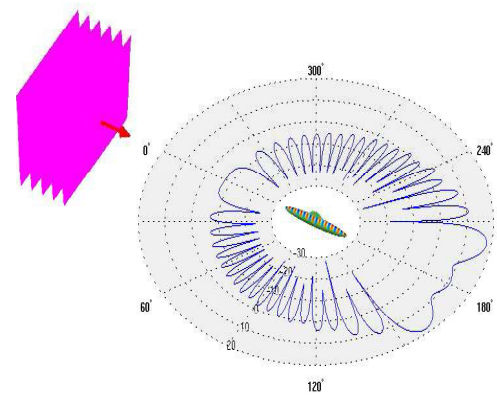


Bistatic RCS

FIG. 6. *Visualizations of the scattered field, surface current, and bistatic RCS induced by a horizontally polarized plane wave impinging on* $\mathbf{cas}(10\lambda)$ *at* 3.0GHz *with* $\widehat{\boldsymbol{d}}_0 = (-1,0,0)^T$.

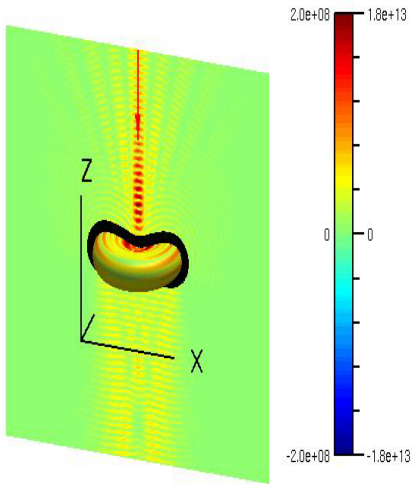


Scattered electric field $|\boldsymbol{\mathcal{E}}(\cdot,t)|$ and surface current $|\boldsymbol{w}(\cdot)|$

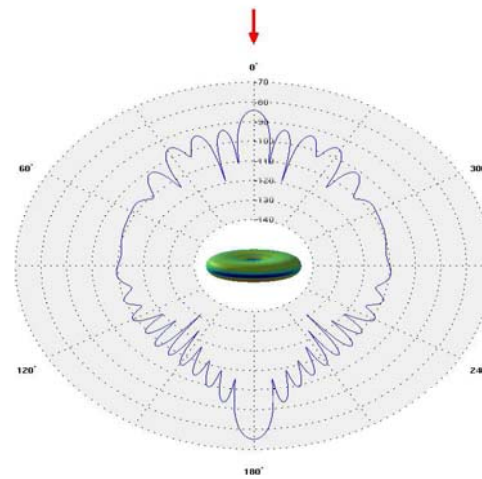


Bistatic RCS

FIG. 7. *Visualizations of the scattered field, surface current, and bistatic RCS induced by a horizontally polarized plane wave impinging on* $\mathbf{sub}(10\lambda)$ *at* 149.9MHz *with* $\widehat{\boldsymbol{d}}_0 = (-1,0,0)^T$.

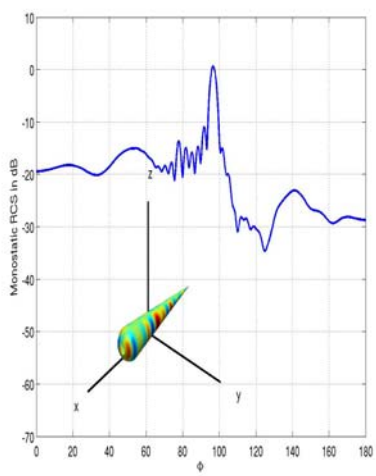


Scattered electric field $|\boldsymbol{\mathcal{E}}(\cdot,t)|$ and surface current $|\boldsymbol{w}(\cdot)|$

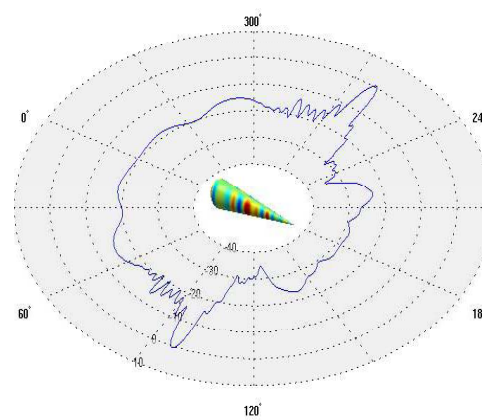


Bistatic RCS

FIG. 8. *Visualizations of the scattered field, surface current, and bistatic RCS induced by a horizontally polarized plane wave impinging on* $\mathbf{ery}(10\lambda)$ *at* 475.9THz *with* $\widehat{\boldsymbol{d}}_0 = (0,0,-1)^T$.



Monostatic RCS and surface current



Monostatic RCS

FIG. 9. *Visualizations of the surface current and monostatic RCS induced by a vertically polarized plane wave impinging on* $\mathbf{cone_sph}(10\lambda)$ *at* 4.35GHz.

REFERENCES

[1] G. E. Antilla and N. G. Alexopoulos, *Scattering from complex three-dimensional geometries by a curvilinear hybrid finite-element-integral equation approach*, J. Opt. Soc. Amer. A, 11 (1994), pp. 1445–1457.

[2] A. Bendali, Y. Boubendir, and M. Fares, *A FETI-like domain decomposition method for coupling finite elements and boundary elements in large-size scattering problems of acoustic scattering*, Comput. & Structures, 85 (2007), pp. 526–535.

[3] O. P. Bruno, *Fast, high-order, high-frequency integral methods for computational acoustics and electromagnetics*, in Topics in Computational Wave Propagation, M. Ainsworth et al., eds., Lect. Notes Comput. Sci. Eng. 31, Springer, Berlin, 2003, pp. 43–82.

[4] O. P. Bruno and L. A. Kunyansky, *A fast, high-order algorithm for the solution of surface scattering problems: Basic implementation, tests, and applications*, J. Comput. Phys., 169 (2001), pp. 80–110.

[5] L. Canino, J. Ottusch, M. Stalzer, J. Visher, and S. Wandzura, *Numerical solution of the Helmholtz equation in 2D and 3D using a high–order Nystrom discretization*, J. Comput. Phys., 146 (1998), pp. 627–663.

[6] W. C. Chew, J. M. Jin, E. Michielssen, and J. Song, *Fast and Efficient Algorithms in Computational Electromagnetics*, Artech House, Norwood, MA, 2001.

[7] D. Colton and R. Kress, *Integral Equation Methods in Scattering Theory*, Wiley, New York, 1983.

[8] D. Colton and R. Kress, *Inverse Acoustic and Electromagnetic Scattering Theory*, Springer, Berlin, 1998.

[9] E. Darve and P. Havé, *Efficient fast multipole method for low-frequency scattering*, J. Comput. Phys., 197 (2004), pp. 341–363.

[10] E. Darve and P. Havé, *A fast multipole method for Maxwell equations stable at all frequencies*, Phil. Trans. Roy. Soc. London Ser. A, 362 (2004), pp. 603–628.

[11] W. Freeden, T. Gervens, and M. Schreiner, *Constructive Approximation on the Sphere*, Oxford University Press, Oxford, UK, 1998.

[12] M. Ganesh and I. G. Graham, *A high-order algorithm for obstacle scattering in three dimensions*, J. Comput. Phys., 198 (2004), pp. 211–242.

[13] M. Ganesh, I. G. Graham, and J. Sivaloganathan, *A pseudospectral three-dimensional boundary integral method applied to a nonlinear model problem from finite elasticity*, SIAM J. Numer. Anal., 31 (1994), pp. 1378–1414.

[14] M. Ganesh, I. G. Graham, and J. Sivaloganathan, *A new spectral boundary integral collocation method for three-dimensional potential problems*, SIAM J. Numer. Anal., 35 (1998), pp. 778–805.

[15] M. Ganesh and S. C. Hawkins, *An efficient surface integral equation method for the time-harmonic Maxwell equations*, ANZIAM J., 48 (2007), pp. C17–C33.

[16] M. Ganesh and S. C. Hawkins, *A spectrally accurate algorithm for electromagnetic scattering in three dimensions*, Numer. Algorithms, 43 (2006), pp. 25–60.

[17] I. G. Graham and I. H. Sloan, *Fully discrete spectral boundary integral methods for Helmholtz problems on smooth closed surfaces in* $\mathbb{R}^3$, Numer. Math., 92 (2002), pp. 289–323.

[18] E. Heikkola, T. Rossi, and J. Toivanen, *A parallel fictitious domain method for the three-dimensional Helmholtz equation*, SIAM J. Sci. Comput., 24 (2003), pp. 1567–1588.

[19] J. Hellmers, E. Eremina, and T. Wriedt, *Simulation of light scattering by biconcave Cassini ovals using the nullfield method with discrete sources*, J. Opt. A: Pure Appl. Opt., 8 (2006), pp. 1–9.

[20] E. F. Knott, J. F. Shaeffer, and M. T. Tuley, *Radar Cross Section*, SciTech Publishing, Raleigh, NC, 2004.

[21] P. Monk, *Finite Element Methods for Maxwell's Equations*, Oxford University Press, Oxford, UK, 2003.

[22] R. Reimer, *Hyperinterpolation on the sphere at the minimal projection order*, J. Approx. Theory, 103 (2000), pp. 272–286.

[23] F. Reitich, K. K. Tamma, and R. R. Namburu, eds., *Special Issue on Computational Electromagnetics*, Comput. Model. Eng. Sci., 5 (2004).

[24] I. H. Sloan and R. S. Womersley, *Constructive polynomial approximation on the sphere*, J. Approx. Theory, 103 (2000), pp. 91–118.

[25] J. M. Song, C. C. Lu, W. C. Chew, and S. W. Lee, *Fast Illinois solver code (FISC)*, IEEE Antennas Propag. Mag., 40 (1998), pp. 27–34.

[26] R. A. R. Tricker, *Introduction to Meteorological Optics*, Elsevier, New York, 1970.

[27] L. Tsang, J. A. Kong, and K. Ding, *Scattering of Electromagnetic Waves: Theories and Applications*, Wiley, New York, 2000.

[28] R. T. Wang and H. C. van de Hulst, *Rainbows: Mie computations and the Airy approximation*, Appl. Optics, 30 (1991), pp. 106–117.

[29] L. Wienert, *Die Numerische Approximation von Randintegraloperatoren für die Helmholtzgleichung im* $\mathbb{R}^3$, Ph.D. thesis, University of Göttingen, Göttingen, Germany, 1990.

[30] A. C. Woo, H. T. Wang, M. J. Schuh, and M. L. Sanders, *Benchmark radar targets for the validation of computational electromagnetics programs*, IEEE Antennas Propag. Mag., 35 (1993), pp. 84–89.

[31] T. Wriedt, J. Hellmers, E. Eremina, and R. Schuh, *Light scattering by single erythrocite: Comparison of different methods*, J. Quant. Spectrosc. Radiat. Transfer, 100 (2006), pp. 444–456.

SIAM J. SCI. COMPUT.
Vol. 29, No. 3, pp. 1244–1267
© 2007 Society for Industrial and Applied Mathematics

A FICTITIOUS DOMAIN METHOD WITH MIXED FINITE ELEMENTS FOR ELASTODYNAMICS*

E. BÉCACHE†, J. RODRÍGUEZ‡, AND C. TSOGKA§

Abstract. We consider in this paper the wave scattering problem by an object with Neumann boundary conditions in an anisotropic elastic body. To obtain an efficient numerical method (permitting the use of regular grids) we follow a fictitious domain approach coupled with a first order velocity stress formulation for elastodynamics. We first observe that the method does not always converge when the $Q_1^{\rm div} - Q_0$ finite element is used. In particular, the method converges for some scattering object geometries but not for others. Note that the convergence of the $Q_1^{\rm div} - Q_0$ finite element method was shown in [E. Bécache, P. Joly, and C. Tsogka, *SIAM J. Numer. Anal.*, 39 (2002), pp. 2109–2132] for the elastodynamic problem in the absence of a scattering object (i.e., without the coupling of the mixed finite elements with the fictitious domain method). Therefore we propose here a modification of the $Q_1^{\rm div} - Q_0$ element following the approach in [E. Bécache, J. Rodríguez, and C. Tsogka, *On the convergence of the fictitious domain method for wave equation problems*, Technical report 5802, INRIA, 2006], where the simpler acoustic case was considered. To study the numerical properties of the new element we carry out a dispersion analysis. Several numerical simulations as well as a numerical convergence analysis show that the proposed method provides a good approximate solution.

Key words. mixed finite elements, fictitious domain method, elastic waves, convergence

AMS subject classifications. 65M60, 65M12, 65M15, 65C20, 74S05

DOI. 10.1137/060655821

1. Introduction.

We consider here the scattering of elastic waves by objects or cracks with a homogeneous Neumann boundary condition. This is of interest in several applications such as ultrasonic nondestructive testing, seismic wave propagation, etc. To develop an efficient numerical method we intend to use a fictitious domain approach. Following this approach, we extend artificially the solution to a fictitious domain with a simple shape, typically a rectangle in 2D. Then the boundary condition on the complex geometry is enforced by introducing an auxiliary unknown that leaves only at the boundary of the object. The key point of the method is that the mesh for the unknowns leaving on the enlarged domain can be chosen independently of the geometry of the object. In particular, one can use regular grids or structured meshes which allows for simple and efficient computations.

The fictitious domain method was initially introduced for stationary problems [3, 30, 24, 18, 20, 27], and it was then successfully generalized to time-dependent problems. In particular, it was used for modeling unsteady incompressible viscous flow with fixed or moving boundaries [25, 21, 22, 23] and in several applications of wave propagation, such as acoustic scattering problems [15, 10, 12], electromagnetic scattering problems [17], and musical acoustics [33, 29, 4]. For both stationary and time-dependent problems the fictitious domain improves the performance of the nu-

*Received by the editors March 30, 2006; accepted for publication (in revised form) January 16, 2007; published electronically May 29, 2007.
http://www.siam.org/journals/sisc/29-3/65582.html
†POEMS, INRIA-Rocquencourt, BP 105, F-78153 Le Chesnay Cédex, France (eliane.becache@inria.fr).
‡POEMS, ENSTA, 32 boulevard Victor, 75739 Paris cedex 15, France (jeronimo.rodriguez@ensta.fr)
§Department of Mathematics, University of Chicago, 5734 University Avenue Chicago, IL 60637 (tsogka@math.uchicago.edu).

merical method in terms of computational time and memory. However, the accuracy of the method is limited by construction due to the lack of regularity of the solution in the extended domain [18].

A main feature of the fictitious domain method is that it can be applied only to essential boundary conditions (i.e., that can be considered as an equality constraint in the functional space). In the case we are interested in, the boundary condition is of the Neumann type for the displacement. To apply the fictitious domain method we have to choose a formulation for which the stress tensor $\boldsymbol{\sigma}$ is included in the unknowns and is sought in a subspace of $(H(\mathrm{div}))^2$: the first order velocity-stress system or the second order stress formulation. In this way the boundary condition becomes of the essential type for $\boldsymbol{\sigma}$. We chose the first order formulation because it can be combined with the perfectly matched absorbing layer model, which is an efficient way of simulating wave propagation in infinite domains [11, 16].

For efficiency reasons we would like to discretize the velocity-stress system with mixed finite elements that are compatible with mass lumping. This will permit us to obtain an explicit in time discretization scheme. To our knowledge this is not so obvious; the main difficulty appearing in this problem is finding a way to take into account the symmetry of the stress tensor. The approach which is usually followed is the relaxed symmetry approach: The symmetry is imposed in a weak sense via the introduction of a Lagrange multiplier [1, 2, 34, 35, 31]. Another approach using spaces of symmetric stress tensors, based on composite elements, was introduced in [28]. None of these, however, are adapted to mass lumping. An answer to this difficulty was proposed in [9], where the $Q_{k+1}^{\mathrm{div}} - Q_k$ finite element has been introduced.

A nonstandard convergence analysis of the $Q_{k+1}^{\mathrm{div}} - Q_k$ elements has been carried out in [7] for their scalar version and in [9] for their elastodynamic vectorial version. However, the convergence analysis presented there deals only with the mixed problem without the scattering object (i.e., without the coupling of the mixed finite elements with the fictitious domain method). Recently, in [10] the authors considered the scalar case and showed through some numerical experiments that this finite element approximation does not always converge when it is coupled with the fictitious domain method. Motivated by this negative result, a new finite element based on the enrichment of the approximation space for the primal unknown (which is the pressure in this case) was introduced. Moreover, in [10] the convergence of this finite element coupled with the fictitious domain method was theoretically proven.

In the present paper we treat the elastodynamic case where the convergence issues are similar to those in the scalar case. Namely, we show through some numerical experiments that the $Q_1^{\mathrm{div}} - Q_0$ element does not always converge when it is coupled with the fictitious domain method (the method converges for some scattering object geometries, while it does not converge for others). Following the same ideas presented in [10] we introduce a modified finite element for the vectorial case, the so-called $Q_1^{\mathrm{div}} - P_1^{disc}$ where we enrich the approximation space for the velocity from Q_0 to P_1 discontinuous functions. This amounts to adding four additional degrees of freedom per element for the velocity and results in introducing spurious modes to the solution. To get rid of this nonphysical part of the solution we propose to attenuate the spurious modes by introducing a damping term in the equations. We study (numerically) the error between the discrete and continuous solutions in terms of the damping parameter (β) and propose a way to determine β in order to minimize this error.

The theoretical convergence of the method is not a straightforward generalization of the one presented in [10] for the scalar case. The main difficulty is due to the lack of regularity of the solution in the extended fictitious domain. However, several

numerical simulations indicate that the method converges, and we conjecture that this is true. The numerical order of convergence obtained for a particular object is in agreement with the conjectured theoretical one.

The outline of the paper is as follows. In section 2 we present the elastodynamic equations and the fictitious domain formulation that we will use in what follows. In section 3.2 we illustrate with numerical experiments that the $Q_1^{\rm div} - Q_0$ element does not always converge. The modified $Q_1^{\rm div} - P_1^{disc}$ element is introduced in section 4 and its dispersion analysis in section 4.3. The proposed method for attenuating the nonphysical part of the solution is then presented in section 4.4, and the difficulties of the theoretical convergence analysis are explained in section 4.2. Numerical results that indicate the convergence of the method are shown in section 4.5. Finally in section 5 we consider a numerical convergence analysis for a particular object.

2. The fictitious domain formulation of the diffraction problem.

2.1. The continuous problem. We consider the diffraction of an elastic wave by a two-dimensional object with a free surface boundary condition (Neumann boundary condition on the velocity field $\mathbf{v}$) on its boundary Γ. The object can be either an obstacle with a closed boundary or a crack with an open boundary (see Figure 2.1), but for the sake of clarity we will consider here only this second configuration. The domain of propagation is denoted by Ω with an exterior boundary Σ (see Figure 2.1), and we assume that $C = \Omega \cup \Gamma$ is a domain of "simple" geometry, typically a rectangle. Under the assumptions of small deformations, wave propagation can be modeled by the velocity-stress formulation of the linear elastodynamic equations (e.g., [9] or [13] for the static case)

$$
\left\{
\begin{array}{rcrcll}
\rho \dfrac{\partial \mathbf{v}}{\partial t} & - & \operatorname{div} \boldsymbol{\sigma} & = & \mathbf{f} & \text{in } \Omega, \quad \text{(a)} \\
\mathbf{A} \dfrac{\partial \boldsymbol{\sigma}}{\partial t} & - & \boldsymbol{\varepsilon}(\mathbf{v}) & = & 0 & \text{in } \Omega, \quad \text{(b)} \\
\boldsymbol{\sigma}\mathbf{n} & & & = & 0 & \text{on } \Gamma, \quad \text{(c)} \\
\mathbf{v} & & & = & 0 & \text{on } \Sigma, \quad \text{(d)}
\end{array}
\right. \tag{2.1}
$$

together with the initial conditions

$$
\mathbf{v}(t=0) = \mathbf{v}_0, \qquad \boldsymbol{\sigma}(t=0) = \boldsymbol{\sigma}_0. \tag{2.2}
$$

In (2.1), the unknowns are the vector $\mathbf{v} \in \mathbb{R}^2$ and the second order symmetric tensor $\boldsymbol{\sigma} \in \mathbb{R}^{2\times 2}$ (i.e., $\boldsymbol{\sigma} = \boldsymbol{\sigma}^t$). These two quantities represent, respectively, the velocity

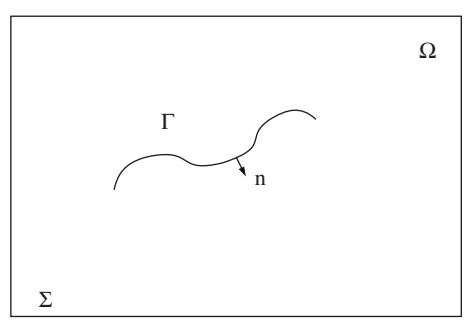


FIG. 2.1. *Geometry of the problem.*

field and the stress tensor. The scalar function ρ and the tensor $\mathbf{A}$ characterize the medium, and the vector field $\mathbf{f}$ represents the external forces. Moreover, we assume that ρ satisfies

$$0 < \rho^- \leq \rho(x) \leq \rho^+ < +\infty,$$

and $\mathbf{A}$ is a fourth order symmetric positive tensor such that

$$0 < \kappa|\boldsymbol{\tau}|^2 \leq \mathbf{A}(x)\boldsymbol{\tau} : \boldsymbol{\tau} \leq \nu|\boldsymbol{\tau}|^2 \quad \forall\ \boldsymbol{\tau} \text{ symmetric}, \quad \boldsymbol{\tau} \neq 0.$$

This formulation is preferred to the classical displacement formulation, because the boundary condition on the crack is natural (of Neumann type) for the displacement while it becomes essential on $\boldsymbol{\sigma}$, and the fictitious domain approach can then be followed.

We also assume that the support of the initial data $(\mathbf{v}_0, \boldsymbol{\sigma}_0)$ and the support of the source $\mathbf{f}$ do not intersect Γ, which means that

$$\text{supp}(\mathbf{v}_0) \cup \text{supp}(\boldsymbol{\sigma}_0) \subset C \setminus \Gamma, \quad \bigcup_{t \leq T} \text{supp}(\mathbf{f}(t)) \subset C \setminus \Gamma. \tag{2.3}$$

The classical variational formulation of (2.1) would be set in some functional spaces that depend on the shape of the obstacle (i.e., depend on Ω). More precisely, the classical formulation is

$$\left\{ \begin{array}{l} \text{Find } (\mathbf{v}(t), \boldsymbol{\sigma}(t)) \in \underline{\underline{X}}_0^{sym} \times \underline{M} \text{ satisfying} \\ \dfrac{\mathrm{d}}{\mathrm{d}t}\displaystyle\int_\Omega \mathbf{A}\boldsymbol{\sigma} : \boldsymbol{\tau}\mathrm{d}x + \int_\Omega \mathrm{div}\boldsymbol{\tau} \cdot \mathbf{v}\mathrm{d}x = 0 \qquad \forall\ \boldsymbol{\tau} \in \underline{\underline{X}}_0^{sym}, \\ \dfrac{\mathrm{d}}{\mathrm{d}t}\displaystyle\int_\Omega \rho\ \mathbf{v} \cdot \mathbf{w}\mathrm{d}x - \int_\Omega \mathrm{div}\boldsymbol{\sigma} \cdot \mathbf{w}\mathrm{d}x = \int_\Omega \mathbf{f} \cdot \mathbf{w}\mathrm{d}x \quad \forall\ \mathbf{w} \in \underline{M}, \\ (\boldsymbol{\sigma}, \mathbf{v})_{t=0} = (\boldsymbol{\sigma}_0, \mathbf{v}_0), \end{array} \right. \tag{2.4}$$

where the functional spaces are defined as

$$\begin{array}{l} \underline{X} = H(\mathrm{div}; \Omega), \quad \underline{\underline{X}} = \underline{X} \times \underline{X}, \quad \underline{\underline{X}}^{sym} = \left\{ \boldsymbol{\tau} \in \underline{\underline{X}} : \boldsymbol{\tau} \text{ is symmetric} \right\}, \\ \underline{\underline{X}}_0^{sym} = \left\{ \boldsymbol{\tau} \in \underline{\underline{X}}^{sym} : \boldsymbol{\tau}\ \mathbf{n} = 0 \text{ on } \Gamma \right\}, \quad M = L^2(\Omega), \quad \underline{M} = M \times M. \end{array} \tag{2.5}$$

The well-posedness of the problem (2.4) results from the classical theory on hyperbolic PDEs.

THEOREM 2.1. [illegible] $\mathbf{f} \in C^0([0,T], \underline{M})$ $\boldsymbol{\sigma}_0 \in \underline{\underline{X}}_0^{sym}(\Omega)$ $\mathbf{v}_0 \in \underline{M}$ [illegible] (2.3) [illegible] (2.4) [illegible] $(\boldsymbol{\sigma}, \mathbf{v}) \in (C^1([0,T], (L^2(\Omega))^{2\times 2}) \cap C^0([0,T], \underline{\underline{X}}_0^{sym}(\Omega))) \times C^1([0,T], \underline{M})$

The fictitious domain formulation of this problem consists in taking into account the boundary condition on Γ in a weak way, by introducing a Lagrange multiplier $\boldsymbol{\lambda}$ defined on Γ. This allows for working in functional spaces (for the volume unknowns) which do not depend any more on the shape of the obstacle. The fictitious domain formulation is then the following (to simplify the notations, we will denote by

$(\boldsymbol{\sigma}(t), \mathbf{v}(t))$ the new unknowns defined now in C):

$$
\text{(2.6)}\quad \begin{cases} \text{Find } (\boldsymbol{\sigma}(t), \mathbf{v}(t), \boldsymbol{\lambda}(t)) \in \underline{\underline{X}}^{sym} \times \underline{M} \times \underline{\mathcal{G}} \text{ satisfying} \\ \dfrac{\mathrm{d}}{\mathrm{d}t} a(\boldsymbol{\sigma}, \boldsymbol{\tau}) + b(\boldsymbol{\tau}, \mathbf{v}) - \langle \boldsymbol{\tau}\ \mathbf{n}, \boldsymbol{\lambda} \rangle_\Gamma = 0 \quad \forall\ \boldsymbol{\tau} \in \underline{\underline{X}}^{sym}, \\ \dfrac{\mathrm{d}}{\mathrm{d}t} (\mathbf{v}, \mathbf{w})_\rho - b(\boldsymbol{\sigma}, \mathbf{w}) = (\mathbf{f}, \mathbf{w}) \quad \forall\ \mathbf{w} \in \underline{M}, \\ \langle \boldsymbol{\sigma}\ \mathbf{n}, \boldsymbol{\mu} \rangle_\Gamma = 0 \quad \forall\ \boldsymbol{\mu} \in \underline{\mathcal{G}}, \\ (\boldsymbol{\sigma}, \mathbf{v})_{t=0} = (\boldsymbol{\sigma}_0, \mathbf{v}_0), \end{cases}
$$

where the functional spaces are now defined as

$$
\text{(2.7)}\quad \begin{aligned} &\underline{X} = H(\mathrm{div}; C), \quad \underline{\underline{X}} = \underline{X} \times \underline{X}, \quad \underline{\underline{X}}^{sym} = \left\{ \boldsymbol{\tau} \in \underline{\underline{X}} : \boldsymbol{\tau} \text{ is symmetric} \right\}, \\ &M = L^2(C), \quad \underline{M} = M \times M, \quad \mathcal{G} = H_{00}^{1/2}(\Gamma), \quad \underline{\mathcal{G}} = \mathcal{G} \times \mathcal{G}, \end{aligned}
$$

the bilinear forms as

$$
\text{(2.8)}\quad \begin{cases} a(\boldsymbol{\sigma}, \boldsymbol{\tau}) = \displaystyle\int_C \mathbf{A}\boldsymbol{\sigma} : \boldsymbol{\tau} \mathrm{d}x & \forall (\boldsymbol{\sigma}, \boldsymbol{\tau}) \in \underline{\underline{X}} \times \underline{\underline{X}}, \\ (\mathbf{v}, \mathbf{w})_\eta = \displaystyle\int_C \eta\ \mathbf{v} \cdot \mathbf{w} \mathrm{d}x & \forall (\mathbf{v}, \mathbf{w}) \in \underline{M} \times \underline{M}, \\ b(\boldsymbol{\tau}, \mathbf{w}) = \displaystyle\int_C \mathrm{div}(\boldsymbol{\tau}) \cdot \mathbf{w} \mathrm{d}x & \forall \underline{\underline{X}} \times \underline{M}, \end{cases}
$$

and the bracket $\langle \boldsymbol{\sigma}\ \mathbf{n}, \boldsymbol{\mu} \rangle_\Gamma$ is the duality product between $\underline{\mathcal{G}}$ and $\underline{\mathcal{G}}'$. Note that, due to assumption (2.3), the data also belong to

$$
\mathbf{f} \in C^0([0,T], \underline{M}), \quad \boldsymbol{\sigma}_0 \in \underline{\underline{X}}(C), \quad \mathbf{v}_0 \in \underline{M}.
$$

In the following we will denote by $(\cdot,\cdot) := (\cdot,\cdot)_1$ the usual $L^2(C)$ scalar product.

. . . . 1 (on the regularity of the solution). For regular data, one can expect more regularity on the solution. However, in general, the space regularity of the solution is at most

$$
\boldsymbol{\sigma}(t) \in (H^{\frac{1}{2}-\varepsilon}(\mathrm{div}, C))^2, \quad \mathbf{v}(t) \in (H^{\frac{1}{2}-\varepsilon}(C))^2, \qquad \varepsilon > 0,
$$

and this is obtained for regular enough data and a regular geometry of the crack. This is due to the fact that the unknowns are defined on the whole domain C without considering the geometry of the obstacle, so that $\mathbf{v}$ has now a jump in C.

The regularity in Ω (i.e., outside the obstacle) is, in general, higher and depends on the geometry of the obstacle. In particular, for data $(\boldsymbol{\sigma}_0, \mathbf{v}_0, \mathbf{f})$ satisfying (2.3), we have

- for a closed boundary

$$
\mathbf{v}_{/\Omega}(t) \in (H^2(\Omega))^2, \quad \boldsymbol{\lambda}(t) \in (H^{3/2}(\Gamma))^2;
$$

- for an open boundary, due to the singular behavior near the tip of the crack [26] (the solution behaves as $\sqrt{r}$, r being here the distance to the tip), we have

$$
\mathbf{v}_{|\Omega}(t) \in (H^{3/2-\varepsilon}(\Omega))^2, \quad \boldsymbol{\lambda}(t) \in (H^{1-\varepsilon}(\Gamma))^2, \qquad \varepsilon > 0.
$$

2.2. The semidiscrete approximation. For the approximation in space of this problem, we introduce finite dimensional spaces $\underline{\underline{X_h}}^{sym} \subset \underline{\underline{X}}^{sym}$, $\underline{M}_h \subset \underline{M}$, and $\underline{\mathcal{G}}_H \subset \underline{\mathcal{G}}$ satisfying the approximation properties

$$\left|\begin{array}{lcll} \lim\limits_{h\to 0} \inf\limits_{\boldsymbol{\tau}_h \in \underline{\underline{X_h}}} \|\boldsymbol{\sigma} - \boldsymbol{\tau}_h\|_{\underline{\underline{X}}} & = & 0 & \forall \boldsymbol{\sigma} \in \underline{\underline{X}}, \\ \lim\limits_{h\to 0} \inf\limits_{\mathbf{w}_h \in \underline{M}_h} \|\mathbf{v} - \mathbf{w}_h\|_{\underline{M}} & = & 0 & \forall \mathbf{v} \in \underline{M}, \\ \lim\limits_{H\to 0} \inf\limits_{\boldsymbol{\mu}_H \in \underline{\mathcal{G}}_H} \|\boldsymbol{\lambda} - \boldsymbol{\mu}_H\|_{\underline{\mathcal{G}}} & = & 0 & \forall \boldsymbol{\lambda} \in \underline{\mathcal{G}}. \end{array}\right. \tag{2.9}$$

The semidiscrete problem is then

(2.10)

$$\left\{\begin{array}{l} \text{Find } (\boldsymbol{\sigma}_h(t), \mathbf{v}_h(t), \boldsymbol{\lambda}_H(t)) \in \underline{\underline{X}}_h^{sym} \times \underline{M}_h \times \underline{\mathcal{G}}_H \text{ satisfying} \\ \dfrac{\mathrm{d}}{\mathrm{d}t} a(\boldsymbol{\sigma}_h, \boldsymbol{\tau}_h) + b(\boldsymbol{\tau}_h, \mathbf{v}_h) - \langle \boldsymbol{\tau}_h\, \mathbf{n}, \boldsymbol{\lambda}_H \rangle_\Gamma = 0 \quad \forall\, \boldsymbol{\tau}_h \in \underline{\underline{X}}_h^{sym}, \\ \dfrac{\mathrm{d}}{\mathrm{d}t} (\mathbf{v}_h, \mathbf{w}_h)_\rho - b(\boldsymbol{\sigma}_h, \mathbf{w}_h) = (\mathbf{f}, \mathbf{w}_h) \quad \forall\, \mathbf{w}_h \in \underline{M}_h, \\ \langle \boldsymbol{\sigma}_h\, \mathbf{n}, \boldsymbol{\mu}_H \rangle_\Gamma = 0 \quad \forall\, \boldsymbol{\mu}_h \in \underline{\mathcal{G}}_H, \\ (\boldsymbol{\sigma}_h, \mathbf{v}_h)_{t=0} = (\boldsymbol{\sigma}_{h,0}, \mathbf{v}_{h,0}), \end{array}\right.$$

where $(\boldsymbol{\sigma}_{h,0}, \mathbf{v}_{h,0}) \in \underline{\underline{X}}_h \times \underline{M}_h$ is an approximation of the exact initial condition.

Now the question is how we can choose the approximate spaces in order to ensure the convergence of $(\boldsymbol{\sigma}_h, \mathbf{v}_h, \boldsymbol{\lambda}_H)$ to $(\boldsymbol{\sigma}, \mathbf{v}, \boldsymbol{\lambda})$.

3. The fictitious domain method using the $\mathcal{Q}_1^{\mathrm{div}} - \mathcal{Q}_0$ element.

3.1. Position of the problem. For the volume unknowns, we introduce a regular mesh $\mathcal{T}_h$ of the rectangular domain C composed of square elements of length h. In [5], the authors introduced for the problem without obstacle new mixed finite elements, the so-called $\mathcal{Q}_{k+1}^{\mathrm{div}} - \mathcal{Q}_k$ elements, inspired by Nédélec's second family [32]. These elements, which are a generalization to elastodynamics of those presented in [6] for the scalar case, are compatible with mass lumping and therefore allow for constructing explicit schemes in time. A nonstandard convergence analysis of these $\mathcal{Q}_{k+1}^{\mathrm{div}} - \mathcal{Q}_k$ elements has been carried out, showing the convergence without the fictitious domain method. Our first choice for the approximation spaces of the problem with an obstacle was naturally the lowest order element $\mathcal{Q}_1^{\mathrm{div}} - \mathcal{Q}_0$ for the stress tensor and the velocity field

$$\left\{\begin{array}{l} \underline{\underline{X}}_h^{sym} = \left\{ \boldsymbol{\tau}_h \in \underline{\underline{X}}^{sym} \;/\; \forall\, K \in \mathcal{T}_h, \tau_{h|K} \in (\mathcal{Q}_1 \times \mathcal{Q}_1)^2 \right\}, \\ \underline{M}_h = \underline{M}_h^0 = \left\{ \mathbf{w}_h \in \underline{M} \;/\; \mathbf{w}_{h\,K} \in (Q_0)^2 \right\}. \end{array}\right. \tag{3.1}$$

We point out that due to the symmetry of the stress tensor, another characterization of $\underline{\underline{X}}_h^{sym}$ is

$$\left\{\begin{array}{l} \underline{\underline{X}}_h^{sym} = \Big\{ \boldsymbol{\tau}_{1,2} \in H^1(C) \;/\; \boldsymbol{\tau}_{1,2|K} \in Q_1 \;\forall\, K \in \mathcal{T}_h \text{ and} \\ \qquad\qquad (\boldsymbol{\tau}_{1,1}, \boldsymbol{\tau}_{2,2}) \in H(\mathrm{div}, C) / (\boldsymbol{\tau}_{1,1}, \boldsymbol{\tau}_{2,2})_{|K} \in (Q_1)^2 \;\forall\, K \in \mathcal{T}_h \Big\}. \end{array}\right. \tag{3.2}$$

The degrees of freedom of the lowest order element are illustrated in Figure 3.1. We recall here the approximation properties for the space $\underline{\underline{X_h}}^{sym}$. Let $\tau \in \underline{\underline{X}}^{sym}$ with

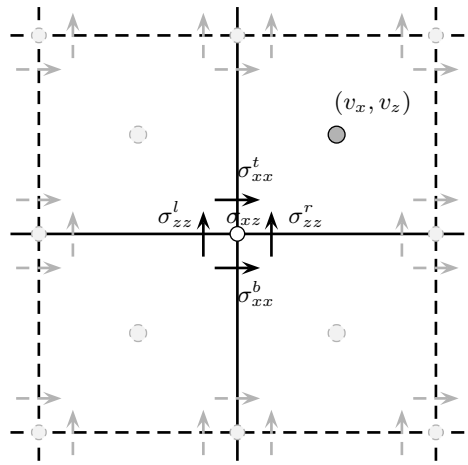


FIG. 3.1. *Degrees of freedom for the mixed finite element defined by* (3.1).

$(\tau_{11}, \tau_{22}) \in H^{1,0} \times H^{0,1}$ (see [9] for the definition of these spaces) and $\tau_{12} = \tau_{21} \in H^1$; then

$$\lim_{h \to 0} \inf_{\boldsymbol{\tau}_h \in \underline{\underline{X_h}}^{sym}} \|\boldsymbol{\tau} - \boldsymbol{\tau}_h\|_{\underline{\underline{X}}} = 0.$$

Moreover, if $(\tau_{11}, \tau_{22}) \in H^{2,1} \times H^{1,2}$ and $\tau_{12} \in H^2$, then

$$\inf_{\boldsymbol{\tau}_h \in \underline{\underline{X_h}}^{sym}} \|\boldsymbol{\tau} - \boldsymbol{\tau}_h\|_{\underline{\underline{X}}} \leq Ch(|\tau_{11}|_{H^{2,1}} + |\tau_{22}|_{H^{1,2}} + |\tau_{12}|_{H^2}). \tag{3.3}$$

For the approximation of the Lagrange multiplier, we introduce a mesh of Γ composed by N curvilinear segments S_j of length H_j, and we set $H = \sup_j H_j$. We assume that this mesh is uniformly regular

$$\exists\, \nu, 0 < \nu \leq 1 \text{ such that: } \forall\ j, 1 \leq j \leq N,\ H_j \geq \nu H. \tag{3.4}$$

We then choose the space of continuous linear piecewise functions:

$$\underline{\mathcal{G}}_H = \left\{ \nu_h \in \underline{\mathcal{G}} \ / \ \forall S_j, j = 1, \ldots, N \ ; \ \nu_{H|S_j} \in (P_1)^2 \right\}. \tag{3.5}$$

The spaces $(\underline{\underline{X}}_h, \underline{M}^0_h, \underline{\mathcal{G}}_H)$ clearly satisfy the approximation properties (2.9).

This choice which seemed to us natural, since the convergence was proven without obstacle, is the one that was used in [8]. However, we have not been able to prove the convergence of the fictitious domain method with these spaces.

The convergence analysis of the fictitious domain method applied to other problems [3, 18, 29] shows that convergence holds if a compatibility condition between the step sizes of the two meshes is satisfied,

$$H \geq \alpha h. \tag{3.6}$$

We will show in what follows some numerical illustrations which seem to indicate that for some special configurations of obstacles, the method does not converge.

Before showing these numerical results, let us briefly recall the main difficulty of the convergence analysis in the case without an object. Introducing the linear

operators

$$\begin{array}{lccccccc} B: & \underline{\underline{X}} & \longrightarrow & \underline{M}' & & & & \\ & \boldsymbol{\sigma} & \mapsto & B(\boldsymbol{\sigma}): & \underline{M} & \longrightarrow & \mathbb{R} \\ & & & & \mathbf{w} & \mapsto & \langle B(\boldsymbol{\sigma}), \mathbf{w}\rangle = b(\boldsymbol{\sigma}, \mathbf{w}), \\ B_h: & \underline{\underline{X}}_h & \longrightarrow & \underline{M}_h' & & & & \\ & \boldsymbol{\sigma}_h & \mapsto & B_h(\boldsymbol{\sigma}_h): & \underline{M}_h & \longrightarrow & \mathbb{R} \\ & & & & \mathbf{w}_h & \mapsto & \langle B_h(\boldsymbol{\sigma}_h), \mathbf{w}_h\rangle = b(\boldsymbol{\sigma}_h, \mathbf{w}_h), \end{array}$$

it is easy to verify that the inclusion

$$\mathrm{Ker}(B_h) \subset \mathrm{Ker}(B) \tag{3.7}$$

is not satisfied and that furthermore the bilinear form $a(.,.)$ is not coercive on $\mathrm{Ker}(B_h)$ (even if it is on $\mathrm{Ker}(B)$), so that our problem does not fit the classical mixed finite element theory (cf. [13, 19]). It was, however, possible to overcome this difficulty when dealing with the problem without the fictitious domain method. When coupled with the fictitious domain method, the same technique cannot be applied.

3.2. Numerical illustrations. We present in what follows some numerical results that illustrate the difficulties related to the convergence of the method that we discussed in the previous section.

The computational domain is the square $[0, 10] \times [0, 10]$ mm^2 composed by a homogeneous isotropic material with density and Lamé coefficients given by

$$\rho = 1000 \text{ Kgr/m}^3, \quad \lambda = 3.45 \times 10^9 \text{ Pa}, \quad \mu = 2.04 \times 10^9 \text{ Pa}. \tag{3.8}$$

We introduce an initial condition on the velocity field centered on $(x_c, z_c) = (5, 5)$ mm,

$$\mathbf{v}((x, z), t = 0) = 0.1 \; F\left(\frac{r}{r_0}\right) \frac{\mathbf{r}}{r},$$

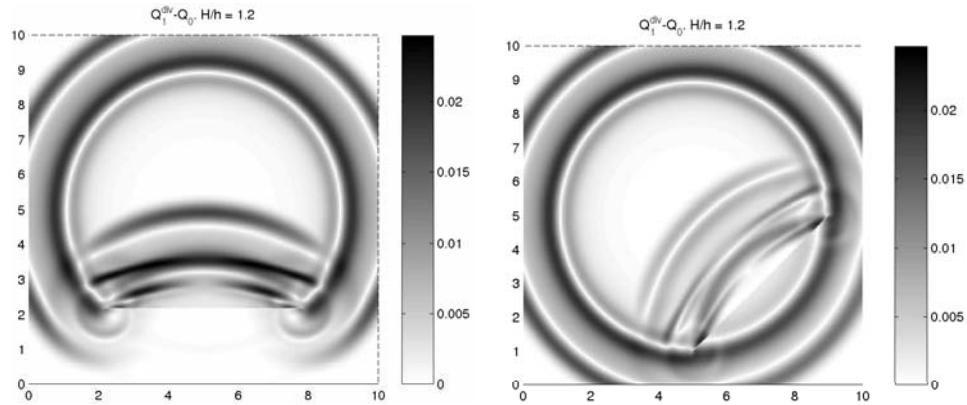
where $F(r)$ is supported in $[0, 1]$ and given by (for $r \in [0, 1]$)

$$F(r) = A_0 - A_1 \cos(2\pi r) + A_2 \cos(3\pi r) - A_3 \cos(6\pi r),$$

with $\mathbf{r} = (x - x_c, z - z_c)^t$, $r = \|\mathbf{r}\|$, $r_0 = 1.5$ mm, and

$$A_0 = 0.35875, \quad A_1 = 0.48829, \quad A_2 = 0.14128, \quad A_3 = 0.01168.$$

We consider a uniform mesh of squares using a discretization step $h = 0.025$ mm. The time discretization is done using a leapfrog scheme with the time step Δt chosen in such a way that the ratio $\Delta t/h$ is equal to the maximal value that ensures the stability. Perfectly matched layers [11, 16] are used to simulate a nonbounded domain in all of the boundaries.



(a) $t = 1.7310\ \mu$s (b) $t = 1.7310\ \mu$s

FIG. 3.2. $Q_1^{\rm div} - Q_0$. *Isotropic medium. Modulus of the velocity field.* $H/h = 1.2$.

In the first experiment we consider a plane horizontal crack

$$(x, z) = (5 + 2\sqrt{2}(2t - 1), 5 - 2\sqrt{2}) \text{ mm}, \qquad t \in [0, 1], \tag{3.9}$$

that we discretize using a uniform mesh of step $H = Rh$. The method converges, and we obtain good results for reasonable values for the parameter R (in the interval $[0.75, 3]$). In Figure 3.2(a) we show a snapshot of the modulus of the velocity field for $R = 1.2$. We can see the incident pressure wave that has reached the boundary of the computational domain, the reflected waves, and the scattered waves created by the two tips of the crack.

In the second experiment we treat a plane diagonal defect given by

$$(x, z) = (5 + 4t, 1 + 4t) \text{ mm}, \qquad t \in [0, 1], \tag{3.10}$$

that is, the same obstacle considered in the previous paragraph rotated by $\pi/4$ radians with respect to (x_c, z_c), the center of the initial condition. We point out that the initial condition satisfies

$$v((x, z), t = 0) \;=\; Q\; v((\tilde{x}, \tilde{z}), t = 0),$$

with

$$(\tilde{x}, \tilde{z})^t \;=\; Q^t\; (x, z)^t \;+\; (x_c, z_c)^t, \qquad Q \;=\; \begin{pmatrix} \cos(\frac{\pi}{4}) & \sin(\frac{\pi}{4}) \\ -\sin(\frac{\pi}{4}) & \cos(\frac{\pi}{4}) \end{pmatrix}.$$

As the domain is isotropic, the velocity field obtained with the diagonal crack, denoted by $\mathbf{v}_{diag}$, and the one obtained with the horizontal one, denoted by $\mathbf{v}_{hor}$, satisfy

$$\mathbf{v}_{hor}(x, z) \;=\; Q\; \mathbf{v}_{diag}(\tilde{x}, \tilde{z}) \quad \text{and} \quad |\mathbf{v}_{hor}|(x, z) \;=\; |\mathbf{v}_{diag}|(\tilde{x}, \tilde{z}).$$

In the same way as in the horizontal case, we discretize the Lagrange multiplier using a uniform grid of step $H = Rh$, with several values for the parameter R. However, this time, the approximated solution does not seem to converge towards the physical solution (see, for instance, Figure 3.2(b) for $R = 1.2$). The incident wave is not completely reflected but also transmitted through the interface. The scattered waves created by the tips of the crack are also poorly approximated.

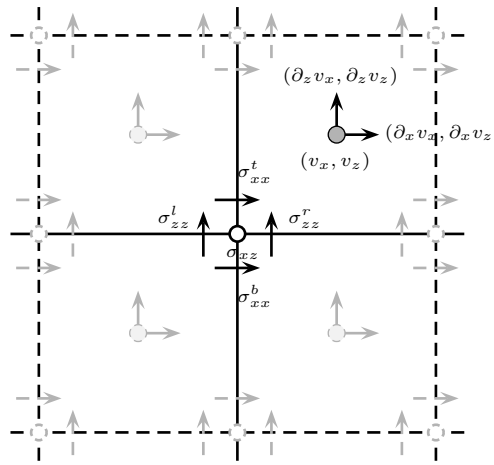


FIG. 4.1. *Degrees of freedom of the modified mixed finite element* $Q_1^{\mathrm{div}} \times P_1^{disc}$.

4. The modified element $Q_1^{\mathrm{div}} - P_1^{disc}$.

4.1. Presentation of the modified element. As explained in section 3.1, one of the difficulties of the convergence analysis of the $Q_1^{\mathrm{div}} \times P_1^{disc}$, even in the absence of an obstacle, was the fact that inclusion (3.7) is not satisfied. In order to avoid this problem, and following the same ideas the authors introduced in [10], we propose to modify the space M_h in such a way that

$$\mathrm{div}\left(\underline{\underline{X}}_h^{sym}\right) \subset \underline{M}_h, \tag{4.1}$$

which easily implies (3.7) and could get rid of the lack of convergence observed in section 3.2 and simplify the analysis. That is why we have chosen to discretize the pressure in the space

$$\underline{M}_h = \underline{M}_h^1 \quad \text{with} \quad \underline{M}_h^1 = \left\{ \mathbf{w}_h \in \underline{M} \ / \ \forall\ K \in \mathcal{T}_h,\ \mathbf{w}_{h|K} \in (P_1(K))^2 \right\}.$$

With this enrichment of the approximation space, we will have six degrees of freedom per element on the unknown $\mathbf{v}_h$ as shown in Figure 4.1. Since $\underline{M}_h^0 \subset \underline{M}_h^1$ we have obviously

$$\inf_{\mathbf{w}_h \in \underline{M}_h} \|\mathbf{v} - \mathbf{w}_h\|_\rho \ \leq \ \inf_{\mathbf{w}_h^0 \in \underline{M}_h^0} \|\mathbf{v} - \mathbf{w}_h^0\|_\rho,$$

and, in consequence, the approximation properties (2.9) are still satisfied.

REMARK 2. Assuming (4.1) and that the density is constant on each element, we have that

$$\boldsymbol{\tau}_h \in \underline{\underline{X}}_h \Longrightarrow \mathbf{w}_h := \frac{1}{\rho}\mathrm{div}(\boldsymbol{\tau}_h) \in \underline{M}_h.$$

Introducing this particular test function in the second equation of (2.10), we obtain

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_C \mathbf{v}_h \cdot \mathrm{div}(\boldsymbol{\tau}_h) - \int_C \frac{1}{\rho}\mathrm{div}(\boldsymbol{\sigma}_h) \cdot \mathrm{div}(\boldsymbol{\tau}_h) = \int_C \frac{1}{\rho}\ \mathbf{f}\ \cdot \mathrm{div}(\boldsymbol{\tau}_h).$$

Deriving with respect to time the first and third equations of (2.10) and using the last expression, we deduce that our variational formulation implies the following second order formulation:

$$\begin{cases} \text{Find } (\boldsymbol{\sigma}_h(t), \tilde{\boldsymbol{\lambda}}_H(t)) \in \underline{\underline{X}}_h \times \underline{\mathcal{G}}_H \text{ such that } \ \forall \ \ (\boldsymbol{\tau}_h, \boldsymbol{\mu}_H) \in \underline{\underline{X}}_h \times \underline{\mathcal{G}}_H, \\ \dfrac{\mathrm{d}^2}{\mathrm{d}t^2} \displaystyle\int_C \mathbf{A}\boldsymbol{\sigma}_h : \boldsymbol{\tau}_h + \int_C \frac{1}{\rho}\mathrm{div}(\boldsymbol{\tau}_h) \cdot \mathrm{div}(\boldsymbol{\sigma}_h) - \int_\Gamma \boldsymbol{\tau}_h \mathbf{n} \ \cdot \tilde{\boldsymbol{\lambda}}_H = \int_C -\frac{1}{\rho} f \cdot \mathrm{div}(\boldsymbol{\tau}_h), \\ \displaystyle\int_\Gamma \boldsymbol{\sigma}_h \mathbf{n} \ \cdot \boldsymbol{\mu}_H = 0, \\ \boldsymbol{\sigma}_h(t=0) = \boldsymbol{\sigma}_{h,0}, \qquad \dfrac{\partial \boldsymbol{\sigma}_h}{\partial t}(t=0) = \mathbf{C}\ \varepsilon(\mathbf{v}_{h,0}), \end{cases}$$

where $\tilde{\boldsymbol{\lambda}}_H = \dfrac{\partial \boldsymbol{\lambda}_H}{\partial t}$. The nature of this problem is close to the ones analyzed in [18, 29].

4.2. Some comments about the convergence of the modified element. From the numerical point of view we observe that the method converges under a compatibility condition of the form (3.6) between the two meshes. In section 4.5, repeating the numerical experiment of section 3.2 using the $Q_1^{\mathrm{div}} \times P_1^{disc}$ element, we observe that the results are similar regardless of the orientation of the crack, concluding that the method converges. In section 5 we compute numerically the order of convergence of the new method for a particular case.

From the theoretical point of view the convergence proof in this case is not a straightforward generalization of the result obtained in [10] for the scalar case. The main difficulty comes from the nonstandard regularity required to obtain the approximation properties for $\underline{\underline{X}}_h^{sym}$ (see (3.3)). Indeed, the maximal regularity (in space) of the stress tensor in the case of a domain with a crack is $\boldsymbol{\sigma} \in (H^{\frac{1}{2}-\varepsilon}(\mathrm{div}, C))^2$, and $\boldsymbol{\sigma}$ is symmetric. This regularity is not sufficient to obtain (3.3), and thus we cannot conclude (see [10] for more details on the differences with the scalar case). However, guided by the results obtained for the scalar case, we conjecture the following result.

CONJECTURE 4.1. [illegible] $\mathbf{f} \in C^0([0,T], \underline{M})$ $\boldsymbol{\sigma}_0 \in \underline{\underline{X}}^{sym}$ $\mathbf{v}_0 \in \underline{M}$ [illegible] (2.3) $(\boldsymbol{\sigma}, \mathbf{v}, \boldsymbol{\lambda})$ [illegible] (2.6). [illegible] $(\boldsymbol{\sigma}_h, \mathbf{v}_h, \boldsymbol{\lambda}_H)$ [illegible] (2.10) [illegible] $(\boldsymbol{\sigma}_{h,0}, \mathbf{v}_{h,0})$ [illegible] $(\boldsymbol{\sigma}_0, \mathbf{v}_0)$ [illegible] (3.6). [illegible] $(\boldsymbol{\sigma}, \mathbf{v}, \boldsymbol{\lambda}) \in C^2\big([0,T], (H^{\frac{1}{2}-\varepsilon}(\mathrm{div}, C))^2 \times (H^{\frac{1}{2}-\varepsilon}(C))^2 \times (H^{1-\varepsilon}(\Gamma))^2\big)$

$$\begin{aligned} &\|\boldsymbol{\sigma}_h - \boldsymbol{\sigma}\|_{C^0([0,T];\underline{\underline{X}}^{sym})} \ + \ \|\mathbf{v}_h - \mathbf{v}\|_{C^0([0,T];\underline{M})} \ + \ \|\boldsymbol{\lambda}_H - \boldsymbol{\lambda}\|_{C^0([0,T];\underline{\mathcal{G}})} \ \leq \\ &\mathcal{C}\left(h^{\frac{1}{2}-\varepsilon} + H^{\frac{1}{2}-\varepsilon}\right)(1+T)\left(\|\boldsymbol{\sigma}\|_{C^2([0,T];(H^{\frac{1}{2}-\varepsilon}(\mathrm{div},C))^2)} \ + \right. \\ &\left. \|\mathbf{v}\|_{C^2([0,T];(H^{\frac{1}{2}-\varepsilon}(C))^2)} \ + \ \|\boldsymbol{\lambda}\|_{C^2([0,T];(H^{1-\varepsilon}(\Gamma))^2)}\right). \end{aligned} \tag{4.2}$$

The numerical results obtained in section 5 confirm this result.

4.3. Dispersion analysis of the modified element. It is useful and classical to perform a dispersion analysis in order to specify the properties of a discrete scheme (in the absence of an obstacle) (e.g., [36, 14]). This consists in studying the behavior of discrete plane waves propagated with the scheme. We will show that four spurious modes that contaminate the discrete solution appear, for the modified element, because four new degrees of freedom per element were added to the velocity field.

Assume that our computational domain $\Omega = \mathbb{R}^2$ is homogeneous and that we use a uniform mesh composed by squares of edge h. In this way, the degrees of freedom of the stress tensor are placed at the vertices of the squares, that is, at the points $(x_i, z_j) = (ih, jh)$. We thus have

$$(\sigma^t_{x,x})_{i,j}, \qquad (\sigma^b_{x,x})_{i,j}, \qquad (\sigma^r_{z,z})_{i,j}, \qquad (\sigma^l_{z,z})_{i,j}, \qquad (\sigma_{x,z})_{i,j}, \qquad (i,j) \in \mathbb{Z} \times \mathbb{Z}.$$

The degrees of freedom for the velocity field are defined at the center of each element, that is, at the points $(x_{i+\frac{1}{2}}, z_{j+\frac{1}{2}}) = ((i+1/2)h, (j+1/2)h)$. The corresponding unknowns are denoted by

$$\begin{array}{llll} (v_x)_{i+\frac{1}{2},j+\frac{1}{2}}, & (\partial_x v_x)_{i+\frac{1}{2},j+\frac{1}{2}}, & (\partial_z v_x)_{i+\frac{1}{2},j+\frac{1}{2}}, & (i,j) \in \mathbb{Z} \times \mathbb{Z}, \\ (v_z)_{i+\frac{1}{2},j+\frac{1}{2}}, & (\partial_x v_z)_{i+\frac{1}{2},j+\frac{1}{2}}, & (\partial_z v_z)_{i+\frac{1}{2},j+\frac{1}{2}}, & (i,j) \in \mathbb{Z} \times \mathbb{Z}. \end{array}$$

The numerical scheme is thus given by

$$\left| \begin{array}{rcl} \dfrac{\mathrm{d}}{\mathrm{d}t}(v_x)_{i+\frac{1}{2},j+\frac{1}{2}} & = & \dfrac{1}{2\rho h}\left\{(\sigma^t_{xx})_{i+1,j} - (\sigma^t_{xx})_{i,j} + (\sigma^b_{xx})_{i+1,j+1} - (\sigma^b_{xx})_{i,j+1}\right. \\ & & \qquad \left. + (\sigma_{xz})_{i+1,j+1} - (\sigma_{xz})_{i+1,j} + (\sigma_{xz})_{i,j+1} - (\sigma_{xz})_{i,j}\right\}, \\ \dfrac{\mathrm{d}}{\mathrm{d}t}(\partial_x v_x)_{i+\frac{1}{2},j+\frac{1}{2}} & = & \dfrac{1}{\sqrt{12}\rho h}\left\{(\sigma_{xz})_{i+1,j+1} + (\sigma_{xz})_{i,j} - (\sigma_{xz})_{i+1,j} - (\sigma_{xz})_{i,j+1}\right\}, \\ \dfrac{\mathrm{d}}{\mathrm{d}t}(\partial_z v_x)_{i+\frac{1}{2},j+\frac{1}{2}} & = & \dfrac{1}{\sqrt{12}\rho h}\left\{(\sigma^b_{xx})_{i+1,j+1} + (\sigma^t_{xx})_{i,j} - (\sigma^t_{xx})_{i+1,j} - (\sigma^b_{xx})_{i,j+1}\right\}, \end{array} \right.$$

$$\left| \begin{array}{rcl} \dfrac{\mathrm{d}}{\mathrm{d}t}(v_z)_{i+\frac{1}{2},j+\frac{1}{2}} & = & \dfrac{1}{2\rho h}\left\{(\sigma_{xz})_{i+1,j} - (\sigma_{xz})_{i,j} + (\sigma_{xz})_{i+1,j+1} - (\sigma_{xz})_{i,j+1}\right. \\ & & \qquad \left. + (\sigma^r_{zz})_{i,j+1} - (\sigma^r_{zz})_{i,j} + (\sigma^l_{zz})_{i+1,j+1} - (\sigma^l_{zz})_{i+1,j}\right\}, \\ \dfrac{\mathrm{d}}{\mathrm{d}t}(\partial_x v_z)_{i+\frac{1}{2},j+\frac{1}{2}} & = & \dfrac{1}{\sqrt{12}\rho h}\left\{(\sigma^l_{zz})_{i+1,j+1} + (\sigma^r_{zz})_{i,j} - (\sigma^l_{zz})_{i+1,j} - (\sigma^r_{zz})_{i,j+1}\right\}, \\ \dfrac{\mathrm{d}}{\mathrm{d}t}(\partial_z v_z)_{i+\frac{1}{2},j+\frac{1}{2}} & = & \dfrac{1}{\sqrt{12}\rho h}\left\{(\sigma_{xz})_{i+1,j+1} + (\sigma_{xz})_{i,j} - (\sigma_{xz})_{i+1,j} - (\sigma_{xz})_{i,j+1}\right\}. \end{array} \right.$$

$$\begin{bmatrix} \frac{A_{1,1}}{2} & 0 & \frac{A_{1,2}}{4} & \frac{A_{1,2}}{4} & \frac{A_{1,3}}{2} \\ 0 & \frac{A_{1,1}}{2} & \frac{A_{1,2}}{4} & \frac{A_{1,2}}{4} & \frac{A_{1,3}}{2} \\ \frac{A_{1,2}}{4} & \frac{A_{1,2}}{4} & \frac{A_{2,2}}{2} & 0 & \frac{A_{2,3}}{2} \\ \frac{A_{1,2}}{4} & \frac{A_{1,2}}{4} & 0 & \frac{A_{2,2}}{2} & \frac{A_{2,3}}{2} \\ \frac{A_{1,3}}{2} & \frac{A_{1,3}}{2} & \frac{A_{2,3}}{2} & \frac{A_{2,3}}{2} & A_{3,3} \end{bmatrix} \begin{bmatrix} \frac{\mathrm{d}}{\mathrm{d}t}(\sigma^t_{xx})_{i,j} \\ \frac{\mathrm{d}}{\mathrm{d}t}(\sigma^b_{xx})_{i,j} \\ \frac{\mathrm{d}}{\mathrm{d}t}(\sigma^r_{zz})_{i,j} \\ \frac{\mathrm{d}}{\mathrm{d}t}(\sigma^l_{zz})_{i,j} \\ \frac{\mathrm{d}}{\mathrm{d}t}(\sigma_{xz})_{i,j} \end{bmatrix} = \begin{bmatrix} (B_1)_{i,j} \\ (B_2)_{i,j} \\ (B_3)_{i,j} \\ (B_4)_{i,j} \\ (B_5)_{i,j} \end{bmatrix},$$

with

$$
\begin{aligned}
(B_1)_{i,j} &= \frac{1}{2h}\left((v_x)_{i+\frac12,j+\frac12}-(v_x)_{i-\frac12,j+\frac12}-\frac{1}{\sqrt{3}}((\partial_z v_x)_{i+\frac12,j+\frac12}-(\partial_z v_x)_{i-\frac12,j+\frac12})\right),\\
(B_2)_{i,j} &= \frac{1}{2h}\left((v_x)_{i+\frac12,j-\frac12}-(v_x)_{i-\frac12,j-\frac12}+\frac{1}{\sqrt{3}}((\partial_z v_x)_{i+\frac12,j-\frac12}-(\partial_z v_x)_{i-\frac12,j-\frac12})\right),\\
(B_3)_{i,j} &= \frac{1}{2h}\left((v_z)_{i+\frac12,j+\frac12}-(v_z)_{i+\frac12,j-\frac12}-\frac{1}{\sqrt{3}}((\partial_x v_z)_{i+\frac12,j+\frac12}-(\partial_x v_z)_{i+\frac12,j-\frac12})\right),\\
(B_4)_{i,j} &= \frac{1}{2h}\left((v_z)_{i-\frac12,j+\frac12}-(v_z)_{i-\frac12,j-\frac12}+\frac{1}{\sqrt{3}}((\partial_x v_z)_{i-\frac12,j+\frac12}-(\partial_x v_z)_{i-\frac12,j-\frac12})\right),\\
(B_5)_{i,j} &= \frac{1}{2h}\left((v_x)_{i+\frac12,j+\frac12}-(v_x)_{i+\frac12,j-\frac12}+(v_x)_{i-\frac12,j+\frac12}-(v_x)_{i-\frac12,j-\frac12}\right)\\
&\quad+\frac{1}{2h}\left((v_z)_{i+\frac12,j+\frac12}-(v_z)_{i-\frac12,j+\frac12}+(v_z)_{i+\frac12,j-\frac12}-(v_z)_{i-\frac12,j-\frac12}\right)\\
&\quad+\frac{1}{\sqrt{12}h}\left((\partial_x v_x)_{i+\frac12,j-\frac12}-(\partial_x v_x)_{i+\frac12,j+\frac12}+(\partial_x v_x)_{i-\frac12,j+\frac12}-(\partial_x v_x)_{i-\frac12,j-\frac12}\right)\\
&\quad+\frac{1}{\sqrt{12}h}\left((\partial_z v_z)_{i-\frac12,j+\frac12}-(\partial_z v_z)_{i+\frac12,j+\frac12}+(\partial_z v_z)_{i+\frac12,j-\frac12}-(\partial_z v_z)_{i-\frac12,j-\frac12}\right).
\end{aligned}
$$

For the sake of simplicity we treat the isotropic case; that is, we assume that

$$
\begin{pmatrix} A_{1,1} & A_{1,2} & A_{1,3}\\ A_{1,2} & A_{2,2} & A_{2,3}\\ A_{1,3} & A_{2,3} & A_{3,3}\end{pmatrix}
=\begin{pmatrix} \frac{2\mu+\lambda}{4\mu(\lambda+\mu)} & -\frac{\lambda}{4\mu(\lambda+\mu)} & 0\\ -\frac{\lambda}{4\mu(\lambda+\mu)} & \frac{2\mu+\lambda}{4\mu(\lambda+\mu)} & 0\\ 0 & 0 & \frac{1}{\mu}\end{pmatrix}, \tag{4.3}
$$

where λ and μ are the Lamé coefficients of the material. The next step is to eliminate the unknowns associated to the stress tensor and to consider a plane wave with frequency ω and wave vector $\boldsymbol{k}=(k_x,k_z)=k(\cos\ \theta,\sin\ \theta)$:

$$
\begin{aligned}
\begin{pmatrix}(v_x)_{i+\frac12,j+\frac12}\\(v_z)_{i+\frac12,j+\frac12}\\(\partial_x v_x)_{i+\frac12,j+\frac12}\\(\partial_z v_z)_{i+\frac12,j+\frac12}\\(\partial_x v_z)_{i+\frac12,j+\frac12}\\(\partial_z v_x)_{i+\frac12,j+\frac12}\end{pmatrix} &= \mathbf{V}\exp\left[i\left(ct-(\cos(\theta)x_{i+\frac12}+\sin(\theta)z_{j+\frac12})\right)\right],\\
\mathbf{V} &= [\ \mathbf{v_x}\ ,\ \mathbf{v_z}\ ,\ \partial_{\mathbf{x}}\mathbf{v_x}\ ,\ \partial_{\mathbf{z}}\mathbf{v_z}\ ,\ \partial_{\mathbf{x}}\mathbf{v_z}\ ,\ \partial_{\mathbf{z}}\mathbf{v_x}\ ]^t,
\end{aligned} \tag{4.4}
$$

where $c=\omega/k$ is the phase velocity and $\mathbf{V}$ is the amplitude vector. Plugging this expression into the numerical scheme, we observe that the couple $(\omega^2,\boldsymbol{V})$ has to satisfy the following eigenvalue problem:

$$
\mathcal{K}_h\boldsymbol{V} = \frac{\omega^2}{k^2}\boldsymbol{V},\quad \text{with } k=|\boldsymbol{k}|, \tag{4.5}
$$

where the nonzero elements of the matrix $\mathcal{K}_h$ are given in Appendix A.

The matrix $\mathcal{K}_h$ is Hermitian; thus, there are six real eigenvalues with six orthogonal eigenvectors. Performing a Taylor expansion we obtain the following results. For the two first couples denoted by $(\omega_T^2, \boldsymbol{V}_T),\ T \in \{P, S\}$, we have

$$\left|\begin{array}{rcl} c_P = \dfrac{\omega_P}{k} & = & \sqrt{\dfrac{\lambda+2\mu}{\rho}} + \mathcal{O}(kh)^2, \\ \mathbf{V_P} & = & [\ \cos\theta\ ,\ \sin\theta\ ,\ 0\ ,\ 0\ ,\ 0\ ,\ 0\ ]^t + \mathcal{O}(kh), \end{array}\right. \tag{4.6}$$

$$\left|\begin{array}{rcl} c_S = \dfrac{\omega_S}{k} & = & \sqrt{\dfrac{\mu}{\rho}} + \mathcal{O}(kh)^2, \\ \mathbf{V_S} & = & [\ -\sin\theta\ ,\ \cos\theta\ ,\ 0\ ,\ 0\ ,\ 0\ ,\ 0\ ]^t + \mathcal{O}(kh). \end{array}\right. \tag{4.7}$$

The phase velocity of the these two waves is a second order approximation of the phase velocity of the continuous waves. The other four solutions of the eigenvalue problem are spurious modes produced by the introduction of the additional degrees of freedom and are given by

$$\left|\begin{array}{rcl} c_{spur1} = \dfrac{\omega_{spur1}}{k} & = & \mathcal{O}(kh), \\ \mathbf{V_{spur1}} & = & [\ 0\ ,\ 0\ ,\ 1\ ,\ 0\ ,\ 0\ ,\ 0\ ]^t + \mathcal{O}(kh), \end{array}\right. \tag{4.8}$$

$$\left|\begin{array}{rcl} c_{spur2} = \dfrac{\omega_{spur2}}{k} & = & \mathcal{O}(kh), \\ \mathbf{V_{spur2}} & = & [\ 0\ ,\ 0\ ,\ 0\ ,\ 1\ ,\ 0\ ,\ 0\ ]^t + \mathcal{O}(kh), \end{array}\right. \tag{4.9}$$

$$\left|\begin{array}{rcl} c_{spur3} = \dfrac{\omega_{spur3}}{k} & = & \sqrt{\dfrac{4\sin^2(\theta)(\lambda+\mu)\mu}{3\rho(\lambda+2\mu)}} + \mathcal{O}(kh)^2, \\ \mathbf{V_{spur3}} & = & [\ 0\ ,\ 0\ ,\ 0\ ,\ 0\ ,\ 1\ ,\ 0\ ]^t + \mathcal{O}(kh), \end{array}\right. \tag{4.10}$$

$$\left|\begin{array}{rcl} c_{spur4} = \dfrac{\omega_{spur4}}{k} & = & \sqrt{\dfrac{4\cos^2(\theta)(\lambda+\mu)\mu}{3\rho(\lambda+2\mu)}} + \mathcal{O}(kh)^2, \\ \mathbf{V_{spur4}} & = & [\ 0\ ,\ 0\ ,\ 0\ ,\ 0\ ,\ 0\ ,\ 1\ ]^t + \mathcal{O}(kh). \end{array}\right. \tag{4.11}$$

We remark that the first two spurious modes propagate with a phase velocity that goes to zero, and thus they are not really visible in the numerical simulations. The phase velocity of the other two modes is nonzero and always smaller than the shear wave velocity. Therefore, these spurious waves will propagate and pollute the physical solution.

4.4. Damping of the spurious modes. As we have seen in the previous section, the modified element gives rise to some spurious modes. In this section we propose a way to damp the amplitude of these modes (without damping the "physical" part of the solution), so that they do not pollute too much the approximate solution.

The approximate space $\underline{M}_h$ can be decomposed as

$$\underline{M}_h = \underline{M}_h^0 \oplus \underline{M}_h^r,$$

where $\underline{M}_h^0$ is the space of piecewise constants on each component and $\underline{M}_h^r$ is its orthogonal complement (for the L^2 scalar product). The space $\underline{M}_h^r$ is composed of P^1 discontinuous functions with a vanishing mean value per element and per component.

From the dispersion analysis, we observe that the main components of the spurious modes (the $\mathcal{O}(1)$ part) belong to $\underline{M}_h^r$. In order to damp this main part, we introduce the L^2 orthogonal projection on $\underline{M}_h^r$, which we denote by $P_{\underline{M}_h^r}$, defined for any $\mathbf{v}_h \in \underline{M}_h$ as

$$P_{\underline{M}_h^r}(\mathbf{v}_h) \in \underline{M}_h^r \quad \text{and} \quad (P_{\underline{M}_h^r}(\mathbf{v}_h), \mathbf{w}_h) = (\mathbf{v}_h, \mathbf{w}_h) \quad \forall\, \mathbf{w}_h \in \underline{M}_h^r.$$

The approximate problem with damping consists in finding $(\boldsymbol{\sigma}_h, \mathbf{v}_h, \boldsymbol{\lambda}_H) \in \underline{\underline{X}}_h^{sym} \times \underline{M}_h \times \underline{\mathcal{G}}_H$ such that

(4.12)
$$\begin{cases} \text{Find } (\boldsymbol{\sigma}_h(t), \mathbf{v}_h(t), \boldsymbol{\lambda}_H(t)) \in \underline{\underline{X}}_h^{sym} \times \underline{M}_h \times \underline{\mathcal{G}}_h \text{ satisfying} \\ \dfrac{\mathrm{d}}{\mathrm{d}t} a(\boldsymbol{\sigma}_h, \boldsymbol{\tau}_h) + b(\boldsymbol{\tau}_h, \mathbf{v}_h) - \langle \boldsymbol{\tau}_h\, \mathbf{n}, \boldsymbol{\lambda}_H \rangle_\Gamma = 0 \qquad \forall\, \boldsymbol{\tau}_h \in \underline{\underline{X}}_h^{sym}, \\ \dfrac{\mathrm{d}}{\mathrm{d}t} (\mathbf{v}_h, \mathbf{w}_h)_\rho + (P_{\underline{M}_h^r}(\mathbf{v}_h), \mathbf{w}_h)_\beta - b(\boldsymbol{\sigma}_h, \mathbf{w}_h) = (\mathbf{f}, \mathbf{w}_h) \qquad \forall\, \mathbf{w}_h \in \underline{M}_h, \\ \langle \boldsymbol{\sigma}_h\, \mathbf{n}, \boldsymbol{\mu}_H \rangle_\Gamma = 0 \qquad \forall\, \boldsymbol{\mu}_h \in \underline{\mathcal{G}}_H. \end{cases}$$

In this system β represents a damping parameter, which is chosen as a positive constant in the applications. The case $\beta = 0$ gives back the nondamped problem, while a strictly positive β corresponds to a dissipative problem. In section 5 we will explain how to choose the value for this parameter.

4.5. Numerical illustrations using the modified element. Let us now show some numerical illustrations of the behavior of the fictitious domain method with the new finite element space. The numerical experiments that we have considered are the same as in section 3.2 and will allow us to compare both finite elements.

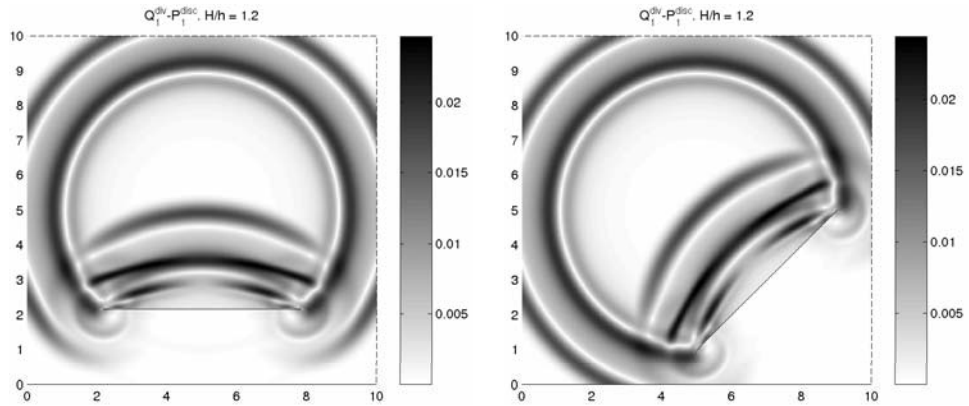
Once again we discretize the horizontal crack defined by (3.9) using a uniform mesh of step $H = Rh$. The results obtained with the new mixed finite element $Q_1^{\mathrm{div}} - P_1^{disc}$ are similar to those given by the $Q_1^{\mathrm{div}} - Q_0$ element. The method converges for reasonable values of the parameter R (in the interval $[0.75, 3]$). In Figure 4.2(a) we can see the results for $R = 1.2$.

We now consider the diagonal crack defined by the expression (3.10). We recall that the continuous problem is a rotation of $\pi/4$ radians with respect to the point $(x_c, z_c) = (5, 5)$.

The Lagrange multiplier is again discretized using an uniform mesh of step $H = Rh$. Contrary to the results obtained with the element $Q_1^{\mathrm{div}} - Q_0$, the ones given by the modified element $Q_1^{\mathrm{div}} - P_1^{disc}$ converge towards the physical solution when choosing reasonable values for the ratio H/h. In Figure 4.2(b), this time the incident wave is almost completely reflected by the obstacle. The scattered waves created by the tips of the crack are well approximated.

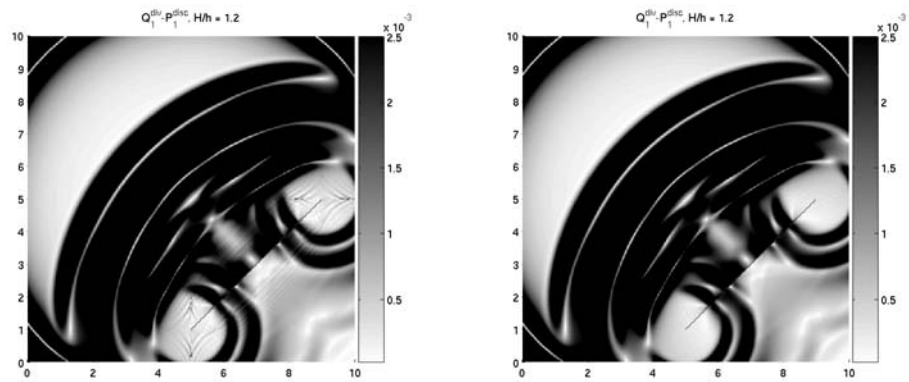
As explained before, the enrichment of the $\underline{M}_h$ approximation space introduces spurious modes in the solution. Although the amplitude of these nonphysical waves goes to zero with the size of the discretization step, it is still significant for a usual choice of the discretization parameters, typically corresponding to 20 points per wavelength. These spurious modes are, for example, visible in the results presented in Figure 4.3(a) where we have changed the scale to amplify the spurious phenomena.

In order to study in more detail these phenomena we represent in Figure 4.4 the evolution in time of the modulus of the velocity field at three points: $\mathbf{x_1} = (x_1, z_1) = (6.5, 3.5)$ mm, $\mathbf{x_2} = (x_2, z_2) = (7.5, 2.5)$ mm, and $\mathbf{x_3} = (x_3, z_3) = (5, 0.5)$ mm. The first two points are centered with respect to the crack, one behind and the other in



(a) $t = 1.7310\ \mu$s (b) $t = 1.7310\ \mu$s

FIG. 4.2. $Q_1^{\text{div}} - P_1^{disc}$. *Isotropic medium. Modulus of the velocity field.* $H/h = 1.2$.



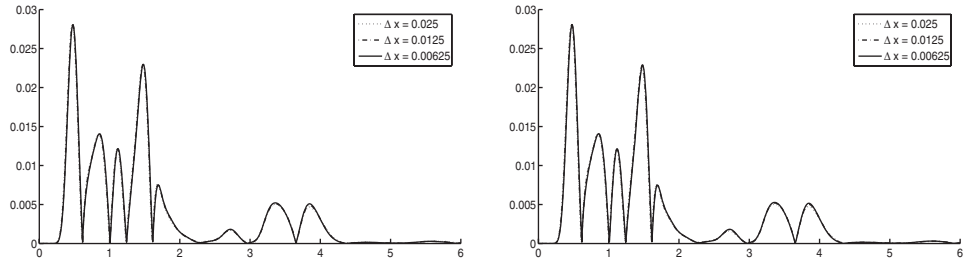
(a) $Q_1^{\text{div}} - P_1^{disc}$. $\beta = 0$ (b) $Q_1^{\text{div}} - P_1^{disc}$. $\beta = 5\rho 10^6$

FIG. 4.3. *Modulus of the velocity field at* $t = 2.5965\ \mu$s.

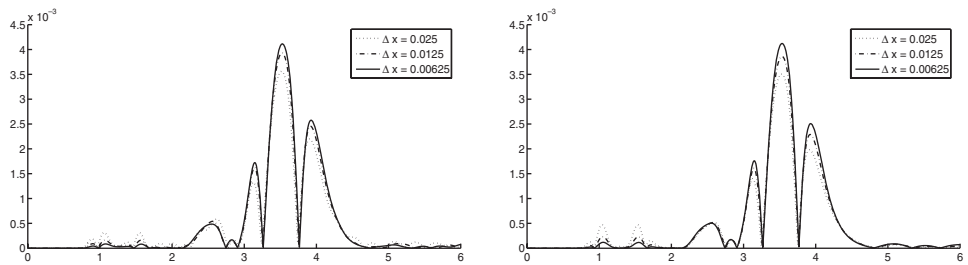
front of it. The third point is located near the lower tip of the crack, where the spurious waves seem stronger. To determine the speed of convergence of the method, we use three different meshes with a space discretization step of $h = 0.025, 0.0125$, and 0.00625 mm, respectively.

The results obtained at the first point (in front of the crack) are already good with the coarse mesh (see Figure 4.4, top left). The amplitude of the spurious waves is very small with respect to the amplitude of the incident ($t \in [0.25, 1]\ \mu$s), reflected ($t \in [0.75, 2]\ \mu$s), and (first) scattered waves ($t \in [2.75, 4.25]\ \mu$s).

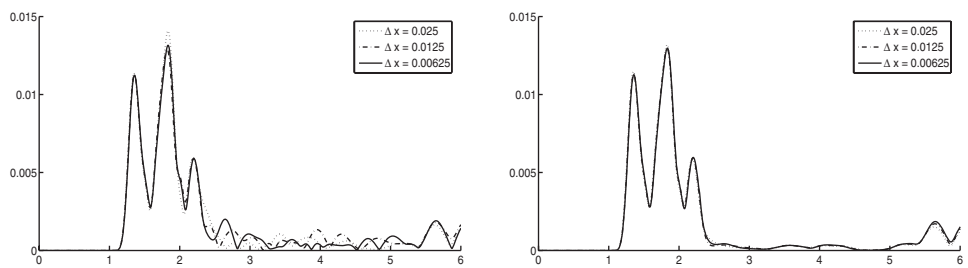
The second point is in the "shadow region," where the amplitude of the physical waves is about 15% smaller than the amplitude of the incident wave (see Figure 4.4, middle left). Consequently the error is more visible. In the time interval $t \in [0, 2]\ \mu$s we can see the part of the incident wave that has been transmitted across the obstacle. The amplitude of those waves goes to zero when we refine the mesh. From $t = 2$ to



(a) $|v|((6.5, 3.5)\ \text{mm}, t)$, $\beta = 0$ (b) $|v|((6.5, 3.5)\ \text{mm}, t)$, $\beta = 5\varrho 10^6$



(c) $|v|((7.5, 2.5)\ \text{mm}, t)$, $\beta = 0$ (d) $|v|((7.5, 2.5)\ \text{mm}, t)$, $\beta = 5\varrho 10^6$

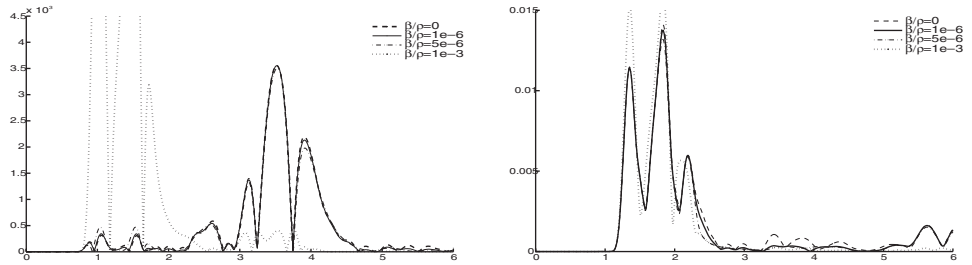


(e) $|v|((5.0, 0.5), t)$, $\beta = 0$ (f) $|v|((5.0, 0.5), t)$, $\beta = 5\varrho 10^6$

FIG. 4.4. *The modulus of the velocity* $|\mathbf{v}|(\mathbf{x_i}, t)$, $t \in [0, 6]$, $i \in \{1, 2, 3\}$, *computed using the* $Q_1^{\text{div}} \times P_1^{disc}$ *element with* $\beta = 0$ *on the left column and* $\beta = 5\varrho 10^6$ *on the right one.*

$t = 4.5$ μs we can see the first group of scattered waves. At the beginning and at the end of this time interval we can see some oscillations that come from the spurious modes introduced by the enrichment of the velocity field discretization space. The error on those amplitudes is about 2% of the amplitude of the incident wave.

At the third point, the spurious phenomena are much stronger (see Figure 4.4, bottom left). Here the amplitude of the incident and scattered waves is about 40% of the amplitude of the actual incident wave. As we can observe, comparing the solutions obtained with the different meshes, the method converges very slowly (see the time interval $t \in [2, 5.5]$ μs). This is due to the spurious modes created by the singularity on the tips of the crack (see also Figure 4.3(a)).



(a) $|v|((7.5, 2.5)\text{ mm}, t)$ (b) $|v|((5.0, 0.5)\text{ mm}, t)$

FIG. 4.5. *The modulus of the velocity* $|\mathbf{v}|(\mathbf{x_i}, t)$, $t \in [0,6]$ μs, $i \in \{2,3\}$, *computed using the* $Q_1^{\text{div}} \times P_1^{disc}$ *element with* $h = 0.025$ mm *and different values of the damping parameter* β.

These spurious phenomena can be reduced using a positive value of β, the damping coefficient. We will explain how to chose β in section 5.1. Let us analyze here the results obtained with $\beta = 5\varrho 10^6$. As we can see in Figure 4.3(b) the results seem to be better than those obtained with $\beta = 0$ (see Figure 4.3(a)). The signal obtained at the first point $\mathbf{x_1}$ is very similar for both choices (see Figure 4.4, top right). The results for the second point are also comparable. We remark that some oscillations on the time intervals $t \in [2,3]$ and $t \in [4.5,6]$ are removed with the damping. It is in the signal recorded on the third point where the effect of the damping of the spurious modes is more efficient (see Figure 4.4, bottom right). The oscillations observed with $\beta = 0$ are completely removed, and the method gives a good solution even with the coarsest mesh.

To illustrate the effect of the damping parameter β on the solution, we present in Figure 4.5 results obtained on the coarsest grid for different values of β. We have made the following observations: When we do not use any damping, the solution is polluted by spurious modes. On the other hand, the amplitude of the transmitted (nonphysical) waves through the crack increases as the value of the damping β increases. This is expected, because the limit case $\beta \to +\infty$ corresponds to seeking the velocity in Q_0, and we know that in this case the method does not converge. There is thus an optimal value for the damping parameter β to be determined so, as the spurious modes are damped, while the transmitted nonphysical wave remains small. In the next section we determine numerically the rate of convergence of the method for a particular geometry, and we discuss a procedure to choose the value of β.

5. Numerical error estimates. In this section we estimate numerically the order of convergence of the method. To do so, we consider solving the elastodynamic system on a disk $\Omega \subset \mathbb{R}^2$ with homogeneous Neumann boundary conditions on its boundary $\Gamma = \delta\Omega$. The geometry of the problem is presented in Figure 5.1.

To compute the solution we extend the unknowns in the domain of simple geometry C (see Figure 5.1) and use the fictitious domain formulation (2.10) with zero force term $\mathbf{f} = 0$ and an initial condition on the velocity field, given by

$$\mathbf{v}((x,z), t=0) = 0.1\ F\left(\frac{r}{r_0}\right)\frac{\mathbf{r} + \mathbf{r}^{\perp}}{r},$$

where $F(\cdot)$, $\mathbf{r}$, and r have been defined in section 3.2 and $r_0 = 1.5$ mm. The domain

of propagation is an isotropic medium with the density and Lamé coefficients given by (3.8). The extended domain C is truncated using perfectly matched layers [11, 16]. We consider the final time equal to $T = 5$ μs when both the pressure and shear waves have reached the boundary.

The fact of having a problem that is rotationally invariant allows us to compute a reference solution solving a one-dimensional problem. More precisely, rewriting (2.1) in polar coordinates

$$
\left|\begin{array}{rcll}
\rho\dfrac{\partial v_r}{\partial t} & = & \dfrac{\partial \sigma_{rr}}{\partial r} + \dfrac{1}{r}\dfrac{\partial \sigma_{r\theta}}{\partial \theta} + \dfrac{1}{r}(\sigma_{rr}-\sigma_{\theta\theta}) & \text{in } [0,R]\times[0,2\pi],\\
\rho\dfrac{\partial v_\theta}{\partial t} & = & \dfrac{\partial \sigma_{r\theta}}{\partial r} + \dfrac{1}{r}\dfrac{\partial \sigma_{\theta\theta}}{\partial \theta} + \dfrac{2}{r}\sigma_{r\theta} & \text{in } [0,R]\times[0,2\pi],\\
\dfrac{\partial \sigma_{rr}}{\partial t} & = & (2\mu+\lambda)\dfrac{\partial v_r}{\partial r} + \lambda\dfrac{v_r}{r} + \lambda\dfrac{1}{r}\dfrac{\partial v_\theta}{\partial \theta} & \text{in } [0,R]\times[0,2\pi],\\
\dfrac{\partial \sigma_{\theta\theta}}{\partial t} & = & (2\mu+\lambda)\left(\dfrac{1}{r}\dfrac{\partial v_\theta}{\partial \theta}+\dfrac{v_r}{r}\right) + \lambda\dfrac{\partial v_r}{\partial r} & \text{in } [0,R]\times[0,2\pi],\\
\dfrac{\partial \sigma_{r\theta}}{\partial t} & = & \mu\left(\dfrac{1}{r}\dfrac{\partial v_r}{\partial \theta} + \dfrac{\partial v_\theta}{\partial r} - \dfrac{v_\theta}{r}\right) & \text{in } [0,R]\times[0,2\pi],\\
\sigma_{rr}=0 \quad \sigma_{r\theta}=0 & & & \text{in } R\times[0,2\pi],
\end{array}\right. \tag{5.1}
$$

with the initial conditions

$$
v_s((r,\theta),t=0) = 0.1F\left(\frac{r}{r_0}\right), \quad s\in\{r,\theta\}, \tag{5.2}
$$

we remark that the solution depends only on r, and the former equations are equivalent to the two following decoupled one-dimensional problems:

$$
\left|\begin{array}{rcll}
\rho\dfrac{\partial v_r}{\partial t} & = & \dfrac{\partial \sigma_{rr}}{\partial r} + \dfrac{1}{r}(\sigma_{rr}-\sigma_{\theta\theta}) & \text{in } [0,R],\\
\dfrac{\partial \sigma_{rr}}{\partial t} & = & (2\mu+\lambda)\dfrac{\partial v_r}{\partial r} + \lambda\dfrac{v_r}{r} & \text{in } [0,R],\\
\dfrac{\partial \sigma_{\theta\theta}}{\partial t} & = & (2\mu+\lambda)\dfrac{v_r}{r} + \lambda\dfrac{\partial v_r}{\partial r} & \text{in } [0,R],\\
\sigma_{rr}=0 & & & \text{in } R,
\end{array}\right. \tag{5.3}
$$

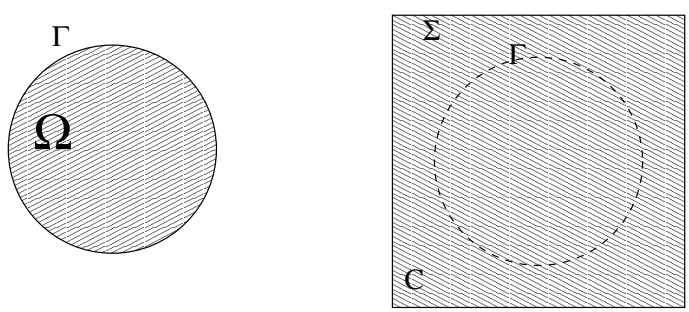
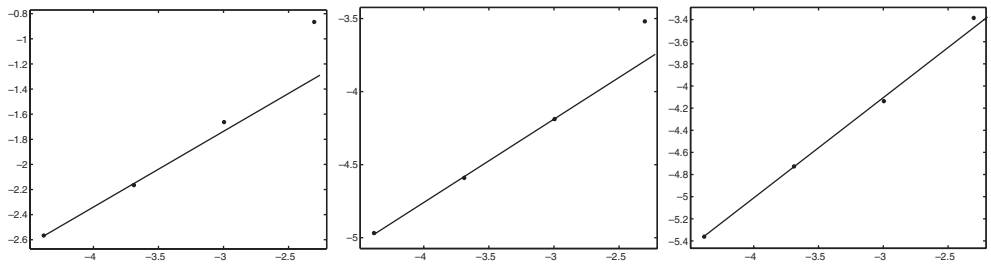


FIG. 5.1. *The geometry of the problem. On the left is the initial domain of propagation* Ω*, and on the right is the extended domain* C *introduced by the fictitious domain formulation of the problem.*



(a) $\sup_{t\leq T} \|\boldsymbol{\sigma} - \boldsymbol{\sigma}_h\|_{\underline{\underline{X}}}$, $s = 0.59$ (b) $\sup_{t\leq T} \|\mathbf{v} - \mathbf{v}_h\|_{\underline{M}}$, $s = 0.55$ (c) $\sup_{t\leq T} \|\boldsymbol{\lambda} - \boldsymbol{\lambda}_h\|_{\underline{L^2}}$, $s = 0.92$

FIG. 5.2. *Numerical error on* $\boldsymbol{\sigma}$, $\mathbf{v}$, *and* $\boldsymbol{\lambda}$ *versus the discretization step.*

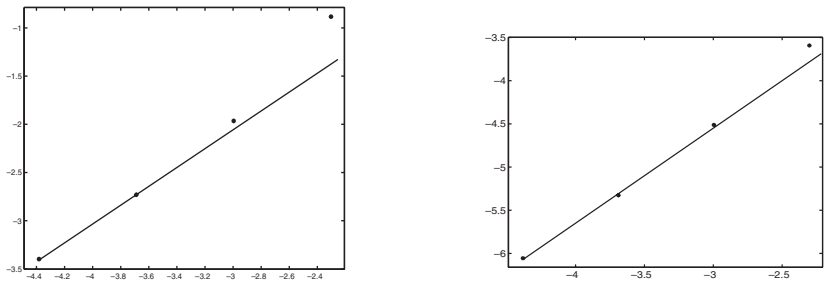
$$
\left|\begin{array}{lcll}
\rho\dfrac{\partial v_\theta}{\partial t} & = & \dfrac{\partial \sigma_{r\theta}}{\partial r} + \dfrac{2}{r}\sigma_{r\theta} & \text{in } [0,R]\times[0,2\pi], \\
\dfrac{\partial \sigma_{r\theta}}{\partial t} & = & \mu\left(\dfrac{\partial v_\theta}{\partial r} - \dfrac{v_\theta}{r}\right) & \text{in } [0,R]\times[0,2\pi], \\
\sigma_{r\theta} y = 0 & & & \text{in } R\times[0,2\pi].
\end{array}\right. \tag{5.4}
$$

We solve numerically those systems using piecewise constant functions for the discretization of the velocity field and continuous linear functions for the stress tensor. For the time discretization we use a leapfrog scheme. The reference solution is obtained using a very fine mesh ($h_{1d} \approx 1/800$). The two-dimensional problem is solved using four different meshes with $h_z = h_z = 1/10, 1/20, 1/40$, and $1/80$. We use the larger time step Δt authorized by the CFL condition. The mesh for the object is uniform and with a discretization step H such that $H/h \approx 4$ for each mesh. In all cases the damping parameter β is equal to zero. For each numerical experiment we compute the difference between the approximated solution and the reference solution. In Figure 5.2 we display the logarithm of the error on the stress tensor, the velocity field, and the Lagrange multiplier versus the logarithm of the discretization step. The rate of convergence is thus given by the slope of the lines. We observe that the order of convergence for $\boldsymbol{\sigma}$ in $L^\infty([0,T],(H(\mathrm{div},C))^2)$ norm and for $\mathbf{v}$ in $L^\infty([0,T],(L^2(C))^2)$ norm are near the values we could expect (i.e., $1/2$). The convergence rate obtained for the Lagrange multiplier $\boldsymbol{\lambda}$ (close to 1) is computed in $L^\infty([0,T];(L^2(\Gamma))^2)$ norm, which means that we recover the conjectured convergence rate $(1/2)$ in $L^\infty([0,T];(H^{\frac{1}{2}}(\Gamma))^2)$ norm.

In Figure 5.3 we display the same results but with the norm of the error now computed in $\widetilde{C} = C/B_b(\Gamma)$, i.e., the domain C restricted from $B_b(\Gamma)$, defined by

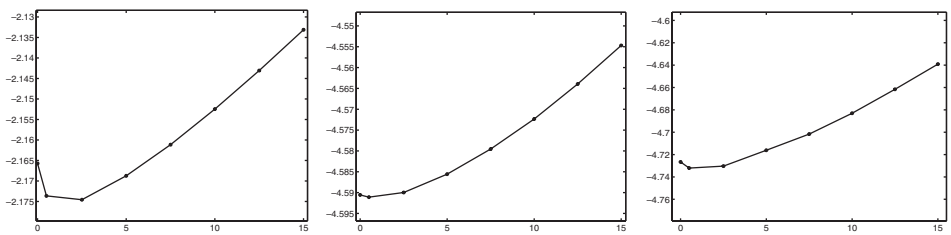
$$
B_b(\Gamma) = \left\{ \mathbf{x} \in C \text{ subject to } \min_{\mathbf{y}\in\Gamma} |\mathbf{x}-\mathbf{y}| \leq b \right\}. \tag{5.5}
$$

In this case, we observe that the convergence rate of the method is higher. This agrees with our intuition in the sense that the elements that we need to remove are the ones in which the solution has less regularity (see Remark 1), i.e., the elements that have nonzero intersection with the boundary Γ.



(a) $\sup_{t\leq T} \|\boldsymbol{\sigma} - \boldsymbol{\sigma}_h\|_{\underline{\underline{H_{\mathrm{div}}}}(\tilde{C})}$. $s = 0.96$. (b) $\sup_{t\leq T} \|\mathbf{v} - \mathbf{v}_h\|_{\underline{L^2}(\tilde{C})}$. $s = 1.05$.

FIG. 5.3. *Numerical error* $\boldsymbol{\sigma}$, $\mathbf{v}$ *versus the discretization step. Here we compute the norm of the error in* $\widetilde{C}$.



(a) $\sup_{t\leq T} \|\boldsymbol{\sigma} - \boldsymbol{\sigma}_h\|_{\underline{\underline{X}}}$. (b) $\sup_{t\leq T} \|\mathbf{v} - \mathbf{v}_h\|_{\underline{M}}$. (c) $\sup_{t\leq T} \|\boldsymbol{\lambda} - \boldsymbol{\lambda}_h\|_{\underline{\mathcal{G}}}$.

FIG. 5.4. *Numerical error on* $\boldsymbol{\sigma}$, $\mathbf{v}$, *and* $\boldsymbol{\lambda}$ *versus the damping parameter* β/ρ.

5.1. Determination of $\boldsymbol{\beta}$. Finally we discuss the influence of the damping parameter β on the convergence results. In order to do so we repeat the experiment described above using the mesh with $h = 1/40$ mm for $\beta/\rho 10^{-6} = 0, 0.5, 2.5, 5, 7.5, 10$, 12.5, and 15. We display in Figure 5.4 the logarithm of the error on $\boldsymbol{\sigma}$, $\mathbf{v}$, and $\boldsymbol{\lambda}$ as a function of the value of $\beta/\rho 10^{-6}$. As we can see, better results are obtained for values of $\beta/\rho 10^{-6}$ between 2.5 and 5. We also observe that the error increases when we choose β too large. The same experiments done for different materials show that the optimal range for β/ρ is independent of the material. It depends only on the number of points per wavelength in the discretization. It should be proportional to $1/\Delta t$:

$$\frac{\beta}{\rho} = \frac{\zeta}{\Delta t},$$

where ζ, an adimensional constant, determined by the previous experiment, should be chosen in the interval $[0.03, 0.06]$.

Conclusions. In this paper we proposed a numerical method for solving the elastodynamic problem in anisotropic media with cracks (or more general objects) of complex geometry. Our approach leads to an efficient numerical method that uses regular grids for the space discretization (due to the fictitious domain formulation) and is explicit in time. The last point is achieved using a mixed finite element that is

compatible with mass lumping for the discretization of the velocity-stress system. Although the theoretical convergence analysis is still an open problem, several numerical experiments suggest that the method provides a good approximate solution.

Appendix. The matrix $\mathcal{K}_h$. We give here the expressions of the nonzero elements of the Hermitian matrix $\mathcal{K}_h$ that appears in the dispersion relation (4.5):

$$(\mathcal{K}_h)_{1,1} = \frac{2}{\rho(kh)^2}\left(\mathcal{A}\sin^2\left(\frac{k_x h}{2}\right) + \mathcal{B}\cos(k_z h)\sin^2\left(\frac{k_x h}{2}\right) + 2\mathcal{D}\cos^2\left(\frac{k_x h}{2}\right)\sin^2\left(\frac{k_z h}{2}\right)\right),$$

$$(\mathcal{K}_h)_{1,2} = \frac{-2\ \mathbf{i}\ \mathcal{D}}{\sqrt{3}\rho(kh)^2}\sin(k_x h)\sin^2\left(\frac{k_z h}{2}\right), (\mathcal{K}_h)_{1,3} = \frac{-2\ \mathbf{i}\ \mathcal{B}}{\sqrt{3}\rho(kh)^2}\sin(k_z h)\sin^2\left(\frac{k_x h}{2}\right),$$

$$(\mathcal{K}_h)_{1,4} = \frac{\mathcal{C}+\mathcal{D}}{\rho(kh)^2}\sin(k_x h)\sin(k_z h), (\mathcal{K}_h)_{1,5} = \frac{-2\ \mathbf{i}\ \mathcal{C}}{\sqrt{3}\rho(kh)^2}\sin(k_z h)\sin^2\left(\frac{k_x h}{2}\right),$$

$$(\mathcal{K}_h)_{1,6} = \frac{-2\ \mathbf{i}\ \mathcal{D}}{\sqrt{3}\rho(kh)^2}\sin(k_x h)\sin^2\left(\frac{k_z h}{2}\right), (\mathcal{K}_h)_{2,2} = \frac{4\mathcal{D}}{3\rho(kh)^2}\sin^2\left(\frac{k_x h}{2}\right)\sin^2\left(\frac{k_z h}{2}\right),$$

$$(\mathcal{K}_h)_{2,4} = \frac{2\ \mathbf{i}\ \mathcal{D}}{\sqrt{3}\rho(kh)^2}\sin(k_z h)\sin^2\left(\frac{k_x h}{2}\right), (\mathcal{K}_h)_{2,6} = \frac{4\mathcal{D}}{3\rho(kh)^2}\sin^2\left(\frac{k_x h}{2}\right)\sin^2\left(\frac{k_z h}{2}\right),$$

$$(\mathcal{K}_h)_{3,3} = \frac{2}{3\rho(kh)^2}\left(\mathcal{A}\sin^2\left(\frac{k_x h}{2}\right) - \mathcal{B}\cos(k_z h)\sin^2\left(\frac{k_x h}{2}\right)\right),$$

$$(\mathcal{K}_h)_{3,4} = \frac{2\ \mathbf{i}\ \mathcal{C}}{\sqrt{3}\rho(kh)^2}\sin(k_x h)\sin^2\left(\frac{k_z h}{2}\right), (\mathcal{K}_h)_{3,5} = \frac{-4\mathcal{C}}{3\rho(kh)^2}\sin^2\left(\frac{k_x h}{2}\right)\sin^2\left(\frac{k_z h}{2}\right),$$

$$(\mathcal{K}_h)_{4,4} = \tfrac{2}{\rho(kh)^2}\left(\mathcal{A}\sin^2\left(\tfrac{k_z h}{2}\right) + \mathcal{B}\cos(k_x h)\sin^2\left(\tfrac{k_z h}{2}\right) + 2\mathcal{D}\cos^2\left(\frac{k_z h}{2}\right)\sin^2\left(\frac{k_x h}{2}\right)\right),$$

$$(\mathcal{K}_h)_{4,5} = \frac{-2\ \mathbf{i}\ \mathcal{B}}{\sqrt{3}\rho(kh)^2}\sin(k_x h)\sin^2\left(\frac{k_z h}{2}\right), (\mathcal{K}_h)_{4,6} = \frac{-2\ \mathbf{i}\ \mathcal{D}}{\sqrt{3}\rho(kh)^2}\sin(k_z h)\sin^2\left(\frac{k_x h}{2}\right),$$

$$(\mathcal{K}_h)_{5,5} = \frac{2}{3\rho(kh)^2}\left(\mathcal{A}\sin^2\left(\frac{k_z h}{2}\right) - \mathcal{B}\cos(k_x h)\sin^2\left(\frac{k_z h}{2}\right)\right),$$

$$(\mathcal{K}_h)_{6,6} = \frac{4\mathcal{D}}{3\rho(kh)^2}\sin^2\left(\frac{k_x h}{2}\right)\sin^2\left(\frac{k_z h}{2}\right),$$

and the coefficients $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$ are given by

$$\mathcal{A} = \frac{8\mu^2 + 8\lambda\mu + \lambda^2}{2\mu+\lambda}, \quad \mathcal{B} = \frac{\lambda^2}{2\mu+\lambda}, \quad \mathcal{C} = \lambda, \quad \mathcal{D} = \mu.$$

REFERENCES

[1] M. Amara and J.M. Thomas, *Equilibrium finite elements for the linear elastic problem*, Numer. Math., 33 (1979), pp. 367–383.

[2] D. N. Arnold, F. Brezzi, and J. Douglas, *PEERS: A new mixed finite element for plane elasticity*, Japan J. Appl. Math., 1 (1984), pp. 347–367.

[3] I. Babuska, *The finite element method with Lagrangian multipliers*, Numer. Math., 20 (1973), pp. 179–192.

[4] E. Bécache, A. Chaigne, G. Derveaux, and P. Joly, *Time-domain simulation of a guitar: Model and method*, J. Acoust. Soc. Amer., 6 (2003), pp. 3368–3383.

[5] E. Bécache, P. Joly, and C. Tsogka, *Eléments finis mixtes et condensation de masse en élastodynamique linéaire.* (i) *construction*, C.R. Acad. Sci. Paris, Série I 325, (1997), pp. 545–550.

[6] E. Bécache, P. Joly, and C. Tsogka, *Etude d'un nouvel élément fini mixte permettant la condensation de masse*, C.R. Acad. Sci. Paris, Série I 324, (1997), pp. 1281–1286.

[7] E. Bécache, P. Joly, and C. Tsogka, *An analysis of new mixed finite elements for the approximation of wave propagation problems*, SIAM J. Numer. Anal., 37 (2000), pp. 1053–1084.

[8] E. Bécache, P. Joly, and C. Tsogka, *Fictitious domains, mixed finite elements and perfectly matched layers for 2d elastic wave propagation*, J. Comput. Acoust., 9 (2001), pp. 1175–1203.

[9] E. Bécache, P. Joly, and C. Tsogka, *A new family of mixed finite elements for the linear elastodynamic problem*, SIAM J. Numer. Anal., 39 (2002), pp. 2109–2132.

[10] E. Bécache, J. Rodríguez, and C. Tsogka, *On the convergence of the fictitious domain method for wave equation problems*, Technical report 5802, INRIA, 2006.

[11] J. P. Bérenger, *A perfectly matched layer for the absorption of electromagnetic waves*, J. Comput. Phys., 114 (1994), pp. 185–200.

[12] V.A. Bokil and R. Glowinski, *An operator-splitting scheme with a distributed Lagrange multiplier based fictitious domain method for wave propagation problems*, J. Comput. Phys., 205 (2005), pp. 242–268.

[13] F. Brezzi and M. Fortin, *Mixed and Hybrid Finite Element Methods*, Springer-Verlag, New York, 1991.

[14] G. Cohen, *Higher-order numerical methods for transient wave equations*, Sci. Comput. XIII, Springer-Verlag, Berlin, 2002, with a foreword by R. Glowinski.

[15] F. Collino, P. Joly, and F. Millot, *Fictitious domain method for unsteady problems: Application to electromagnetic scattering*, J. Comput. Phys., 138 (1997), pp. 907–938.

[16] F. Collino and C. Tsogka, *Application of the PML absorbing layer model to the linear elastodynamic problem in anisotropic heteregeneous media*, Geophysics, 66 (2001), pp. 294–307.

[17] S. Garcès, *Application des Méthodes de Domaines Fictifs à la Modélisation des Structures Rayonnantes Tridimensionnelles*, Ecole Nationale Supérierure de l'Aéronautique et de l'Espace, 1997.

[18] V. Girault and R. Glowinski, *Error analysis of a fictitious domain method applied to a Dirichlet problem*, Japan J. Indust. Appl. Math., 12 (1995), pp. 487–514.

[19] V. Girault and P. A. Raviart, *Finite element methods for Navier-Stokes equations*, Springer Ser. Comput. Math. 5, Springer-Verlag, Berlin, 1986, theory and algorithms.

[20] R. Glowinski and Y. Kuznetsov, *On the solution of the Dirichlet problem for linear elliptic operators by a distributed Lagrange multiplier method*, C. R. Acad. Sci. Paris Sér. I Math., 327 (1998), pp. 693–698.

[21] R. Glowinski, T. W. Pan, T. I. Hesla, D. D. Joseph, and J. Periaux, *A distributed Lagrange multiplier/fictitious domain method for flows around moving rigid bodies: Application to particulate flow*, Internat. J. Numer. Methods Fluids, 30 (1999), pp. 1043–1066.

[22] R. Glowinski, T. W. Pan, T. I. Hesla, D. D. Joseph, and J. Periaux, *A distributed Lagrange multiplier/fictitious domain method for the simulation of flow around moving rigid bodies: Application to particulate flow*, Comput. Methods Appl. Mech. Engrg., 184 (2000), pp. 241–267.

[23] R. Glowinski, T. W. Pan, T. I. Hesla, D. D. Joseph, and J. Periaux, *A fictitious domain approach to the direct numerical simulation of incompressible viscous fluid past moving rigid bodies: Application to particulate flow*, J. Comput. Methods Phys., 169 (2001), pp. 363–426.

[24] R. Glowinski, T. W. Pan, and J. Periaux, *A fictitious domain method for Dirichlet problem and applications*, Comput. Methods Appl. Mech. Engrg., 111 (1994), pp. 283–303.

[25] R. Glowinski, T. W. Pan, and J. Periaux, *A fictitious domain method for external incompressible flow modeled by navier-stokes equations*, Comput. Methods Appl. Mech. Engrg., 112 (1994), pp. 133–148.

[26] P. Grisvard, *Problèmes aux limites dans les polygones. Mode d'emploi*, EDF Bull. Direction Études Rech. Sér. C. Math. Inform., 1 (1986), pp. 21–59.

[27] E. Heikkola, Y. A. Kuznetsov, P. Neittaanmäki, and J. Toivanen, *Fictitious domain methods for the numerical solution of two-dimensional scattering problems*, J. Comput. Phys., 145 (1998), pp. 89–109.

[28] C. Johnson and B. Mercier, *Some equilibrium finite element methods for two-dimensional elasticity problems*, Numer. Math., 30 (1978), pp. 103–116.

[29] P. Joly and L. Rhaouti, *Analyse numérique - Domaines fictifs, éléments finis H(div) et condition de Neumann : le problème de la condition inf-sup*, C.R. Acad. Sci. Paris, Série I 328, (1999), pp. 1225–1230.

[30] Y. A. KUZNETSOV, *Fictitious component and domain decomposition methods for the solution of eigenvalue problems*, in Proceedings of the Seventh International Symposium on Computing Methods in Applied Sciences and Engineering Versailles, 1985, North–Holland, Amsterdam, 1986, pp. 155–172.

[31] M. E. MORLEY, *A family of mixed finite elements for linear elasticity*, Numer. Math., 55 (1989), pp. 633–666.

[32] J.C. NÉDÉLEC, *A new family of mixed finite elements in* $\mathbb{R}^3$, Numer. Math., 50 (1986), pp. 57–81.

[33] L. RHAOUTI, *Domaines Fictifs pour la Modélisation d'un Probème d' Interaction Fluide-Structure: Simulation de la Timbale*, Ph.D. thesis, Université Paris IX, 1999.

[34] R. STENBERG, *On the construction of optimal mixed finite element methods for the linear elasticity Problem*, Numer. Math., 48 (1986), pp. 447–462.

[35] R. STENBERG, *A family of mixed finite elements for the elasticity problem*, Numer. Math., 53 (1988), pp. 513–538.

[36] N. TORDJMAN, *Eléments Finis d'Ordre Élevé vec Condensation de Masse pour l'Équation des Ondes*, Ph.D. thesis, Université Paris IX, 1995.

SIAM J. SCI. COMPUT.
Vol. 29, No. 3, pp. 1268–1288
© 2007 Society for Industrial and Applied Mathematics

A NEW ITERATIVE METHOD FOR SOLVING LARGE-SCALE LYAPUNOV MATRIX EQUATIONS*

V. SIMONCINI†

Abstract. In this paper we propose a new projection method to solve large-scale continuous-time Lyapunov matrix equations. The new approach projects the problem onto a much smaller approximation space, generated as a combination of Krylov subspaces in A and A^{-1}. The reduced problem is then solved by means of a direct Lyapunov scheme based on matrix factorizations. The reported numerical results show the competitiveness of the new method, compared to a state-of-the-art approach based on the factorized alternating direction implicit iteration.

Key words. matrix equations, Krylov subspace, model order reduction, iterative methods, large-scale equations

AMS subject classifications. 65F10, 65F30, 15A06

DOI. 10.1137/06066120X

1. Introduction.

Given the time-invariant linear system

$$\mathbf{x}'(t) = A\mathbf{x}(t) + B\mathbf{u}(t), \qquad \mathbf{x}(0) = x_0, \tag{1.1}$$

with $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times s}$, with $s \le n$, the solution of the associated continuous-time Lyapunov equation

$$AX + XA^T + BB^T = 0 \tag{1.2}$$

has a fundamental role in many application areas such as signal processing and system and control theory. We assume that $s \ll n$ and that A is dissipative; that is, $x^T(A + A^T)x < 0$ for all real vectors x. The symmetric solution X, called the reachability Gramian, carries important information on the stability and energy of the linear system (1.1) and on the feasibility of order reduction techniques [1], [4], [9].

The numerical solution of the Lyapunov matrix equation has been addressed in a large body of literature, and well-established methods exist for small dimensional problems [2], [20]. Research has recently focused mostly on schemes for medium- to large-scale Lyapunov equations, for which methods based on matrix factorizations become too demanding, in terms of both computational costs and memory requirements. For dense problems, efforts have been devoted to the development of matrix iterative methods based on the sign function, possibly exploiting the problem structure; see, e.g., [3], [6], and references therein.

In the case of sparse matrix A, projection-type methods based on Krylov subspaces, using either polynomials or rational functions of A, have been employed; see, e.g., [23], [24], [25], [26], [28], [32], and the general treatment in [1]. In particular, since the work by Ellner and Wachspress in [12], the alternating direction implicit (ADI) iteration has found great applicability in the solution of (1.2), and software implementing an efficient variant is now available; see, e.g., [30]. A common feature of all of these projection methods is that the approximate solution $\widehat{X}$ is given in factored form, $\widehat{X} = ZZ^T$, with Z of low column rank. The possibility of obtaining a

*Received by the editors May 29, 2006; accepted for publication (in revised form) February 7, 2007; published electronically May 29, 2007.
http://www.siam.org/journals/sisc/29-3/66120.html
†Dipartimento di Matematica, Università di Bologna, Piazza di Porta S. Donato 5, I-40127 Bologna, Italy and CIRSA, Ravenna, Italy (valeria@dm.unibo.it).

good low rank approximate solution is supported by recent theoretical results showing that a rapid decay of the eigenvalues of the symmetric and positive semidefinite exact solution may be observed under certain conditions on A; see [16], [29], [37]. This property can be fully exploited when dealing with large problems, since storing the whole, in general, full matrix $\widehat{X}$ would require a significant amount of memory space. Other methods have been investigated, such as that in [22], which, however, do not build a low rank approximation.

In spite of these recent efforts, projection methods that are directly based on Krylov subspace approaches (cf. [32], [23]) may be slow, requiring the generation of a very large approximation space to obtain a sufficiently accurate solution; see the experiments reported in [28]. On the other hand, ADI-type methods need the a priori evaluation of some parameters which may be hard to tune. In addition, memory and CPU time requirements remain an issue, for these methods perform various large matrix factorizations.

In this paper we propose a new projection-type method for large-scale problems that is, in general, faster than approaches based on the ADI iteration and does not require estimation of parameters. The new method projects the problem onto a small approximation space, generated as a combination of Krylov subspaces in A and A^{-1}. The reduced problem is then solved by means of a standard scheme for small Lyapunov equations. The method is equipped with a natural stopping criterion, which can be computed inexpensively. The performance of the new method is compared on benchmark problems to that of the cyclic implementation of the factorized ADI method available in [30].

The paper is organized as follows. Section 2 describes the general subspace projection framework, which characterizes the new approach and several other reduction methods. Section 3 and its subsections describe the new method and its algorithmic aspects. Section 4 reviews the ADI iteration and its efficient variant to prepare the notation for the numerical experiments reported in section 5. Finally, in section 6 we provide some conclusive remarks.

The following notation is used throughout. The square identity and zero matrices are denoted by I and O, respectively; rectangular portions will be identified by using subscripts. The only exceptions are the (block) columns of the identity, denoted by e_i, the ith column (resp., E_i, the ith block of $2s$ columns of the identity matrix). Matlab notation is used whenever possible [27]. $\|\cdot\|$ indicates the 2-norm for vectors and the induced norm for matrices, while $\|\cdot\|_F$ denotes the Frobenius norm; that is, $\|A\|_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{m} a_{i,j}^2$ for $A \in \mathbb{R}^{n\times m}$. Range$(V)$ indicates the space spanned by the columns of V. Finally, exact arithmetic is assumed throughout.

2. Projection methods for the Lyapunov equation. In this section we review a general framework associated with projection methods for the Lyapunov equation. This will allow us to simplify the presentation of the new approach in section 3.

Consider a subspace K of $\mathbb{R}^n$, and let V be a matrix whose orthonormal columns span K, with $B = VE$ for some matrix E, so that the columns of B belong to K. We project the Lyapunov equation onto this subspace K: We set $H = V^TAV$ and seek an approximate solution in K in the form $\widehat{X} = VYV^T$. The projected problem may be obtained by imposing that the residual $R = A\widehat{X} + \widehat{X}A^T + BB^T$ be orthogonal to the approximation space K, the so-called Galerkin condition. In matrix terms, this can be written as[1] $V^TRV = O$. Since V has orthonormal columns, imposing $V^TRV = 0$

[1]This condition can be derived by first defining the matrix inner product $\langle X, Y\rangle = \mathrm{tr}(XY^T)$ and then imposing $\langle R, P\rangle = 0$ for any $P = VGV^T$ [32].

gives the following projected Lyapunov equation:

$$V^T AVY + YV^T A^T V + EE^T = 0, \tag{2.1}$$

whose solution Y defines $\widehat{X}$. Since we assume that A is dissipative, $V^T AV$ is stable, and the equation (2.1) admits a unique solution. The projection is effective if a good approximation is attained for a small dimensional subspace, so that (2.1) can be cheaply solved by means of procedures based on matrix factorizations; cf. [2], [20]. In turn, this effectiveness is more likely to take place whenever the exact solution can be accurately approximated by low rank matrices. An additional theoretical motivation for using such a technique is given by the exponential form of the analytical solution:

$$X = \int_0^\infty e^{tA} BB^T e^{tA^T} dt. \tag{2.2}$$

Note that the use of this formulation has led to different approximation procedures such as that in [17]. The quantity $W = e^{tA}B$ can be approximated in K as $\widehat{W} = Ve^{tH}E$. Therefore, an approximation to X can be obtained as

$$\widehat{X} = \int_0^\infty \widehat{W}\widehat{W}^T \, dt = V \left(\int_0^\infty e^{tH} EE^T e^{tH^T} \, dt \right) V^T =: VYV^T.$$

Using the fact that $H = V^T AV$ is stable, the following result can be readily established; see, e.g., [9].

PROPOSITION 2.1. [illegible] $Y = \int_0^\infty e^{tH} EE^T e^{tH^T} dt$ [illegible] (2.1).

Although the procedure above holds independently of the specific approximation space K, its [illegible] crucially depends on K. The presentation outlined above was first introduced in [32] for $B \in \mathbb{R}^n$ and K corresponding to the Krylov subspace $\mathcal{K}_k(A, B) = \text{span}\{B, AB, \ldots, A^{k-1}B\}$. Further developments of Krylov subspace methods for $B \in \mathbb{R}^{n \times s}$ and using conditions on the residual different from the Galerkin one were discussed in [23]. Reasons for expecting good accuracy in the approximation for a sufficiently large dimension of the Krylov subspace comes from the approximation properties of the exponential. Let V_k be such that $\text{Range}(V_k) = \mathcal{K}_k(A, B)$, and $H_k = V_k^T AV_k$. If $\widehat{W}_k = V_k e^{tH} E$ is a good approximation to $W = e^{tA}B$, we expect the corresponding integral to approximate X. In particular, assuming for the sake of the presentation $t = 1$, if the numerical range of A is contained in the complex disk $|z + \rho| \le \rho$ for some $\rho > 0$, then a bound of the form $\| \exp(A)v - V_k \exp(H_k)e_1 \| = O((\frac{e\rho}{k})^k)$ holds for $k \ge 2\rho$; see [21]. More accurate bounds can be obtained in the symmetric case [21], [40], [10].

Assume now that a sequence of approximation subspaces K_k of dimension k is given, with $K_k \subset K_{k+1}$, and satisfies $AK_k \subset K_{k+1}$. Note that these conditions on K_k are reasonably general.

Let the columns of V_k span K_k, $\underline{H}_k = V_{k+1}^T AV_k$, and let $X_k = V_k Y_k V_k^T$ be the approximate solution to (1.2) in K_k, obtained by imposing the Galerkin condition. The associated residual $R_k = AX_k + X_k A^T + BB^T$ satisfies (see, e.g., [32])

$$R_k = V_{k+1} \underline{H}_k Y_k V_k^T + V_k Y_k \underline{H}_k^T V_{k+1}^T + V_k EE^T V_k^T =: V_{k+1} \widehat{R} V_{k+1}^T. \tag{2.3}$$

The selection $K_k = \mathcal{K}_k(A, B)$ fulfills all conditions above, and in addition it can be shown that $\|R_k\| = \|E_{k+1}^T \underline{H}_k Y_k\|$, so that the quality of the approximation X_k can be measured by computing the residual norm, without explicitly evaluating the residual matrix. This setting has been heavily exploited to devise competitive approaches and to derive theoretical properties; see, e.g., [24], [23], [31].

3. The new iterative procedure. The general description of the previous section suggests that the approximation space K could be enriched to contain more information than that of the regular Krylov subspace $\mathcal{K}_k(A, B)$. To this end, starting with the pair $\{B, A^{-1}B\}$, we propose to generate a sequence of approximation spaces that contain information on both A and A^{-1}, by adding two vectors at a time, one multiplied by A and one by A^{-1}; that is,

$$\text{span}\{B, A^{-1}B, AB, A^{-2}B, A^2B, A^{-3}B, \dots\},$$

where the span is intended blockwise; see, e.g., [33]. As discussed in section 2, once the approximation space is generated, the original problem is projected onto this space, and the resulting small dimensional problem is solved. Our approach is inspired by a related acceleration procedure for the approximation of matrix functions developed in [11]; see the discussion at the end of section 3.2.

An outline of the algorithm is given next. Implementation details of the approach are provided in section 3.1.

Given B, A, set $V_1 = \texttt{gram_sh}([B, A^{-1}B])$, $\mathcal{V}_0 = \emptyset$.

For $m = 1, 2, \dots,$

1. $\mathcal{V}_m = [\mathcal{V}_{m-1}, V_m]$.
2. Set $\mathcal{T}_m = \mathcal{V}_m^T A \mathcal{V}_m$ and $E = \mathcal{V}_m^T B$.
3. Solve $\mathcal{T}_m Y + Y \mathcal{T}_m^T + EE^T = 0$ and set $Y_m = Y$.
4. If converged, then $X_m = \mathcal{V}_m Y_m \mathcal{V}_m^T$ and stop.
5. Set $V_m^{(1)}$: first s columns of V_m; $V_m^{(2)}$: second s columns of V_m.
6. $V'_{m+1} = [AV_m^{(1)}, A^{-1}V_m^{(2)}]$.
7. $\widehat{V}_{m+1} \leftarrow$ orthogonalize V'_{m+1} w.r.t. $\mathcal{V}_m$.
8. $V_{m+1} = \texttt{gram_sh}(\widehat{V}_{m+1})$.

At each iteration of this process, two new blocks of s vectors are added to the space, so that the space dimension increases by $2s$. Unless breakdown occurs, at the mth iteration the method has constructed an orthonormal basis of dimension $2sm$, given by the columns of the matrix $\mathcal{V}_m = [V_1, V_2, \dots, V_m]$, $V_i \in \mathbb{R}^{n \times 2s}$, and $\mathbf{K}_m := \text{Range}(\mathcal{V}_m)$. The orthogonalization is performed first with respect to the previous basis vectors and then within the new block of $2s$ vectors (function `gram_sh`; see section 3.1). At each iteration (cf. step 3), a small dimensional Lyapunov equation with matrices of size $2sm \times 2sm$ is solved. Only at convergence (step 4), the approximate solution to X is constructed as

$$X_m = \mathcal{V}_m Y_m \mathcal{V}_m^T, \tag{3.1}$$

with Y_m symmetric. This step can be omitted if X_m is not required explicitly. The following important properties are a consequence of the subspace definition.

PROPOSITION 3.1. [illegible] $m \geq 1$ [illegible] $\mathbf{K}_m$ [illegible] $\mathbf{K}_m = \mathcal{K}_{2m}(A, A^{-m}B)$ [illegible] $\mathcal{K}_{2m}$ [illegible]

$$A\mathbf{K}_m \subseteq \mathbf{K}_{m+1}. \tag{3.2}$$

We observe that the results of Proposition 3.1 ensure that $\mathcal{T}_m$ is block upper Hessenberg, since $V_k^T A V_j = O$ for $k > j + 1$, $j = 1, 2, \dots$. In particular, property (3.2) allows us to employ the framework of section 2. Moreover, since A is dissipative, $\mathcal{T}_m$ is stable, so that Y_m is positive (semi)definite.

To simplify the presentation, we limit the following discussion to the case $s = 1$; however, the stated relations formally hold unchanged for $s \geq 1$, by replacing scalar

quantities with $s \times s$ matrices. We derive some recursive relations that can be used to significantly reduce the computational cost of the basic algorithm. Let

$$\underline{\mathcal{H}}_m = \begin{bmatrix} \mathcal{H}_m \\ \chi_m E_m^T \end{bmatrix} \in \mathbb{R}^{2(m+1)\times 2m}$$

collect all orthonormalization coefficients; that is,

$$\widehat{V}_{m+1} = [AV_m^{(1)}, A^{-1}V_m^{(2)}] - \mathcal{V}_m \mathcal{H}_m [e_{2m-1}, e_{2m}], \tag{3.3}$$

$$V_{m+1}\chi_m = \widehat{V}_{m+1}. \tag{3.4}$$

Here χ_m is an upper 2×2 triangular matrix such that V_{m+1} has orthonormal columns; note that $\mathcal{H}_m$ is block upper Hessenberg with 2×2 blocks. Let also $\underline{\mathcal{T}}_m := \mathcal{V}_{m+1}^T A \mathcal{V}_m$. The computation of $\mathcal{T}_m$ seems to require additional matrix-vector products with A and extra inner products of long vectors. We next derive a recursion for $\underline{\mathcal{T}}_m$ that completely avoids this expensive step.

PROPOSITION 3.2. [illegible] $\ell^{(k)} = (\ell_{ij})$ [illegible] 2×2 [illegible] $V_k = \hat{V}_k \ell^{(k)}$ [illegible]
(3.3) $k = 1, \dots, m$ [illegible]

$$\underline{\mathcal{T}}_m = (t_{i,j})_{i=1,\dots,2m+2, j=1,\dots,2m} \qquad \mathcal{H}_m = (h_{i,j})_{i=1,\dots,2m, j=1,\dots,2m}.$$

[illegible]

$$t_{:,2k-1} = h_{:,2k-1}, \quad k = 1, \dots, m,$$

[illegible]

$$\begin{aligned}
(k=1) \quad & t_{:,2} = h_{:,1}(\ell_{11}^{(1)})^{-1}\ell_{12}^{(1)} + e_1(\ell_{11}^{(1)})^{-1}\ell_{22}^{(1)}, \qquad t_{:,4} = (e_2 - \underline{\mathcal{T}}_1 h_{1:2,2})\ell_{22}^{(2)}, \\
& \rho^{(2)} = (\ell_{11}^{(2)})^{-1}\ell_{12}^{(2)}, \\
(1 < k \le m) \quad & t_{:,2k} = t_{:,2k} + t_{:,2k-1}\rho^{(k)}, \\
& t_{:,2k+2} = (e_{2k} - \underline{\mathcal{T}}_k h_{1:2k,2k})\ell_{22}^{(k+1)}, \qquad \rho^{(k+1)} = (\ell_{11}^{(k+1)})^{-1}\ell_{12}^{(k+1)}.
\end{aligned}$$

[illegible]. For $k \ge 1$, we write $V_k = [V_k^{(1)}, V_k^{(2)}] \in \mathbb{R}^{n\times 2}$. Using (3.3), we immediately have $AV_k^{(1)} = V_{k+1}\chi_k e_1 + \mathcal{V}_k \mathcal{H}_k e_{2k-1} = \mathcal{V}_{k+1}\underline{\mathcal{H}}_k e_{2k-1}$, so that the odd columns of $\mathcal{T}_m$ satisfy

$$\mathcal{V}_m^T A \mathcal{V}_m e_{2k-1} = \mathcal{V}_m^T A V_k^{(1)} = \begin{bmatrix} I_{2k+1} \\ O \end{bmatrix} \underline{\mathcal{H}}_k e_{2k-1} = \mathcal{H}_k e_{2k-1}.$$

The even columns need to take into account that the recurrence comes from multiplications by A^{-1}, and thus the term $AV_k^{(2)}$ is not explicitly available. However, from (3.3) we obtain

$$V_k^{(2)} = A\widehat{V}_{k+1}^{(2)} + A\mathcal{V}_k \mathcal{H}_k e_{2k},$$

which gives a recurrence for the $(2k+2)$th column $A\widehat{V}_{k+1}^{(2)}$. Hence,

$$\mathcal{V}_{m+1}^T A \widehat{V}_{k+1}^{(2)} = \mathcal{V}_{m+1}^T V_k^{(2)} - \mathcal{V}_{m+1}^T A \mathcal{V}_k \mathcal{H}_k e_{2k} = e_{2k} - \begin{pmatrix} \underline{\mathcal{T}}_k \\ O_{m-k,2k} \end{pmatrix} \mathcal{H}_k e_{2k}.$$

To complete the derivation, we recall that $V_{k+1} = \widehat{V}_{k+1}\ell^{(k+1)}$, with $\ell^{(k+1)} = (\ell_{ij})$ upper triangular, so that $V_{k+1}^{(2)} = \widehat{V}_{k+1}^{(1)}\ell_{12} + \widehat{V}_{k+1}^{(2)}\ell_{22}$, from which $AV_{k+1}^{(2)} = A\widehat{V}_{k+1}^{(1)}\ell_{12} + A\widehat{V}_{k+1}^{(2)}\ell_{22} = AV_{k+1}^{(1)}\ell_{11}^{-1}\ell_{12} + A\widehat{V}_{k+1}^{(2)}\ell_{22}$. Therefore,

$$\mathcal{V}_{m+1}^T A V_{k+1}^{(2)} = \mathcal{V}_{m+1}^T \mathcal{V}_{k+2} \underline{\mathcal{H}}_{k+1} e_{2k+1}\ell_{11}^{-1}\ell_{12} + \left(e_{2k} - \begin{pmatrix} \underline{\mathcal{T}}_k \\ O_{m-k,2k} \end{pmatrix} \mathcal{H}_k e_{2k} \right) \ell_{22}.$$

The recursion thus follows. ∎

The recurrence for $\underline{\mathcal{T}}_m$ shows that the lower diagonal 2×2 block has a zero second row. This fact can be used to simplify the computation of the residual norm.

PROPOSITION 3.3. [illegible] $m \geq 1$ [illegible] $\mathbf{K}_m = \mathrm{Range}(\mathcal{V}_m)$ [illegible] X_m [illegible] (3.1) [illegible] $R_m = AX_m + X_m A^T + BB^T$ [illegible] $\mathrm{Range}(R_m) = \mathrm{Range}(\mathcal{V}_{m+1})$ [illegible] $\|R_m\| = \|e_{2m+1}^T \underline{\mathcal{T}}_m Y_m\|$ [illegible]. Since $A\mathcal{V}_m = \mathcal{V}_{m+1}L$, for some matrix L, the first result follows from (2.3). Therefore, $R_m = \mathcal{V}_{m+1}\widehat{R}_m\mathcal{V}_{m+1}^T$, with

$$\widehat{R}_m = \mathcal{V}_{m+1}^T R_m \mathcal{V}_{m+1} = \begin{pmatrix} 0 & \mathcal{V}_m^T R_m V_{m+1} \\ V_{m+1}^T R_m \mathcal{V}_m & 0 \end{pmatrix}.$$

Using $V_{m+1}^T R_m \mathcal{V}_m = E_{m+1}^T \underline{\mathcal{T}}_m Y_m$, it follows that $\|R_m\| = \|\widehat{R}_m\| = \|V_{m+1}^T R_m \mathcal{V}_m\| = \|E_{m+1}^T \underline{\mathcal{T}}_m Y_m\| = \|e_{2m+1}^T \underline{\mathcal{T}}_m Y_m\|$. ∎

The matrix Y_m in the solution $X_m = \mathcal{V}_m Y_m \mathcal{V}_m^T$ is often numerically positive semidefinite [29], [16], [37]. If this is so, it is possible to generate a lower rank X_m as $X_m = Z_m Z_m^T$, with Z_m of rank lower than $2m$, the dimension of Y_m. More precisely, let $Y_m = WDW^T$ be the eigendecomposition of the $2m \times 2m$ matrix Y_m, with D diagonal having diagonal entries sorted in decreasing order, and let $W_k D_k W_k^T$ be the approximate decomposition obtained by truncating to the first k columns of W, with $k \ll 2m$. Then $Y_m \approx W_k D_k W_k^T$, and the approximation is determined by using $Z_m = V_m W_k D_k^{\frac{1}{2}}$. The number k of columns to retain may be determined by inspecting the diagonal elements of D. In our implementation, we discarded the columns of W corresponding to diagonal elements of D less than 10^{-12}; a more conservative tolerance could also be employed.

Due to the form of the approximate solution, it is also possible to monitor at low cost the amount by which the approximate solution varies as the iterations proceed. Indeed, letting

$$Y_m = \begin{bmatrix} \widehat{Y}_m & y_m \\ y_m^T & \eta_m \end{bmatrix},$$

with $\widehat{Y}_m$ of size $(2m-2)\times(2m-2)$, we have

$$X_m - X_{m-1} = [\mathcal{V}_{m-1}, V_m] \begin{bmatrix} \widehat{Y}_m - Y_{m-1} & y_m \\ y_m^T & \eta_m \end{bmatrix} [\mathcal{V}_{m-1}, V_m]^T =: \mathcal{V}_m G_m \mathcal{V}_m^T,$$

so that

$$\frac{\|X_m - X_{m-1}\|_F}{\|X_{m-1}\|_F} = \frac{\|G_m\|_F}{\|Y_{m-1}\|_F}, \tag{3.5}$$

where both G_m and Y_{m-1} have small dimension. We do not advocate using this quantity as a stopping criterion; however, we shall sometimes refer to it in our experiments for comparison purposes.

3.1. Implementation details. An implementation of the algorithm associated with the procedure outlined above is given by the function Krylov-plus-inverted-Krylov (hereafter K-PIK). For the sake of simplicity, we report here the case $s = 1$. The general case $s \geq 1$ is depicted in the appendix. The two codes differ only in the matrix indexing. Both codes are available from the author upon request.

```
function Z=K_PIK(A,rhs,m_max,k_max,tol)

nrmb=norm(rhs,'fro')^2;
nrma=norm(A,'fro');

[LA,UA]= lu(A);               % For sym matrices use UA=chol(-A); LA=-UA';
[ibeta,U(:,1:2)] = gram_sh([rhs, UA\ (LA\ rhs)]);

for j=1:m_max,

  % Expand the approximation space
  jms=(j-1)*2+1;j1s=(j+1)*2;js=j*2;js1=js+1;
  Up(:,1) = A*U(:,jms);     Up(:,2) = UA\ (LA\ U(:,js));

  %New Basis Block
  for il=1:2           % Reorthogonalization
      k_min=max(1,j-k_max);
      for kk=k_min:j
         k1=(kk-1)*2+1; k2=kk*2;
         coef= U(1:n,k1:k2)'*Up;
         H(k1:k2,jms:js) = H(k1:k2,jms:js)+ coef;
         Up = Up - U(:,k1:k2)*coef;
      end
  end
  if (j<=m_max)
      [hinv,U(1:n,js1:j1s)]=gram_sh(Up);    H(js1:j1s,jms:js)=inv(hinv);
  end
  I = speye(js+2);

  % Update the T recurrence
  if (j==1),
    l(1:js+1,j)=[ H(1:3,1), speye(3,1)]*ibeta(1:2,2)/ibeta(1,1);
  else
    l(1:js+2,j)=l(1:js+2,j) + H(1:js+2,js-1)*rho;
  end
  T(1:js+2,1:2:js)=H(1:js+2,1:2:js); T(1:js+2,2:2:js)=l(1:js+2,1:j);
  l(1:js+2,j+1)=( I(1:js+2,j*2)-T(1:js+2,1:js)*H(1:js,js))*hinv(2,2);
  rho = hinv(1,2)/hinv(1,1);

  % Solve small Lyapunov equation
  Y = lyap(T(1:js,1:js),beta^2*speye(js,1)*speye(js,1)');

  % Compute residual and test for convergence
  cc(1:2,1) = H(js1:j1s,js-1); cc(1:2,2) = l(js1:j1s,j);
  nrmx = norm(Y,'fro');
  er2(j) = norm(cc*Y(js-1:js,:))/(nrmb+nrma*nrmx);
  if (er2(j)<tol), break, end

end

% Reduced rank solution   tol = 1e-12
[uY,sY]=eig(Y); [sY,id]=sort(diag(sY));
sY=flipud(sY); uY=uY(:,id(end:-1:1)); is=sum(abs(sY)>1e-12);
Y0 = uY(:,1:is)*diag(sqrt(sY(1:is)));
Z = U(:,1:js)*Y0;
```

A few comments are in order. The LU factorization of A is employed for medium-size matrices, and the Cholesky factorization of $-A$ should be used for A symmetric. For large matrices, a preconditioned iterative method could be employed to solve systems with A at each iteration, where the preconditioner could be generated once for all; see section 5.3 for some experiments.

Function `gram_sh` performs the modified Gram–Schmidt orthogonalization of the new block [14]. The same process is used for the orthogonalization with respect to older vectors in the basis. Note that this reduces to a procedure similar to the block Lanczos algorithm (see [15]) when the problem is symmetric, by setting `k_max = 2`, so that only orthogonalization with respect to the previous two blocks is enforced. In particular, for A symmetric, the whole matrix $\mathcal{V}_m$ is needed only at convergence to recover the solution factor. Therefore, one can either store the columns that are not needed in the orthogonalization process or use a "two-pass" procedure, in which the basis vectors are first discarded. At convergence, the short-term iteration is performed once again to recover the basis vectors and update the final solution. Although this procedure is expensive, essentially doubling the computational cost of the generation of the approximation space, the memory requirements become extremely limited, allowing one to tackle very large problems, in the symmetric case. It is well known that if orthogonality is not enforced explicitly, as is the case when using short-term recurrences for A symmetric, loss of orthogonality of the columns of $\mathcal{V}_m$ may affect the computation of the matrix $\mathcal{H}_m$ and of the residual norm in Proposition 3.3. If one is willing to keep the whole basis in fast memory, orthogonalization of all basis vectors can be explicitly performed. Problems increase for $s > 1$. We refer to [19] for a recent presentation of these issues and for safeguard strategies that can be implemented to mitigate the problem in the block setting.

The function `lyap` computes the numerical solution of the small dimensional Lyapunov equation by means of a Schur decomposition-type method; in our experiments, we used the algorithm proposed by Sorensen and Zhou [38], which implements the Bartels–Stewart method with only real arithmetic; other alternatives may be considered as, e.g., in the SLICOT package ([5], [36], [41]). Since the matrix $\mathcal{T}_m$ is stable, other procedures such as the Hammarling method could also be employed [20]. Note also that the fact that $\mathcal{T}_m$ is block upper Hessenberg significantly reduces the computational work of the method. We should remark, however, that as the approximation subspace increases, this step becomes the most computationally demanding task of the algorithm.

The stopping criterion is

$$\frac{\|AX_m + X_m A^T + BB^T\|}{2\|A\|_F\|Y_m\|_F + \|B\|_F^2} \leq \texttt{tol}, \tag{3.6}$$

where `tol` is a chosen fixed threshold. The numerator is obtained by using the computationally inexpensive relation of Proposition 3.3. We observe that the output matrix is the memory-conserving factor Z_m of $X_m = Z_m Z_m^T$.

3.2. Finite termination and convergence. For the sake of simplicity, in this section we mostly limit our discussion to the case $s = 1$. In exact arithmetic, the procedure just described has finite termination, since for m such that $2m \geq n$ the generated vectors span the whole space. However, since two vectors at a time are added to the current basis, loss of rank may occur during the orthogonalization with respect to the older basis vectors, so that the next basis pair using V_{m+1} cannot be built. We next show that this implies convergence, as expected by the result of Proposition 3.1.

PROPOSITION 3.4. [illegible] $m-1$ [illegible] $B \in \mathbb{R}^n$ [illegible] m [illegible] $\widehat{V}_{m+1}$ [illegible] (3.3) [illegible] $\mathcal{V}_m \cup \{\widehat{V}_{m+1}\}$ [illegible] A [illegible] B [illegible] $\mathrm{Range}(\mathcal{V}_m \cup \{\widehat{V}_{m+1}\})$ [illegible]. We recall that $\mathrm{Range}(\mathcal{V}_m) = \mathcal{K}_{2m}(A, A^{-m}B)$. Therefore, the given assumption requires that there exist $\alpha_i \in \mathbb{R}$, $i = -m-1, \ldots, m$, not all zero such that

$$\alpha_{-m}A^{-m}B + \cdots + \alpha_{-1}A^{-1}B + \alpha_0 B + \alpha_1 AB + \cdots + \alpha_{m-1}A^{m-1}B + \alpha_m A^m B + \alpha_{-m-1}A^{-m-1}B = 0.$$

Assume first $\alpha_{-m-1} = 0$ and $\alpha_m \neq 0$. Then $A(A^{m-1}B) = \sum_{j=0}^{2m-1} \beta_j A^j A^{-m} B$, which shows that $A\mathcal{V}_m = \mathcal{V}_m \mathcal{T}_m$, and thus $\mathcal{V}_m$ is invariant for A. Using $X_m = \mathcal{V}_m Y_m \mathcal{V}_m^T$, we obtain

$$AX_m + X_m A^T + BB^T = \mathcal{V}_m(\mathcal{T}_m Y_m + Y_m \mathcal{T}_m + EE^T)\mathcal{V}_m^T = O.$$

Next, assume that $\alpha_{-m-1} \neq 0$ and $\alpha_m \neq 0$. Then

$$A^{-1}A^{-m}B = \sum_{j=0}^{2m-1} \beta_j A^j A^{-m} B + \beta_{2m} A^m B,$$

or, equivalently, $A^{-m}B - \sum_{j=1}^{2m} \beta_j A^j A^{-m} B = \beta_{2m} A(A^m B)$, so that $A[\mathcal{V}_m, V_m^{(1)}] = [\mathcal{V}_m, V_m^{(1)}]\widehat{\mathcal{T}}_m$, where $\widehat{\mathcal{T}}_m$ is the restriction of $\mathcal{T}_{m+1}$ to the first $(2m+1)$ columns and rows. Therefore, $[\mathcal{V}_m, V_m^{(1)}]$ is invariant for A. As in the previous case, writing $X'_m = [\mathcal{V}_m, V_m^{(1)}] Y'_m [\mathcal{V}_m, V_m^{(1)}]^T$, where Y'_m is the solution to the Lyapunov equation with coefficient matrix $\widehat{\mathcal{T}}_m$, we obtain $AX'_m + X'_m A^T + BB^T = [\mathcal{V}_m, V_m^{(1)}](\widehat{\mathcal{T}}_m Y'_m + Y'_m \widehat{\mathcal{T}}_m + EE^T)[\mathcal{V}_m, V_m^{(1)}]^T = O$, and the proof is completed. ∎

If B is not a vector, then loss of rank while generating the basis may occur without convergence taking place. In this case, deflation strategies should be employed, as in standard block Krylov subspace methods [19].

In section 2 we discussed the relation between the approximation to the exponential and the projected Lyapunov equation. The new approach implicitly approximates $W = \exp(A)B$, with $\widehat{W} = \mathcal{V}_m \exp(\mathcal{T}_m)E$ in $\mathbf{K}_m = \mathcal{K}_{2m}(A, A^{-m}B)$, and results in this context can be used to theoretically motivate our approach. For A symmetric and B a vector, it is shown by Druskin and Knizhnerman in [11] that the choice $\mathcal{K}_{2m}(A, A^{-m}B)$ yields an improved approximation to the exponential, over the standard Krylov subspace $\mathcal{K}_{2m}(A, B)$. More precisely, they show that the quality of the approximation to the exponential (and in fact to a whole class of functions) in $\mathcal{K}_{2m}(A, B)$ corresponds to that in $\mathcal{K}_{\sqrt{2m}}(A, A^{-m}B)$. Therefore, using $\mathcal{K}_{\sqrt{2m}}(A, A^{-m}B)$ allows one to achieve the same approximation quality as a regular Krylov space with a much lower subspace dimension.

We explicitly observe that the method proposed in [11] aims at approximating [illegible]. In addition, the authors suggest to fix a priori the number k of matrix-vector products with A^{-1}, so as to implicitly build the approximation space $\mathrm{span}\{A^{-k}v, A^{-k+1}v, \ldots, Av, \ldots, A^{m-1}v, \ldots\}$, where m (but not k) is allowed to increase. Here we propose to increase the subspace dimension in "both directions" until convergence. This can be easily done thanks to the recurrence in Proposition 3.2 for the matrix $\mathcal{T}_m$.

4. The Cholesky-factorized ADI iteration. The Lyapunov equation was shown to be a "model problem" for the classical ADI iteration; see [12], [42], [13], [26]. More recently, a Cholesky-factorized (CF) version has made the method more appealing for large-scale computation, for the approximate solution can be written as the product of two low rank matrices [25], [28], [30]. We next review various implementation details of this method to clarify its use in our numerical comparisons in section 5. Finally, we should mention that for the CF-ADI method to be derived, A is required only to be stable; that is, its eigenvalues should lie in the left-half plane, whereas projection-type methods need dissipativity, to ensure that the matrix V^TAV in (2.1) is stable.

Given a set of parameters $p_1, p_2, \ldots, p_\ell$, the basic ADI iteration determines an approximate solution as (cf., e.g., [25])

$$X_0 = 0,\ X_j = -2p_j(A+p_jI)^{-1}BB^T(A+p_jI)^{-T} + (A+p_jI)^{-1}(A-p_jI)X_{j-1}(A-p_jI)^T(A+p_jI)^{-T}, \quad j=1,\ldots,\ell.$$

It can be shown that X_j is symmetric and positive (semi)definite, so that it can be represented as $X_j = Z_jZ_j^T$, and is iteratively built one set of s columns at a time. The actual implementation, hereafter CF-ADI, requires s solutions with $A+p_jI$ per iteration; see, e.g., [25, Algorithm 2]. A cyclic procedure is obtained by continuing the iteration above and cyclically using the parameters, and this is also referred to as the cyclic low rank Smith(ℓ) method [28]. The recurrence can be stopped as soon as the residual matrix $AZ_jZ_j^T + Z_jZ_j^TA^T + BB^T$ is sufficiently small in some norm. It was shown in [28] that

$$\|AZ_jZ_j^T + Z_jZ_j^TA^T + BB^T\|_F = \|[AZ_j, Z_j, B][Z, AZ_j, B]^T\|_F = \|r\mathcal{I}r^T\|_F, \tag{4.1}$$

where r is the triangular square matrix of the "economy-size" QR factorization of the matrix $[AZ_j, Z_j, B]$, and $\mathcal{I}$ is a permutation matrix. Due to memory limitations, the previous quantities cannot be computed explicitly for large problems; therefore, a criterion based on the current iterate is commonly employed. More precisely, if $Z_j = [Z_{j-1}, z_j]$, then using $\|Z_jZ_j^T - Z_{j-1}Z_{j-1}^T\|_F = \|z_j\|^2$, the iteration is stopped if $\|z_j\|$ (absolute) or $\|z_j\|/\|Z_{j-1}\|_F$ (relative) is sufficiently small, which implies that the recurrence stagnates. Note that neither of these last two stopping criteria ensures that convergence has taken place.

The ADI parameters represent the key ingredient for the real success of the method. Optimal parameters are known analytically when A has a real spectrum; see, e.g., the discussion in [13]. The situation is significantly more complicated for a complex spectrum. In both cases, however, quite accurate information on the spectral region is required. The computation of quasi-optimal parameters has raised considerable attention; see, e.g., [39], [13], [34], and references therein. In particular, an economical procedure was proposed by Penzl in [28]. In our experiments, we used the function[2] `lp_para` in the LYAPACK package [30] to compute these suboptimal ADI parameters.

Another important implementation aspect is related to the solutions with $A+p_jI$, $j=1,\ldots,\ell$. Each matrix $A+p_jI$ can be factorized once for all and the Cholesky factor stored and used cyclically when p_j is employed; cf., e.g., [30]. For the sake of comparison, we use this strategy in Example 5.1, where factorizing the matrix requires limited

[2]The Arnoldi procedures in `lp_para` were modified to include reorthogonalization, so as to limit the occurrence of unstable shifts.

memory resources. Since this procedure is clearly too memory-consuming in large-scale applications, we opted for solving systems with $A+p_jI$ repeatedly at occurrence.

An additional difficulty is given by the use of complex arithmetic when A is non-symmetric and the ADI parameters are complex conjugate. In the original CF-ADI iteration, conjugate parameters can be coupled [25]. In the cyclic version, coupling does not seem to be doable, and the whole computation is carried out in complex arithmetic to get a complex Z_j. Note that it is possible to recover a real factor Z_j at the cost of additional computation [30]; alternatively, Gugercin, Sorensen, and Antoulas suggest only dealing with real parameters, even when the matrix has a complex spectrum [18]. To fully explore the properties of the method, in our experiments with the cyclic CF-ADI method we decided to use complex parameters and thus allowed complex arithmetic throughout the iteration. Further considerations regarding the typical behavior of this method are reported in the next section.

5. Numerical experiments. In this section we report on our numerical experience with the new method described in section 3. Comparisons with the cyclic CF-ADI method are also considered. All reported experiments were obtained using Matlab 7.1 (R3) [27], on a PC with a 2 GHz processor, and 2 GBytes of RAM.

Both CPU time (in seconds) and number of iterations (as defined next) are used to measure the cost of the approaches. For the CF-ADI method, the number of iterations (counted as s times the number of cycles times the number of parameters) coincides with the number of shifted systems to be solved and also with the number of long vectors that need to be stored. For the K-PIK method, each (block) iteration involves s system solutions with A and $2s$ new vectors entering the basis. Therefore, for the Krylov approach the final memory requirements for the approximation space are given by at most $2s$ times the number of iterations.

In all tests with the new method, we also report the rank of the final approximate solution as discussed in section 3. The cost of this computation is included in the overall cost of the method. For the CF-ADI iteration, we did not attempt to compute the factor rank to limit the computational costs. Nonetheless, we remark that a procedure has been proposed in [18] to more cheaply determine a lower rank factor that is updated at each iteration. For the sake of generality, we do not reproduce this implementation, as it requires further conditions and additional computation per iteration, although it may provide significant memory savings when the ADI solution space is large. In general, this procedure does not improve the convergence rate of the method; more details can be found in [18]; see also [34] for a recent thorough analysis of the method.

Unless explicitly stated, the CF-ADI complex parameters were determined with search parameters $(\ell, k, m) = (10, 40, 20)$, where ℓ is the number of ADI parameters,[3] k is the number of Arnoldi iterations, and m is the number of inverted-Arnoldi iterations in the procedure `lp_para`; these are the parameters suggested in [28] for similar problems. We stress that the cost of generating the ADI parameters was taken into account in the performance evaluation. Indeed, unless some tuning on (ℓ, k, m) is needed (cf., for instance, Example 5.2), this preprocessing employs only a low percentage of the whole cost of the algorithm. Our numerical experience showed that the computation of the parameters was also sensitive to the choice of the starting vector that was used for generating spectral information. ADI parameters determined with the same values (ℓ, k, m) but with a different starting vector for the two Krylov subspaces were significantly different, affecting the performance of the ADI iteration.

[3] This number may need to be adjusted to accommodate complex conjugates.

. As already mentioned, a comparison between the new method and the cyclic CF-ADI method faces the problem of the different stopping criteria used in the two cases, namely, (3.6) and the one discussed in section 4. Indeed, the fact that the two stopping criteria are satisfied in the two cases does not ensure that the quality of the approximate solution is comparable. For this reason we decided to take the "user point of view"; that is, we measure what the user can really handle during the run of the algorithm. To this end, we fix a certain tolerance and evaluate the performance of the two methods by monitoring the associated "standard" stopping criterion. More precisely,

$$\text{K-PIK:} \qquad \frac{\|AX_m + X_m A^T + BB^T\|}{2\|A\|_F\|Y_m\|_F + \|B\|_F^2} \le \texttt{tol}, \tag{5.1}$$

$$\text{Cyclic CF-ADI:} \qquad \frac{\|z_j\|}{\|Z_{j-1}\|_F} \le \sqrt{\texttt{tol}}. \tag{5.2}$$

We recall that (5.2) corresponds to $\|X_j - X_{j-1}\|_F / \|X_{j-1}\|_F \le \texttt{tol}$. In all examples considered, we used $\texttt{tol} = 10^{-10}$. In some cases, the final residual norm (4.1) of the CF-ADI iteration was also computed, but this cost was not included in the total computational requirements of the method, because in most cases it was overwhelming. We found the CF-ADI residual norm often to be above the requested residual tolerance.

5.1. The case $s = 1$. In this section we focus on the case where B is a vector, that is, $s = 1$, to emphasize some implementation issues.

5.1. This example is taken from [28, Example 6.3] and describes a model of heat flow with convection in the given domain. The associated parabolic equation is given by

$$\mathbf{x}' = \mathbf{x}_{xx} + \mathbf{x}_{yy} - 10x\mathbf{x}_x - 1000y\mathbf{x}_y + \mathbf{b}(x,y)\mathbf{u}(t)$$

on the unit square, with Dirichlet boundary conditions. The 4900×4900 matrix A resulting from the centered finite difference discretization of the differential terms is nonsymmetric with complex eigenvalues and has 24220 nonzero entries. Vector b was taken to be the vector of all ones. This choice allows one to completely replicate the reported experiments. The results in the following table show the performance of the new method, compared to the cyclic CF-ADI iteration. For the sake of completeness, in this example we also report the performance of the CF-ADI method when all matrix factorizations are carried out at the beginning of the computation and thus stored, so that at each ADI iteration only triangular solutions are performed (CF-ADI_m column).

	K-PIK	CF-ADI	CF-ADI_m
CPU time (s)	1.1	14	9
# iterations	19	80	80
Dim. approx. space	38	80	80
Solution rank	35	80	80

The reported values show the competitiveness of the new method on this medium-size problem, with low CPU time and memory requirements. A posteriori, we computed the residual norm of the CF-ADI solution and found that it was above 10^{-5}. Following (3.5), we also computed the relative norm of the variation in the solution of the new method, which for this example was $5.7 \cdot 10^{-8}$. All of these numbers show

that the quality of the approximate solutions significantly differs, depending on the employed measure. In this particular case, we observed in a separate experiment that letting the K-PIK solution difference (3.5) go below the threshold 10^{-10} took only less than one additional second of CPU time.

EXAMPLE 5.2. This example is a three-dimensional variant of the previous problem, and the differential equation is given by

$$\mathbf{x}' = \mathbf{x}_{xx} + \mathbf{x}_{yy} + \mathbf{x}_{zz} - 10x\mathbf{x}_x - 1000y\mathbf{x}_y - 10\mathbf{x}_z + \mathbf{b}(x, y)\mathbf{u}(t)$$

on the unit cube, with Dirichlet boundary conditions. The nonsymmetric matrix resulting from centered finite difference discretization with 18 nodes in each direction is of size 5832×5832 and has 38880 nonzero entries. Once again, b was taken to be the vector of all ones. The results are shown in the first part of the following table.

n		K-PIK	CF-ADI (10,40,20)	CF-ADI (16,60,40)
5832	CPU time (s)	15	149	112
	# iterations	56	110	85
	Dim. approx. space	112	110	85
	Solution rank	47	110	85
		K-PIK	CF-ADI (16,60,40)	CF-ADI (30,60,60)
10648	CPU time (s)	43	336	236
	# iterations	45	85	65
	Dim. approx. space	90	85	65
	Solution rank	45	85	65

The reported numbers are consistent with those of the previous experiments. It is worth noticing that at completion, for the new method the difference $\|X_m - X_{m-1}\|/\|X_{m-1}\|$ was equal to about $2 \cdot 10^{-8}$ in both problems (cf. (3.5)).

The behavior for different CF-ADI parameters is also reported. The computed parameters for the two choices are also depicted in Figure 5.1, where the symbol "×"

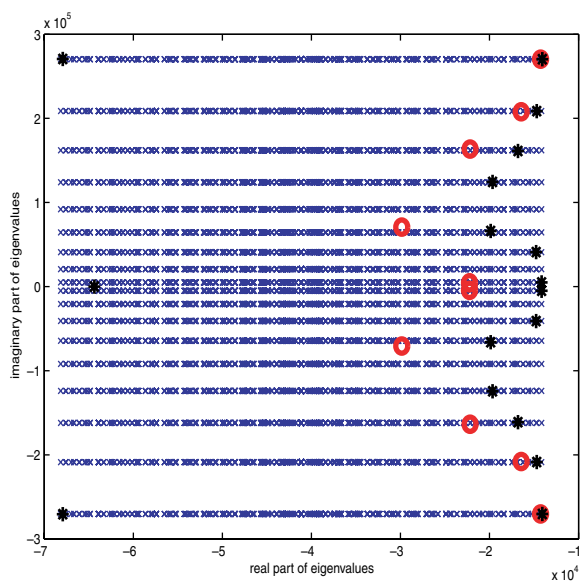


FIG. 5.1. *Example* 5.2. *Spectrum of matrix A (×), ADI parameters for* $(\ell, k, m) = (10, 40, 20)$ *(symbol o) and for* $(\ell, k, m) = (16, 60, 40)$ *(symbol *).*

denotes the eigenvalues of A, "o" the ADI parameters for $(\ell, k, m) = (10, 40, 20)$, and "*" the case $(\ell, k, m) = (16, 60, 40)$. The last choice is clearly more successful in capturing the relevant spectral boundary, suggesting better performance.

In the table above we also report the results for a run of the same problem with a slightly finer discretization, leading to a 10648×10648 nonsymmetric matrix A with 71632 nonzero entries. The results are consistent with those for the coarser mesh, confirming the difficulty of the CF-ADI preprocessing in finding effective ADI parameters even for different discretizations of the same problem. For $\ell = 30$, the CF-ADI stopping criterion was monitored at every iteration.

EXAMPLE 5.3. With this example we start our tests on symmetric problems. Here we consider the parabolic equation

$$\mathbf{x}' = \mathbf{x}_{xx} + \mathbf{x}_{yy} + \mathbf{x}_{zz} + \mathbf{b}(x, y)\mathbf{u}(t)$$

with homogeneous boundary conditions and discretize the 3D Laplace operator with centered finite differences in the unit cube. The resulting symmetric matrix for the given discretization mesh has size $n = 27000$. The vector b is the vector of all ones. Note that both methods can take great advantage of the symmetry of A. In particular, the CF-ADI is all performed in real arithmetic, with $\ell = 10$ real parameters. The results are reported below.

	K-PIK	CF-ADI
CPU time (s)	132	202
# iterations	8	20
Dim. approx. space	16	20
Solution rank	14	20

The numbers confirm the competitiveness of the new method, which does not require any preprocessing to gain spectral information.

EXAMPLE 5.4. We next consider a benchmark problem stemming from a semidiscretized heat transfer problem for optimal cooling of steel profiles [7]. The associated linear time-invariant system is given by

$$\begin{aligned} M\mathbf{x}'(t) &= N\mathbf{x}(t) + B_0\mathbf{u}(t), \quad \text{with} \quad t > 0, \\ \mathbf{x}(0) &= \mathbf{x}_0, \\ \mathbf{y}(t) &= C_0\mathbf{x}(t), \end{aligned}$$

with M symmetric positive definite, N symmetric negative definite, and B_0 of small column rank. We considered two tests available in the data set in [8], corresponding to different mesh resolutions, with dimensions $n = 1357$ (file `rail_1357_c60`) and $n = 5177$ (`rail_5177_c60`). In our experiments, we used the first column of B_0 for each data set.

The numerical treatment of this problem requires the discussion of some additional details. For M nonsingular, the problem can be recast as in (1.1). Using $M = LL^T$, we have $A = L^{-1}NL^{-T}$, $B = L^{-1}B_0$, and $C = C_0L^{-1}$. Clearly, none of these matrices is computed explicitly; rather, the following operations are performed:

$$Av = L^{-1}(N(L^{-T}v)), \qquad A^{-1}v = L^T(N^{-1}(Lv)).$$

Thus, matrix-vector products with A require the sequential solution of two triangular systems. To limit the computational cost, the stopping criterion (5.1) was used with $\|A\|_F \approx \|N\|_F$. In the case of the ADI iteration, the system solution is given as $(A + pI)^{-1}v = L^T(N + pM)^{-1}Lv$, therefore requiring the solution of $\ell = 10$ systems with coefficient matrix $N + pM$ at each ADI cycle.

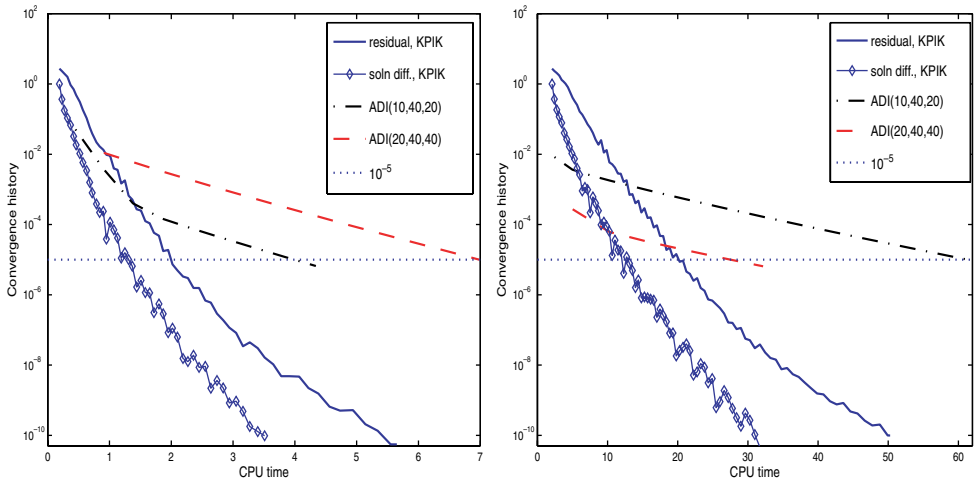


FIG. 5.2. *Example* 5.4. *Convergence history of the two methods. Left:* $n = 1357$. *Right:* $n = 5177$.

The results for both problems are summarized in the following table.

n		K-PIK	CF-ADI (10,40,20)	CF-ADI (20,40,40)
1357	CPU time (s)	5.67	4.36	6.96
	# iterations	59	90	140
	Dim. approx. space	118	90	140
	Solution rank	12	90	140
	Final residual F-norm	$< 10^{-10}$(†)	$2.8 \cdot 10^{-5}$	$2.8 \cdot 10^{-5}$
5177	CPU time (s)	50	62	32
	# iterations	86	220	120
	Dim. approx. space	172	220	120
	Solution rank	13	220	120
	Final residual F-norm	$< 10^{-10}$(†)	$9.9 \cdot 10^{-5}$	$9.9 \cdot 10^{-5}$

†: Before rank truncation

We observe that the first problem is quite small, and the K-PIK method is penalized, since its auxiliary costs (reorthogonalizations and the solution of a small Lyapunov equation) may be significant.

In this set of experiments, $\|B\| \approx 10^{-7}$ and $\|A\|_F \approx 10^{-3}$, so that $\|X\| \approx 10^{-3}$. These figures make the two stopping criteria behave very differently and seem to show that the CF-ADI method performs better than Krylov, when the parameters are properly chosen (note, however, that the performance of the parameters is dimension-dependent). We investigate further this example by reporting the convergence curves in Figure 5.2. There, we show the behavior of the relative difference $\|X_m - X_{m-1}\|/\|X_{m-1}\|$ for both methods and also the residual norm for the Krylov approach. Clearly, had the K-PIK convergence been measured in terms of the solution difference, the iteration would have been stopped much earlier, with significant savings in both memory and CPU time requirements. In the table above we also report the final relative residual norm (cf. (5.1)) of the CF-ADI iteration. According to this measure, the accuracy of the CF-ADI solution is not satisfactory, for a tolerance `tol`$=10^{-10}$.

5.2. The case $s \geq 1$. The algorithm for the general case $s \geq 1$ is reported in the appendix. In Table 5.1 we display the performance of the K-PIK and CF-ADI

TABLE 5.1
Experiments when B has multiple columns ($s \geq 1$).

	s	K-PIK			CF-ADI		
		CPU time	# its	dim. space	CPU time	# its	dim. space
Example 5.4	1	5.95	12	24	31.66	6	120
`tol`=10^{-5}	2	8.08	10	40	30.83	5	200
	4	11.11	7	56	40.20	5	400
	7	18.12	6	84	54.22	5	700
Example 5.2	1	38.95	34	68	588.68	5	150
`tol`=10^{-8}	2	50.50	33	132	633.41	5	300
	4	90.69	33	264	722.92	5	600
	7	204.91	32	448	857.57	5	1050

methods for $s = 1, 2, 4, 7$ on the largest matrices of two of the test problems. To make the comparison easier to interpret, and in light of the discussion related to Example 5.4, the relative change in the iterative solution is used as the stopping criterion for both methods. The original data in Example 5.4 included up to 7 columns of B, and these are all considered in the experiment. For Example 5.2 we picked up the first $s \leq 7$ columns of the matrix $B \in \mathbb{R}^{n\times 10}$ with random normal entries (random number generator initialized as `randn(''state'',0)`).

The K-PIK method clearly outperforms the CF-ADI procedure by keeping both CPU time and memory allocation limited. We recall that for the CF-ADI method, memory requirements may be reduced at some additional cost per iteration [18]. As s increases, the dimension of the "small" Lyapunov equation to be solved in K-PIK may become significant. To save computational time, the numerical solution of this equation may be thus performed periodically. Note that the number of iterations decreases as s increases; this is typical of "block" Krylov subspace methods.

5.3. The solution of the linear system with A. The K-PIK method requires solving s systems with A at each iteration, and this was performed by a direct method in the previous examples. For the sake of generality, in our experiments we have used natural orderings for the entries of A; however, it is well known that reordering techniques can dramatically improve the performance and memory requirements of a sparse direct solver. In practice, an appropriate reordering technique may allow the LU factorization to generate much sparser factors, which in turn implies faster solution with the triangular factors at each iteration. A typical such case is depicted in the following table, where K-PIK is used for the largest matrices in Examples 5.2 and 5.3, with $s = 1$ and with or without reordering of the matrix entries before the computation starts. The stopping tolerance for the Lyapunov solver is fixed at 10^{-10}, and the stopping criterion is (5.1). The functions `symamd` and `symrcm` are the Matlab implementation of "symmetric minimum degree" and of "reverse Cuthill–McKee" permutations, respectively. The former is particularly well suited for symmetric problems.

Example 5.3	Natural ordering	132 secs (8 iterations)
(sym. pb)	`symamd` ordering	31 secs (8 iterations)
Example 5.2	Natural ordering	43 secs (45 iterations)
(nonsym. pb)	`symrcm` ordering	22 secs (45 iterations)

If direct system solvers become prohibitively expensive, the K-PIK method can be implemented so that preconditioned iterative methods are used to solve the linear systems with A. In Table 5.2 we report the CPU time of K-PIK and $s = 1$, when the systems with A are solved by the bcgstab method with incomplete LU-type preconditioning (Matlab functions `luinc` and `cholinc`). The threshold used in the in-

TABLE 5.2
K-PIK on two test problems, with $s = 1$ and iterative solution of systems with A. Cholesky or LU preconditioners with fill-in threshold (Matlab functions `cholinc` *and* `luinc`*).*

	Inner tolerance	CPU time	# its	Avg. # inner its	Comments
Example 5.3	10^{-8}	26.19	8	17	Fill-in threshold 10^{-2}
(sym. pb)	10^{-5}	19.39	9	10	Fill-in threshold 10^{-2}
	10^{-3}	14.64	10	6	Fill-in threshold 10^{-2}
	10^{-3}	12.19	10	6	`symamd` permut., thr. 10^{-2}
Example 5.2	10^{-8}	22.92	45	4.5	Fill-in threshold 10^{-2}
(nonsym. pb)	10^{-5}	17.75	45	3	Fill-in threshold 10^{-2}
	10^{-3}	15.06	45	2	Fill-in threshold 10^{-2}
	10^{-3}	10.81	45	2	`symrcm` permut., thr. 10^{-2}
	10^{-3}	17.87	45	6	`symrcm` permut., thr. 10^{-1}

complete factorization allowed us to decrease the memory requirements for the factor by at least one order of magnitude. The stopping tolerance for the Lyapunov solver is again 10^{-10}, whereas various tolerances are tested for the iterative solution with A. All of these results significantly improve the performance of the corresponding direct method.

The first lines in Table 5.2 display results with a symmetric matrix. Due to the iterative solution of the system with A, so-called inexact solutions, this symmetry is lost. Nonetheless, the computation of the Lyapunov solver is still carried out by means of a short-term recurrence as if the operator were symmetric. This fact should explain the slight change in the total number of iterations as the inner system is solved less accurately, although the significant improvement in computational time more than compensates the higher number of iterations. We refer to [35] for pointers to similar so-called inner-outer methods in the linear system setting. The second set of experiments is with a nonsymmetric solver, for which full orthogonalization is performed. Note that the method does not seem to be particularly sensitive to a quite inaccurate solution of systems with A, showing that a rough approximation of the action of A^{-1} is sufficient to maintain fast convergence.

6. Conclusions. In this paper we have presented a new projection method for determining low rank approximate solutions to large-scale Lyapunov algebraic equations with a dissipative coefficient matrix. Our experiments show that the new algorithm is extremely competitive with the state-of-the-art solver cyclic CF-ADI, and it does not require parameter estimation. The new approach provides a natural, cheaply computable a posteriori stopping criterion. Note that such a measure is missing in the cyclic CF-ADI iteration. Moreover, the new method provides an economical procedure to further decrease the rank of the final approximate factor. The approach effectively reduces the original problem to one of much smaller dimension in a way that information on the problem augments as the dimension of the approximation space increases. This viewpoint is completely different from the CF-ADI type of iteration, where spectral information on the problem is gathered beforehand and then used repeatedly. This difference is reminiscent of the differences between semi-iterative and fully nonstationary methods in the solution of systems of linear equations [14]. We have tested the new approach on medium-size problems, while even greater benefits can be obtained when solving larger problems. In this case, the advantages observed in section 5.3 in using preconditioned iterative system solvers may be even more significant. We also mention that if the dimension of the approximation space becomes

large, then the costly step of solving the projected Lyapunov equation can be done periodically and not at every iteration. This small variant may significantly reduce the overall cost of the method.

The general framework described in section 2 suggests that further acceleration techniques could be derived by constructing new approximation spaces that satisfy the given constraints. Moreover, several computational issues deserve deeper investigation, such as the possibility of implementing truncation, restarting, or some other memory-conserving strategy. Finally, a complete round-off error analysis of the procedure would help fully understand the behavior of the method in finite precision arithmetic.

Appendix. In this appendix we describe the Matlab implementation of the K-PIK method when B is a tall $n \times s$ matrix. A direct method is used to solve systems with A.

```
function X=K_PIK(A,rhs,m,tol,k_max,period)

t=cputime;
[n,sh]=size(rhs);
nrmb=norm(rhs,'fro')^2;  nrma=norm(A,'fro');
Y=[]; odds=[];
[LA,UA]=lu(A); % For sym matrices use UA = chol(-A); LA = -UA';

s=2*sh;
rhs1=UA\(LA\rhs);
[ibeta,U(:,1:s)]=gram_sh([rhs, rhs1]);
VV = U;
H=sparse((m+1)*s,m*s);
L=sparse((m+1)*s,m*s);
beta=inv(ibeta); beta = beta(1:sh,1:sh); beta2=beta*beta';

for j=1:m,

    jms=(j-1)*s+1;j1s=(j+1)*s;js=j*s;js1=js+1; jsh=(j-1)*s+sh;
    Up(:,1:sh) = A*U(:,jms:jsh);
    Up(:,sh+1:s) = UA\(LA\U(:,jsh+1:js));

%new bases block (modified gram)
    for l=1:2
        k_min=max(1,j-k_max);
        for kk=k_min:j
            k1=(kk-1)*s+1; k2=kk*s;
            coef= U(1:n,k1:k2)'*Up;
            H(k1:k2,jms:js) = H(k1:k2,jms:js)+ coef; % U(1:n,k1:k2)'*Up;
            Up = Up - U(:,k1:k2)*coef; %H(k1:k2,jms:js);
        end
    end

  if (j<=m)
       [hinv,U(1:n,js1:j1s)]=gram_sh(Up);
       H(js1:j1s,jms:js)=inv(hinv);
  end

  I=speye(js+s);
  if (j==1),
      L(1:j*s+sh,(j-1)*sh+1:j*sh) = \ldots
      [ H(1:s+sh,1:sh)/ibeta(1:sh,1:sh), speye(s+sh,sh)/ibeta(1:sh,1:sh)]\ldots
                                  *ibeta(1:s,sh+1:s);
  else
      L(1:j*s+s,(j-1)*sh+1:j*sh) = L(1:j*s+s,(j-1)*sh+1:j*sh)\ldots
                                  + H(1:j*s+s,jms:jms-1+sh)*rho;
```

```
  end

  odds = [odds, jms:(jms-1+sh)];   % store the odd block columns
  evens = 1:js; evens(odds)=[];
  T(1:js+s,odds)=H(1:js+s,odds);   %odd columns

  T(1:js+sh,evens)=L(1:js+sh,1:j*sh);   %even columns
  L(1:j*s+s,j*sh+1:(j+1)*sh) = \ldots
       ( I(1:j*s+s,(js-sh+1):js)- T(1:js+s,1:js)*H(1:js,js-sh+1:js))\ldots
                                                  *hinv(sh+1:s,sh+1:s);
  rho = hinv(1:sh,1:sh)\hinv(1:sh,sh+1:s);

  I = speye(j*s);
  VV=U(:,1:j*s);
  k=j;

if (rem(j,period)==0)
  Y0=Y;
  Y = lyap2solve(full(T(1:js,1:js)),speye(k*s,sh)*beta2*speye(k*s,sh)');

  cc = [H(js1:j1s,js-s+1:js-sh), L(js1:j1s,(j-1)*sh+1:j*sh)];
  nrmx = norm(Y,'fro');
  er2(k) = norm(cc*Y(js-s+1:js,:),'fro')/(nrmb+2*nrma*nrmx);
  er2_dif(k) =\ldots
      norm([Y(1:js-s,1:js-s)-Y0, Y(1:js-s,js-s+1:js);Y(js-sh:js,:)],'fro')/nrmx;

  tot_time(j)=  cputime-t;
  disp([k,er2(k), er2_dif(k)])

  if (er2(k)<tol ),
  break
  end
 end
end

  [uY,sY]=eig(Y); [sY,id]=sort(diag(sY));
  sY=flipud(sY); uY=uY(:,id(end:-1:1));
  trunc_tol=1e-12;
  is=sum(abs(sY)>trunc_tol);
  Y0 = uY(:,1:is)*diag(sqrt(sY(1:is)));
  X = U(:,1:js)*Y0;
  er2(j+1) = er2(j);
  tot_time(j+1)=  cputime-t;
  total_time=cputime-t;

return
```

Acknowledgments. We thank Peter Benner and Jens Saak for insightful conversations and for pointing to [8]. We thank the referees for their comments. All reported results were obtained using the computer facilities of the SINCEM Laboratory at CIRSA, Ravenna.

REFERENCES

[1] A. C. Antoulas, *Approximation of Large-Scale Dynamical Systems*, Adv. Des. Control, SIAM, Philadelphia, 2005.

[2] R. H. Bartels and G. W. Stewart, *Algorithm* 432: *Solution of the Matrix Equation* $AX + XB = C$, Comm. ACM, 15 (1972), pp. 820–826.

[3] U. Baur and P. Benner, *Factorized solution of Lyapunov equations based on hierarchical matrix arithmetic*, Computing, 78 (2006), pp. 211–234.

[4] P. Benner, *Control Theory*, in Handbook of Linear Algebra, Leslie Hogben, ed., Chapman & Hall/CRC, 2006.

[5] P. Benner, D. Kressner, and V. Sima, *The SLICOT Toolboxes - A Survey*, Technical report, 2006.
[6] P. Benner, E. S. Quintana-Ortí, and G. Quintana-Ortí, *Solving Stable Sylvester Equations via Rational Iterative Schemes*, J. Sci. Comput., 28 (2006), pp. 51–83.
[7] P. Benner and J. Saak, *Efficient numerical solution of the LQR-problem for the heat equation*, PAMM, Proc. Appl. Math. Mech., 4 (2004), pp. 648–649.
[8] *Oberwolfach Model Reduction Benchmark Collection*, http://www.imtek.de/simulation/benchmark, 2003.
[9] M. J. Corless and A. E. Frazho, *Linear Systems and Control - An Operator Perspective*, Pure Appl. Math. 254, Marcel Dekker, New York, 2003.
[10] V. Druskin and L. Knizhnerman, *Two polynomial methods of calculating functions of symmetric matrices*, Comput. Math. Math. Phys., 29 (1989), pp. 112–121.
[11] V. Druskin and L. Knizhnerman, *Extended Krylov subspaces: Approximation of the matrix square root and related functions*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 755–771.
[12] N. S. Ellner and E. L. Wachspress, *New ADI model problem applications*, in Proceedings of 1986 ACM Fall Joint Computer Conference, Dallas, IEEE Press, Los Alamitos, CA, 1986, pp. 528–534.
[13] N. S. Ellner and E. L. Wachspress, *Alternating direction implicit iteration for systems with complex spectra*, SIAM J. Numer. Anal., 28 (1991), pp. 859–870.
[14] G. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, MD, 1996.
[15] G. H. Golub and R. Underwood, *The block Lanczos method for computing eigenvalues*, in Mathematical Software III, J. R. Rice, ed., Academic, New York, 1977, pp. 364–377.
[16] L. Grasedyck, *Existence of a low rank of H-matrix approximant to the solution of a Sylvester equation*, Numer. Linear Algebra Appl., 11 (2004), pp. 371–389.
[17] T. Gudmundsson and A. J. Laub, *Approximate solution of large sparse Lyapunov equations*, IEEE Trans. Automat. Control, 39 (1994), pp. 1110–1114.
[18] S. Gugercin, D. C. Sorensen, and A. C. Antoulas, *A modified low-rank Smith method for large-scale Lyapunov equations*, Numer. Algorithms, 32 (2003), pp. 27–55.
[19] M. H. Gutknecht, *Block Krylov space methods for linear systems with multiple right-hand sides: An introduction*, in Modern Mathematical Models, Methods and Algorithms for Real World Systems, A. H. Siddiqi, I. S. Duff, and O. Christensen, eds., Anamaya, New Delhi, to appear.
[20] S. J. Hammarling, *Numerical solution of the stable, non-negative definite Lyapunov equation*, IMA J. Numer. Anal., 2 (1982), pp. 303–323.
[21] M. Hochbruck and C. Lubich, *On Krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal., 34 (1997), pp. 1911–1925.
[22] M. Hochbruck and G. Starke, *Preconditioned Krylov subspace methods for Lyapunov matrix equations*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 156–171.
[23] I. M. Jaimoukha and E. M. Kasenally, *Krylov subspace methods for solving large Lyapunov equations*, SIAM J. Numer. Anal., 31 (1994), pp. 227–251.
[24] K. Jbilou and A.J. Riquet, *Projection methods for large Lyapunov matrix equations*, Linear Algebra Appl., 415 (2006), pp. 344–358.
[25] J.-R. Li and J. White, *Low Rank solutions of Lyapunov equations*, SIAM J. Matrix Anal. Appl., 24 (2002), pp. 260–280.
[26] A. Lu and E. L. Wachspress, *Solution of Lyapunov equations by alternating direction implicit iteration*, Comput. Math. Appl., 21 (1991), pp. 43–58.
[27] The MathWorks, Inc., *MATLAB* 7, September 2004.
[28] T. Penzl, *A cyclic low-rank Smith method for large sparse Lyapunov equations*, SIAM J. Sci. Comput., 21 (2000), pp. 1401–1418.
[29] T. Penzl, *Eigenvalue decay bounds for solutions of Lyapunov equations: The symmetric case*, Systems Control Lett., 40 (2000), pp. 139–144.
[30] T. Penzl, *Lyapack Users Guide*, Technical report SFB393/00-33, TU Chemnitz, 09107 Chemnitz, D, 2000. Available from `http://www.tu-chemnitz.de/sfb393/sfb00pr.html`.
[31] M. Robbé and M. Sadkane, *A convergence analysis of GMRES and FOM for Sylvester equations*, Numer. Algorithms, 30 (2002), pp. 71–89.
[32] Y. Saad, *Numerical solution of large Lyapunov equations*, in Proceedings of the International Symposium MTNS-89, Vol. III, M. A. Kaashoek, J. H. van Schuppen, and A. C. Ran, eds., Signal Processing, Scattering, Operator Theory, and Numerical Methods, Birkhauser, Boston, 1990, pp. 503–511.
[33] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.
[34] J. Sabino, *Solution of Large-Scale Lyapunov Equations via the Block Modified Smith Method*, Ph.D. thesis, Rice University, Houston, 2006.

[35] V. Simoncini and D. B. Szyld, *Recent computational developments in Krylov subspace methods for linear systems*, Numer. Linear Algebra Appl., 14 (2007), pp. 1–59.
[36] *The Control and Systems Library SLICOT*, http://www.slicot.de/, 2006.
[37] D. Sorensen and Y. Zhou, *Bounds on Eigenvalue Decay Rates and Sensitivity of Solutions to Lyapunov Equations*, Technical report 02-07, Rice University, Houston, TX, 2002.
[38] D. C. Sorensen and Y. Zhou, *Direct methods for matrix Sylvester and Lyapunov equations*, J. Appl. Math., 2003 (2003), pp. 277–303.
[39] G. Starke, *Fields of values and the ADI method for non-normal matrices*, Linear Algebra Appl., 180 (1993), pp. 199–218.
[40] H. Tal-Ezer, *Spectral methods in time for parabolic problems*, SIAM J. Numer. Anal., 26 (1989), pp. 1–11.
[41] A. van den Boom, A. Brown, F. Dumortier, A. Geurts, S. Hammarling, R. Kool, M. Vanbegin, P. Van Dooren, and S. Van Huffel, *SLICOT, a subroutine library for control and system theory*, in Preprints of the IFAC Symposium on CADCS, Swansea, UK, July 1991, pp. 89–94.
[42] E. L. Wachspress, *Iterative Solution of the Lyapunov Matrix Equations*. Appl. Math. Lett., 1 (1988), pp. 87–90.

SIAM J. SCI. COMPUT.
Vol. 29, No. 3, pp. 1289–1314
© 2007 Society for Industrial and Applied Mathematics

PARALLEL SYMBOLIC FACTORIZATION FOR SPARSE LU WITH STATIC PIVOTING*

LAURA GRIGORI†, JAMES W. DEMMEL‡, AND XIAOYE S. LI§

Abstract. This paper presents the design and implementation of a memory scalable parallel symbolic factorization algorithm for general sparse unsymmetric matrices. Our parallel algorithm uses a graph partitioning approach, applied to the graph of $|A|+|A|^T$, to partition the matrix in such a way that is good for sparsity preservation as well as for parallel factorization. The partitioning yields a so-called separator tree which represents the dependencies among the computations. We use the separator tree to distribute the input matrix over the processors using a block cyclic approach and a subtree to subprocessor mapping. The parallel algorithm performs a bottom-up traversal of the separator tree. With a combination of right-looking and left-looking partial factorizations, the algorithm obtains one column structure of L and one row structure of U at each step. The algorithm is implemented in C and MPI. From a performance study on large matrices, we show that the parallel algorithm significantly reduces the memory requirement of the symbolic factorization step, as well as the overall memory requirement of the parallel solver. It also often reduces the runtime of the sequential algorithm, which is already relatively small. In general, the parallel algorithm prevents the symbolic factorization step from being a time or memory bottleneck of the parallel solver.

Key words. sparse LU factorization, parallel symbolic factorization, memory scalability

AMS subject classifications. 65F50, 65F05

DOI. 10.1137/050638102

1. Introduction.

We consider solving a sparse unsymmetric system of linear equations $Ax = b$ using a direct method, which first factorizes the matrix A into the product of a unit lower triangular matrix L and an upper triangular matrix U, and then solves $LUx = b$ via two triangular solutions. One distinct property of sparse LU factorization is the notion of fill-in. That is, a zero element in the original matrix A can become nonzero in L and U. The purpose of [illegible] is to compute the nonzero structure of the factors L and U, which contains the original nonzero elements in A as well as the filled elements. When there is no need to pivot during numerical factorization, the nonzero structure of L and U can be computed without referring to the numerical values of the matrix. This allows us to allocate sufficient memory and to organize computations before numerical factorization. Computationally, symbolic factorization is usually much faster than numerical factorization, and hence there has not been much motivation to parallelize this step. However, as numerical factorization has been made more and more scalable, symbolic factorization

*Received by the editors August 12, 2005; accepted for publication (in revised form) December 1, 2006; published electronically June 1, 2007. This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy.

http://www.siam.org/journals/sisc/29-3/63810.html

†INRIA Futurs, Parc Club Orsay Universite, 4 rue Jacques Monod, Bat G, 91893 Orsay, France (Laura.Grigori@inria.fr).

‡Computer Science Division and Mathematics Department, University of California at Berkeley, Berkeley, CA 94720-1776 (demmel@cs.berkeley.edu). The work of this author was supported in part by Department of Energy grants DE-FC02-01ER25478 and DE-FC02-06ER25786, and National Science Foudation grant CNS-0325873

§Lawrence Berkeley National Laboratory, One Cyclotron Road, Mail Stop 50F-1650, Berkeley, CA 94720-8139 (XSLi@lbl.gov). The work of this author was supported by the Director, Office of Advanced Scientific Computing Research, Division of Mathematical, Information, and Computational Sciences of the U.S. Department of Energy under contract DE-AC03-76SF00098.

can become a serious bottleneck. In particular, a sequential algorithm requires the nonzero structure of the matrix A to be assembled onto one processor, which leads to a serious bottleneck in memory (if not timewise). We have therefore designed and implemented a memory scalable symbolic factorization algorithm for distributed memory machines. The parallel algorithm and the software are suitable for sparse LU factorization with static pivoting [19].

We developed this parallel algorithm in the context of the SuperLU_DIST [19] solver, which is a widely used sparse direct solver for distributed memory parallel computers. SuperLU_DIST contains several distinct steps in the solution process:

1. Choose a permutation matrix P_1 and diagonal matrices D_1 and D_2 so that the diagonal entries of $P_1 D_1 A D_2$ are ± 1 and the magnitudes of the off-diagonal entries are bounded by 1. The above equilibration and permutation help ensure accuracy of the computed solution. Currently, this is achieved using the HSL routine MC64 [5]. The purpose of the large-diagonal permutation is to decrease the probability of encountering small pivots during factorization.
2. Reorder the equations and variables by using a heuristic that chooses a permutation matrix P_2 so that the number of fill-in elements in the factors L and U of $P_2 P_1 D_1 A D_2 P_2^T$ is reduced.
3. Perform the symbolic factorization to identify the locations of the nonzero entries of L and U.
4. Compute the numerical values of L and U.
5. Perform triangular solutions.
6. Perform a few steps of an iterative method like iterative refinement if the solution is not accurate enough.

In the current version of SuperLU_DIST, the numerical factorization and triangular solutions are performed in parallel, but the first three steps are done sequentially. Our new parallel symbolic factorization algorithm uses ParMetis [17] (parallel Metis) to find a sparsity-preserving ordering (second step) and performs the subsequent structural analysis in parallel using a distributed input matrix (third step). Therefore, once we have integrated this algorithm in the solver, the only remaining sequential bottleneck will be in the first step, for which parallelization work is underway [20].

There are few results published in the literature on the parallelization of the symbolic factorization. Most of them consider the Cholesky factorization of symmetric positive definite (SPD) matrices. For these matrices, the symbolic factorization can be expressed by a tree structure, called the elimination tree, that represents the transitive reduction of the graph of the Cholesky factor L. The approach used in [22, 23] consists of computing in parallel first the structure of the elimination tree and then the symbolic factorization. While the latter step can be easily parallelized, the parallelization of the former is a difficult task and leads to modest speedup. A parallel symbolic factorization algorithm is also presented in [15] and is part of the PSPASES direct solver for SPD matrices. The algorithm is driven by the two-dimensional (2D) distribution of the data (designed for the numerical factorization step) and the elimination tree of A. The authors report that the parallel symbolic factorization takes relatively little time compared to the other steps of the solver.

In the case of unsymmetric matrices, the transitive reduction of the graphs of L and U leads to general directed acyclic graphs. The parallelization of the symbolic factorization of unsymmetric matrices is even more difficult than that of symmetric matrices, since it does not get the benefit of using a tree structure. A parallel algorithm for the factorization of unsymmetric matrices is presented in [2]. It is based on a

sequential algorithm that exhibits ideal parallelism but has two main drawbacks. It needs the input matrix to be replicated on every processor's local memory and hence is not scalable in memory. The sequential algorithm on which the parallel formulation is based is much slower in practice than more recent and improved sequential symbolic factorization algorithms.

The difficulty in formulating a scalable parallel algorithm lies in the small computation-to-communication ratio and in the sequential data dependency. Our primary goal in this research is to develop a parallel algorithm that provides memory scalability. At the same time, even if the serial runtime of the symbolic factorization is already relatively small, we still want to achieve reasonable speedup; otherwise it will dominate runtime with a larger number of processors. Our purpose is to obtain enough speedup to prevent this step from being a computational bottleneck. For this, our algorithm exploits two types of parallelism. The first type of parallelism relies on a graph partitioning approach, to reduce the fill-in and to permute the matrix to a suitable form for parallel execution. The graph partitioning is applied on the symmetrized matrix $|A| + |A|^T$, where $|A|$ denotes the matrix of absolute values of the entries of matrix A. The second type of parallelism is based on a block cyclic distribution of the matrix, which is a standard technique used in parallel matrix computations. To further decrease the runtime, we exploit the fact that as the elimination proceeds, the remaining submatrices in L and U have more and more nonzero elements, and at some point we can treat the remaining submatrix as a full matrix.

The choice of using a graph partitioning approach is in accordance with the state-of-the-art in parallelizing the ordering algorithms. The algorithms based on a local greedy strategy, such as the minimum-degree ordering algorithm, have proved to be difficult to parallelize. For example, the parallelization of one of the minimum-degree ordering algorithm's variants was discussed by Chen, Gilbert, and Toledo [1]. The parallel algorithm was designed for shared memory machines, and only limited speedups were obtained. The main sources of inefficiency come from insufficient parallelism and load imbalance. Nested dissection, on the other hand, is an algorithm that uses a top-down divide-and-conquer paradigm. It first computes a vertex separator of the entire graph that divides the matrix into two disconnected parts. The matrix is reordered such that the variables corresponding to the separator are ordered after those in the disconnected parts. This splitting-reordering process is then recursively applied to the two disconnected parts, respectively. The main advantage is that the reordered form of the matrix is suitable for parallel computation. The state-of-the-art nested dissection algorithms use multilevel partitioning [12, 13, 16]. A widely used, highly parallel code is ParMetis [17], which we will use as our ordering front-end preceding the symbolic factorization.

The rest of the paper is organized as follows. Section 2 briefly reviews several sequential symbolic factorization algorithms. Section 3 discusses the parallel symbolic factorization algorithm. Section 4 describes several techniques to identify dense submatrices in the factors L and U and thus reduce the runtime of the symbolic factorization. Section 5 presents the experimental results obtained when applying the algorithm on real world matrices, and section 6 concludes the paper.

2. Unsymmetric symbolic factorization. In this section we present briefly the various sequential algorithms described in the literature [6, 9, 10, 21], with an emphasis on the algorithm used in our parallel approach. The differences between the existing algorithms lie mostly in three aspects: (1) whether the algorithm uses the structure of A alone or also uses the partly computed structure of L and U; (2) dif-

ferent ways to improve the runtime of the algorithm; and (3) whether the algorithm computes the nonzero structure of the factors by rows, by columns, or by both rows and columns. As noted in [9], the algorithms' complexities are comparable. In practice, the number of operations performed is greater than $nnz(L+U)$, but is much smaller than the number of floating-point operations required for numerical factorization. However, the experiments performed on small sized matrices [9] and discussion of aspect (1) showed that the algorithms that use the structure of A alone are slower than the algorithms that use the structure of A and the partly computed structure of L and U. The other results addressing aspect (2) showed that improvements based on approaches like, for example, supernodes and symmetric pruning (we will describe these approaches later in this section) significantly reduced the symbolic factorization time.

Let A be a sparse unsymmetric $n \times n$ matrix and let $nnz(A)$ denote the number of its nonzero elements. In what follows, $Str(A)$ refers to the nonzero structure of the matrix A. a_{ij} represents the element of row i and column j of matrix A, while $A(k\colon r, i\colon j)$ represents the submatrix consisting of elements in rows k through r and columns i through j of A. When $k = 1$ and $r = n$, we use the notation $A(:, i\colon j)$, which represents columns i through j of A. $G(A)$ denotes the directed graph associated with matrix A. This graph has n vertices and an edge (i, j) for each nonzero element a_{ij}.

Our parallel algorithm is based on a variant that computes progressively the nonzero structure of L by columns and that of U by rows, using at each step the partial structure previously computed. That is, the structure of the ith column of L is formed by the union of $Str(A(i\colon n, i))$ with a subset of the previous columns of L. This subset of columns is determined from $Str(U(:, i))$. And the structure of the ith row of U is given in a similar way by the union of $Str(A(i, i\colon n))$ with a subset of the previous rows of U. The following theorem states how the structure of L can be determined by those of A and U.

THEOREM 2.1 (Rose and Tarjan [21]).

$$Str(L(:,i)) = Str(A(i\colon n, i) \cup \bigcup \{Str(L(i\colon n, j)) \mid (j,i) \in G(U(1\colon i-1,:))\}.$$

To decrease the runtime of the algorithm, we use two different improvements. The first improvement, called symmetric reduction or symmetric pruning, was proposed by Eisenstat and Liu [6]. It is based on the observation from Theorem 2.1 that only the structures of a subset of the columns $L(:, j)$ need to be considered in computing $Str(L(:, i))$. This subset is determined by the symmetric elements of the factors, which are used to remove some (but not all) redundant edges in the graphs of L and U. For example, if both l_{kj} and u_{jk} are nonzero and $k < i$, we must have $Str(L(k\colon n, j) \subseteq Str(L(:, k))$. Then, when computing $Str(L(:, i))$, to avoid redundant work, it is sufficient to consider only $Str(L(:, k))$. The symmetric reduction of the structure of $U(j, :)$ (structure of $L(:, j)$) is obtained by removing all nonzeros u_{jt} (nonzeros l_{tj}) for which l_{kj} and u_{jk} are nonzeros and $t > k$.

The second improvement is based on the notion of supernode, used in one of the more recent algorithms [4] and also in SuperLU_DIST [19]. In SuperLU_DIST, the symbolic factorization computes the structure of L and U by columns. A supernode groups together successive columns of L with the same nonzero structure, such that they can be treated as a dense submatrix. At the end of each step, the symbolic factorization algorithm decides whether the new column i of L belongs to the same

supernode as column $i-1$ of L. If it does, then it is not necessary to store the structure of column i in memory. That is, for each supernode, only the structure of its first column is stored. This leads to an important reduction in the memory needs of the symbolic factorization. Furthermore, supernodes also help reduce the runtime of the symbolic factorization. This is because to compute $Str(L(:,i))$, the previous supernodes are used instead of the columns $1,\ldots,i-1$.

In our variant, a supernode groups together successive columns of L having the same structure and successive rows of U having the same structure. This definition is appropriate for our algorithm that computes the structure of L by columns and that of U by rows. The symmetric pruning for each supernode consists of finding the first symmetric nonzero that does not belong to the diagonal block of the supernode. This is equal to the first symmetric nonzero of the last column of L and the last row of U of the supernode.

From a memory point of view, the usage of supernodes leads to a decrease in the memory needs of the symbolic factorization algorithm. From a computation point of view, the supernodes used in our variant do not lead to removing more redundant work in Theorem 2.1 than symmetric pruning alone does. For example, consider the computation of the structure of L and a supernode formed by indices $s, s+1,\ldots,t$. Denote by i the first symmetric nonzero of this supernode, where $i > t$. When supernodes are used, the structure of the first column of the supernode $L(:,s)$ is used in the computation of the structure of all the columns $L(:,j)$, with $t < j \leq i$ and $u_{sj} \neq 0$. Consider now that only symmetric pruning is used. For every index k such that $s \leq k < t$, the first symmetric nonzero corresponds to index $k+1$; that is, $l_{k+1,k}$ and $u_{k,k+1}$ are nonzero. Hence, all the entries after $k+1$ will be pruned. And the symmetric pruning information corresponding to column t of L and row t of U is the same as the symmetric pruning information corresponding to the supernode. For all the indices j with $t < j \leq i$ and $u_{tj} \neq 0$, the structure of column $L(:,t)$ is included in the structure of all the columns $L(:,j)$. But the structure of column t of L is the same as the structure of column s of L, and the structure of row t of U is the same as the structure of row s of U. Hence, computing the structure of $L(:,j)$ involves the same number of operations when symmetric pruning is used or when supernodes and symmetric pruning are used. However, since the symmetric pruning information corresponding only to the first column of L and row of U of a supernode is stored, the usage of supernodes helps decrease the time and the memory necessary for storing the symmetric pruning information.

Algorithm 1 presents this variant in a so-called left-looking approach. When it arrives at the ith step of computation, $Str(L(:,1{:}i-1))$ and $Str(U(1{:}i-1,:))$ are known. $Str(L(:,i))$ is formed using column i of the pruned structure of $U(1{:}i-1,:)$ (denoted as $prU(1{:}i-1,:)$). $Str(U(i,:))$ is formed using row i of the pruned structure of $L(:,1{:}i-1)$ (denoted as $prL(:,1{:}i-1)$). Then the first symmetric nonzero in $Str(L(:,i))$ and $Str(U(i,:))$ is determined. The pruned column i of L is added to prL, and the pruned row i of U is added to prU. For the sake of clarity, this algorithm does not use supernodes. In Section 3 we will also use a right-looking approach, in which, for example, for the structure of L, at the ith step of computation, the structure of column i is computed and then is used to update the partially computed structure of columns $i+1$ to n that need it.

The time spent in step 1 of Algorithm 1 is bounded by the number of operations needed to multiply the matrix $L(:,1{:}i-1)$ by the vector $prU(:,i)$ at each step i, which is written as $ops(L \cdot prU)$ [9]. Similarly, the time spent in step 2 is bounded by $ops(prL \cdot U)$. The complexity of the entire algorithm is dominated by these two steps.

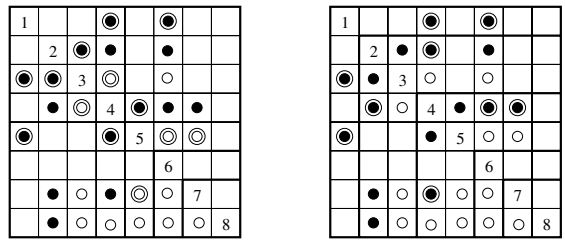


FIG. 2.1. *Example matrix A that illustrates Algorithm* 1 *using only symmetric pruning (left) and using symmetric pruning and supernodes (right).*

Algorithm 1. Left-Looking Symbolic Factorization.

for $i := 1$ to n **do**
  Compute $Str(L(:,i)) =$
  $Str(A(i{:}\,n,i)) \cup \bigcup \{Str(L(i{:}\,n,j)) \mid (j,i)$ is an edge of $G(prU(1{:}\,i-1,:))\}$
  Compute $Str(U(i,:)) =$
  $Str(A(i,i{:}\,n)) \cup \bigcup \{Str(U(j,i{:}\,n)) \mid (i,j)$ is an edge of $G(prL(:,1{:}\,i-1))\}$
  $k := n$
  determine $k := \min(j)$ such that l_{ji} and u_{ij} are nonzeros
  add $Str(L(i{:}\,k,i))$ to prL and $Str(U(i,i{:}\,k))$ to prU
end for

We illustrate Algorithm 1 in Figure 2.1 on an example matrix A with nonzero diagonal and of order 8. The matrix at the left shows the results obtained when only symmetric pruning is used, while the matrix at the right shows how Algorithm 1 performs when supernodes are used in addition to symmetric pruning. We suppose that the first 5 columns of L and the first 5 rows of U have been computed and that the algorithm arrives at the computation of the 6th column of L. The dark circles represent nonzero elements of the original matrix A, and the empty circles represent fill-in elements of the factors. The elements belonging to prL and prU are doubly circled.

Consider first that only symmetric pruning is used (left-hand side of Figure 2.1). For example, l_{32} and u_{23} are nonzeros; then for the index 2, only these two elements belong to prL and prU. The structure of $prU(:,6)$ is formed by the elements u_{16} and u_{56}. Hence, to compute the structure of $L(:,6)$, the algorithm forms the union of structures of $A(6{:}\,8,6)$, $L(6{:}\,8,1)$, and $L(6{:}\,8,5)$.

Consider now that supernodes are used in addition to symmetric pruning (right-hand side of Figure 2.1). Nodes 2 and 3 form a supernode, because columns 2 and 3 of L have the same structure and rows 2 and 3 of U have the same structure. Similarly, nodes 4 and 5 form another supernode. Symmetric pruning information corresponding only to the first column of L and row of U of a supernode is added to prL and prU. For example, l_{42} and u_{24} are nonzeros; then only elements l_{42} and u_{24} belong to prL and, respectively, prU (we store only the off-diagonal block indices). The symmetric nonzeros considered for pruning are l_{42} and u_{24}, because 4 is the first node not in the supernode formed by 2 and 3.

To compute the structure of $L(:,6)$, the algorithm forms the union of structures of $A(6{:}\,8,6)$, $L(6{:}\,8,1)$, and $L(6{:}\,8,4)$. Since indices 4 and 5 belong to the same supernode, the structure of $L(6{:}\,8,4)$ is the same as the structure of $L(6{:}\,8,5)$. Hence,

from a computation point of view, forming the structure of column 6 of L using supernodes and symmetric pruning is equivalent to the previous approach that uses only symmetric pruning. But the usage of supernodes decreases the memory needs. Only the columns of L and rows of U corresponding to indices $1, 2$, and 4 are stored.

3. Parallel unsymmetric symbolic factorization. We now describe our parallel unsymmetric symbolic factorization algorithm. Our algorithm uses the partly computed $Str(L)$ and $Str(U)$ as the algorithm progresses and exploits two types of parallelism. The first type comes from the graph partitioning approach, which is used as a fill-reducing ordering step. This approach partitions the matrix in a suitable way for parallel factorization and also provides a tree structure representing the dependencies among the computations. The second type of parallelism depends on a block cyclic distribution of the data.

The partitioning approach uses the graph of the symmetrized matrix $|A| + |A|^T$ to identify a vertex separator that splits the graph into two disconnected subgraphs. If the vertices of the separator are ordered after the vertices in the disconnected subgraphs, a blocked matrix suitable for parallel computation is obtained. The partitioning defines a binary tree structure, called the [illegible]. Each node of this tree corresponds to a separator. The root of the tree corresponds to the separator from the first level partitioning. Then each branch corresponds to one of the disconnected subgraphs. The partitioning can then be applied recursively on each subgraph. Figure 3.1 illustrates a matrix partitioned in two levels with four disconnected subgraphs. Its corresponding separator tree is depicted in Figure 3.2, where we number the levels of the binary tree starting with 0 at the bottom. This numbering does not reflect the top-down graph partitioning approach. Instead, it reflects the bottom-up traversal of the tree during our symbolic factorization algorithm. In Figure 3.1, the patterned blocks illustrate portions of the matrix that can contain nonzero elements. The meaning of the different patterns used will be described later in this section.

The separator tree plays an important role in sparse factorizations because it shows the parallelism from the partition of the matrix and identifies dependencies among the computations. If $Str(L(:, j))$ is needed in computing $Str(L(:, i))$, then the node owning index j belongs to the subtree rooted at the node owning index i.

In order to perform the symbolic factorization in parallel, the nodes of the separator tree are assigned to processors using the standard subtree-to-subcube algorithm [8, 11]. This algorithm considers that the separator tree is well balanced. That is, each partitioning leads to two disconnected graphs of roughly the same size. In practice, unbalanced trees can be obtained, and a better mapping of nodes to processors can be used. We plan to address this problem in our future work.

The subtree-to-subcube algorithm maps nodes to processors during a top-down traversal of the separator tree. It starts by assigning all the p processors to the root. Then it assigns $p/2$ processors to each of the two subtrees of the root. This process is continued until only one processor is assigned to a subtree. The advantage of this algorithm is that the communication in a subtree is restricted to only the processors assigned to this subtree, thus hopefully ensuring a low communication cost. Each node of the tree is associated with a range of indices. All the processors assigned to a node will cooperate to compute the structure of the columns of L and of the rows of U whose indices are associated with the node. The indices are distributed among those processors using a one-dimensional (1D) block cyclic distribution along the diagonal, in a block arrow shape.

Consider again Figures 3.1 and 3.2. Suppose that we have 4 processors. The

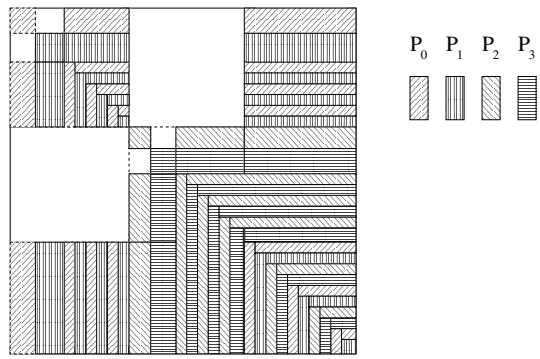


FIG. 3.1. *Example matrix and data distribution for the parallel symbolic factorization algorithm executed on* 4 *processors.*

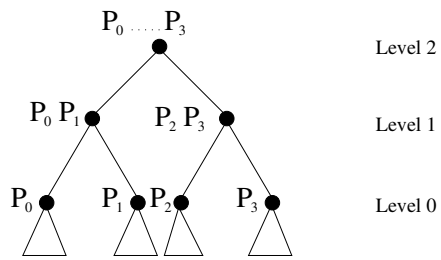


FIG. 3.2. *Separator tree corresponding to the example matrix in Figure* 3.1.

subtree-to-subcube mapping is illustrated in Figure 3.2 in which we label the processors assigned to each node of the separator tree. The 1D block cyclic data distribution is illustrated in Figure 3.1, where we use different patterns for different processors.

Algorithm 2 describes the main steps of the parallel symbolic factorization algorithm. Each processor performs a bottom-up traversal of the separator tree to participate in factorizing the nodes it owns. It starts with an independent symbolic factorization phase for the indices associated with its leaf node, using a left-looking algorithm similar to Algorithm 1. Then at each higher level of the tree, each processor is involved in computing one node D that it owns at this level. The computation of each D consists of two phases. In the first phase, named factorization, the columns of L and rows of U corresponding to the indices associated with D are updated in a right-looking manner by the columns of L and rows of U belonging to the predecessors of D in the separator tree. In the second phase, named factorization, all the processors participate in completing the symbolic factorization of node D, using a combination of right-looking and left-looking partial symbolic factorization methods.

We now use Figure 3.3 to illustrate the execution of Algorithm 2 step by step. Consider Level 1 of the tree. The computation of the right node D (doubly circled in the figure) is assigned to processors P_2, P_3. We use the following notation to distinguish the indices and their corresponding columns of L and rows of U associated with each node. The indices corresponding to the right node of Level 1 are marked in patterned white, and the indices to be used to update this node and that correspond to the leaf nodes assigned to P_2 and P_3 are in dark gray. All the other indices, not involved in the computation considered here, are displayed in light gray.

Algorithm 2. Parallel Symbolic Factorization.

let $myPE$ be my processor number, P be the total number of processors
compute symbolic factorization of leaf node in a left-looking fashion
for each level from $l := 1$ to $logP$ in the separator tree **do**
  let μ and ν be the first and last indices of node D that $myPE$ owns
  for $i := \mu$ to ν such that $myPE$ owns index i **do**
    init $Str(L(:, i)) := Str(A(i{:}\, n, i))$
    init $Str(U(i, :)) := Str(A(i, i{:}\, n))$
  end for
  Inter-level Symbolic Factorization ($myPE$, l) (Algorithm 3)
  Intra-level Symbolic Factorization ($myPE$, l) (Algorithm 4)
end for

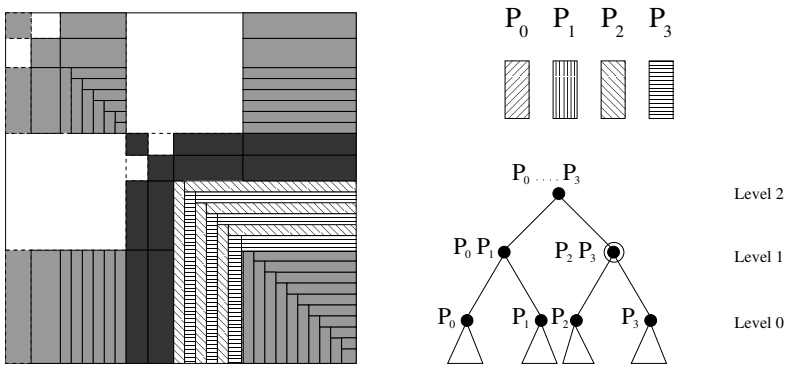


FIG. 3.3. *Example illustrating the parallel symbolic factorization (Algorithm 2) on the example matrix in Figure 3.1 for the doubly circled node assigned to processors P_2, P_3. Its associated indices are marked in patterned white, and the indices to be used to update this node and that correspond to the leaf nodes assigned to P_2 and P_3 are in dark gray. All the other indices are displayed in light gray.*

At this stage, processor P_2 has already computed the structures of L and U of the indices associated with the left leaf node, and processor P_3 has done so with the right leaf node. During inter-level factorization, processors P_2 and P_3 will determine which columns of L and which rows of U from the leaf nodes are needed for updating node D. This data is first exchanged between P_2 and P_3, and node D is then updated in a right-looking fashion. During intra-level factorization, the factorization of node D is completed using a block cyclic algorithm, where the factorization of each block is performed in a left-looking fashion, and then the block is used to update the subsequent blocks in a right-looking fashion.

Algorithm 3 details the inter-level factorization. Consider the computation of node D at level l, which is assigned to processors $P_r, \ldots, P_q$. This phase performs all the updates that the indices of node D must receive from the indices below node D. Only the indices associated with the nodes in the subtree rooted at D can contribute to these updates. Each processor determines the source of the updating data (columns of L and rows of U) that it owns and that belongs to nodes of the subtree rooted at D. Then the data is copied in a send buffer. Because of sparsity, it is very likely that this processor needs to send data to only a small subset of the processors that own node D. Several implementations of this collective communication are possible. In our current implementation, we use a simple scheme that assumes that the receive

buffer of each processor is large enough to store all the incoming messages. We have tested this scheme on the matrices in our test set and we have observed that the size of the receive buffer represents a small fraction of the total memory needs. Thus, we expect that this scheme will not affect the memory scalability of our algorithm.

Each processor can determine the destination processors to send the data. After a global exchange among the processors owning node D, each processor also knows from whom the data needs to be received. Afterwards, each processor sends its data to destination processors in a nonblocking manner, and then posts all the necessary receives. Since we need to release the send/receive buffers, the received data has to be consumed right away. This is achieved by using a right-looking approach, as described in Algorithm 5, to update the indices associated with node D. In our example, the indices that contribute to this update are shown in dark gray. Processors P_1 and P_2 exchange necessary parts of columns of L and rows of U corresponding to these indices.

Algorithm 3. Inter-level Symbolic Factorization.

Input: $myPE$: my processor number, l : level in the separator tree
let D be the node at level l owned by processor $myPE$
let $P_r, \ldots, P_q$ be the processors assigned to node D
let μ and ν be the first and last indices of node D
copy data (columns of L, rows of U) to be sent to $P_r, \ldots, P_q$ in $sendBuf$
determine the processors that $myPE$ needs to exchange data with
for each processor P in $P_r, \ldots, P_q$ to which data needs to be sent **do**
  nonblocking Send $sendBuf$ to P
end for
for each processor P in $P_r, \ldots, P_q$ from which data needs to be received **do**
  Recv data (columns of L, rows of U) in $recvBuf$
  Right-Looking update of $U(\mu{:}\,\nu, :)$ and $L(:, \mu{:}\,\nu)$ using $recvBuf$
end for
Wait for all nonblocking Sends to complete

Algorithm 4 details the intra-level factorization. The goal is to complete the factorization of node D by performing all the necessary updates between the indices associated with node D. The algorithm is a block variant of right-looking factorization. It provides a general framework to perform asynchronously a right-looking factorization of a sparse matrix. The algorithm proceeds as follows. Each processor $myPE$ iterates over the blocks of node D that it owns. Let $P_r, \ldots, P_q$ be the $q-r+1$ processors that are assigned to node D in 1D block cyclic fashion. Assume processor $myPE$ arrives at the computation of block K.

First, processor $myPE$ has to receive a number of messages from the processors that computed blocks $K-q+r, \ldots, K-1$ to update its blocks in a right-looking fashion. To determine the number of messages that have to be received, we use an auxiliary array, $noMsgsToRcv$, that is initialized to 0. This array keeps global information on the number of messages that every processor needs to receive. Locally, each processor stores the number of messages that were already received in the variable $myMsgs$.

Processor $myPE$ will receive the array $noMsgsToRcv$ from the owner of block $K-1$, and $noMsgsToRcv[myPE]$ gives the number of messages $myPE$ has to receive at this step of computation. This is implemented as follows. Processor $myPE$ posts a nonblocking receive from the owner of block $K-1$, anticipating the array $noMsgsToRcv$. The condition variable $rcvdArray$ is set when this nonblocking re-

ceive is completed. While waiting for this message, $myPE$ iterates over a loop, in which a nonblocking receive for data is posted at each iteration. When a message is received, if it corresponds to data, the later blocks of node D owned by $myPE$ are updated in a right-looking fashion. The variable $myMsgs$ is incremented by one. If the message received corresponds to the array $noMsgsToRcv$, then $myPE$ knows how many messages must be received. If all the messages were received, the last posted data receive is canceled. If not, $myPE$ continues with receiving all the necessary data messages.

Second, processor $myPE$ has to complete the factorization of block K itself. This is performed using a left-looking approach, similar to Algorithm 1. The difference is that the algorithm iterates from the first index of block K (μ_K) to the last index of block K (ν_K), instead of iterating from 1 to n as in Algorithm 1, where n is the order of the matrix A.

Third, processor $myPE$ needs to identify all the processors to which block K should be sent. For each such processor P, the value of $noMsgsToRcv[P]$ is incremented. And if K is not the last block of node D, $myPE$ sends the array $noMsgsToRcv$ to the owner of block $K+1$. Since the matrix is sparse, the data needs to be sent to only a small subset of processors. Empirically, we have observed that this number is usually 2 or 3. In particular, in symmetric cases, the data needs to be sent to only the processor owning the next block.

The block cyclic distribution of columns of L and rows of U helps maintain load balance. Note that we need the rows of U to determine what columns of L need to be sent to which processors. Similarly, we need the columns of L to determine to which processors we need to send the rows of U. It is easy to determine this communication pattern because of the arrow shape layout.

4. Exploiting density in the separators. In this section, we describe one optimization technique that can easily improve the runtime of symbolic factorization. It is well known that toward the end of the elimination, the factored matrices have more and more nonzero elements. In addition, we also observed that the submatrices associated with the separators have more and more nonzero elements. A symbolic factorization algorithm that computes $Str(L)$ by columns and $Str(U)$ by rows can easily identify the density in the factors. Moreover, since our parallel algorithm uses a 1D distribution of the data along the diagonal, this detection can be implemented without any communication.

4.1. Separator with dense diagonal block. Consider Algorithm 2 when it arrives at the intra-level computation of node D consisting of indices $\mu, \ldots, \nu$. Let B be the diagonal block submatrix associated with the separator, that is, $B = L(\mu{:}\,\nu, \mu{:}\,\nu) + U(\mu{:}\,\nu, \mu{:}\,\nu)$. Once the structures of column μ of L and row μ of U have been computed, the sizes $r = nnz(L(\mu{:}\,\nu, \mu))$ and $q = nnz(U(\mu, \mu{:}\,\nu))$ are known. The product rq gives a lower bound on the fill in B. If the product rq is $(\nu-\mu)^2$, B would be completely full. Then, there is no need to compute the subsequent rows of L and columns of U corresponding to the indices $\mu+1$ to ν. Note that even if B is not entirely full, we can use the ratio $rq/(\nu-\mu)^2$ as a parameter to control [illegible], that is, padding zeros in B to increase the size of supernodes.

The following theorems show that, given a full B on the diagonal, the off-diagonal block structures for L and U can also be easily determined. In fact, the L part is formed by the dense subrows that end at column ν, and the U part is formed by the dense subcolumns that end at row ν, as depicted in Figure 4.1. These structures are often referred to as [illegible] structures.

Algorithm 4. Intra-level Symbolic Factorization.

Input: $myPE$: my processor number, l : level in the separator tree
let D be node of level l owned by processor $myPE$
let $P_r, \ldots, P_q$ be the processors assigned to node D
let μ and ν be the first and last indices of node D
let N_l be number of blocks node D is divided into
initialize array $noMsgsToRcv[]$ to 0
initialize variable $myMsgs$ to 0
for each block K from 1 to N_l that $myPE$ owns **do**
  let μ_K and ν_K be the first and last indices of block K
  if K is not the first block **then**
    $rcvdArray = FALSE$
    post nonblocking Recv for array $noMsgsToRcv$ from owner of block $K-1$
    post nonblocking Recv for data in $recvBuf$
    while $myMsgs \neq noMsgsToRcv[myPE]$ *and* $rcvdArray \neq TRUE$ **do**
      Wait for a receive to complete
      if message received corresponds to data **then**
        increment variable $myMsgs$ by one
        Right-Looking update of $L(:, \mu_K{:}\nu)$ and $U(\mu_K{:}\nu, :)$ using $recvBuf$
      else
        /* message received corresponds to $noMsgsToRcv[]$*/
        Cancel nonblocking receive of data if $myMsgs = noMsgsToRcv[myPE]$
        $rcvdArray = TRUE$
      end if
    end while
  end if
  Left-Looking symbolic factorization to compute $L(:, \mu_K{:}\nu_K)$ and $U(\mu_K{:}\nu_K, :)$
  if K is not the last block **then**
    for each processor P in $P_r, \ldots, P_q$ to which $myPE$ needs to send data **do**
      increment $noMsgsToRcv[P]$ by one
      Send $L(:, \mu_K{:}\nu_K)$ and $U(\mu_K{:}\nu_K, :)$ to P
    end for
    Send $noMsgsToRcv[]$ to owner of block $K+1$
  end if
end for

Algorithm 5. Right-Looking Update of $L(:, \mu{:}\nu)$, $U(\mu{:}\nu, :)$.

Input: $recvBuf$, $L(:, \mu{:}\nu)$, $U(\mu{:}\nu, :)$, $myPE$
Output: Updated $L(:, \mu{:}\nu)$, $U(\mu{:}\nu, :)$
initialize prL, prU to empty
for each index j of $recvBuf$ **do**
  let k be such that l_{jk} and u_{kj} are nonzeros
  if there is an i such that $l_{ij} \neq 0$, $i \leq k$, $\mu \leq i \leq \nu$, $myPE$ owns index i **then**
    add pruned column j of L to prL
  end if
  if there is an i such that $u_{ji} \neq 0$, $i \leq k$, $\mu \leq i \leq \nu$, $myPE$ owns index i **then**
    add pruned row j of U to prU
  end if
end for
for $i := \mu$ to ν such that $myPE$ owns index i **do**
  Add to $Str(L(:, i))$ the union $\cup \{Str(L(i{:}n, j)) \mid (j, i)$ is an edge of $G(prU)\}$
  Add to $Str(U(i, :))$ the union $\cup \{Str(U(j, i{:}n)) \mid (i, j)$ is an edge of $G(prL)\}$
end for

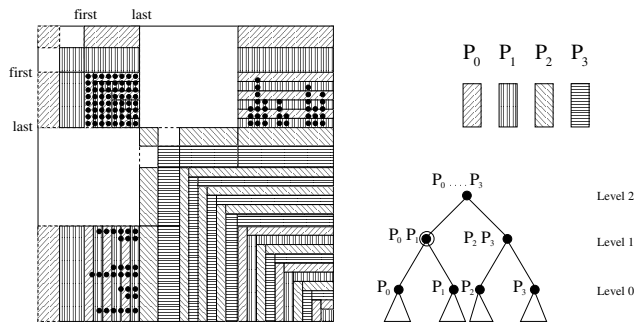


FIG. 4.1. *Example of the skyline structures of L and U when the separator has a dense diagonal block. The left node of level* 1 *(doubly circled) has the dense diagonal block from* μ *to* ν.

THEOREM 4.1. [illegible] $A = LU$ [illegible] $u_{i,i+1}$ [illegible] $i = \mu, \ldots, \nu - 1$ [illegible] a_{jk} [illegible] j [illegible] $\mu \leq k \leq \nu$ [illegible] $j > \nu$ [illegible] l_{jt} [illegible] t [illegible] $k < t \leq \nu$

[illegible]. First, l_{jk} is nonzero because a_{jk} is nonzero (here as elsewhere, we do not consider cancellation). Since $u_{i,i+1}$ is nonzero for all i, $\mu \leq i \leq \nu - 1$, then by Theorem 2.1, $Str(L(:,i)) \subseteq Str(L(:,i+1))$ for all i, $\mu \leq i \leq \nu - 1$. Hence l_{jt} is nonzero for all t from k to ν. □

THEOREM 4.2. [illegible] $A = LU$ [illegible] $l_{i+1,i}$ [illegible] $i = \mu, \ldots, \nu - 1$ [illegible] a_{jk} [illegible] j [illegible] $\mu \leq j \leq \nu$ [illegible] $k > \nu$ [illegible] u_{tk} [illegible] t [illegible] $j < t \leq \nu$

[illegible]. The proof is similar to the proof of Theorem 4.1. □

We explain now how the nonzero structure of a range of indices of L corresponding to node D is computed, using the fact that the diagonal B is full. The computation of the nonzero structure of U is similar. The key is to compute for each column i ($\mu \leq i \leq \nu$) a set $FNZ(i)$ that contains the row indices of the rows of A that have the first nonzero in column i. More formally, FNZ is defined as

$$FNZ(i) = \{j \mid a_{ji} \neq 0 \text{ and } a_{jt} = 0 \text{ for all } t \in (\mu{:}\, i-1)\}.$$

The computation of FNZ is achieved as follows. Let processors $P_r, \ldots, P_q$ be assigned to node D. First, every processor determines locally the column index i of the first nonzero in row j of A owned by this processor. Then a parallel reduction operation is applied among all the processors $P_r, \ldots, P_q$ to determine the first nonzero column index for each entire row j. This information is used to compute the structure FNZ. Then the structure of column i of L is simply the union of $FNZ(k)$ for $\mu \leq k \leq i$.

Note that the sets $FNZ(\mu), \ldots, FNZ(\nu)$ are disjoint. If data is well balanced among processors, the total computational cost is dominated by $O(nnz(L)/P)$ plus the cost of a call to *MPI_Allreduce*. In intra-level symbolic factorization, Algorithm 4 tries to identify dense diagonals after computing the structure of each index. When a dense column and row are found, this computation is performed for the rest of the separator.

4.2. Separators with dense structures belonging to a path of the separator tree. One further optimization is to identify the dense separators on a path of the separator tree. Consider again Algorithm 2 when it arrives at the intra-level

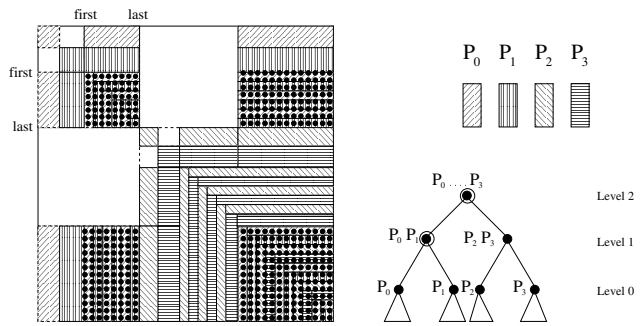


FIG. 4.2. *Example of the dense structures of L and U when there are dense separators on a path to the root. The nodes of the tree corresponding to the dense separators are doubly circled.*

computation of node D consisting of variables $\mu, \ldots, \nu$. Let B be the submatrix containing the variables associated with all the nodes on the path from node D to the root. Define $r = nnz(L(:, \mu))$ and $q = nnz(U(\mu, :))$. The product rq represents a lower bound on the number of fill in the submatrix B. If r and q are the same and equal the order of B, then we know B is full. That is, all the submatrices associated with the nodes on the path from D to the root are dense. This is illustrated in Figure 4.2. The processor that detects this μ column sends messages to all the other processors involved in the nodes on the path from D to the root. The rest of the algorithm is simply to generate the indices of these dense submatrices.

5. Experimental results. In this section, we present experimental results for our parallel symbolic factorization algorithm applied to real world matrices. We tested the algorithm on an IBM SP RS/6000 distributed memory machine at NERSC (National Energy Research Scientific Computing Center). The system contains 2944 compute processors distributed among 184 compute nodes. Each processor is clocked at 375 Mhz and has a peak performance of 1.5 GFlops. Each node of 16 processors has 16 to 64 Gbytes of shared memory.

We use several medium to large size matrices that are representative of a variety of application domains. The matrices and their characteristics are summarized in Table 5.1, which includes the matrix order, the number of nonzeros, the structural symmetry, and the application domain. We show the average number of nonzeros per column of original matrix A, as a measure of sparsity. The matrices are available from University of Florida Sparse Matrix collection [3] (BBMAT, STOMACH, PRE2, TORSO1, TWOTONE, MARK3JAC140SC, G7JAC200SC, LHR71C) and the accelerator design project at the Stanford Linear Accelerator Center [18] (DDS.QUAD.K-SM, DDS15.K-SM). To analyze the memory scalability of our algorithm, we also use matrices obtained from the 11-point discretization of the Laplacian operator on three-dimensional (3D) grids (REG3D).

The goal of the experiments is twofold. First, we want to study the performance of the new parallel symbolic factorization algorithm itself. Second, we want to examine the performance improvement of the entire solution process of SuperLU_DIST gained by parallelizing the symbolic factorization step. Our performance criteria include both runtime and memory usage. As stated in the introduction, our goal is to prevent the symbolic factorization step from being a bottleneck (especially in terms of memory) in the entire solution process. We show here that this goal is indeed achieved.

For the second study, we present detailed performance results for the first six

TABLE 5.1
Benchmark matrices.

Matrix	Order	$nnz(A)$	Structural sym.	$nnz(A)/n$	Application domain
REG3D	729000	7905960	1.0	10.8	3D Laplacian cubic grid size 90
DDS.QUAD.K-SM	380698	15844364	1.0	41.6	accelerator design
DDS15.K-SM	834575	13100653	1.0	15.7	accelerator design
STOMACH	213360	3021648	.85	14.2	bioengineering
BBMAT	38744	1771722	.53	45.7	fluid flow
PRE2	659033	5834044	.33	8.8	circuit simulation
TORSO1	116158	8516500	.42	73.3	bioengineering
TWOTONE	120750	1206265	.24	10.0	circuit simulation
MARK3JAC140SC	64089	376395	.07	5.9	economic modeling
G7JAC200SC	59310	717620	.03	12.1	economic modeling
LHR71C	70304	1528092	.00	21.7	chemical engineering

matrices in our test set. We compare the performance of two configurations of SuperLU_DIST: one is referred to as SLU_SFseq in which the serial symbolic factorization algorithm is used as before, and the other is referred to as SLU_SFpar in which the new parallel symbolic factorization algorithm is used. We keep the other phases the same as much as possible, including the scaling and the prepivoting using the sequential routine MC64 [5] from the HSL collection [14], the fill-reducing ordering using PARMETIS applied to the structure of $|A| + |A|^T$, the numerical LU factorization, and the triangular solutions.

Because of the different data structures used in symbolic factorization, the redistribution of the matrix from the storage format used in symbolic factorization to the storage format used in numeric factorization is different. We will report separately the timings for symbolic factorization ("SFseq" for SLU_SFseq and "SFpar" for SLU_SFpar) and the timings for redistribution ("RDseq" for SLU_SFseq and "RDpar" for SLU_SFpar). Another difference between the two configurations lies in the supernode definitions. SLU_SFseq groups in a supernode the columns of L with the same structure, whereas SLU_SFpar groups in a supernode the indices corresponding to the columns of L with the same structure and to the rows of U with the same structure. Thus, the definition in SLU_SFpar is more restrictive, and tends to give more supernodes of smaller sizes. This difference may lead to different runtimes of numerical factorization and triangular solutions, but empirically we observed that the differences are very small, and thus we report the average time obtained by these steps in the two solvers.

The matrix A is distributed among processors in a block row format, which is consistent with the input format required by PARMETIS. In SLU_SFseq, the distributed matrix is gathered such that every processor has a copy of it. Then MC64 is called on one processor to determine a row permutation, which in turn is broadcast to every other processor. The structure of $|A| + |A|^T$ is computed in parallel using the distributed input matrix, and then PARMETIS is called. The symbolic factorization step is performed redundantly on each processor. The distributed structure of L and U is then set up among the processors with a 2D block-cyclic distribution over a 2D processor grid, which will be used in numerical factorization.

In SLU_SFpar, the matrix is gathered on one processor only to perform the first step (calling MC64) sequentially. All the other steps involve the distributed input matrix. (Although this means that the solver as tested is not yet memory scalable,

the work in progress that parallelizes the first step [20] should solve this problem.) The structure of $|A| + |A|^T$ is computed in parallel, and then ParMetis is called. The data is then distributed in the arrow shaped format among the processors, and parallel symbolic factorization is performed. Afterwards, the data is redistributed from this arrow shaped format to the 2D block-cyclic distribution over a 2D processor grid, which will be used in numerical factorization.

Notice that SuperLU_DIST can be used on any number of processors. But the partitioning performed by ParMetis is only applicable to a number of processors that is a power of two. Therefore, in our code, we have another data redistribution phase when the number of processors to be used is not a power of two. That is, we choose the largest power of two that is at most the total number of processors and distribute the data to this subset of processors for parallel ordering and symbolic factorization. After that, we repopulate the matrix to the full set of processors.

To perform the same numeric factorization in both configurations, we turned off supernode relaxation in SuperLU_DIST. This is because the relaxation is determined using the elimination tree that we do not have in SLU_SFpar when the input matrix is distributed. Ideally, each processor can locally compute a partial elimination tree associated with each leaf node of the separator tree, and perform relaxation locally. But we have not implemented this scheme. In [4] it was shown that supernode relaxation leads to improvements of 5% to 15% in the running time of sequential SuperLU. We expect that similar improvements can be obtained in the running time of the distributed numeric factorization.

The experimental results for the first six matrices in our test set are presented in the runtime plots of Figure 5.1 and in the memory usage plots of Figure 5.2.

For the runtime performance, we display in each plot of Figure 5.1 the total time of the solver when using sequential or parallel symbolic factorization (SLU_SFseq and SLU_SFpar), the runtime of numerical factorization (Factor), parallel reordering (ParMetis), sequential and parallel symbolic factorization (SFseq and SFpar), and redistribution (RDseq and RDpar).

For memory usage evaluation, we report in the plots in Figure 5.2 the maximum memory requirement per processor throughout the entire solution process (except MC64; see below). The peak is likely to occur in the redistribution routine from the symbolic factorization data structure to the numeric factorization data structure, or in the numeric factorization routine. The redistribution routine needs as input the nonzero structures of L and U. Memory is allocated to store these structures along with the space for the numerical values of L and U. Several auxiliary arrays are used in this routine when placing the numerical values of A into their corresponding positions of the factors L and U. Those arrays and the input nonzero structures of L and U are deallocated on return. The memory need in the numerical factorization routine is given by the 2D storage of structural and numerical information of the factors plus several supplementary arrays needed for communication.

We ignore for the moment the memory needs of MC64, which is dominated by the size of the matrix A, because in the future this step will be replaced by a parallel algorithm [20]. We report only the memory needs of the symbolic factorization step and of the numeric factorization step. The memory needs of the other steps of the algorithm are dominated by the size of the distributed matrix A, the size of the distributed factor L, or the size of the distributed factor U, which are not the bottlenecks in the solver. Therefore, we do not report memory statistics for any of the other steps.

Our parallel symbolic factorization algorithm uses two kinds of data, replicated data and distributed data. The replicated data consists of two arrays of size n, where

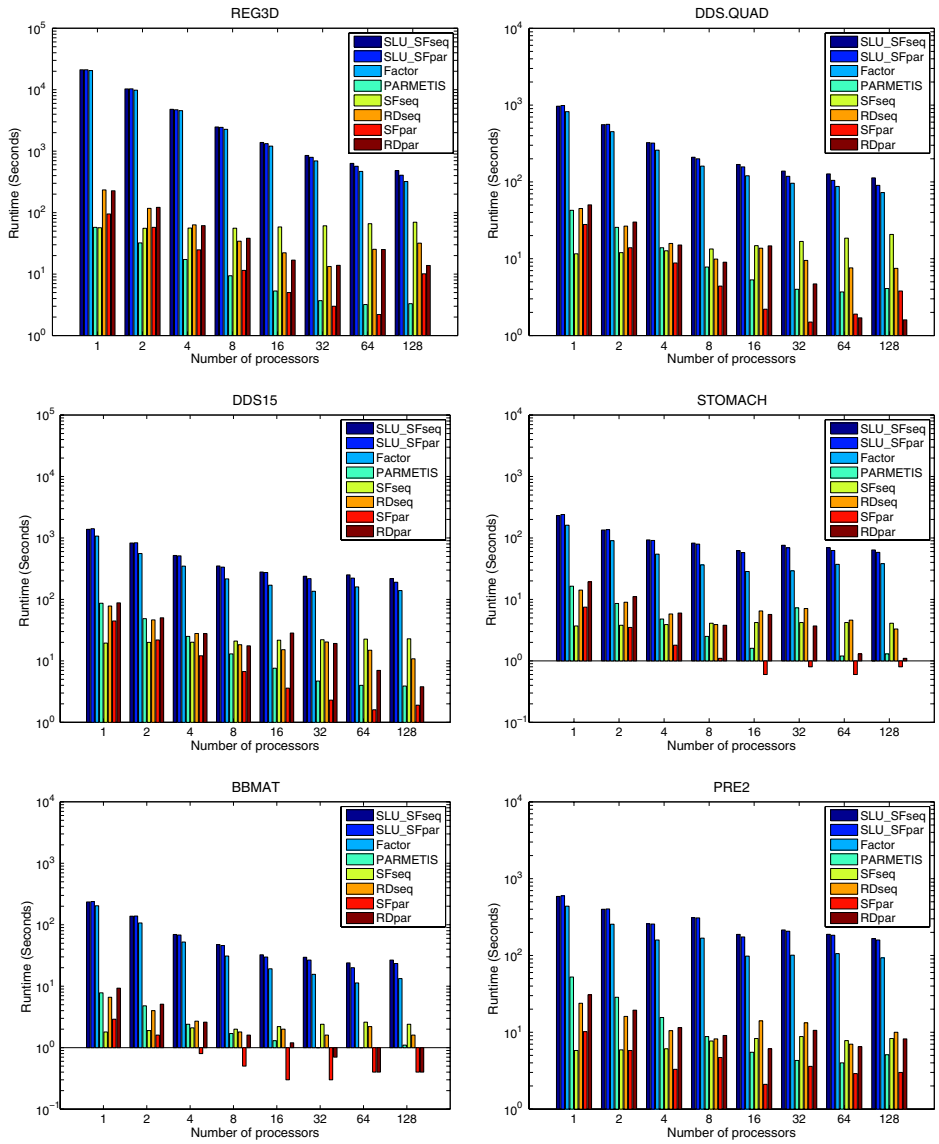


FIG. 5.1. *Runtimes (in logarithmic scale) for parallel reordering (*PARMETIS*), sequential and parallel symbolic factorization and redistribution (SFseq and SFpar, RDseq and RDpar), numeric factorization (Factor), and the entire solvers (SLU_SFseq and SLU_SFpar) for matrices in our test set.*

n is the order of the matrix. The first replicated array is used as a marker during the computation of $Str(L)$ and $Str(U)$. The second replicated array is a mapping array that shows for each variable its owner and its local index. That is, every variable has a global index which is known by all the processors and a local index which is known and used only by its owner. To set up the communication in the algorithms, each processor needs to determine the owners of certain variables. This local-global index translation is achieved by this mapping array. The distributed data are those whose size per processor is expected to decrease when increasing the number of processors.

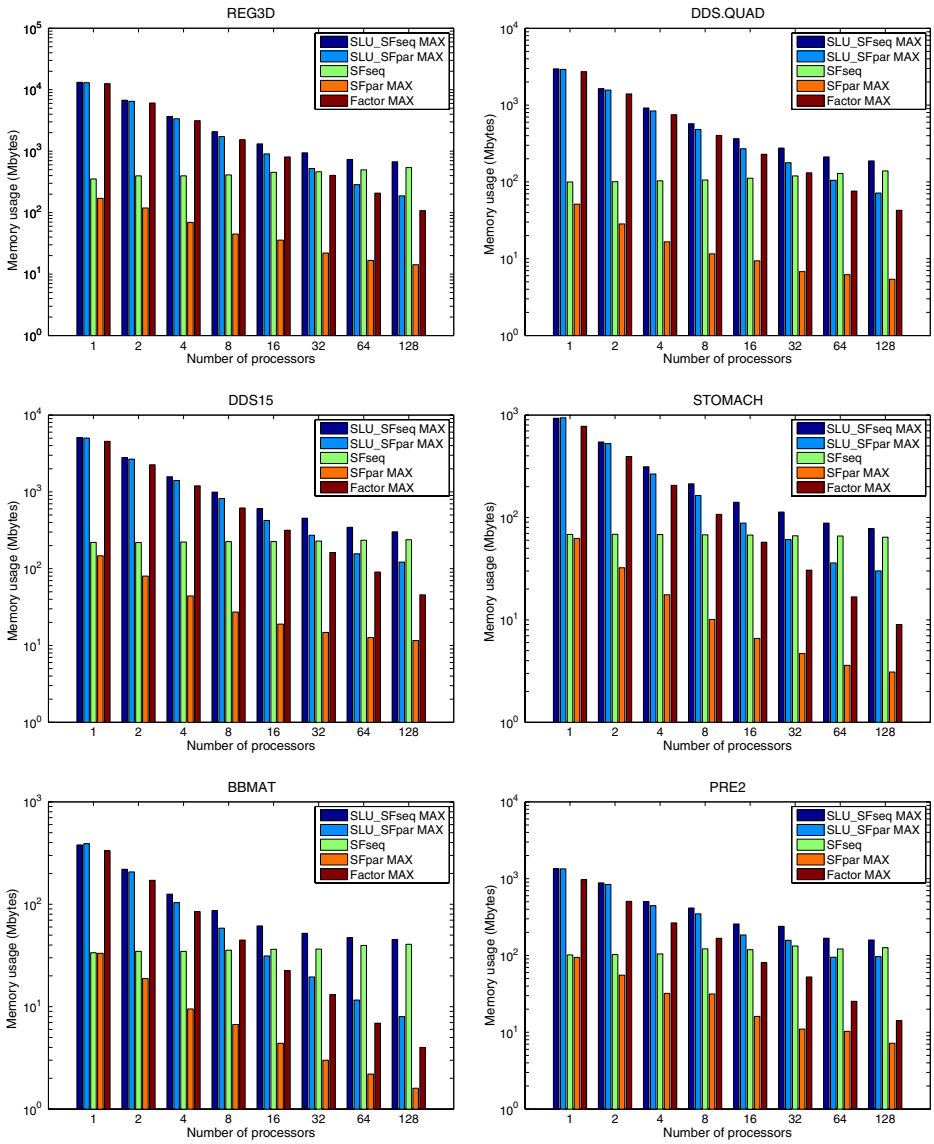


FIG. 5.2. *Memory usage (in logarithmic scale) for sequential and parallel symbolic factorization (SFseq and SFpar MAX), numeric factorization (Factor), and the entire solvers (SLU_SFseq and SLU_SFpar) for matrices in our test set.*

These are the arrays used to store the $Str(L)$ and $Str(U)$, the arrays used to store the pruned structures of L and U, and the buffers needed for communication.

In summary, the data reported is as follows: maximum memory used by the solver using sequential and parallel symbolic factorization (SLU_SFseq MAX and SLU_SFpar MAX), memory used during sequential and parallel symbolic factorization (SFseq and SFpar MAX), and maximum memory used during numerical factorization (Factor MAX).

Notice that the ordering results of PARMETIS differ using different number of processors. (They are usually worse with increasing number of processors.) Conse-

quently, the runtime and the memory usage corresponding to the sequential symbolic factorization change with different numbers of processors.

Since the algorithm behaviors are very different for the matrices that are structurally symmetric compared to those that are very unsymmetric, we first report the results separately for these two classes of problems.

5.1. Performance with structurally symmetric matrices. In this section we discuss the results with the structurally symmetric matrices. In fact, our symmetric test matrices are also positive definite, so there is no need to call MC64 (that can destroy the symmetry of the input matrix). The code was compiled using 64-bit addressing mode so that each process can use more than 2 GBytes of memory. In the parallel symbolic factorization algorithm, the block size used in the 1D block cyclic layout is 50.

In our parallel algorithm, the symmetrically pruned graph determines the communication pattern. For symmetric matrices, we obtain the benefit of symmetric reduction to the maximum extent. That is, the pruned graph contains only the first off-diagonal nonzero element in the first off-diagonal block, and is in fact the same as the supernodal elimination tree. Then, every variable is used in the computation of at most one other variable. The communication volume is therefore bounded by the number of nonzeros in $Str(L)$ and $Str(U)$. Thus the algorithm needs a low communication volume in both inter-level and intra-level factorizations. On the other hand, this also implies that the algorithm has a limited parallelism in the intra-level factorization, where a right-looking block cyclic approach is used.

We first examine the results for the model problem REG3D. We use a cubic grid of size $NX = NY = NZ = 90$ with an 11-point discretization. We see in the plot REG3D of Figure 5.1 that the runtime of SFpar continues to decrease with increasing number of processors up to 64, achieving very good speedup—30.0 on 64 processors. The redistribution routine implemented in SLU_SFpar is usually a little faster than that in SLU_SFseq, with differences larger on more processors. Overall, the entire SLU_SFpar solver is about 16% faster than SLU_SFseq on 128 processors.

The plot REG3D in Figure 5.2 reports the memory usage. On 32 processors, the memory need of SFseq becomes larger than that of the per-processor need in numeric factorization. On 128 processors, the memory need of SFseq is 5 times more than that of the per-processor need in numeric factorization. Thus, sequential symbolic factorization is indeed the bottleneck for memory scalability of SuperLU_DIST. Our parallel symbolic factorization algorithm achieves a good memory scalability. For example, on 128 processors, the memory need of SFpar is 38-fold smaller than that of SFseq. The reduction in the symbolic factorization phase also leads to significant reduction in the overall memory usage. On 128 processors, the maximum per-processor memory need of SLU_SFpar is only about 28% of that required by SLU_SFseq. Therefore, SLU_SFpar can solve a much larger problem.

We notice that on 8 and 16 processors, the entire memory needs of SLU_SFseq are larger than 16 GBytes, which is the available memory on most of the compute nodes of IBM SP RS/6000 at NERSC. In order to fit the problem in memory, we had to use more compute nodes than necessary. For example, SLU_SFseq on 16 processors was run on two compute nodes, with only 8 processors per node used. In addition, the runtime can slow down severely when the memory used per node is too large (causing disk swapping). In our experiments, this happened when SLU_SFseq was run on 32 processors and when SLU_SFpar was run on 16 processors. Hence, to obtain the best runtimes, we have used for these two runs two times more nodes than necessary.

We now examine the results for the irregular problems from the accelerator design. The plots DDS.QUAD.K-SM and DDS15.K-SM in Figure 5.1 show the runtimes of SLU_SFpar and SLU_SFseq for the two matrices. Again, SFpar exhibits good performance, with a maximum speedup of 14.1 on 64 processors for the matrix DDS15.K-SM. The time of SFpar is always smaller than the time spent in PARMETIS. On a small number of processors, the times spent in redistribution are comparable for RDseq and RDpar. But with increasing number of processors, the time spent in RDpar can be up to four times smaller than the time spent in RDseq. Overall, the entire SLU_SFpar solver is about 20% faster than SLU_SFseq on 128 processors for DDS.QUAD.K-SM.

The plots DDS.QUAD.K-SM and DDS15.K-SM in Figure 5.1 compare the memory usage of SLU_SFpar and SLU_SFseq and various steps in the solvers. We see again that with increasing number of processors, SFseq becomes the memory bottleneck of the SLU_SFseq solver. The parallel algorithm reduces the memory usage by more than 20-fold in symbolic factorization. Overall, the maximum per-processor memory need by the entire SLU_SFpar solver is only about 38–40% of that needed by SLU_SFseq.

5.2. Performance of structurally unsymmetric matrices. The plots BBMAT, STOMACH, and PRE2 in Figures 5.1 and 5.2 show the results with the unsymmetric matrices. Since these matrices are smaller than the symmetric ones, each processor needs no more than 2 GBytes of memory, and the code could succeed when compiled with 32-bit addressing mode. In the parallel symbolic factorization algorithm, the block size used in the 1D block cyclic layout is 50.

When comparing the runtimes of the two solvers SLU_SFpar and SLU_SFseq, we see that the time spent in numeric factorization dominates the time spent in any other step, which is true even on 128 processors. The runtime of SFseq is modest for all the matrices. This is probably because the sizes of those unsymmetric matrices are still quite moderate. The parallel algorithm further reduces this time. In particular, the time of SFpar is always smaller than the time spent in PARMETIS. The times spent for redistribution (RDseq and RDpar) are comparable on smaller numbers of processors (less than 16). But on larger numbers of processors, RDpar can be much faster than RDseq (up to 5-fold for BBMAT).

When comparing the memory usage of the sequential symbolic factorization of SLU_SFseq and the parallel symbolic factorization of SLU_SFpar, we observe that the memory usage is significantly reduced for all the matrices. It continues to decrease when increasing the number of processors up to 128. The parallel algorithm reduces the memory usage by up to 25-fold in symbolic factorization. Overall, the maximum per-processor memory usage of the entire SLU_SFpar ranges between 18% (BBMAT) and 60% (PRE2) of that of the SLU_SFseq solver.

5.3. Summary. Figure 5.3 summarizes the runtime and the memory usage of our parallel symbolic factorization algorithm for the first six matrices in our test set. In addition, the performance of our algorithm applied to matrices with higher asymmetry is shown in Figure 5.4.

For runtime scaling, we show the speedup obtained by our parallel symbolic factorization algorithm with respect to the sequential symbolic factorization algorithm. We see that for symmetric matrices, the runtime continues to decrease with increasing number of processors up to 32 for DDS.QUAD.K-SM, and up to 64 for REG3D and DDS15.K-SM. A maximum speedup of 30.0 on 64 processors is obtained for REG3D. We found that the technique for identifying dense separators plays an important role for symmetric matrices, significantly reducing the number of operations in symbolic factorization. The matrix REG3D represents the best case in terms of exploiting the

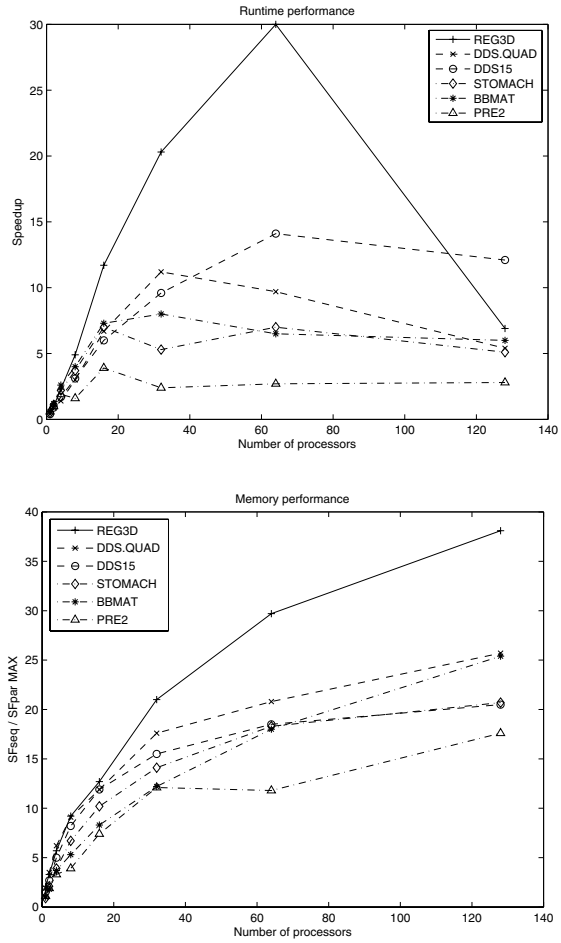


FIG. 5.3. *Speedup and memory saving of the parallel symbolic factorization algorithm for the first six matrices in our test set.*

dense separators, but we observe a big performance drop on 128 processors. After further investigation, we found that the technique of identifying dense separators was much less effective on 128 processors.

For five of our unsymmetric matrices, the time of the parallel symbolic factorization decreases when increasing the number of processors up to 16. The best speedups for SFpar are for matrices BBMAT (8-fold on 32 processors) and LHR71C (9-fold on 32 processors). These three matrices are relatively denser than the other matrices, and the technique of exploiting dense separators is more effective. For matrix PRE2, which is more unsymmetric and has very little dense structure, the algorithm achieves a speedup of almost 4 on 16 processors. For the other three matrices in our test set (TORSO1, TWOTONE, and G7JAC200SC), SFpar achieves only a speedup of two, even with increasing number of processors. However, for these matrices the time of SFpar is generally smaller than the time of PARMETIS. For TORSO1, the time spent in the graph partitioning phase can be 10 times larger than the time spent in SFpar.

For memory scaling, we display in Figures 5.3 and 5.4 the decrease in the mem-

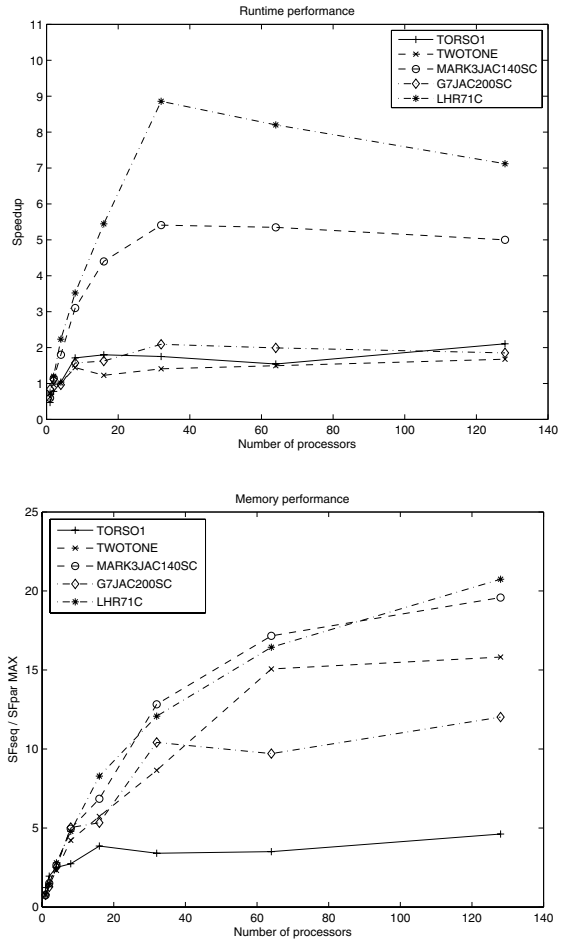


FIG. 5.4. *Speedup and memory saving of the parallel symbolic factorization algorithm for the last five matrices in our test set.*

ory usage of the parallel symbolic factorization algorithm SFpar with respect to the memory needs of the sequential symbolic factorization algorithm SFseq. The memory usage is significantly reduced for all the matrices, with the exception of TORSO1. In the former case, the reduction of memory is observed all the way up to 128 processors. The largest reduction is obtained with REG3D for which on 128 processors, the memory need of SFpar is 38-fold smaller than that of SFseq. For the unsymmetric matrices, the largest reductions occur with matrices BBMAT (up to 25-fold) and LHR71C (up to 21-fold). Note that on a large number of processors, for the unsymmetric matrices, the memory used to store the replicated data may become more than what is needed to store the distributed data. For example, for STOMACH the replicated data represents 77% of the memory needs on 128 processors. Still, these matrices help show that the parallel algorithm exhibits good memory scalability.

For TORSO1, the parallel symbolic factorization uses only 4.6 times less memory than the sequential symbolic factorization. However, in this case PARMETIS leads to very unbalanced partitioning, as quantified by differences in the sizes of the separators

(in terms of the number of their indices and the number of their nonzero elements of the input matrix) at each level of the separator tree. For example, we have found that on 64 processors, the largest ratio of the maximum number of indices per separator relative to the level's average number of indices per separator is obtained at the second level of the separator tree, where it is 9.6. On 128 processors this ratio is 9.9, and it is also obtained at the second level of the separator tree. At the lower level of the separator tree, the ratio of the maximum number of nonzero elements in a partition relative to the average number of nonzero elements at this level is equal to 24.9 on 64 processors and 43.3 on 128 processors. As we mentioned already earlier, our data distribution algorithm is not the best choice for unbalanced separator trees and can lead to disparity in the per-processor memory usage in such cases and to low overall memory reduction rates. Indeed, in the case of TORSO1, we find that the maximum per-processor memory usage is 7.6 times larger than the average memory usage on 64 processors and is 8.5 times larger on 128 processors, consistent with the above hypothesis. We plan to address this problem in our future work.

5.4. Scalability analysis. Recall that our main goal of this research is to develop a parallel symbolic factorization algorithm that is memory scalable. We now quantify the memory scalability of SFpar using the matrices arising from the 11-point discretization of the Laplacian operator on 3D cubic grids. The notion of isoefficiency is useful when studying the scalability of a parallel algorithm. A parallel algorithm is considered highly scalable if it keeps a constant efficiency as the number of processors increases while the problem size per processor is kept constant, where the problem size could be the size of the dataset or the amount of work. For a memory scalability study, the suitable quantity would be the dataset size, which is the size of the L and U factors in this case. The memory efficiency on P processors is computed as the ratio of the memory usage of the sequential algorithm (SFseq) over the product of the maximum per-processor memory usage of the parallel algorithm (SFpar) and P. That is, Mem. eff. $= \frac{mem(SFseq)}{max.mem(SFpar)\times P}$. Therefore, we choose the dimensions of the cubic grids in such a way that the nonzeros of L and U are maintained roughly constant per processor while increasing the number of processors. Theoretically, when nested dissection ordering is used, for each matrix of order n, the number of nonzeros is on the order of $O(n^{4/3})$ while the amount of work is on the order of $O(n^2)$ [7].

Note that since SFseq and SFpar use very different data organizations (SFseq is column oriented, whereas SFpar is both column and row oriented), the total memory requirements are very different. In fact, when run on one processor, SFpar usually uses less memory than SFseq. Therefore, for ease of understanding, we will use a simulated single processor run of SFpar as the sequential code baseline. Since the PARMETIS ordering gives different numbers of fill-ins using different numbers of processors, we do not use the actual run of SFpar on one processor. Instead, we compute a lower bound of the memory need of Algorithm 1 based on the amount of fill in L and U after PARMETIS ordering using P processors. This algorithm stores the structure of L by columns and the structure of U by rows. It also assumes that the input matrix A is stored by columns below the diagonal and by rows above the diagonal. The memory needs are as follows. Two arrays of size n and two arrays of size equal to the supernodal structure of L and U store the structure of the factors. Two arrays of size n and two arrays, each of size twice the number of supernodes, store the pruned graphs of L and U. Two arrays, each of size n, are used to identify supernodes. One array, of size n, is used as a marker during the computation of the structure of L and U. Two arrays of size n and two arrays of size equal to the number of nonzeros of the

TABLE 5.2
Time and memory usage of the parallel symbolic factorization algorithm when the average number of nonzeros of L and U per processor is kept constant.

P	Grid size	Flops/P (10^9)	$nnz(L+U)$/P (10^6)	Time	Mem. SFpar MAX	Mem. SFpar AVG	Mem. SFseq	Mem. eff. (%)
1	29	5.5	9.5	.8	5.3	5.3	5.2	98
2	34	8.6	10.2	.9	5.0	4.9	9.4	94
4	39	12.2	10.0	.8	5.3	4.8	17.6	83
8	46	16.2	9.9	.8	6.0	4.8	34.9	73
16	53	21.2	9.7	.7	5.9	5.0	63.0	67
32	62	29.6	9.8	.8	6.8	5.5	111.1	51
64	74	46.0	10.7	1.1	8.8	7.2	220.2	39
128	85	55.8	10.0	9.8	12.1	9.2	390.2	25

original matrix A are used to store the structure of the input matrix.

Table 5.2 reports the size of the grids used, the number of operations and the size of the factors per process, the time of SFpar, and the memory used in the algorithm. The last two columns give the computed memory lower bound of a sequential symbolic factorization and the memory efficiency. We see that the algorithm achieves very respectable efficiencies ranging from 89% on 2 processors to 25% on 128 processors. The decrease in efficiency with increasing number of processors is mainly due to the two duplicated arrays of size n, which increase with increasing problem size. For example, the size of the two arrays on 64 processors is 3.2 MBytes, and on 128 processors is 4.9 MBytes. This represents almost half of the per-processor memory need. But since now the memory demand in the symbolic factorization phase is not a bottleneck anymore, there is no point in reducing this memory usage at the expense of the extra runtime.

The parallel runtime is almost constant up to 64 processors. But we observe a significant increase on 128 processors. With further study, we found that this is due to the same reasons as were previously observed with matrix REG3D. That is, the technique of identifying dense separators was less effective on 128 processors than on a smaller number of processors.

6. Conclusions. We have presented a parallel symbolic factorization algorithm for sparse Gaussian elimination with static pivoting. The algorithm is suitable when the equations and variables are preordered by a nested dissection algorithm, from which a binary separator tree is available, and our symbolic factorization algorithm exploits parallelism exposed by this separator tree.

The performance of the parallel algorithm depends on the partitioning and reordering (e.g., by PARMETIS), the structural symmetry, and the density of the matrix. The more symmetric the matrix is, the smaller the symmetrically pruned graph is, which leads to a smaller amount of communication. The denser the matrix, the denser the separators, which the parallel algorithm exploits for faster structural analysis. For the symmetric test cases, the parallel symbolic factorization algorithm SFpar has achieved a 14.1-fold speedup on 64 processors when compared with the sequential symbolic factorization algorithm SFseq (DDS15.K-SM; see Figure 5.3), and the entire solver SLU_SFpar is up to 20% faster than the solver SLU_SFseq (DDS.QUAD.K-SM; see Figure 5.1). For the unsymmetric cases, the parallel algorithm SFpar has achieved a 9-fold speedup on 32 processors (LHR71C; see Figure 5.4), and the entire solver SLU_SFpar is up to 16.7% faster than the solver SLU_SFseq on 64 processors (BBMAT; see Figure 5.1).

On one processor, SFpar can be slower than SFseq. But on a larger number of processors, SFpar can be much faster than SFseq.

In the redistribution step, both versions achieve good speedup. During this step, the communication patterns for numerical factorization and triangular solutions are determined. In SLU_SFseq, since every processor has a copy of the reduced structures of L and U, there is no communication when computing the communication pattern. In SLU_SFpar, with the distributed reduced structures of L and U, there are fewer accesses to the data on each processor, but there are more all-to-all communications to determine the communication pattern. This explains why on smaller numbers of processors, RDseq is faster, while on large number of processors, RDpar is faster (up to four times faster for several matrices).

The SFpar algorithm exhibits very good memory scalability. For many matrices, on one processor the memory usage of SFpar is less than that of SFseq. For all the matrices, with the exception of one (TORSO1), the maximum per-processor memory usage of SFpar continues to decrease with increasing number of processors up to 128. For the unsymmetric test cases, on 128 processors, the SFpar maximum per-processor memory reduction is up to 25-fold compared with SFseq, and the entire SLU_SFpar solver has up to a 5-fold reduction in maximum per-processor memory need compared with the entire SLU_SFseq solver (BBMAT; see Figure 5.2).

For matrix TORSO1, PARMETIS leads to an unbalanced partitioning and a corresponding unbalanced separator tree. Our data distribution algorithm is not well suited for unbalanced separator trees, and SFpar leads to a modest reduction of the memory requirement of the symbolic factorization. The imbalance in the partitioning can be related to the fact that the graph partitioning is applied on the symmetrized input matrix $|A|+|A|^T$. Two possible solutions can be envisaged, which could remedy this problem. First, one could try to use a partitioning algorithm that accounts for the asymmetry of the input matrix. Second, one could use a better mapping of the data to processors that takes into account that the separator tree is unbalanced. We will leave these options to be addressed in the future.

Notwithstanding that, the experimental results show that SFpar greatly increases the problem size the parallel SuperLU solver can handle, and the symbolic factorization effectively removes a memory bottleneck in many, if not most, common cases.

Acknowledgment. The authors thank the anonymous reviewers for their careful reading and the suggestions for improving the paper.

REFERENCES

[1] T.-Y. CHEN, J. R. GILBERT, AND S. TOLEDO, *Toward an efficient column minimum degree code for symmetric multiprocessors*, in Proceedings of the Ninth SIAM Conference on Parallel Processing for Scientific Computing, SIAM, Philadelphia, 1999.

[2] M. COSNARD AND L. GRIGORI, *A parallel algorithm for sparse symbolic LU factorization without pivoting on out-of-core matrices*, in Proceedings of the 15th International Conference on Supercomputing, ACM Press, New York, 2001, pp. 146–153.

[3] T. DAVIS, *University of Florida Sparse Matrix Collection*, NA Digest, vol. 92, no. 42, October 16, 1994, NA Digest, vol. 96, no. 28, July 23, 1996, and NA Digest, vol. 97, no. 23, June 7, 1997; http://www.cise.ufl.edu/research/sparse/matrices.

[4] J. W. DEMMEL, S. C. EISENSTAT, J. R. GILBERT, X. S. LI, AND J. W. H. LIU, *A supernodal approach to sparse partial pivoting*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 720–755.

[5] I. S. DUFF AND J. KOSTER, *On algorithms for permuting large entries to the diagonal of a sparse matrix*, SIAM J. Matrix Anal. Appl., 22 (2001), pp. 973–996.

[6] S. C. EISENSTAT AND J. W. H. LIU, *Exploiting structural symmetry in unsymmetric sparse symbolic factorization*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 202–211.

[7] A. George and J. W. H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice–Hall, Englewood Cliffs, NJ, 1981.

[8] A. George, J. W. H. Liu, and E. Ng, *Communication results for parallel sparse Cholesky factorization on a hypercube*, Parallel Comput., 10 (1989), pp. 287–298.

[9] J. R. Gilbert and J. W. H. Liu, *Elimination structures for unsymmetric sparse LU factors*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 334–352.

[10] A. Gupta, *Improved symbolic and numerical factorization algorithms for unsymmetric sparse matrices*, SIAM J. Matrix Anal. Appl., 24 (2002), pp. 529–552.

[11] A. Gupta, G. Karypis, and V. Kumar, *Highly scalable parallel algorithms for sparse matrix factorization*, IEEE Trans. Parallel Distrib. Syst., 8 (1997), pp. 502–520.

[12] B. Hendrickson and R. Leland, *An improved spectral graph partitioning algorithm for mapping parallel computations*, SIAM J. Sci. Comput., 16 (1995), pp. 452–469.

[13] B. Hendrickson and R. Leland, *A multilevel algorithm for partitioning graphs*, in Proceedings of SuperComputing '95, ACM Press, New York, 1995.

[14] HSL, *A Collection of Fortran Codes for Large-Scale Scientific Computation*, http://www.cse.clrc.ac.uk/nag/hsl/ (2004).

[15] M. Joshi, G. Karypis, V. Kumar, A. Gupta, and F. Gustavson, *PSPASES: An efficient and scalable parallel sparse direct solver*, in Proceedings of the Ninth SIAM Conference on Parallel Processing for Scientific Computing, SIAM, Philadelphia, 1999.

[16] G. Karypis and V. Kumar, *A parallel algorithm for multilevel graph partitioning and sparse matrix ordering*, J. Parallel Distrib. Comput., 48 (1998), pp. 71–95.

[17] G. Karypis, K. Schloegel, and V. Kumar, ParMeTiS*: Parallel Graph Partitioning and Fill-Reducing Matrix Ordering, Version* 3.1, University of Minnesota, http://www-users.cs.umn.edu/∼karypis/metis/parmetis/ (August 2003).

[18] K. Ko, N. Folwell, L. Ge, A. Guetz, V. Ivanov, L. Lee, Z. Li, I. Malik, W. Mi, C. Ng, and M. Wolf, *Electromagnetic Systems Simulation—From Simulation to Fabrication*, sciDAC report, Stanford Linear Accelerator Center, Menlo Park, CA, 2003.

[19] X. S. Li and J. W. Demmel, *SuperLU_DIST: A scalable distributed-memory sparse direct solver for unsymmetric linear systems*, ACM Trans. Math. Software, 29 (2003), pp. 110–140.

[20] J. Riedy and J. W. Demmel, *Parallel Bipartite Matching for Sparse Matrix Computation*, manuscript, University of California at Berkeley, Berkeley, CA.

[21] D. J. Rose and R. E. Tarjan, *Algorithmic aspects of vertex elimination on directed graphs*, SIAM J. Appl. Math., 34 (1978), pp. 176–197.

[22] A. van Heukelum, *Symbolic Sparse Cholesky Factorization Using Elimination Trees*, Master's thesis, Department of Mathematics, Utrecht University, Utrecht, The Netherlands, 1999.

[23] E. Zmijewski and J. R. Gilbert, *A parallel algorithm for sparse symbolic Cholesky factorization on a multiprocessor*, Parallel Comput., 7 (1988), pp. 199–210.

SIAM J. SCI. COMPUT.
Vol. 29, No. 3, pp. 1315–1337
© 2007 Society for Industrial and Applied Mathematics

COMPATIBLE COARSE NODAL AND EDGE ELEMENTS THROUGH ENERGY FUNCTIONALS*

F. MUSY†, L. NICOLAS‡, AND R. PERRUSSEL†‡

Abstract. We propose new algorithms for the setup phase of algebraic multigrid (AMG) solvers for linear systems coming from edge element discretization. The construction of coarse levels is performed by solving an optimization problem with a Lagrange multiplier method: we minimize the energy of coarse bases under a constraint linking coarse nodal and edge element bases. On structured meshes, the resulting AMG method and the geometric multigrid method behave similarly as preconditioners. On unstructured meshes, the convergence rate of our method compares favorably with the AMG method of Reitzinger and Schöberl.

Key words. multigrid, constrained optimization, edge finite elements

AMS subject classifications. 65F10, 65N55, 65N30

DOI. 10.1137/050645750

1. Introduction.

Edge element discretization plays a key role in computational electromagnetism; it can be implemented for evaluating the electric or magnetic field using the vector wave equation or for evaluating the magnetic potential using the eddy current formulation. As this finite element discretization leads to sparse but generally large linear systems, efficient methods are required for the solution. Multilevel techniques have been introduced by Hiptmair [5] and Arnold, Falk, and Winther [1]; they are shown to be optimal on a hierarchy of nested grids. However, the inherent need of hierarchical finite element meshes in geometric multigrid methods is very restricting for industrial applications. It is more convenient to make use of a grey-box multilevel algorithm which needs a single grid but takes into account additional information on the initial problem.

For this purpose, algebraic multilevel methods have already been developed. Beyond the use of specific smoothers as proposed in [1, 5], Reitzinger and Schöberl [14] have highlighted an essential geometric compatibility relation which means that the gradient of a coarse nodal element function must belong to the coarse edge element space. Based on this relation, Reitzinger and Schöberl achieved an efficient method, which is now widespread in the computational electromagnetism community [7, 9, 10, 18, 17]. However, the combinatorial approach used there does not lead to an optimal convergence rate. Some improvements have been proposed by Bochev and coworkers [2, 6] based on smoothed aggregation techniques [15] and a compatibility with a larger class of nodal prolongation operators.

Here, we propose a somewhat different approach. Following ideas from [16] and [19], we construct coupled nodal and edge coarse bases by energy minimization, and we enforce the compatibility relation as a constraint. A connection can be made with

*Received by the editors November 21, 2005; accepted for publication (in revised form) December 28, 2006; published electronically June 1, 2007.
http://www.siam.org/journals/sisc/29-3/64575.html

†CNRS, UMR5208, Institut Camille Jordan, Villeurbanne, F-69622, France, and Ecole Centrale de Lyon, Ecully, F-69134, France; Université de Lyon, Lyon, F-69003, France (francois.musy@ec-lyon.fr, ronan.perrussel@inria.fr).

‡CNRS, UMR5005, Laboratoire Ampère, Villeurbanne, F-69622, France, and Ecole Centrale de Lyon, Ecully, F-69134, France; Université de Lyon, Lyon, F-69003, France (laurent.nicolas@ec-lyon.fr, ronan.perrussel@ec-lyon.fr).

the smoothed aggregation method [8], which also involves energy minimization. In order to simplify the notation, we present our construction in a two-level framework.

In section 2, the problem and the properties of the considered finite element space are reviewed, especially the essential geometric relation. The constrained minimization problem is presented in section 3. Then we introduce the resulting linear system, with Lagrange multipliers and the edge prolongation matrix as unknowns. Section 4 describes in detail the conditions for obtaining a well-posed minimization problem. First, we clarify the construction of the edge function supports from a given graph and the coarse nodal function supports, and we point out the satisfied constraints. Then, we prove that the coarse graph has to satisfy a connectivity condition if the implicit constraints are not enforced, and we propose a construction of the coarse nodal function supports together with a suitable definition of the coarse graph. Finally, in section 5, we evaluate the efficacy of various versions of the algorithm corresponding to different choices of energy norm. We give numerical results for 2D and 3D problems on structured and unstructured meshes. On a hierarchy of nested meshes, the edge bases being computed from the geometric nodal bases at each level, the resulting AMG method, and the geometric multigrid method behave similarly as preconditioners. On unstructured meshes, the convergence rate of our method is better than the method of Reitzinger and Schöberl [14]. However, because of the solution of the constrained minimization problem, we pay for this improvement with a large increase in the computing cost for the edge bases.

2. Definition of the continuous problem and its discretization.

2.1. Formulation. The following problem has to be solved on a domain $\Omega \subset \mathbb{R}^d$, $d = 2, 3$:

$$\begin{cases} \text{Find } \mathbf{E} \in V \text{ such that } a(\mathbf{E}, \mathbf{E}') = F(\mathbf{E}') \ \forall \mathbf{E}' \in V_0, \\ \text{with } a(\mathbf{E}, \mathbf{E}') = \displaystyle\int_\Omega \delta \operatorname{curl} \mathbf{E} \cdot \operatorname{curl} \mathbf{E}' + \int_\Omega \gamma \mathbf{E} \cdot \mathbf{E}'. \end{cases} \tag{2.1}$$

V is an affine subspace of $\mathbf{H}(\operatorname{curl}, \Omega)$ [11], taking into account essential boundary conditions; V_0 is the vector subspace parallel to V; F is a linear form on V_0, which describes the source term; δ and γ are strictly positive functions, so that a is coercive; F, δ, and γ depend on the applications under consideration.

This formulation includes many static and transient electromagnetic models: potential vector formulation for magnetostatic or eddy currents and electric or magnetic field formulation in the transient case. For instance, for an eddy current formulation in the transient case discretized by an implicit Euler scheme with time parameter $\triangle t$, $\mathbf{E}$ is the electric field, δ is equal to $\triangle t/\mu$ with μ the magnetic permeability, and γ is the electric conductivity.

2.2. Finite element space and properties. Problem (2.1) is discretized by using the lowest order edge elements introduced by Nédélec [13], which are conforming in $\mathbf{H}(\operatorname{curl}, \Omega)$. We consider triangular and tetrahedral meshes. For a tetrahedron K, the local polynomial space V_K is defined by

$$V_K = \{x \mapsto \mathbf{p} \times \mathbf{x} + \mathbf{q}, \ \mathbf{x} \in K, \text{ and } \mathbf{p}, \mathbf{q} \in \mathbb{R}^3\}. \tag{2.2}$$

The symbol $\times$ denotes the cross-product.

The local degrees of freedom which permit us to ensure the conformity in $\mathbf{H}(\mathrm{curl}, \Omega)$ are given by path integrals along the edges of the element:

$$\mathbf{E}_h \mapsto \int_e \mathbf{E}_h \cdot \mathbf{t} \, \mathrm{ds}, \tag{2.3}$$

where e is an edge of K, $\mathbf{t}$ is a tangential vector to e, and $\mathbf{E}_h$ belongs to V_K. Based on this local description, an edge finite element space can be defined on a mesh of the domain. V_h will denote the edge finite element space, defined on a mesh T_h of the domain Ω, and taking into account essential boundary conditions.

Therefore using this finite element space, we are led to solve a linear system:

$$Ax = f. \tag{2.4}$$

A fundamental property links the edge elements to the nodal P_1-Lagrange elements: the gradient of a nodal function belongs to the space of edge functions. If we denote the finite element bases as follows,

(i) $(\phi_p^h)_{p=1,\ldots,N^h}$ is the nodal basis,

(ii) $(\lambda_i^h)_{i=1,\ldots,E^h}$ is the edge basis,

this property can be recast as

$$\mathrm{grad}\, \phi_p^h = \sum_{i=1}^{E^h} G_{ip}^h \lambda_i^h \qquad \forall p \in \{1, \ldots, N^h\}, \tag{2.5}$$

where G^h is known and is the edge-node incidence matrix of the mesh T_h; in what follows, the superscript h for the fine level will be opposed to H used for the coarse level. G^h is a discrete analogue of the gradient operator on this mesh. Moreover, if the domain is contractible, the subspace spanned by the $\mathrm{grad}\, \phi_p^h$'s coincides exactly with the kernel of the curl operator in the edge element space [4].

Our aim, as highlighted in [14], is to preserve the analogous compatibility relation for coarse bases, which is natural with a hierarchy of nested grids. Thus, a coherent representation of the kernel of the curl operator is preserved, which permits the efficient use of the smoothers proposed by Hiptmair [5] and by Arnold, Falk, and Winther [1]. Therefore the gradient of a coarse nodal function must be a combination of coarse edge functions.

Let us introduce the following notation:

(i) $(\phi_n^H)_{n=1,\ldots,N^H}$ is the coarse nodal basis,

(ii) $(\lambda_e^H)_{e=1,\ldots,E^H}$ is the coarse edge basis.

Then the compatibility condition is written algebraically as

$$\mathrm{grad}\, \phi_n^H = \sum_{e=1}^{E^H} G_{en}^H \lambda_e^H \qquad \forall n \in \{1, \ldots, N^H\}, \tag{2.6}$$

where G^H is the discrete analogue of the gradient operator and is the edge-node incidence matrix of a coarse graph. This operator G^H or equivalently the associated coarse graph has to be constructed before the computation of the coarse edge basis. The way to obtain the coarse graph from the fine graph is postponed to section 4.

These coarse bases are constructed so as to satisfy the inclusion of finite element spaces, the "coarse" being included in the "fine," which is expressed by the following

algebraic relations:

$$\phi_n^H = \sum_{p=1}^{N^h} \alpha_{pn} \phi_p^h \qquad \forall n \in \{1, \dots, N^H\}, \tag{2.7a}$$

$$\lambda_e^H = \sum_{i=1}^{E^h} \beta_{ie} \lambda_i^h \qquad \forall e \in \{1, \dots, E^H\}. \tag{2.7b}$$

According to relations (2.5), (2.6), and (2.7), the components α_{pn} and β_{ie} of the coarse bases must satisfy

$$\sum_{e=1}^{E^H} \sum_{i=1}^{E^h} \beta_{ie} G_{en}^H \lambda_i^h = \sum_{p=1}^{N^h} \sum_{i=1}^{E^h} G_{ip}^h \alpha_{pn} \lambda_i^h \qquad \forall n \in \{1, \dots, N^H\}, \tag{2.8a}$$

or equivalently

$$\sum_{e=1}^{E^H} \beta_{ie} G_{en}^H = \sum_{p=1}^{N^h} G_{ip}^h \alpha_{pn} \qquad \forall i \in \{1, \dots, E^h\},\ \forall n \in \{1, \dots, N^H\}. \tag{2.8b}$$

Finally, if α denotes the matrix of components α_{in} and β the matrix of components β_{ie}, the compatibility relation (2.6) can be written, also under assumptions (2.5) and (2.7), as

$$\beta G^H = G^h \alpha. \tag{2.9}$$

3. Overview of the coarse bases construction.

3.1. Energy minimization problems. The domain Ω is decomposed into overlapping subdomains Ω_n^H, for n in $\{1, \dots, N^H\}$; this decomposition enables us to restrict the support of the coarse nodal basis functions.

In the same way, Ω is decomposed into overlapping subdomains $\mathcal{U}_e$, for e in $\{1, \dots, E^H\}$ in order to localize the supports of the coarse edge basis functions. The subdomain $\mathcal{U}_e$ will be the intersection of two subdomains Ω_l^H and Ω_m^H.

Given the subdomains Ω_n^H, the choice of the matrix G^H determines the definition of the subdomains $\mathcal{U}_e$; we will clarify this point in subsection 4.1.

Two minimization problems under constraints are solved successively. In analogy with the nodal element case [16], we first solve

$$\begin{cases} \text{Find } (\phi_n^H)_{n=1,\dots,N^H} \text{ minimizing } \displaystyle\sum_{n=1}^{N^H} c(\phi_n^H, \phi_n^H) \text{ under the constraints:} \\ \displaystyle\sum_{n=1}^{N^H} \phi_n^H(x) = 1 \quad \forall x \in \overline{\Omega} \quad \text{and} \quad \operatorname{supp}(\phi_n^H) \subset \overline{\Omega_n^H} \quad \forall n \in \{1, \dots, N^H\}. \end{cases} \tag{3.1}$$

The bilinear form c can be, for instance, $c(\phi, \psi) = \int_\Omega \gamma \operatorname{grad} \phi \cdot \operatorname{grad} \psi$ or more simply the bilinear form associated with the Laplacian of the graph, i.e., $(G^h)^t G^h$. Here the superscript t denotes transposition.

Next, from the coarse nodal basis $(\phi_n^H)_{n=1,\dots,N^H}$ we compute the coarse edge basis $(\lambda_e^H)_{e=1,\dots,E^H}$ by solving the problem

$$\text{(3.2)}\qquad \begin{cases} \text{Find } (\lambda_e^H)_{e=1,\dots,E^H} \text{ minimizing } \displaystyle\sum_{e=1}^{E^H} b(\lambda_e^H, \lambda_e^H) \text{ under the constraints:} \\ \operatorname{grad} \phi_n^H = \displaystyle\sum_{e=1}^{E^H} \lambda_e^H G_{en}^H \quad \forall n \in \{1,\dots,N^H\}, \\ \operatorname{supp}(\lambda_e^H) \subset \overline{U_e} \quad \forall e \in \{1,\dots,E^H\}. \end{cases}$$

The letter b denotes a scalar product on V_h, which can be the bilinear form $a(\cdot,\cdot)$ from (2.1) or variants, which will be introduced in subsection 5.2 and implemented in numerical examples in section 5.

Problem (3.1) can be solved by using the method described in [16]. This is the reason why we mainly concentrate our attention on problem (3.2).

We introduce algebraic notation which encodes the support constraints. First for all e in $\{1,\dots,E^H\}$, I_e is a subset of the set of indices of the fine edge basis functions whose support is included in $\overline{U_e}$. We assume that, for all e, I_e is nonempty, and we define

$$\text{(3.3)}\qquad \widetilde{M} = \sum_{e=1}^{E^h} |I_e|.$$

The term $|Z|$ denotes the number of elements in a finite set Z. Then for e in $\{1,\dots,E^H\}$ with $I_e = \{i_1,\dots,i_{|I_e|}\}$, we introduce the projection operator which keeps only the components indexed by I_e:

$$\text{(3.4)}\qquad \begin{aligned} &Q_e : \mathbb{R}^{E^h} \to \mathbb{R}^{|I_e|}, \\ &(Q_e x)_k = x_{i_k} \quad \forall x \in \mathbb{R}^{E^h},\ \forall k \in \{1,\dots,|I_e|\}. \end{aligned}$$

The term x_{i_k} denotes the i_kth component of x. It is straightforward that the transposed operation Q_e^t is defined by

$$\text{(3.5)}\qquad \begin{aligned} &Q_e^t : \mathbb{R}^{|I_e|} \to \mathbb{R}^{E^h}, \\ &(Q_e^t y)_i = \begin{cases} 0 \text{ if } i \notin I_e, \\ y_k \text{ if } i = i_k \in I_e, \end{cases} \qquad \forall y \in \mathbb{R}^{|I_e|}. \end{aligned}$$

Secondly, in order to restrict the number of constraints in problem (3.2), we introduce sets J_n for all n in $\{1,\dots,N^H\}$. The sets J_n, like the sets I_e, are subsets of indices of the fine edge basis functions.

The sets J_n will be carefully defined with the help of support conditions, so as to decrease as much as possible the number of constraints coming from (2.8b) and to obtain a well-posed problem; the construction of the J_n's is described in subsection 4.2. We introduce an integer for denoting the number of constraints :

$$\text{(3.6)}\qquad M = \sum_{n=1}^{N^H} |J_n|.$$

Since some J_n's can be empty, we introduce the set

$$F = \{n \in \{1, \dots, N^H\} \mid J_n \neq \emptyset\}. \tag{3.7}$$

Similarly to the operator Q_e from set I_e, we introduce a canonical projection operator associated with J_n for n in F:

$$R_n : \mathbb{R}^{E^h} \to \mathbb{R}^{|J_n|}. \tag{3.8}$$

Let K be the matrix whose coefficients are $b(\lambda_j^h, \lambda_i^h)$. In order to obtain a matrix form of the minimization problem, we define matrices which are restrictions to the subdomains under consideration of the matrix K:

$$K_e = Q_e K Q_e^t. \tag{3.9}$$

We also define

$$\xi_n = R_n G^h \alpha_{\bullet n} \quad \forall n \in F \quad \text{and} \quad \beta_e = Q_e \beta_{\bullet e} \quad \forall e \in \{1, \dots, E^H\}, \tag{3.10}$$

where $\alpha_{\bullet n}$ denotes the nth column of α, and $\beta_{\bullet e}$ the eth column of β.

After solving problem (3.1), which gives the matrix α, we compute the β_e's by solving the problem

$$\begin{cases} \text{Minimize } \displaystyle\sum_{e=1}^{E^H} \beta_e^t K_e \beta_e \text{ under the constraints:} \\ R_n \left(\displaystyle\sum_{e=1}^{E^H} G_{en}^H Q_e^t \beta_e \right) = \xi_n \qquad \forall n \in F. \end{cases} \tag{3.11}$$

3.2. Solution of problem (3.11). The constrained problem (3.11) can be solved by a Lagrange multiplier method. Let us introduce column vectors ξ and ρ of $\mathbb{R}^M$ and matrices D and T.

The components of the vector ρ are Lagrange multipliers relative to the constraints in (3.11), and ξ is the vector defined by (3.10). If we suppose that $F = \{i_1, \dots, i_{|F|}\}$, ξ and ρ are defined blockwise as

$$\xi = \begin{pmatrix} \xi_{i_1} \\ \vdots \\ \xi_{i_{|F|}} \end{pmatrix}, \quad \rho = \begin{pmatrix} \rho_{i_1} \\ \vdots \\ \rho_{i_{|F|}} \end{pmatrix} \quad \text{with } \xi_{i_n}, \rho_{i_n} \in \mathbb{R}^{|J_n|}. \tag{3.12}$$

The matrix D of dimension $(\widetilde{M}, \widetilde{M})$ is the block-diagonal matrix whose diagonal blocks are the matrices K_e, $e \in \{1, \dots, E^H\}$. Finally, the matrix T maps a vector from $\mathbb{R}^M$ to $\mathbb{R}^{\widetilde{M}}$ in the following way:

$$T : \rho \mapsto \begin{pmatrix} Q_1 \left(\displaystyle\sum_{n \in F} G_{1n}^H R_n^t \rho_n \right) \\ \vdots \\ Q_{E^H} \left(\displaystyle\sum_{n \in F} G_{E^H n}^H R_n^t \rho_n \right) \end{pmatrix}. \tag{3.13}$$

The minimization problem (3.11) can now be written

$$\text{Find } \bar{\beta}_c \in \mathbb{R}^{\widetilde{M}} \text{ minimizing } \bar{\beta}^t D \bar{\beta} \text{ in } \mathbb{R}^{\widetilde{M}} \text{ under the constraint } T^t \bar{\beta} = \xi, \tag{3.14}$$

or by introducing Lagrange multipliers:

$$\begin{cases} \text{Find a critical point } (\bar{\beta}_c, \rho_c) \in \mathbb{R}^{\widetilde{M}} \times \mathbb{R}^M \text{ of the Lagrangian } \mathcal{L} \text{ defined by} \\ \mathcal{L}(\bar{q}\beta, \rho) = \dfrac{1}{2} \bar{\beta}^t D \bar{\beta} + \rho^t (\xi - T^t \bar{\beta}). \end{cases} \tag{3.15}$$

This critical point must satisfy the system of equations

$$\begin{pmatrix} D & -T \\ T^t & 0 \end{pmatrix} \begin{pmatrix} \bar{\beta} \\ \rho \end{pmatrix} = \begin{pmatrix} 0 \\ \xi \end{pmatrix}. \tag{3.16}$$

This linear system can be solved in the following way:

— First, the vector ρ_c of Lagrange multipliers is determined by applying an iterative method to the system

$$T^t D^{-1} T \rho = \xi. \tag{3.17}$$

As we will see in subsection 4.2, the matrix $T^t D^{-1} T$ in (3.17) is symmetric positive definite; the construction of the R_n's will precisely ensure this property.

— Second, we return to the computation of $\bar{\beta}_c$ by solving

$$D \bar{\beta} = T \rho_c. \tag{3.18}$$

The matrix $T^t D^{-1} T$ will not be assembled, and its multiplication by a vector can be made efficiently; details will be given in subsection 5.1.

4. Elements required by the construction.

4.1. Algebraic decomposition into subdomains. Associated with the fine nodal basis $(\phi_p^h)_{p=1,\dots,N^h}$, we give an oriented simple connected graph by its set of vertices indexed by $\{1, \dots, N^h\}$ and its set of edges $\mathcal{S}^h \subset \{1, \dots, N^h\}^2$. The orientations of the edges are arbitrary; there are no loops, i.e., no elements (p, p) in $\mathcal{S}^h$, and if (p, q) belongs to $\mathcal{S}^h$, (q, p) does not belong to $\mathcal{S}^h$. By convention, p is the origin and q is the end of the edge (p, q). The edges could be the geometric edges of the mesh, but we do not restrict ourselves to this situation: starting from the second finest grid, we will have no more geometric edges.

The edges are numbered by $i \in \{1, \dots, E^h\}$, where $E^h = |\mathcal{S}^h|$; the ith edge is denoted by $(p(i), q(i))$ and conversely if $p = p(i)$ and $q = q(i)$, we will write $i = \overline{pq}^h$.

Let us now give the precise definition of the edge-node incidence matrix G^h of size $E^h \times N^h$:

$$G_{il}^h = \begin{cases} -1 & \text{if } l = p(i), \\ +1 & \text{if } l = q(i), \\ 0 & \text{otherwise.} \end{cases} \tag{4.1}$$

Two assumptions on the fine bases are supposed to be satisfied at the beginning. We suppose that the fine nodal basis $(\phi_p^h)_{p=1,\dots,N^h}$ satisfies the property

$$\sum_{p=1}^{N^h} \phi_p^h(x) = 1 \qquad \forall x \in \overline{\Omega}. \tag{4.2}$$

We assume that there exists a set of subdomains $(\Omega_p^h)_{p=1,\dots,N^h}$ of Ω such that

$$\Omega = \bigcup_{p=1}^{N^h} \Omega_p^h \quad \text{and} \quad \left(\bigcup_{q \neq p} \Omega_q^h \right)^c \neq \emptyset \qquad \forall p \in \{1, \dots, N^h\}, \tag{4.3}$$

where the superscript c is the standard set-complement, and satisfying

$$\begin{aligned} &\operatorname{supp}(\phi_p^h) \subset \overline{\Omega_p^h} \qquad \forall p \in \{1, \dots, N^h\}, \\ \text{and } &\operatorname{supp}(\lambda_i^h) \subset \overline{\Omega_p^h \cap \Omega_q^h} \qquad \text{if } i = \overline{pq}^h. \end{aligned} \tag{4.4}$$

Assumptions (4.3) and (4.4) on the subdomains $(\Omega_p^h)_{p=1,\dots,N^h}$ enable us to control the supports of the nodal (edge) functions and to avoid the linear dependence of these functions.

4.1.1. Notations and principle of the construction for the coarse nodal basis. Several steps have to be considered:

1. In order to localize the supports of the coarse nodal functions $(\phi_n^H)_{n=1,\dots,N^H}$, we introduce sets $(L_n)_{n=1,\dots,N^H}$ of indices in $\{1, \dots, N^h\}$ such that

$$\bigcup_{n=1}^{N^H} L_n = \{1, \dots, N^h\}, \tag{4.5}$$

and we will write

$$\Omega_n^H = \bigcup_{p \in L_n} \Omega_p^h. \tag{4.6}$$

Then it is equivalent to state $\operatorname{supp}(\phi_n^H) \subset \overline{\Omega_n^H}$ and to require the unknowns (α_{pn}) in (2.7a) to satisfy

$$\alpha_{pn} = 0 \qquad \forall n \in \{1, \dots, N^H\},\ \forall p \in \{1, \dots, N^h\} \setminus L_n. \tag{4.7}$$

We introduce a reciprocal set-valued function $\widetilde{L}$ defined by

$$\widetilde{L}_p = \left\{ n \in \{1, \dots, N^H\} \mid p \in L_n \right\} \qquad \forall p \in \{1, \dots, N^h\}. \tag{4.8}$$

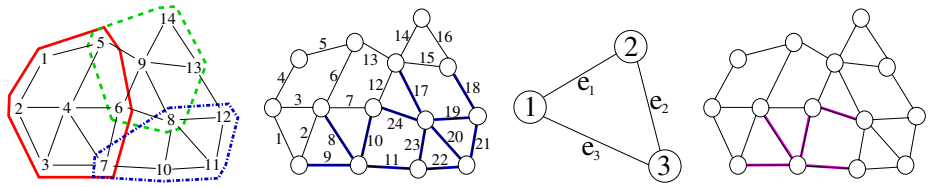
Because of (4.5), $\widetilde{L}_p$ is nonempty. Observe that (4.7) can be rewritten as

$$\alpha_{pn} = 0 \qquad \forall p \in \{1, \dots, N^h\},\ \forall n \in \{1, \dots, N^H\} \setminus \widetilde{L}_p. \tag{4.9}$$

An illustration of these definitions is given in Figure 4.1. For the fine graph in Figure 4.1(a), we set $L_1 = \{1, 2, 3, 4, 5, 6, 7\}$, $L_2 = \{5, 6, 8, 9, 13, 14\}$, and $L_3 = \{7, 8, 10, 11, 12\}$. One obtains, for instance, the sets $\widetilde{L}_7 = \{1, 3\}$ and $\widetilde{L}_4 = \{1\}$.

2. The constant preservation constraint of (3.1) is obtained by enforcing

$$\sum_{n=1}^{N^H} \alpha_{pn} = 1 \qquad \forall p \in \{1, \dots, N^h\}. \tag{4.10}$$



(a) Fine graph with node indices. Sets $(L_n)_{n=1,\dots,3}$ are surrounded.

(b) Fine graph with edge indices. Set C_3 is represented by bold edges.

(c) Coarse graph.

(d) Set I_{e_3} is represented by bold edges.

FIG. 4.1. *Representation of the fine and coarse graphs, sets* $(L_n)_{n=1,\dots,3}$, C_3, *and* I_{e_3}.

Indeed, we substitute (2.7a) into the constant preservation constraint and we use (4.2):

$$\sum_{n=1}^{N^H} \phi_n^H(x) = \sum_{n=1}^{N^H} \left(\sum_{p=1}^{N^h} \alpha_{pn} \phi_p^h(x) \right)$$
$$= \sum_{p=1}^{N^h} \phi_p^h(x) \left(\sum_{n=1}^{N^H} \alpha_{pn} \right) = \sum_{p=1}^{N^h} \phi_p^h(x) = 1.$$

3. We compute the prolongation matrix α with the encoded support constraint (4.7) by minimizing the energy functional from (3.1) under the constant preservation constraint (4.10). A method is given in [16].

4.1.2. More notation and definitions: How to relate coarse and fine, edge and nodal elements. For all n in $\{1,\dots,N^H\}$, we let C_n be a set of fine edge indices; these fine edges start or end in a node whose index is in L_n:

$$C_n = \left\{ i \in \{1,\dots,E^h\} \mid i = \overline{pq}^h \text{ with } p \text{ or } q \in L_n \right\}. \tag{4.11}$$

By (4.4) and (4.6), for i in C_n, the support of the fine edge basis function λ_i^h is included in the nth coarse nodal domain:

$$i \in C_n \implies \operatorname{supp}(\lambda_i^h) \in \overline{\Omega_n^H}. \tag{4.12}$$

We say that the fine edge function λ_i^h contributes to the gradient of the coarse nodal function ϕ_n^H if i belongs to C_n. Indeed, from (2.5) and (2.7a), we get

$$\operatorname{grad} \phi_n^H = \sum_{p=1}^{N^h} \alpha_{pn} \left(\sum_{i=1}^{E^h} G_{ip}^h \lambda_i^h \right) = \sum_{i=1}^{E^h} (G^h \alpha_{\bullet n})_i \lambda_i^h,$$

and from (4.1), (4.7), and (4.11),

$$(G^h \alpha_{\bullet n})_i = 0 \qquad \forall n \in \{1,\dots,N^H\},\ \forall i \in \{1,\dots,E^h\} \setminus C_n. \tag{4.13}$$

We introduce the reciprocal set-valued function $\widetilde{C}$:

$$\widetilde{C}_i = \left\{ n \in \{1,\dots,N^H\} \mid i \in C_n \right\} \qquad \forall i \in \{1,\dots,E_h\}. \tag{4.14}$$

In Figure 4.1(b), the fine edges are numbered, set C_3 is highlighted, and we can draw out, for instance, the set $\widetilde{C}_8 = \{1, 3\}$.

According to (4.8), (4.11), and (4.14) we infer

$$\widetilde{C}_i = \widetilde{L}_p \cup \widetilde{L}_q \quad \text{if } i = \overline{pq}^h. \tag{4.15}$$

As a consequence, $\widetilde{C}_i$ cannot be empty, and the constant preservation constraint (4.10) with (4.9) becomes

$$\sum_{n \in \widetilde{C}_i} \alpha_{pn} = \sum_{n \in \widetilde{C}_i} \alpha_{qn} = 1 \quad \text{if } i = \overline{pq}^h. \tag{4.16}$$

By analogy with the beginning of subsection 4.1, we let $\mathcal{S}^H$ be the set of edges of a simple oriented connected graph whose vertices are indexed by $\{1, \dots, N^H\}$; the elements of $\mathcal{S}^H$ are indexed by $e \in \{1, \dots, E^H\}$, and the inverse of the bijection $e \mapsto (n(e), m(e))$ is $(n, m) \mapsto \overline{nm}^H$. Similarly to the matrix G^h, we introduce the coarse edge-node incidence matrix G^H of dimension $E^H \times N^H$ with $E^H = |\mathcal{S}^H|$.

The set $\mathcal{S}^H$ has to satisfy a compatibility relation with the sets C_n

$$C_n \cap C_m \neq \emptyset \qquad \forall (n, m) \in \mathcal{S}^H. \tag{4.17}$$

For e in $\{1, \dots, E^H\}$, we define the nonempty index set I_e of the coarse edge corresponding to $e = \overline{nm}^H$:

$$I_e = C_n \cap C_m \quad \text{if } e = \overline{nm}^H. \tag{4.18}$$

Then from (4.12), if i belongs to I_e, the fine edge function λ_i^h has its support included in $\overline{\mathcal{U}_e}$:

$$i \in I_e \implies \operatorname{supp}(\lambda_i^h) \subset \overline{\mathcal{U}_e} \quad \text{with } \mathcal{U}_e = \Omega_n^H \cap \Omega_m^H.$$

The coarse graph in Figure 4.1(c) is related to the fine graph in Figure 4.1(a). Set I_{e_3} is represented in Figure 4.1(d).

In order to satisfy the support constraint $\operatorname{supp}(\lambda_e^H) \subset \overline{\mathcal{U}_e}$, we impose on the unknowns (β_{ie}) in (2.7b) the constraints

$$\beta_{ie} = 0 \qquad \forall e \in \{1, \dots, E^H\},\ \forall i \in \{1, \dots, E^h\} \setminus I_e. \tag{4.19}$$

By introducing the reciprocal set-valued function $\widetilde{I}$ defined by

$$\widetilde{I}_i = \big\{ e \in \{1, \dots, E^H\} \mid i \in I_e \big\} \qquad \forall i \in \{1, \dots, E^h\}, \tag{4.20}$$

the relation (4.19) can be rewritten as

$$\beta_{ie} = 0 \qquad \forall i \in \{1, \dots, E^h\},\ \forall e \in \{1, \dots, E^H\} \setminus \widetilde{I}_i. \tag{4.21}$$

The set $\widetilde{I}_i$ might be empty for some $i \in \{1, \dots, E^h\}$; i.e., some fine edge function might not contribute to any coarse edge function, but, as the sets I_e are never empty, we get

$$\bigcup_{i=1}^{E^h} \widetilde{I}_i = \{1, \dots, E^H\}.$$

4.1.3. Graph- and set-theoretical properties of $\widetilde{I}$ and $\widetilde{C}$. We will show that it is not necessary to enforce the constraints (2.8b) for all (i,n). We need the following lemma, which is a direct consequence of (4.14), (4.18), and (4.20).

LEMMA 4.1. $\widetilde{I}_i$

$$\overline{mn}^H \in \widetilde{I}_i \Leftrightarrow (m,n) \in \mathcal{S}^H \qquad \{m,n\} \subset \widetilde{C}_i.$$

Thus the edges indexed by $\widetilde{I}_i$ are the edges of $\mathcal{S}^H$ whose extremity nodes are indexed by $\widetilde{C}_i$. From this result, we will first deduce that the relations in (2.8b) are always fulfilled in the case $n \notin \widetilde{C}_i$.

PROPOSITION 4.2. i $\{1,\dots,E^h\}$ n $\{1,\dots,N^H\} \setminus \widetilde{C}_i$ (2.8b) (i,n) . According to (4.13), the (i,n) coefficient of the right-hand side of (2.8b) vanishes.

Conversely, according to (4.21) the left-hand side of (2.8b) is given by

$$\sum_{e \in \widetilde{I}_i} \beta_{ie} G^H_{en}.$$

It vanishes if $\widetilde{I}_i$ is empty. If $e = \overline{lm}^H$ belongs to $\widetilde{I}_i$, Lemma 4.1 implies that l and m belong to $\widetilde{C}_i$. However, for G^H_{en} not to vanish, m and l must be equal to n, and this contradicts the assumption $n \notin \widetilde{C}_i$. ∎

Secondly, the following proposition points out that some relations in (2.8b) are linearly dependent.

PROPOSITION 4.3. i $\{1,\dots,E^h\}$ m $\widetilde{C}_i$ (2.8b) (i,n) n $\widetilde{C}_i \setminus \{m\}$ (i,m) .

. It is sufficient to prove the equality

$$\sum_{n \in \widetilde{C}_i} \left(\sum_{e=1}^{E^H} \beta_{ie} G^H_{en} \right) = \sum_{n \in \widetilde{C}_i} \left(\sum_{r=1}^{N^h} G^h_{ir} \alpha_{rn} \right).$$

For $i = \overline{pq}^h$, according to (4.1) and (4.16), the right-hand side vanishes since

$$\sum_{n \in \widetilde{C}_i} \left(\sum_{r=1}^{N^h} G^h_{ir} \alpha_{rn} \right) = \sum_{n \in \widetilde{C}_i} \alpha_{qn} - \sum_{n \in \widetilde{C}_i} \alpha_{pn}.$$

For the left-hand side, according to (4.21), we have

$$\sum_{n \in \widetilde{C}_i} \left(\sum_{e=1}^{E^H} \beta_{ie} G^H_{en} \right) = \sum_{e \in \widetilde{I}_i} \beta_{ie} \left(\sum_{n \in \widetilde{C}_i} G^H_{en} \right).$$

It vanishes if $\widetilde{I}_i = \emptyset$ and also if $\widetilde{I}_i \neq \emptyset$ since for all e in $\widetilde{I}_i$, by Lemma 4.1 and the definition of G^H,

$$\sum_{n \in \widetilde{C}_i} G^H_{en} = \sum_{n=1}^{N^H} G^H_{en} = 0. \qquad ∎$$

4.2. How to choose the R_n's in order to simplify the computational process and to have a unique coarse edge basis. The choice of the index set J_n and therefore the projections R_n can now be explicitly defined.

All the matrices $(K_e)_{e=1,\dots,E^H}$ are assumed to be symmetric positive definite; from the definition of the matrix D (subsection 3.2), it is clear that D^{-1} is symmetric positive definite. Therefore, in order to obtain a unique solution of (3.17), the matrix T must have full column rank, which has to be enforced by a proper choice of the R_n's. If T does not satisfy this property, the set of β satisfying (2.9) can be empty, and the existence of a solution to (3.17) is not ensured.

We define the reciprocal set-valued function $\widetilde{J}$:

$$\widetilde{J}_i = \big\{n \in \{1,\dots,N^H\} \mid i \in J_n\big\} \qquad \forall i \in \{1,\dots,E^h\}. \tag{4.22}$$

Since the sets $\widetilde{I}_i$ and $\widetilde{J}_i$ may be empty, we introduce the set $\widetilde{F}$:

$$\widetilde{F} = \big\{i \in \{1,\dots,E^h\} \mid \widetilde{I}_i \neq \emptyset \text{ and } \widetilde{J}_i \neq \emptyset\big\}. \tag{4.23}$$

Then, for i in $\widetilde{F}$, we can define $\widetilde{G}^{H,i}$ the matrix of dimensions $|\widetilde{I}_i|\times|\widetilde{J}_i|$ extracted from the matrix G^H by keeping only the rows $e \in \widetilde{I}_i$ and the columns $n \in \widetilde{J}_i$.

PROPOSITION 4.4. [illegible] T [illegible]

$$\ker(\widetilde{G}^{H,i}) = \{0\} \qquad \forall i \in \widetilde{F}, \tag{4.24a}$$

$$\forall i \in \{1,\dots,E^h\}, \qquad \widetilde{I}_i = \emptyset \Rightarrow \widetilde{J}_i = \emptyset. \tag{4.24b}$$

[illegible]. Let ρ be in $\mathbb{R}^M$. For n in F defined in (3.7), we observe that

$$R_n^t \rho_n = \sum_{i \in J_n} \rho_{n,i} u_i, \tag{4.25}$$

where u_i is the ith vector of the canonical basis of $\mathbb{R}^{E^h}$. We let T operate on ρ, and for that purpose we look at the blockwise result. Let e be in $\{1,\dots,E^H\}$, by the definition of T in (3.13):

$$(T\rho)_e = Q_e\left(\sum_{n\in F} G^H_{en}\left(\sum_{i\in J_n}\rho_{n,i}u_i\right)\right) \tag{4.26a}$$

$$= \sum_{n=1}^{N^H}\sum_{i\in J_n} G^H_{en}\rho_{n,i}Q_e(u_i). \tag{4.26b}$$

From (4.22), the components of ρ indexed by the set of pairs (n,i) with n in $\{1,\dots,N^H\}$ and i in J_n can equivalently be indexed by i in $\{1,\dots,E^h\}$ and n in $\widetilde{J}_i$. Thus $(T\rho)_e$ can be rewritten as

$$(T\rho)_e = \sum_{i=1}^{E^h}\left(\sum_{n\in\widetilde{J}_i} G^H_{en}\rho_{n,i}\right)Q_e(u_i), \tag{4.26c}$$

but $Q_e(u_i)$ is equal to 0 if i is not in I_e. Then

$$(T\rho)_e = \sum_{i\in I_e} \left(\sum_{n\in \widetilde{J}_i} G^H_{e,n}\rho_{n,i} \right) Q_e(u_i). \tag{4.26d}$$

Since the vectors $(Q_e(u_i))_{i\in I_e}$ are linearly independent in $\mathbb{R}^{|I_e|}$, we infer from relation (4.26d)

$$T\rho = 0 \Leftrightarrow \sum_{n\in \widetilde{J}_i} G^H_{e,n}\rho_{n,i} = 0 \qquad \forall e \in \{1,\dots,E^H\},\ \forall i \in I_e, \tag{4.27a}$$

Remarking that $i \in I_e$ corresponds to $e \in \widetilde{I}_i$, (4.27a) becomes

$$T\rho = 0 \Leftrightarrow \sum_{n\in \widetilde{J}_i} G^H_{e,n}\rho_{n,i} = 0 \qquad \forall i \in \widetilde{F},\ \forall e \in \widetilde{I}_i, \tag{4.27b}$$

which can be written, according to the definition of $\widetilde{G}^{H,i}$, as

$$T\rho = 0 \Leftrightarrow \widetilde{G}^{H,i}\rho^i = 0 \qquad \forall i \in \widetilde{F}, \tag{4.27c}$$

where $\rho^i \in \mathbb{R}^{|\widetilde{J}_i|}$ is of components $\rho_{n,i}$ for n in $\widetilde{J}_i$. Finally, one obtains

$$T\rho = 0 \Leftrightarrow \rho^i \in \ker(\widetilde{G}^{H,i}) \qquad \forall i \in \widetilde{F}. \tag{4.28}$$

We can now prove the sufficiency of (4.24). If we assume that condition (4.24b) is satisfied, then

$$\widetilde{F} = \left\{ i \in \{1,\dots,E^h\} \mid \widetilde{J}_i \neq \emptyset \right\},$$

and ρ, vector of $\mathbb{R}^M$, is defined by the knowledge of its blocks ρ^i for all i in $\widetilde{F}$. Condition (4.24a) associated with relation (4.28) enables us to conclude

$$T\rho = 0 \Rightarrow \rho = 0.$$

Then, conditions (4.24) are sufficient for T to be one-to-one.

Conversely, assume that

$$\exists i_0 \in \widetilde{F}, \qquad \ker(\widetilde{G}^H_{i_0}) \neq \{0\},$$

Let ρ be in $\mathbb{R}^M \setminus \{0\}$ such that

$$\rho^j = \{0\} \quad \forall j \neq i_0 \quad \text{and} \quad \rho^{i_0} \in \ker(\widetilde{G}^H_{i_0}) \setminus \{0\}.$$

By relation (4.28), ρ is in $\ker(T)$. Hence, condition (4.24a) is necessary for T to be one-to-one.

In the same way, assume that

$$\exists i_0 \in \{1,\dots,E^h\}, \qquad \widetilde{I}_{i_0} = \emptyset \quad \text{and} \quad \widetilde{J}_{i_0} \neq \emptyset,$$

Let ρ in $\mathbb{R}^M \setminus \{0\}$ be such that

$$\rho^j = \{0\} \qquad \forall j \neq i_0.$$

Since i_0 is not in $\widetilde{F}$, ρ is in $\ker(T)$ from (4.28). Condition (4.24b) is necessary for T to be one-to-one. ∎

In order to reduce the number of enforced constraints, from Propositions 4.2 and 4.3, a suitable choice of sets $\widetilde{J}_i$ is

$$\widetilde{J}_i = \widetilde{C}_i \setminus \{m\}, \text{ with } m \in \widetilde{C}_i, \ \forall i \in \{1, \dots, E^h\}.$$

For such a choice, the condition of Proposition 4.4 can be given in terms of connectivity of induced subgraphs of the graph of edges in $\mathcal{S}^H$. More precisely, for any fine edge i of $\mathcal{S}^h$, let $\mathcal{S}^{H,i}$ be the induced subgraph of the graph of edges in $\mathcal{S}^H$, whose vertices are indexed by $\widetilde{C}_i$ and edges by $\widetilde{I}_i$.

COROLLARY 4.5. [illegible] $\widetilde{J}_i = \widetilde{C}_i \setminus \{m\}$ $i = 1, \dots, E^h$ [illegible] i [illegible] $\{1, \dots, E^h\}$ [illegible] $\mathcal{S}^{H,i}$ [illegible]. Observe first that $\widetilde{J}_i$ is empty iff the subgraph $\mathcal{S}^{H,i}$ reduces to one vertex, which happens by Lemma 4.1 if $\widetilde{I}_i$ is empty.

Assume now $\widetilde{I}_i$ is nonempty. By Lemma 4.1 the edge-node incidence matrix of the subgraph is deduced from G^H by keeping only the rows $e \in \widetilde{I}_i$ and the columns $n \in \widetilde{C}_i$. By deleting one column of index $m \in \widetilde{C}_i$, the obtained matrix $\widetilde{G}^{H,i}$ satisfies the condition (4.24a) of Proposition 4.4 iff the subgraph $\mathcal{S}^{H,i}$ is connected. ∎

The following subsection is devoted to the construction of index sets L_n and an incidence matrix G^H which satisfy the connectivity condition on the induced subgraph $\mathcal{S}^{H,i}$.

4.3. Construction of the index sets L_n. We start from an $N^h \times N^h$ nodal symmetric matrix B^h. When passing from a fine level to a coarser level, we just require

$$B^h_{pp} \neq 0 \qquad \forall p \in \{1, \dots, N^h\} \tag{4.29}$$

and a compatibility condition with the set $\mathcal{S}^h$:

$$(p, q) \in \mathcal{S}^h \implies B^h_{pq} \neq 0. \tag{4.30}$$

This condition means that B^h_{pq} must not vanish on the pairs (p, q) corresponding to the edges belonging to $\mathcal{S}^h$, but it might also be different from zero on other pairs. In other words, there may be more edges (p, q) determined by the condition $B^h_{pq} \neq 0$ than elements (p, q) or (q, p) in $\mathcal{S}^h$.

As Reitzinger and Schöberl do in [14], we define a map:

$$\begin{aligned} \text{ind} : \ & \mathbb{R}^{N^h} \to \mathbb{R}^{N^H} \\ & p \mapsto \text{ind}(p). \end{aligned} \tag{4.31}$$

The inverse images of n by the mapping ind make up a partition of $\{1, \dots, N^h\}$ into sets. The set H_n is the aggregate of fine indices indexed by n:

$$H_n = \big\{p \in \{1, \dots, N^h\} \mid \text{ind}(p) = n\big\}. \tag{4.32}$$

An example of a fine graph with the partition of the nodes is given in Figure 4.2(a). The H_n's make up an arbitrary partition of $\{1, \dots, N^h\}$, but there are algorithms for choosing good partitions, for instance the aggregation algorithm proposed in [15, section 5].

For every fine index p in $\{1,\dots,N^h\}$, we define

$$\mathcal{N}_p = \{q \in \{1,\dots,N^h\} \mid B^h_{pq} \neq 0\}. \tag{4.33}$$

This is the set of all the B^h-neighbors of the nodal variable p and the variable itself. From the compatibility relation (4.30), we remark that

$$(p,q) \in \mathcal{S}^h \implies q \in \mathcal{N}_p. \tag{4.34}$$

For all n in $\{1,\dots,N^H\}$, we will write

$$L_n = \bigcup_{p \in H_n} \mathcal{N}_p. \tag{4.35}$$

Observe that the set $\widetilde{L}_p$ defined in (4.8) is related to $\mathcal{N}_p$ as follows:

$$\widetilde{L}_p = \mathrm{ind}(\mathcal{N}_p). \tag{4.36}$$

We also infer from (4.33) and the symmetry of B^h

$$p \in \mathcal{N}_q \Leftrightarrow q \in \mathcal{N}_p.$$

Some sets L_n corresponding to the partition of Figure 4.2(a) are represented in Figure 4.2(b).

REMARK 4.6. B [7] (4.29) (4.30) $B^h = (G^h)^t G^h$ $\alpha^t B^h \alpha$

4.4. Definition of a compatible coarse edge incidence matrix G^H. By analogy with Reitzinger and Schöberl in [14], we choose $\mathcal{S}^H$ such that (n,m) or (m,n) is an edge if there exists $p \in H_n$ and $q \in H_m$ defined in (4.32) such that p and q are B^h-neighbors, i.e.,

$$\begin{aligned}\mathcal{S}^H = \{(n,m) \in \{1,\dots,N^H\}^2 \mid n \neq m,\ \exists p,q \in \{1,\dots,N^h\} \\ \text{with } \mathrm{ind}(p) = n,\ \mathrm{ind}(q) = m,\ p < q \text{ and } B^h_{pq} \neq 0\}.\end{aligned} \tag{4.37}$$

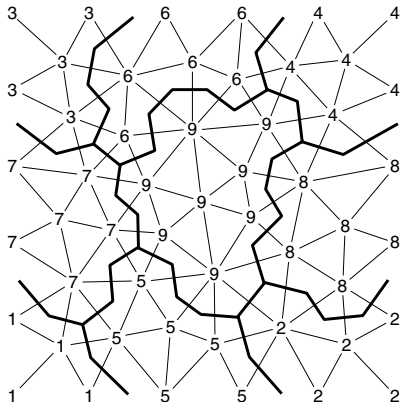
From this definition, we easily verify that the connectivity property of the fine graph $\mathcal{S}^h$ remains for the coarse graph $\mathcal{S}^H$. Observe also that $B^h_{pq} \neq 0$ implies that p is in $L_{\mathrm{ind}(p)} \cap L_{\mathrm{ind}(q)}$. Then $C_{\mathrm{ind}(p)} \cap C_{\mathrm{ind}(q)}$ is nonempty, and $\mathcal{S}^H$ satisfies condition (4.17).

The coarse graph coming from the graph partition of Figure 4.2(a) is represented in Figure 4.2(c). Some sets $\widetilde{L}_p$ are also represented in Figure 4.2(d).

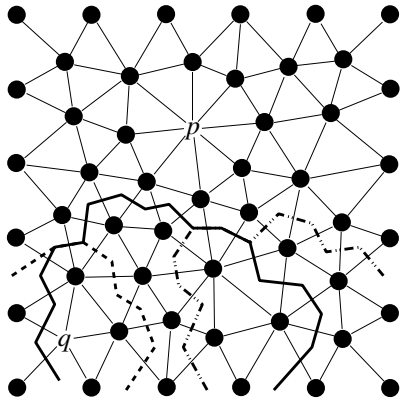
For proving the connectivity of the induced graph $\mathcal{S}^{H,i}$, we need an intermediate result.

LEMMA 4.7. $p \in \{1,\dots,N^h\}$ $\widetilde{L}_p$ $\mathrm{ind}(p)$ $\widetilde{L}_p$ $\mathcal{S}^H$. Let n be in $\widetilde{L}_p \setminus \{\mathrm{ind}(p)\}$. By (4.36), there exists p_n in $\mathcal{N}_p$ such that $n = \mathrm{ind}(p_n)$; then

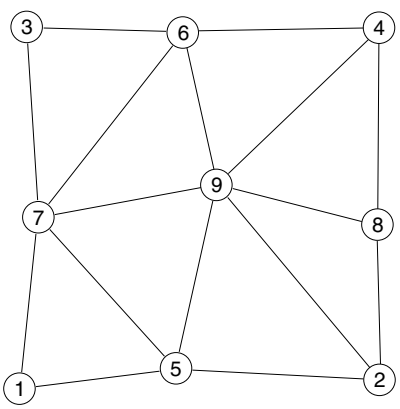
$$B^h_{pp_n} \neq 0 \quad \text{and} \quad \mathrm{ind}(p) \neq \mathrm{ind}(p_n).$$



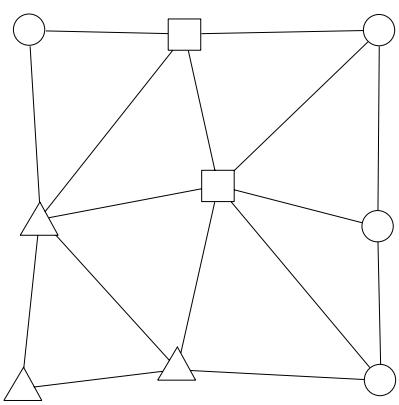
(a) Initial graph and partition.



(b) Representation of L_1, L_5, and L_2.



(c) Coarse graph for partition in (a).



(d) Representation of $\widetilde{L}_q$ (△) and $\widetilde{L}_p$ ( ). Nodes p and q are shown in (b).

FIG. 4.2. (a) *Initial graph and partition of nodes in sets* H_n*;* (b) *some sets* L_n*;* (c) *construction of the coarse graph corresponding to the partition in* (a)*, and* (d) *representation of the sets* $\widetilde{L}_q$ *and* $\widetilde{L}_p$*, referring to* (b).

From definition (4.37) of S^H, we conclude that $(n, \mathrm{ind}(p))$ or $(\mathrm{ind}(p), n)$ is in $\mathcal{S}^H$. ∎

PROPOSITION 4.8. [illegible] $\{1, \dots, E^h\}$ [illegible] $\mathcal{S}^{H,i}$ [illegible] Let $i = \overline{pq}^h$; then from (4.15), $\widetilde{C}_i = \widetilde{L}_p \cup \widetilde{L}_q$. From (4.34) and (4.36), we infer

$$\{\mathrm{ind}(p), \mathrm{ind}(q)\} \subset \widetilde{L}_p \cap \widetilde{L}_q.$$

Assume first that $\mathrm{ind}(p) \neq \mathrm{ind}(q)$. Lemma 4.7 implies that $\mathrm{ind}(p)$ and $\mathrm{ind}(q)$ are connected and, for all n in $\widetilde{C}_i \setminus \{\mathrm{ind}(r)\}$ with $r \in \{p, q\}$, n and $\mathrm{ind}(r)$ are connected.

Assume now $\mathrm{ind}(p) = \mathrm{ind}(q) = n$ with $|\widetilde{C}_i| > 1$. From Lemma 4.7, we infer

$$\forall m \in \widetilde{C}_i \setminus \{n\}, \quad n \text{ and } m \text{ are connected.} \qquad \square$$

REMARK 4.9. [illegible] [14] [illegible] $L_n = H_n$ [illegible] n [illegible] $\{1, \ldots, N^H\}$ [illegible] G^H [illegible] (4.37) [illegible] $|\widetilde{L}_p| = 1$ [illegible] p [illegible] $|\widetilde{C}_i| = 1$ [illegible] $|\widetilde{C}_i| = 2$ [illegible] $|\widetilde{I}_i| = 1$ [illegible]

REMARK 4.10. [illegible] τ^h [illegible] τ^H [illegible] L_n [illegible] τ^h [illegible] n [illegible] G^H [illegible] τ^H [illegible] 4.5 [illegible]

5. Numerical experiments.

5.1. Matrix multiplication algorithm. In order to solve the linear system (3.17), a nonpreconditioned conjugate gradient algorithm is used. Thus only the multiplication of $T^t D^{-1} T$ by a vector of $\mathbb{R}^M$ is required. We can compute $T^t D^{-1} T \rho = \widetilde{\rho}$ in three steps:

- Step 1: For $e = 1, \ldots, E^H$, compute

$$b_e = Q_e \left(\sum_{n \in F} G^H_{en} R^t_n \rho_n \right) = Q_e (R^t_m \rho_m - R^t_n \rho_n) \quad \text{for } e = \overline{nm}^H. \tag{5.1}$$

- Step 2: For $e = 1, \ldots, E^H$, solve

$$K_e x_e = b_e. \tag{5.2}$$

- Step 3: For $n = 1, \ldots, N^H$, compute

$$\widetilde{\rho}_n = R_n \left(\sum_{e=1}^{E^H} G^H_{en} Q^t_e x_e \right). \tag{5.3}$$

Step 1 requires $\widetilde{M}$ additions, and the number of arithmetical operations in Step 3 is bounded by $\max_n (\sum_{e=1}^{E^H} |G^H_{en}|) M$. The most expensive part is the solution of all the local problems: $K_e x_e = b_e$ for all e in $\{1, \ldots, E^H\}$.

Observe that such solutions of local problems are also required for the computation of the prolongation matrix $\bar{\beta}_c$ from the Lagrange multiplier vector ρ_c by using (3.18). For an exact solution of local problems, a factorization of each matrix K_e must be done.

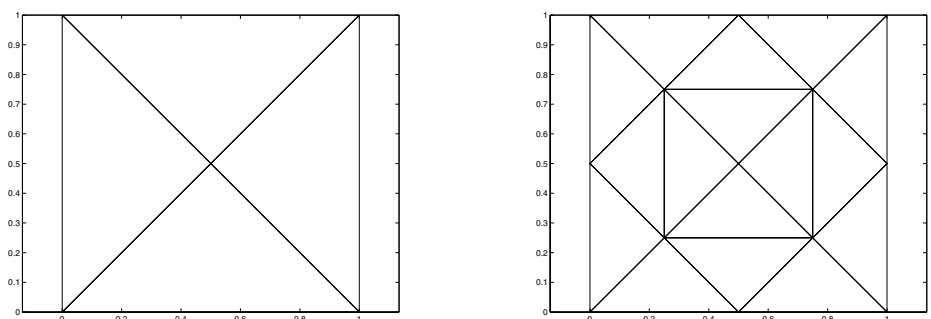
5.2. Choice of the bilinear form *b*. In the following tables, different bilinear forms b are used for the minimization. For convenience, we define a few abbreviations:

- A refers to the bilinear form a of the problem. With this choice, spatial variations of γ and δ together with mesh heterogeneity are completely taken into account for the construction of the finite element basis.
- $A + G^h M_\phi^{-1} (G^h)^t$ refers to the bilinear form defined from this matrix, M_ϕ being the mass matrix on nodal elements, with mass lumping; hence M_ϕ is diagonal. This choice enables us to improve the conditioning of local matrices K_e, as will be noted for two-dimensional (2D) simulations.

- $S + \mathrm{reg}(\eta)$ refers to the bilinear form $\int_\Omega \mathrm{curl}\, E \cdot \mathrm{curl}\, E'$ added to a local regularization depending on the parameter η; more precisely, the matrix of the local problem is $S_e + \eta \max(\mathrm{diag}(S_e))\,\mathrm{Id}$, where the matrix Id, is the identity matrix and S_e is the local matrix computed from $\int_\Omega \mathrm{curl}\, E \cdot \mathrm{curl}\, E'$. The focus is to improve the conditioning of the local matrices, the essential part of the original bilinear form a being kept.
- Id refers to the use of the identity matrix for the local problems. Then no factorization is needed, but the original problem is no longer taken into account in the definition of b.

The annotation GSsym means that one symmetric Gauss–Seidel iteration is performed in the local systems instead of an exact solution in order to significantly reduce the complexity.

5.3. Structured meshes and constant coefficients. In order to validate the proposed choices of bilinear form b, we begin the numerical experiments with structured meshes and constant coefficients in the problem. The coefficients δ and γ are set equal to 1. A sequence of nested meshes $(\tau_k^h)_{k=0,\dots,4}$ is constructed by a regular refinement starting from a simple triangular or tetrahedral mesh on the unit square and the unit cube (see Figures 5.1 and 5.2). The source term is given by nonhomogeneous boundary conditions. For each level, the edge prolongation matrix β is computed by solving system (3.16) with the right-hand-side member ξ defined from the standard nodal prolongation operator [3, Chapter 6]. The sets L_n together with the edge-node incidence matrices are defined from the meshes as described in Remark 4.10.

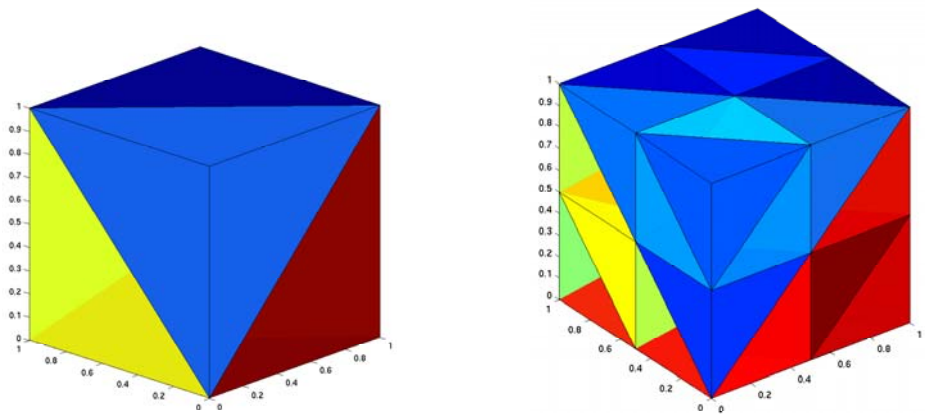


(a) (b)

FIG. 5.1. (a) *Initial mesh* τ_0^h *and* (b) *first refinement* τ_1^h.

5.3.1. Dimensions. For the finest level, the number of Lagrange multipliers M given by the vector ρ is reported in Table 5.1 for the 2D case and Table 5.2 for the 3D case. The number of unknowns E^h for each problem is also recalled. Observe that the number of multipliers is less than twice the number of unknowns in the 2D case. The ratio is slightly larger in three dimensions except for problems with few unknowns.

5.3.2. Computation of Lagrange multipliers. The multipliers are initialized to zero. The number of iterations required to divide the Euclidean norm of the residual by 10^3 during the iterative process is reported in Table 5.3 for the 2D case and Table



(a) (b)

FIG. 5.2. (a) *Initial mesh* τ_0^h *and* (b) *first refinement* τ_1^h.

TABLE 5.1
Number (Nb) of Lagrange multipliers and unknowns—2D, structured meshes.

	τ_1^h	τ_2^h	τ_3^h	τ_4^h
M = Nb of multipliers	40	152	592	2336
E^h = Nb of unknowns	20	88	368	1504

TABLE 5.2
Number of Lagrange multipliers and unknowns—3D, structured meshes.

	τ_1^h	τ_2^h	τ_3^h	τ_4^h
M	164	1060	7544	56752
E^h	26	316	3032	26416

TABLE 5.3
Number of iterations for the multiplier computation (division of the Euclidean norm of the residual by 10^3*)—2D, structured meshes.*

Norm	τ_1^h	τ_2^h	τ_3^h	τ_4^h
A	5	42+11	64+42+14	79+64+42+14
A (GSsym)	11	12+11	12+12+11	11+11+12+11
$A + G^h M_\phi^{-1} (G^h)^t$	4	9+7	9+8+7	9+8+8+7
$A + G^h M_\phi^{-1} (G^h)^t$ (GSsym)	4	9+7	9+8+7	9+8+8+7
$S + \text{reg}(0.1)$	6	19+10	19+19+9	18+19+19+10
Id	2	2+2	2+2+2	2+2+2+2

5.4 for the 3D case. These numbers are given for several choices of bilinear forms b. We also give the number of iterations for each level: finest level $+ \cdots +$ coarsest level.

In the 2D case, the behavior of the algorithm with the original matrix A is not good since the number of iterations increases with the fineness of the initial mesh. The surprising thing is that the number of iterations becomes constant when a Gauss–Seidel iteration is used instead of an exact solution. For the other cases, the behavior is relatively homogeneous, and the number of iterations is almost constant. In three

TABLE 5.4
Number of iterations for the multiplier computation (division of the Euclidean norm of the residual by 10^3)—3D, structured meshes.

Norm	τ_1^h	τ_2^h	τ_3^h	τ_4^h
A	24	32+13	46+16+15	61+20+17+17
A (GSsym)	23	25+13	32+16+15	39+20+18+17
$A + G^h M_\phi^{-1} (G^h)^t$	41	93+21	181+27+17	381+29+21+16
$A + G^h M_\phi^{-1} (G^h)^t$ (GSsym)	26	42+18	56+21+15	62+22+16+14
$S + \text{reg}(0.1)$	20	22+13	29+15+14	36+18+15+15
Id	3	3+3	3+3+3	3+3+3+3

TABLE 5.5
Number of iterations of the conjugate gradient preconditioned by a multilevel method (division of the Euclidean norm of the residual by 10^{10})—2D, structured meshes.

Norm	τ_1^h	τ_2^h	τ_3^h	τ_4^h
A	5	6	7	7
A (GSsym)	5	6	7	9
$A + G^h M_\phi^{-1} (G^h)^t$	5	6	7	8
$A + G^h M_\phi^{-1} (G^h)^t$ (GSsym)	5	6	7	8
$S + \text{reg}(0.1)$	5	6	7	7
Id	5	7	9	11
Geometric	5	6	7	7

TABLE 5.6
Number of iterations of the conjugate gradient preconditioned by a multilevel method (division of the Euclidean norm of the residual by 10^{10})—3D, structured meshes.

Norm	τ_1^h	τ_2^h	τ_3^h	τ_4^h
A	4	7	10	11
A (GSsym)	4	7	10	11
$A + G^h M_\phi^{-1} (G^h)^t$	4	7	10	11
$A + G^h M_\phi^{-1} (G^h)^t$ (GSsym)	4	7	10	11
$S + \text{reg}(0.1)$	4	7	10	11
Id	4	7	10	11
Geometric	4	7	10	11

dimensions, we notice a slow increase in the number of iterations, and the behavior for the original matrix A is not as bad as in the 2D case. The behavior for the matrix Id is the best in both cases.

5.3.3. Solution of system (2.4)**.** System (2.4) is solved by a preconditioned conjugate gradient algorithm. The preconditioner is a multilevel method which uses one pre- and one post-smoothing step by Arnold's smoother [1]; on the coarse grid, a direct solver is used. The conjugate gradient method stops when the Euclidean norm of the residual has been divided by 10^{10}. Table 5.5 gives the results for the 2D case and Table 5.6 for the 3D case. The term "geometric" corresponds to the classical geometric multigrid.

Once the multipliers are computed, and for all choices of bilinear forms b, the efficacy of the geometric and algebraic multigrid methods are comparable.

5.4. Unstructured meshes and varying coefficients. We consider a 2D problem whose structure is pictured in Figure 5.3; γ is set equal to 1 and δ takes

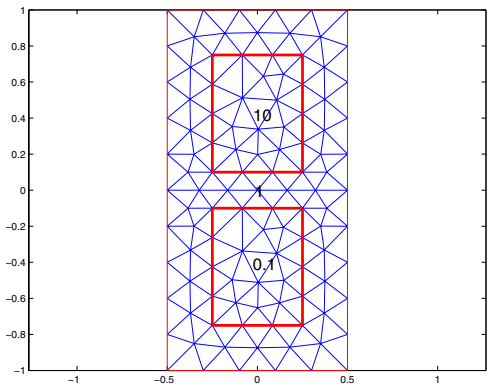


FIG. 5.3. *Geometry of the problem with the values of δ.*

TABLE 5.7
Number of unknowns on each level for several choices of h_{max}.

Method	$h_{\max} = 0.2$	0.1	0.05	0.025
new	276+31	1005+114+6	3843+405+23	15957+1467+82+3
RS	276+31	1005+114+11	3843+405+29	15957+1467+151+21

TABLE 5.8
Number of Lagrange multipliers.

$h_{\max} = 0.2$	0.1	0.05	0.025
536	2105+179	8049+759	31249+2946+118

the values displayed in this figure.

A mesh generator gives an initial unstructured mesh. We control only the maximal diameter $h_{\max}$ of elements, and the generator takes the interfaces into account.

The matrix B^h for the initial level is given by $(G^h)^t G^h$, the simplest choice underlined in Remark 4.6. The nodal prolongation matrix α is obtained from the solution of (3.1) with the bilinear form $c(\cdot,\cdot)$ associated with $(G^h)^t G^h$. The sets L_n and the matrix G^H are defined as in (4.35) and (4.37) with the map ind constructed from the aggregation algorithm given in [15].

5.4.1. Number of unknowns. In Table 5.7, the number of unknowns on each level is given for several choices of $h_{\max}$. The notation "new" denotes our algorithm, and RS denotes the algorithm of Reitzinger and Schöberl given in [14].

5.4.2. Computation of the Lagrange multipliers.

The number of multipliers for every mesh and every level is given in Table 5.8. This number is roughly twice the number of unknowns for every level.

The Euclidean norm of the residual is divided by 10^3. The number of iterations required to compute the Lagrange multipliers with different norms are gathered in Table 5.9. Compared to the results of Table 5.3, we can observe a significant increase for all methods, but the results are similarly ordered.

TABLE 5.9

Number of iterations for the multiplier computation (division of the Euclidean norm of the residual by 10^3*)—unstructured meshes. X means that the stopping criterion was not reached after* 1000 *iterations.*

Norm	$h_{\max} = 0.2$	0.1	0.05	0.025
A	387	X+134	X+407	X+X+136
A (GSsym)	415	396+X	360+X	348+450+X
$A + G^h M_\phi^{-1} (G^h)^t$	62	103+31	117+46	120+59+26
$A + G^h M_\phi^{-1} (G^h)^t$ (GSsym)	94	128+37	112+58	113+59+27
$S + \mathrm{reg}(0.1)$	74	212+32	319+65	326+200+27
Id	8	13+7	13+12	14+15+2

TABLE 5.10

Number of iterations of the conjugate gradient preconditioned by a multilevel method (division of the Euclidean norm of the residual by 10^{10}*).*

	$h_{\max} = 0.2$	0.1	0.05	0.025
A	11	12	18	45
A (GSsym)	12	12	17	25
$A + G^h M_\phi^{-1} (G^h)^t$	12	13	17	26
$A + G^h M_\phi^{-1} (G^h)^t$ (GSsym)	12	13	18	27
$S + \mathrm{reg}(0.1)$	12	12	16	25
Id	13	13	18	28
R. S.	14	20	33	56

5.4.3. Solution of system (2.4)**.** The multilevel method is used as a preconditioner; the results for the different norms are gathered in Table 5.10.

The choices Id and $A+G^h M_\phi^{-1} (G^h)^t$ (GSsym) are the most efficient because they provide us with a performing preconditioner and the lowest computational cost for the setup phase.

The prolongation matrix β built by our method provides us with a more efficient preconditioner than the method proposed by Reitzinger and Schöberl in [14]. Thus in this simple example, the number of iterations approximately grows like $h_{\max}^{-0.75}$ for the method of Reitzinger and Schöberl and like $h_{\max}^{-0.5}$ for the choice Id. This validates the theoretical interest of our algebraic method.

However, the construction of the matrix β remains really more expensive than in the Reitzinger and Schöberl algebraic multigrid method, and the solution of the initial problem with our method is finally more time-consuming. In order to improve the efficacy for the solution of problem (3.11), an algorithm that avoids the Lagrange multiplier computation can be employed [12].

Acknowledgment. We thank Michelle Schatzman for many fertile discussions.

REFERENCES

[1] D. N. ARNOLD, R. S. FALK, AND R. WINTHER, *Multigrid in* $\mathbb{H}(\mathrm{div})$ *and* $\mathbb{H}(\mathrm{curl})$, Numer. Math., 85 (2000), pp. 197–217.

[2] P. B. BOCHEV, C. J. GARASI, J. J. HU, A. C. ROBINSON, AND R. S. TUMINARO, *An improved algebraic multigrid method for solving Maxwell's equations*, SIAM J. Sci. Comput., 25 (2003), pp. 623–642.

[3] S. C. BRENNER AND L. RIDGWAY SCOTT, *The Mathematical Theory of Finite Element Methods*, 2nd ed., Appl. Math. 15, Springer-Verlag, New York, 2002.

[4] P. W. Gross and P. R. Kotiuga, *Electromagnetic Theory and Computation: A Topological Approach*, Mathematical Sciences Research Institute Publications 48, Cambridge University Press, Cambridge, UK, 2004.
[5] R. Hiptmair, *Multigrid method for Maxwell's equations*, SIAM J. Numer. Anal., 36 (1998), pp. 204–225.
[6] J. J. Hu, R. S. Tuminaro, P. B. Bochev, C. J. Garasi, and A. C. Robinson, *Toward an h-independent algebraic multigrid method for Maxwell's equations,* SIAM J. Sci. Comput., 27 (2006), pp. 1669–1688.
[7] M. Kaltenbacher and S. Reitzinger, *Algebraic multigrid methods for nodal and edge based discretizations of Maxwell's equations*, Internat. Compumag Soc. Newsletter, 9 (2002), pp. 15–23.
[8] J. Mandel, M. Brezina, and P. Vaněk, *Energy optimization of algebraic multigrid bases*, Computing, 62 (1999), pp. 205–228.
[9] T. Mifune, T. Iwashita, and M. Shimasaki, *A fast solver for fem analyses using the parallelized algebraic multigrid method*, IEEE Trans. Magnetics, 38 (2002), pp. 369–372.
[10] T. Mifune, T. Iwashita, and M. Shimasaki, *New algebraic multigrid preconditioning for iterative solvers in electromagnetic finite edge-element analyses*, IEEE Trans. Magnetics, 39 (2003), pp. 1677–1680.
[11] P. Monk, *Finite Element Methods for Maxwell's Equations*, Numerical Mathematics and Scientific Computation, Oxford University Press, New York, 2003.
[12] F. Musy, L. Nicolas, and R. Perrussel, *Gradient-prolongation commutativity and graph theory*, C. R. Acad. Sci. Paris, 341 (2005), pp. 707–712.
[13] J. C. Nédélec, *Mixed finite elements in* $\mathbb{R}^3$, Numer. Math., 35 (1980), pp. 315–341.
[14] S. Reitzinger and J. Schöberl, *An algebraic multigrid method for finite element discretizations with edge elements*, Numer. Linear Algebra Appl., 9 (2002), pp. 223–238.
[15] P. Vaněk, J. Mandel, and M. Brezina, *Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems*, Computing, 56 (1996), pp. 179–196.
[16] W. L. Wan, T. F. Chan, and B. Smith, *An energy-minimizing interpolation for robust multigrid methods*, SIAM J. Sci. Comput., 21 (2000), pp. 1632–1649.
[17] K. Watanabe and H. Igarashi, *On robustness of edge-based finite-element analysis using algebraic multigrid method*, COMPEL, 24 (2005), pp. 408–417.
[18] K. Watanabe, H. Igarashi, and T. Honma, *Comparison of geometric and algebraic multigrid methods in edge-based finite-element analysis*, IEEE Trans. Magnetics, 41 (2005), pp. 1672–1675.
[19] J. Xu and L. Zikatanov, *On an energy minimizing basis for algebraic multigrid methods*, Comput. Vis. Sci., 7 (2004), pp. 121–127.

SIAM J. SCI. COMPUT.
Vol. 29, No. 3, pp. 1338–1354
© 2007 Society for Industrial and Applied Mathematics

BIVARIATE SPLINES OF VARIOUS DEGREES FOR NUMERICAL SOLUTION OF PARTIAL DIFFERENTIAL EQUATIONS*

XIAN-LIANG HU†, DAN-FU HAN†, AND MING-JUN LAI‡

Abstract. Bivariate splines with various degrees are considered in this paper. A matrix form of the extended smoothness conditions for these splines is presented. Upon this form, the multivariate spline method for numerical solution of partial differential equations (PDEs) proposed by Awanou, Lai, and Wenston in [*The multivariate spline method for scattered data fitting and numerical solutions of partial differential equations*, in Wavelets and Splines, G. Chen and M. J. Lai, eds., Nashboro Press, Brentwood, TN, 2006, pp. 24–76] is generalized to obtain a new spline method. It is observed that, combined with prelocal refinement of triangulation and automatic degree raising over triangles of interest, the new spline method of bivariate splines of various degrees is able to solve linear PDEs very effectively and efficiently.

Key words. bivariate splines, smoothness condition, splines of various degrees, numerical solution of Poisson and biharmonic equations

AMS subject classifications. 65D07, 65D15, 35Q30, 75D05

DOI. 10.1137/060667207

1. Introduction.

Let $\Omega \subset \mathbf{R}^2$ be a polygonal domain and $\triangle$ be a given triangulation of Ω. A standard spline space $S_d^r(\triangle)$ is the space of all smooth piecewise polynomial functions of degree d and smoothness r over $\triangle$ with $r < d$. That is,

$$S_d^r(\triangle) = \{s \in C^r(\Omega),\, s|_t \in \mathbf{P}_d, \quad t \in \triangle\},$$

where $\mathbf{P}_d$ denotes the space of polynomials of degree d, t denotes a triangle, and $\Omega = \bigcup\{t,\ t \in \triangle\}$. Such a spline space has been studied for a long time in the literature and in practice. (See [6] and references therein.)

In this paper we would like to consider bivariate splines of various degrees d_t, $t \in \triangle$, where d_t is a positive integer for each t. That is, for a given triangulation $\triangle$ and a degree vector $\mathbf{d} = \{d_t,\, t \in \triangle\}$, we let

$$\mathcal{S}_{\mathbf{d}}^r(\triangle) = \{s \in C^r(\Omega),\, s|_t \in \mathbf{P}_{d_t}, \quad t \in \triangle\}.$$

Our study of such a spline space $\mathcal{S}_{\mathbf{d}}^r(\triangle)$ is motivated by our interest in numerically solving partial differential equations (PDEs) more effectively and efficiently. For a PDE with a singularity at the boundary of the domain Ω, a finite element method requires a much finer mesh at the singularity. Otherwise, one has to increase the degrees of finite elements. Instead of increasing degrees globally, we would like to increase the degrees locally and combine this with local triangulation refinement near the singularity to efficiently solve the PDE. For continuous finite elements, there are several constructions of finite elements of various degrees available in the literature, e.g., [15] and [16]. More recently, optimal meshes of p- and hp-finite element methods

*Received by the editors August 9, 2006; accepted for publication (in revised form) January 11, 2007; published electronically June 7, 2007.
http://www.siam.org/journals/sisc/29-3/66720.html
†Department of Mathematics, Zhejiang University, Hangzhou, 310027, China (huxl98@yahoo.com.cn, mhdf2@zju.edu.cn). The work of the second author was partially supported by the National Basic Research Program under grant 2005CB32170X.
‡Department of Mathematics, The University of Georgia, Athens, GA 30602 (mjlai@math.uga.edu).

for PDEs with singularity are discussed in [13], and a constructive scheme for p-finite elements is explained in [14]. However, the implementation of such p-finite elements is very complicated. Recently, in [1], Awanou, Lai, and Wenston introduced a multivariate spline method for numerical solution of PDEs which overcomes the difficulty of the implementation. The researchers in [1] streamlined the process by skipping the construction of smooth finite elements. Instead, they simply used discontinuous piecewise polynomial functions over a triangulation and treated any desired smoothness conditions across edges of triangles as side constraints. Together with the side constraints of boundary conditions, they applied a Lagrange multiplier method to solve the system associated with the PDE of interest. Another key ingredient was to use a special iterative algorithm to solve the associated linear system. (See [1] for more details.) Smooth spline spaces with various degrees were also considered in [1], where a degree reduction condition is introduced and incorporated with standard smoothness conditions acting as constraints. In other words, let $d = \max\{d_t,\, t \in \triangle\}$, $\mathcal{S}^r_{\mathbf{d}}(\triangle) \subset S^r_d(\triangle)$. As the largest degree d among d_t, $t \in \triangle$, becomes large, the size of the linear system arising from the spline space $S^r_d(\triangle)$ grows rapidly. This will not lead to an efficient algorithm. Hence in this paper we propose a different approach. To keep the given degrees over various triangles, we need a new smoothness condition for polynomial pieces over two adjacent triangles with different degrees. Then we use the new smoothness conditions in the place of the original smoothness conditions in the multivariate spline method for numerical solution of PDEs proposed by Awanou, Lai, and Wenston in [1]. For simplicity, we shall refer to the original multivariate spline method for PDEs as the ALW spline method. To solve a PDE with singularity on the boundary of the domain Ω, we do not know in advance which degrees we should use to achieve the given accuracy of the solution. We adopt a strategy of combining local refinement and local degree raising to effectively solve PDEs with singularity on the boundary.

The present paper is organized as follows: A matrix form of the new smoothness conditions for splines of various degrees is considered in section 2. In section 3, we shall explain how to use bivariate splines of various degrees for numerical solution of PDEs. Numerical examples are shown in section 4 to illustrate that our new spline method is more efficient and effective. Finally we make some remarks about our spline method in section 5.

2. A matrix form of the new smoothness condition. Let T be a fixed triangle in $\triangle$. It is known that any polynomials of degree d can be expressed as the following so-called B-form:

$$p(x,y) = \sum_{i+j+k=d} c_{i,j,k} B^d_{i,j,k}(x,y), \tag{2.1}$$

where $c_{i,j,k}$ are Bézier coefficients of p and $B^d_{i,j,k}(x,y)$ are Bernstein polynomials of degree d, which are both sorted in lexicographical order with respect to their Bézier ordinate (i,j,k). That is, the order of indices (i,j,k) is actually arranged by the following ordering function:

$$q(i,j,k) = \binom{j+k+1}{2} + k + 1. \tag{2.2}$$

For example, when $d = 3$, the linear order of Bézier coefficients is as follows:

$$c_{3,0,0}, c_{2,1,0}, c_{2,0,1}, c_{1,2,0}, c_{1,1,1}, c_{1,0,2}, c_{0,3,0}, c_{0,2,1}, c_{0,1,2}, c_{0,0,3}.$$

Note that all the Bézier coefficients are sorted by this ordering function in the remainder of this paper.

The smoothness condition of polynomial patches over two adjacent triangular domains is an important and useful tool in the study of multivariate splines. It can be found in the literature, e.g., [3] in the bivariate setting and [2] in the multivariate setting. A geometric interpretation of bivariate C^r smoothness conditions for general $r \geq 0$ is given in [4]. For convenience, we include the smoothness conditions in this paper.

LEMMA 2.1. $T := \langle v_1, v_2, v_3 \rangle$ $\tilde{T} := \langle v_4, v_2, v_3 \rangle$ $e := \langle v_2, v_3 \rangle$

$$p_d(v) := \sum_{i+j+k=d} c_{ijk} B^d_{ijk}(v) \qquad \widetilde{p}_d(v) := \sum_{i+j+k=d} \tilde{c}_{ijk} \tilde{B}^d_{ijk}(v),$$

$B^d_{ijk}(v)$ $\tilde{B}^d_{ijk}(v)$ T $\tilde{T}$ D^n_u u

$$D^n_u p_d(v) = D^n_u \widetilde{p}_d(v) \qquad \forall v \in \langle v_2, v_3 \rangle$$

$n = 0, \ldots, r$

$$\widetilde{c}_{n,j,k} = \sum_{\mu+\nu+\kappa=n} c_{\mu,j+\nu,k+\kappa} B^n_{\mu\nu\kappa}(\lambda), \qquad j+k = d-n, \qquad n = 0, \ldots, r, \tag{2.3}$$

λ v_4 T

In practice, when applying the spline space $S^r_d(\triangle)$ for numerical solution of PDEs as in [1] and [7], the smoothness conditions (2.3) are assembled into matrix form for all interior edges of $\triangle$:

$$H\mathbf{c} = 0, \tag{2.4}$$

where $\mathbf{c}$ is a vector of Bézier coefficients of a polynomial of degree d over triangle T for all $T \in \triangle$. To extend the smoothness conditions to the case of different degrees on adjacent patches, we first need the following matrix form of the de Casteljau algorithm.

LEMMA 2.2. $\lambda = (\lambda_1, \lambda_2, \lambda_3)$ (x, y) T (2.1)

$$P_d(\lambda) = \sum_{i+j+k=d-r} c^{(r)}_{i,j,k}(\lambda) B^{d-r}_{i,j,k}(\lambda) \quad \forall r = 0, 1, \ldots, d, \tag{2.5}$$

$c^{(0)}_{i,j,k} = c_{i,j,k}$

$$c^{(r+1)}_{i,j,k}(\lambda) = \lambda_1 c^{(r)}_{i+1,j,k}(\lambda) + \lambda_2 c^{(r)}_{i,j+1,k}(\lambda) + \lambda_3 c^{(r)}_{i,j,k+1}(\lambda), \tag{2.6}$$

$$\mathbf{c}^{(r+1)}(\lambda) = \mathcal{D}_{\lambda,d-r} \mathbf{c}^{(r)}(\lambda) \tag{2.7}$$

$\mathbf{c}^{(r)}(\lambda) = \{c^{(r)}_{i,j,k}(\lambda)\}_{i+j+k=d-r}$ $r = 1, \ldots, d$ q (2.2) $D_{\lambda,d}$ $d(d+1)/2 \times (d+1)(d+2)/2$

$$
\begin{array}{ccccccccccccccc}
\lambda_1 & \lambda_2 & \lambda_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \lambda_1 & 0 & \lambda_2 & \lambda_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \lambda_1 & 0 & \lambda_2 & \lambda_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \lambda_1 & 0 & 0 & \lambda_2 & \lambda_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \lambda_1 & 0 & 0 & \lambda_2 & \lambda_3 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \lambda_1 & 0 & 0 & \lambda_2 & \lambda_3 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \lambda_1 & 0 & 0 & 0 & \lambda_2 & \lambda_3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda_1 & 0 & 0 & 0 & \lambda_2 & \lambda_3 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda_1 & 0 & 0 & 0 & \lambda_2 & \lambda_3 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda_1 & 0 & 0 & 0 & \lambda_2 & \lambda_3
\end{array}
$$

FIG. 2.1. *The hierarchical structure of matrix* $\mathcal{D}_{\lambda,d}$.

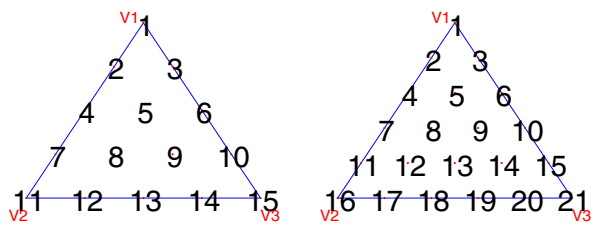


FIG. 2.2. *The hierarchical structure of Bézier ordinates.*

. The first part of this lemma is the standard de Casteljau algorithm of triangular B-form, and (2.7) is easy to verify by writing (2.6) in its matrix form. □

Let us point out an interesting property of the matrix $\mathcal{D}_{\lambda,d}$. First we present the matrix $\mathcal{D}_{\lambda,d}$ with degrees d from 1 to 4 as shown in Figure 2.1. The top-left 1×3 submatrix represents $\mathcal{D}_{\lambda,1}$, the top-left 3×6 matrix represents $\mathcal{D}_{\lambda,2}$, etc. Observe that the matrix has a nice pattern. This is because of the definition of the ordering function q; i.e., the Bézier ordinates are sorted line by line incrementally. In Figure 2.2, the ordinate ordering for Bézier ordinates of $d = 4$ indexed by 1–15 is a part of the ordinate ordering for $d = 5$ indexed by 1–21. Then it is easy to see that $\mathcal{D}_{\lambda,d}$ is a block of $\mathcal{D}_{\lambda,d+1}$ when (2.6) is written into (2.7). This property is referred to as a hierarchical property.

On the other hand, if the Bézier ordinate indices are ordered in another form as shown in Figure 2.3 for $d = 4$ and $d = 5$, we find that this new ordering does not share the hierarchical property as in Figure 2.2.

Based on the above matrix form of the de Casteljau algorithm, we can express the standard smoothness conditions in the following matrix form version.

LEMMA 2.3. $\tilde{\mathbf{c}}$ $\mathbf{c}$ $\{\tilde{c}_{ijk}\}_{i+j+k=d}$ $\{c_{ijk}\}_{i+j+k=d}$

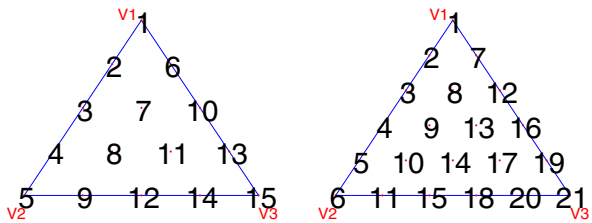


FIG. 2.3. *A new ordering of Bézier ordinates.*

$$D_u^n p_d(v) = D_u^n \widetilde{p}_d(v) \quad \forall v \quad \langle v_2, v_3 \rangle$$

$$P_{n,d}\tilde{\mathbf{c}} = P_{0,d-n}\left\{\prod_{l=1}^{n} \mathcal{D}_{\lambda,d+1-l}\right\}\mathbf{c} \qquad \forall n = 0,\dots,r, \tag{2.8}$$

$P_{n,d}$ $(d-n+1)\times(d+1)(d+2)/2$

$$P_{n,d}(i,j) = \begin{cases} 1, & j = q(n,i-1,d+1-n-i), \quad i = 1,\dots,d+1-n, \\ 0 & \end{cases}$$

$n = 0,\dots,r$

. We need to prove that (2.8) is equivalent to (2.3). By the definition of $P_{n,d}$, it has only one nonzero entry in each row. So we name it the picking matrix, whose geometric meaning is to pick out the Bézier coefficients on the nth line parallel to common edge e from all the Bézier coefficients of degree d. Then we have

$$\{\tilde{c}_{n,j,k}\}_{j+k=d-n} = P_{n,d}\tilde{\mathbf{c}} \qquad \forall n = 0,\dots,r. \tag{2.9}$$

By (2.5), for all $n = 0,\dots,r$, it is true that

$$\sum_{\mu+\nu+\kappa=n} c_{\mu,j+\nu,k+\kappa} B^n_{\mu\nu\kappa}(\lambda) = \sum_{\mu+\nu+\kappa=n} c^{(0)}_{\mu,j+\nu,k+\kappa} B^n_{\mu\nu\kappa}(\lambda) = c^{(n)}_{0,j,k}(\lambda), \quad j+k = d-n.$$

It can also be written in vector form. That is, applying (2.7) for n times, we get

$$\begin{aligned} \{c^{(n)}_{0,j,k}(\lambda)\}_{j+k=d-n} &= P_{0,d-n}\mathbf{c^{(n)}} \\ &= P_{0,d-n}\mathcal{D}_{\lambda,d+1-n}\mathbf{c^{(n-1)}} \\ &= \cdots \\ &= P_{0,d-n}\left\{\prod_{l=1}^{n} \mathcal{D}_{\lambda,d+1-l}\right\}\mathbf{c}. \end{aligned} \tag{2.10}$$

Combining (2.3), (2.9), and (2.10), we have (2.8). □

Next we express the well-known degree raising algorithm for the B-form in matrix form as in Lemma 2.2.

LEMMA 2.4. [illegible] p [illegible] d [illegible] T [illegible] (2.1) [illegible] $c_{ijk}^{[d]}$ c_{ijk} [illegible]

$$p = \sum_{i+j+k=d+1} c_{ijk}^{[d+1]} B_{ijk}^{d+1},$$

[illegible]

$$c_{ijk}^{[d+1]} = (ic_{i-1,j,k}^{[d]} + jc_{i,j-1,k}^{[d]} + kc_{i,j,k-1}^{[d]})/(d+1) \tag{2.11}$$

[illegible] $i+j+k = d+1$ [illegible] $\mathbf{c}^{[d+1]} = \{c_{ijk}^{[d+1]}\}_{i+j+k=d+1}$ [illegible] $\mathbf{c}^{[d]} = \{c_{ijk}^{[d]}\}_{i+j+k=d}$ [illegible] (2.2) [illegible]

$$\mathbf{c}^{[d+1]} = \frac{1}{d+1} A_d \mathbf{c}^{[d]}, \tag{2.12}$$

[illegible] A_d [illegible] $(d+2)(d+3)/2 \times (d+1)(d+2)/2$ [illegible]

[illegible]. It is easy to see that the degree raising algorithm (2.11) can be expressed in a matrix form (2.12). It is trivial to prove that the pattern of A_d is hierarchical. □

We now extend (2.8) to the case of different degrees on the neighboring triangles and get the desired explicit smoothness condition.

THEOREM 2.5. [illegible] $d_1 \geq d_2$ [illegible]

$$p_{d_1}(v) := \sum_{i+j+k=d_1} c_{ijk} B_{ijk}^{d_1}(v) \qquad \widetilde{p}_{d_2}(v) := \sum_{i+j+k=d_2} \widetilde{c}_{ijk} \tilde{B}_{ijk}^{d_2}(v).$$

[illegible] $\mathbf{c}$ [illegible] $\tilde{\mathbf{c}}$ [illegible] $\{c_{ijk}\}_{i+j+k=d_1}$ [illegible] $\{\widetilde{c}_{ijk}\}_{i+j+k=d_2}$ [illegible] q [illegible]

$$D_u^n p_{d_1}(v) = D_u^n \widetilde{p}_{d_2}(v) \qquad \forall v \in \langle v_2, v_3 \rangle$$

[illegible]

$$P_{n,d_1} \left\{ \frac{d_2!}{d_1!} \prod_{d=d_2}^{d_1-1} A_d \right\} \tilde{\mathbf{c}} = P_{0,d_1-n} \left\{ \prod_{l=1}^{n} \mathcal{D}_{\lambda,d_1+1-l} \right\} \mathbf{c} \qquad \forall n = 0, \ldots, r. \tag{2.13}$$

[illegible]. We separate the proof into two cases: $d_1 = d_2$ and $d_1 > d_2$. For the first case, it is easy to see that (2.13) is reduced to (2.8).

For the second case, we raise the degree d_2 of polynomial to d_1 by applying (2.11) for $d_1 - d_2$ times to get

$$\tilde{\mathbf{c}}^{[d_1]} = \frac{d_2!}{d_1!} \left\{ \prod_{d=d_2}^{d_1-1} A_d \right\} \tilde{\mathbf{c}}.$$

Note that Bézier coefficients $\tilde{\mathbf{c}}^{[d_1]}$ have the same degree as d_1. Then we can apply the usual smoothness conditions, i.e., (2.8). Thus the smoothness conditions for various degrees follow. □

They are the new smoothness conditions which will be used in the rest of the paper.

3. A new spline method for numerical solution of PDEs. In [1], Awanou, Lai, and Wenston proposed a multivariate spline method (i.e., the ALW method) for numerical solution of Poisson and biharmonic equations. The most striking advantage of the ALW method is that one can directly use piecewise polynomial functions for solving Poisson and biharmonic equations without constructing finite elements, macroelements, and locally supported basis functions. The ALW spline method can be explained briefly as follows: It uses piecewise polynomials over a triangulation and treats smoothness conditions as side constraints together with boundary conditions to minimize the Euler–Lagrange functionals associated with Poisson or biharmonic equations. An iterative method is introduced to solve the special linear systems arising from the discretization using bivariate spline functions. The method is shown to be very convenient for numerical solution of Poisson and biharmonic equations. In this section, our main purpose is to improve the ALW spline method by using bivariate splines of various degrees through adaptive procedure.

3.1. A unified model for Poisson and biharmonic equations. First let us introduce some necessary notation. Let L_k be a symmetric partial differential operator of order $2k$ with $k \geq 1$. In the bivariate setting, let

$$D^\alpha = \left(\frac{\partial}{\partial x}\right)^{\alpha_1} \left(\frac{\partial}{\partial y}\right)^{\alpha_2} \quad \forall \alpha = (\alpha_1, \alpha_2)$$

and let $|\alpha| = \alpha_1 + \alpha_2$. We intend to find approximate solutions of the following boundary value problem:

$$\left\{ \begin{array}{lcll} L_k u & = & f & \text{in } \Omega, \\ D^\alpha u & = & D^\alpha g & \text{on } \partial\Omega, \ |\alpha| \leq k-1, \end{array} \right. \tag{3.1}$$

where Ω is a polygonal domain in $\mathbf{R}^2$ and $f \in L^2(\Omega)$, $g \in C^{k-1}(\Omega)$. Note that model problem (3.1) includes the Poisson equation in the case $L_1 = -\Delta$ and the biharmonic equation in the case $L_2 = \Delta^2$.

The weak formulation for model problem (3.1) reads as follows: Find $u \in H^k(\Omega)$ which satisfies its boundary condition such that

$$a(u, v) = \langle f, v \rangle \quad \forall v \in H_0^r(\Omega), \tag{3.2}$$

where $a(u, v)$ is the bilinear form defined by

$$a(u, v) = \left\{ \begin{array}{ll} \displaystyle\int_\Omega \nabla u \cdot \nabla v dx dy, & k = 1, \\ \displaystyle\int_\Omega \triangle u \triangle v dx dy, & k = 2, \end{array} \right.$$

and $\langle f, v \rangle = \int_\Omega f(x, y) v(x, y) dx dy$ is the L_2 inner product of f and v. Here $H^k(\Omega)$ and $H_0^k(\Omega)$ are standard Sobolev spaces.

It is known (cf. [8]) that model problem (3.2) has a unique solution u in $H^k(\Omega)$. The proof can be adapted to show that there is a unique solution s_u in spline space $S_{\mathbf{d}}^r(\triangle) \subset H^k(\Omega)$, where $r = k - 1$ and s_u satisfies

$$a(S_u, v) = \langle f, v \rangle \quad \forall v \in S_{\mathbf{d}}^r(\triangle) \tag{3.3}$$

and the boundary conditions.

Based on the standard calculus of variations, model problem (3.1) is the Euler–Lagrange equation of the following energy functional:

$$E_r(s) = \frac{1}{2}a(s,s) - \langle f, s\rangle.$$

It is known that the weak solution of the model problem is the minimizer of the energy functional $E(s)$ in the following spline space (cf. [8, section 8.2.3]):

$$V = \{s \in S^r_{\mathbf{d}}(\triangle),\, D^\alpha s = D^\alpha g_h \quad \text{on } \partial\Omega,\, |\alpha| \le k-1\},$$

where $g_h \in S^r_{\mathbf{d}}(\triangle)$ is a spline such that $g_h|_{\partial\Omega}$ is a good approximation of the Dirichlet boundary condition g. For simplicity let us assume that g is a function of piecewise polynomial of degree $\le d$. Then the weak solution u satisfies

$$E_r(u) = \min_{s\in V} E_r(s). \tag{3.4}$$

Also any minimizer satisfying (3.4) is the weak solution.

3.2. Spline solution for (3.4). Next we discuss how to approximate weak solutions using spline spaces $S^r_{\mathbf{d}}(\triangle)$, which is also denoted as $S^r_{\mathbf{d}}$ if no confusion arises. We shall compute the approximation $s_u \in S^r_{\mathbf{d}}$ satisfying

$$E_r(s_u) = \min_{s\in S^r_{\mathbf{d}}\cap V} E_r(s).$$

Let us write a spline function $s \in S^r_{\mathbf{d}}$ in the following piecewise polynomial format:

$$s(x,y)|_t = \sum_{i+j+k=d_t} c^t_{i,j,k} B^t_{i,j,k}(x,y), \quad (x,y) \in t \in \triangle.$$

Let $\mathbf{c} = (c^t_{i,j,k},\, i+j+k = d_t,\, t \in \triangle)$ be the B-coefficient vector associated with s. The above energy functional can be written as

$$\tilde{E}_r(\mathbf{c}) = \frac{1}{2}\mathbf{c}^T K_r \mathbf{c} - \mathbf{c}^T M \mathbf{f}$$

with respect to B-coefficient vector $\mathbf{c}$, subject to the smoothness conditions $H\mathbf{c} = 0$ according to (2.13) and boundary condition $B\mathbf{c} = G$, as treated in [1]. Here, $\mathbf{K}_r = \mathrm{diag}(K^t_r,\, t \in \triangle)$ is the stiffness matrix with blocks

$$K^t_1 = \left[\int_t \nabla B^t_{ijk} \nabla B^t_{lmn} dxdy\right]^{l+m+n=d_t}_{i+j+k=d_t} \qquad \forall t \in \triangle$$

when $r = 1$, while K_r is the bending matrix in case $r = 2$, with

$$K^t_2 = \left[\int_t \Delta B^t_{ijk} \Delta B^t_{lmn} dxdy\right]^{l+m+n=d_t}_{i+j+k=d_t} \qquad \forall t \in \triangle$$

as its blocks. Similarly, let $\mathbf{M} = \mathrm{diag}(M_t,\, t \in \triangle)$ be the mass matrix with blocks

$$M_t = \left[\int_t B^t_{ijk} B^t_{lmn} dxdy\right]^{l+m+n=d_t}_{i+j+k=d_t} \qquad \forall t \in \triangle$$

and let $\mathbf{f}$ be the B-coefficient vector of the spline S_f interpolating f at the domain points of t of degree d_t for $t \in \triangle$. By using a Lagrange multiplier method, we let

$$\tilde{L}_r(\mathbf{c}, \alpha, \beta) = \frac{1}{2}\mathbf{c}^T\mathbf{K}_r\mathbf{c} - \mathbf{c}^T\mathbf{M}\mathbf{f} + \alpha H\mathbf{c} + \beta(B\mathbf{c} - G)$$

and compute local minimizers. It is easy to see that we need to solve the following system:

$$\begin{bmatrix} B^T & H^T & \mathbf{K}_r \\ 0 & 0 & B \\ 0 & 0 & H \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \mathbf{c} \end{bmatrix} = \begin{bmatrix} \mathbf{M}\mathbf{f} \\ G \\ 0 \end{bmatrix}. \tag{3.5}$$

Note that the existence and uniqueness of s_u imply that there exists a unique $\mathbf{c}$ satisfying the above linear system. Thus the iterative method in [1] can be applied and is effective in solving this linear system. If we denote $L^T = [B^T H^T]$, $\lambda = [\lambda_1^T \lambda_2^T]^T$, $\mathbf{F} = \mathbf{M}\mathbf{f}$ and the vector G absorbs the zero vector, then system (3.5) becomes

$$\begin{bmatrix} L^T & \mathbf{K}_r \\ 0 & L \end{bmatrix} \begin{bmatrix} \lambda \\ \mathbf{c} \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ G \end{bmatrix}.$$

For any fixed $\epsilon > 0$, given an initial guess $\lambda^{(0)}$, e.g., $\lambda^{(0)} = 0$, we first compute

$$c^{(1)} = \left(\mathbf{K_r} + \frac{1}{\epsilon}L^T L\right)^{-1} \left(F + \frac{1}{\epsilon}L^T b - L^T\lambda^{(0)}\right),$$

and for $k = 1, 2, \ldots,$ we iteratively compute

$$c^{(k+1)} = \left(\mathbf{K_r} + \frac{1}{\epsilon}L^T L\right)^{-1} \left(\mathbf{K_r}c^{(k)} + \frac{1}{\epsilon}L^T b\right). \tag{3.6}$$

The matrix $(\mathbf{K_r} + \frac{1}{\epsilon}L^T L)$ can be proved to be invertible, and the iterative algorithm (3.6) is convergent (cf. [1] for more details). For large-scale problems, the conjugate gradient method for the above iteration is preferred.

3.3. Adaptive algorithms. For the case $r = 0$ or $r = 1$, it is known that when $d_t = d$ for all $t \in \triangle$ with $d \geq 3r + 2$, the spline space S_d^r possesses the optimal approximation order (cf. [5]). So we require $\min_{t\in\triangle} d_t \geq 3r + 2$. What we need to do now is decide the degree vector $\mathbf{d}$ triangle by triangle. It depends on an error indicator and an adaptive procedure. For most problems, an a posteriori error estimate combined with a robust adaptive strategy is necessary as in adaptive finite element methods. So robust adaptive procedures are introduced here, and we leave the a posteriori error estimate to the numerical examples. For a better understanding of the performance of spline spaces with various degrees, we pay attention only to the performances of the local mesh refinement and the local degree raising. Let us present these two adaptive procedures in the following pseudocodes.

ALGORITHM 3.1 (hVersion).

```
mesh = initialize the Mesh;
TOL = 1e-5;
STEP = 1;
MAXSTEP = 30;
refine_percent = 0.8;
```

```
while STEP <MAXSTEP
    x = solve PDE with fixed degree method on mesh;
    error = posteriori_estimate(mesh,x);
    if sum(error) < TOL
        STEP = MAXSTEP + 1;
        break;
    end
    tris = indicate_by_fraction(error,refine_percent);
    mesh = refine_mesh(mesh,tris);
    STEP = STEP + 1;
end
```

In the above algorithm, the PDE is solved by any fixed degree algorithms such as the finite element method in Example 4.1 and the ALW spline method in Example 4.2. We use the fixed fraction strategy for mesh refinement, in which some fraction of elements with highest error are marked for refinement. It appears to be the most robust and convenient strategy among error equidistribution, fixed threshold, error density equidistribution, and fixed fraction strategies (cf. [11]), and has been proved to be convergent in energy norm for the Poisson equation in [12]. The marked elements are refined by Rivara's algorithm, which bisects the prescribed triangle. It was originally developed in [10] to produce a conforming triangulation. Rivara's algorithm is given as follows:

```
find the midpoint P of the longest side in T;
Bisect T with P;
while P is a nonconforming point for T' different to T
    find the midpoint Q of the longest side in T';
    bisect T' with Q;
    if P != Q
        join P and Q;
        P = Q
    end
end,
```

which can be illustrated as in Figure 3.1.

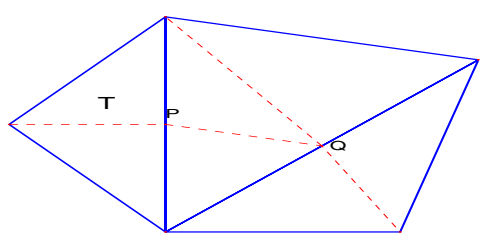


FIG. 3.1. *Conforming refinement using Rivara's algorithm.*

ALGORITHM 3.2 (pVersion).

```
mesh = initMesh;
TOL = 1e-5;
STEP = 1;
MAXSTEP = 50;
refine_percent = 0.8;
```

```
while STEP < MAXSTEP
    x = solve (3.5) with extended smoothness condition (2.13) on mesh;
    error = posteriori_estimate(mesh,x);
    error = smooth_error(mesh,error);
    if max(error) < TOL
        STEP = MAXSTEP + 1;
        break;
    end
    tris = refine_by_fraction(error,refine_percent);
    degree(tris) = degree(tris) + 1;
    STEP = STEP + 1;
end
```

In the above algorithms, we have adopted an error smoothing scheme which is as follows:

```
step 1: compute the mean error on each node according to its
        adjacent elements.
step 2: recompute the mean error on each element
        according to its adjacent nodes.
```

It is important for the convergence of our p-version algorithm to have such an error smoothing scheme because it is not desired that the degree vary too rapidly for adjacent triangles.

We shall use these two adaptive procedures in the next section where numerical results are presented.

4. Numerical examples. We first present our numerical experiments for the Poisson equation.

4.1. Consider the Poisson equation on an L-shaped domain:

$$\begin{cases} -\triangle u &= f & \text{in } \Omega, \\ u &= g & \text{on } \partial\Omega, \end{cases}$$

where $\Omega = \{(x,y),\ (x,y) \in (-1,1)\times(-1,1)\backslash(-1,0)\times(-1,0)\}$. Note that our spline method does not require the information of an analytic solution. For a clear comparison, let

$$u(x,y) = r^{\frac{2}{3}} \sin\left(\frac{2\theta+\pi}{3}\right)$$

be an analytic function, where $r(x,y) = \sqrt{(x^2+y^2)}$ and θ is the angle between vector (x,y) and the position direction of axis x with $\tan(\theta) = y/x$. We use the standard finite element method and our spline method to approximate $u(x,y)$ by solving the Poisson equation with $f = -\Delta u$ inside Ω and $g = u$ on the boundary.

Since we do not use any information about the analytic solution during the adaptive procedure, it is standard to introduce an a posteriori error estimator. We prefer the explicit residual a posteriori error esitmator given in [9] for the Poisson equation. That is, for any triangle T, its L_2 error indicator is

$$\eta_T := \left\{ |T|^4 ||f + \triangle u_h||^2_{0,2;T} + \frac{1}{2}\sum_{e\in\partial T} |e|^3 ||R_e||^2_{0,2;e} \right\}^{\frac{1}{2}}, \tag{4.1}$$

where $|T|$ is the minimal diameter of circles that contain triangle T, $|e|$ is the length of edge e, and R_e stands for the jump of $\frac{\partial u_h}{\partial n}$ on the internal edges or the jump of $\frac{\partial u_h}{\partial n} - g$ on the boundary edges.

The properties of solution for this problem have been discussed theoretically in [15], and its p/hp finite element solution is presented in [14]. For both methods we begin with a special triangulation as shown in Figure 4.1.

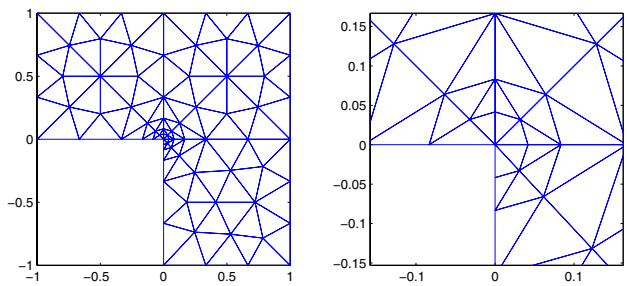


FIG. 4.1. *At the left is an original triangulation, which is amplified at the right.*

We use such a specially designed mesh, which is generated by hand, to capture the corner singularity. The full automatic hp adaptive procedure proposed in [14] required a more accurate reference solution and hence is too expensive to use. Since we pay attention only to the performance of our p-version spline method, a prebuilt mesh is enough.

In Table 4.1 we list the performance of a continuous fixed degree finite element method based on Algorithm 3.1 (hVersion) as well as the performance of our spline method with local degree raising (Algorithm 3.2) starting with continuous bivariate splines of degree 3, referred to as $S_3^0(\triangle)$. We show the number of triangles and the number of unknown coefficients of the finite element solutions and spline solutions which achieve a given tolerance $5e-3$, $5e-4$, or $5e-5$, respectively. We use cubic, quartic, and quintic finite element methods according to different tolerances because the low order finite element method yields too many elements, which is beyond the ability of our machine when the tolerance is small. In addition, we show the total CPU time for computing the finite element solution and spline solution for each of the tolerances. Our machine is equipped with Intel Celeron 2.0GHz and 512M memory.

TABLE 4.1
The performance of fixed order finite element methods and our spline method.

	$5e-3$		$5e-4$		$5e-5$	
	Cubic	Spline	Quartic	Spline	Quintic	Spline
Number of elements	872	116	953	180	1078	216
Number of unknowns	8720	2942	14295	4361	22638	6681
CPU time	29.21s	9.83s	45.98s	16.71s	72.89s	32.46s

From Table 4.1, it is clear to see that our spline method with local degree raising yields higher efficiency due to the smaller size of its system matrix. We also plot in Figure 4.2 the approximated solutions of both approaches whose tolerance is $5e-3$ and in Figure 4.3 the differences of both solutions against the exact analytic solution.

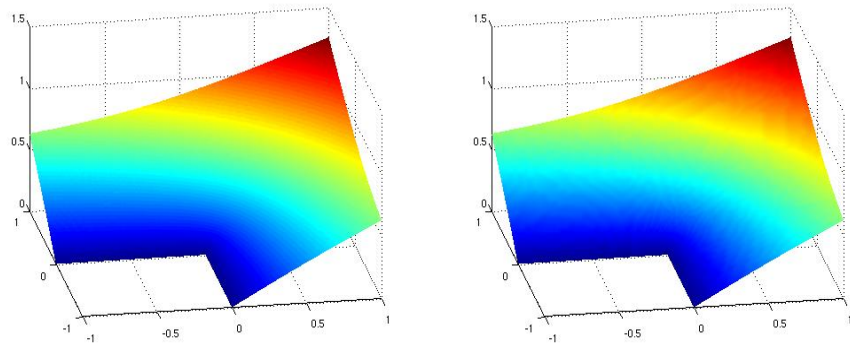


FIG. 4.2. *The approximate solution of the cubic finite element (left) and our spline method (right) under tolerance* $5e-3$.

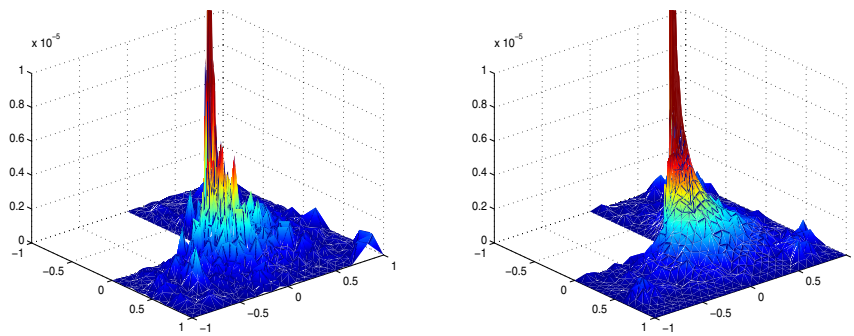


FIG. 4.3. *The error of approximate solution of the cubic finite element (left) and our spline method (right) under tolerance* $5e-3$.

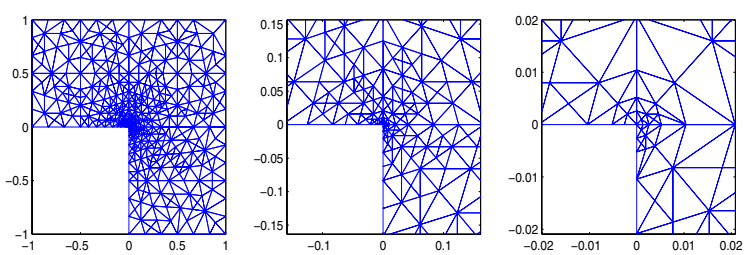


FIG. 4.4. *The resulting triangulation (left) for finite element solution and corner amplification (middle and right).*

From the figures in Figure 4.3, we can see that the error plot of our spline solution is less bumpy than that of the finite element solution and hence, that the surface of our spline solution is closer to the surface of the exact solution that of the finite element solution.

In addition, we present in Figure 4.4 the triangulation of the finite element solution

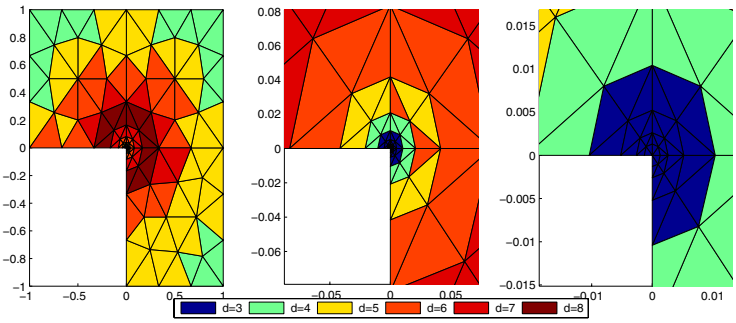


FIG. 4.5. *The triangulation and degrees for our spline solution and two corner amplifications.*

when the given tolerance is $5e-3$, and in Figure 4.5 the degrees as well as the triangulation of the spline solution when the given tolerance is $5e-3$, where we can see that the highest degree in our spline solution is 8, which happens in a few triangles around the singularity.

Next we present our numerical results for biharmonic equations.

EXAMPLE 4.2. We consider a biharmonic problem on an L-shaped domain:

$$\begin{cases} \Delta^2 u &= f & \text{in } \Omega, \\ u &= g & \text{on } \partial\Omega, \\ \frac{\partial}{\partial \mathbf{n}} u &= h & \text{on } \partial\Omega, \end{cases}$$

where Ω is the same L-shaped domain as in the previous example. We test the analytic solution

$$u(x,y) = r^{\frac{5}{2}} \sin\left(\frac{5\theta}{2}\right),$$

where $r(x,y) = \sqrt{(x^2+y^2)}$ and θ is the angle between vector (x,y) and the position direction of axis x with $\tan(\theta) = y/x$. Then f, g, h can be computed directly from u.

If we denote $[\cdot]_e$ as the jump of a function from inside T to outside through edge e, then the L_2 error indicator for any triangle T is the following explicit residual a posteriori estimator, as proposed in [9]:

(4.2)
$$\eta_T := \left\{ |T|^4 ||\triangle^2 u_h - f||^2_{0,2;T} + \sum_{e\in\partial T} [|e|\,||[\triangle u_h]_e||^2_{2;e} + |e|^3 ||[n_e \cdot \nabla \triangle u_h]_e||^2_{2;e}] \right\}^{\frac{1}{2}},$$

where the means of $|T|$ and $|e|$ are the same as above and n_e is the unit outward normal to edge $e \in \partial T$. The behavior of the true solution is very similar to the one in Example 4.1. We first use $S^1_5(\triangle)$ based on the special triangulation as in Figure 4.1 to solve this biharmonic problem by using the ALW spline method in [1] together with Algorithm 3.1. Then we start with $S^1_5(\triangle)$ using Algorithm 3.2 to solve this problem again. For each tolerance $1e-3$, $1e-4$, $1e-5$, we show in Table 4.2 the number of triangles, the number of unknown coefficients, and the total computational times of two spline solutions.

TABLE 4.2
The performance of the ALW spline method and our spline method.

	$1e-3$		$1e-4$		$1e-5$	
	ALW	New	ALW	New	ALW	New
Number of elements	210	116	370	122	716	128
Number of unknowns	4410	3766	7770	4484	15036	5192
Total CPU time	13.45s	15.93s	31.52s	27.61s	75.73s	39.10s

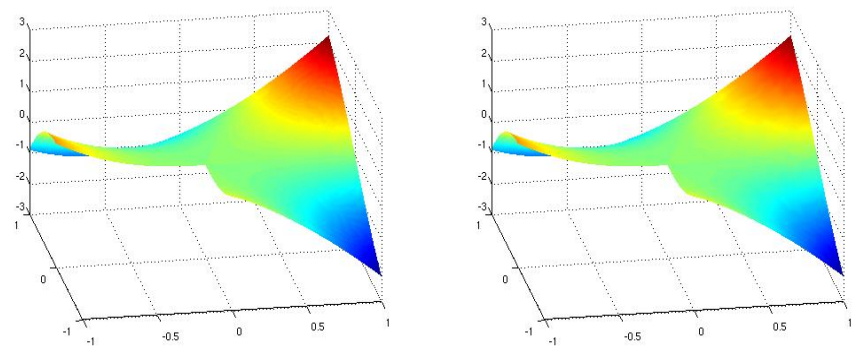


FIG. 4.6. *The approximate solution of the ALW spline method (left) and our spline method (right) under tolerance* $1e-5$.

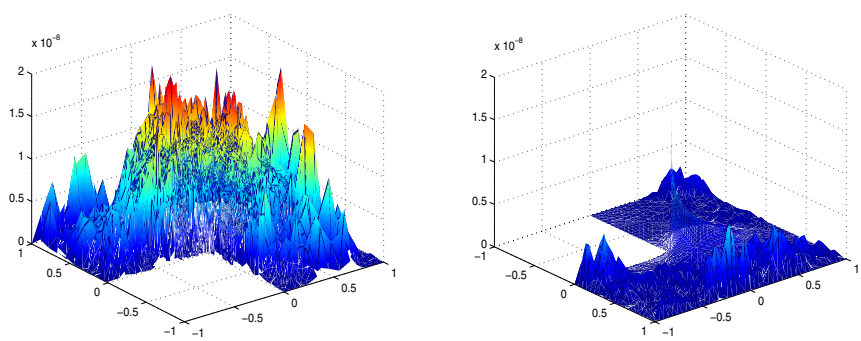


FIG. 4.7. *The error of approximate solution of the ALW spline method (left) and our spline method (right) under tolerance* $1e-5$.

Table 4.2 clearly shows that our spline method is much more efficient than the ALW spline method using h-adaptivity when the tolerance is small. We also plot in Figure 4.6 the approximated solutions of both approaches whose tolerance is $1e-5$ and in Figure 4.7 the differences of both solutions against the exact analytic solution. Comparing the figures in Figure 4.7, we can see that our spline solution is closer to the exact solution than the ALW solution.

In addition, we present in Figure 4.8 the triangulation of the ALW spline solution when the given tolerance is $1e-5$ and in Figure 4.9 the degrees as well as the triangulation of the spline solution with the same tolerance, where we can see that the highest degree in our spline solution is 12, which happens in a few triangles close to the singularity.

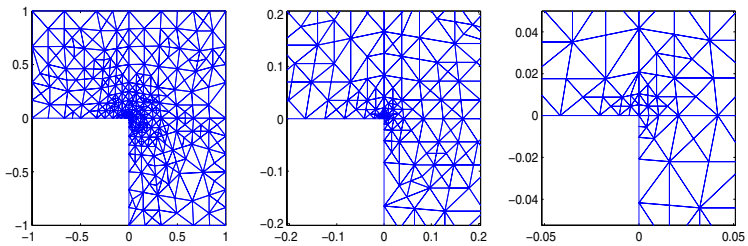


FIG. 4.8. *The resulting triangulation (left) for the ALW spline method and corner amplifications (middle and right).*

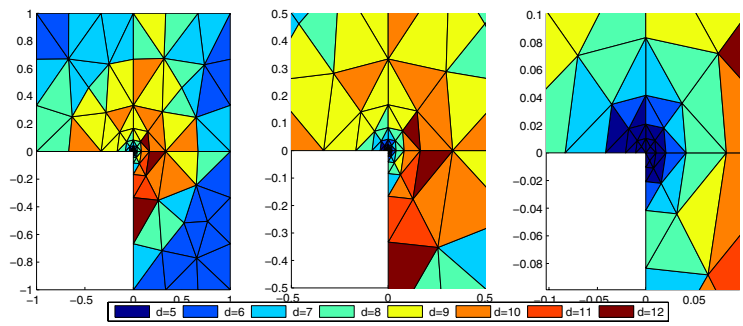


FIG. 4.9. *The triangulation and degrees for our spline solution and two corner amplifications.*

In general, we do not have an analytic solution available to guide our adaptive procedures. Our adaptive algorithms work for these cases. On the other hand, more accurate a posteriori error estimators are preferred for deciding whether to raise degrees or refine the triangle. In fact, a full automatic strategy that combines Algorithms 3.1 and 3.2 is favorable. It is left for future research.

5. Conclusion and remarks. In this paper, we give a new smoothness condition for polynomials with difference degrees over two adjacent triangular patches. Based on the new smoothness conditions we implement bivariate splines of various degrees for numerical solution of PDEs. The p-version adaptive strategy enables us to efficiently solve Poisson and biharmonic equations with singularity on the boundary.

We offer the following remarks:

REMARK 5.1. It is easy to see that the new smoothness conditions can be generalized to deal with polynomials over adjacent simplices in $\mathbf{R}^s$, $s \geq 3$.

REMARK 5.2. It is also easy to see that our spline method can be generalized to numerically solve trivariate partial differential equations, e.g., three-dimensional Poisson equations and three-dimensional biharmonic equations.

REMARK 5.3. It is possible to use our spline method together with local adaptive procedures (h-version) to solve nonlinear PDEs such as two-dimensional Navier–Stokes equations. It will be useful to capture the details for eddies of fluid flows. We are currently working on this research problem.

REFERENCES

[1] G. Awanou, M. J. Lai, and P. Wenston, *The multivariate spline method for scattered data fitting and numerical solutions of partial differential equations*, in Wavelets and Splines, G. Chen and M. J. Lai, eds., Nashboro Press, Brentwood, TN, 2006, pp. 24–76.

[2] C. de Boor, *B-form basics*, in Geometric Modeling: Algorithms and New Trends, G. Garin, ed., SIAM, Philadelphia, 1987, pp. 131–148.

[3] G. Farin, *Triangular Bernstein-Bézier patches*, Comput. Aided Geom. Design, 3 (1986), pp. 83–127.

[4] M. J. Lai, *Geometric interpretation of smoothness conditions of triangular polynomial patches*, Comput. Aided Geom. Design, 14 (1997), pp. 191–199.

[5] M. J. Lai and L. L. Schumaker, *On the approximation power of bivariate splines*, Adv. Comput. Math., 9 (1998), pp. 251–279.

[6] M. J. Lai and L. L. Schumaker, *Spline Functions over Triangulations*, Cambridge University Press, Cambridge, UK, 2007.

[7] M. J. Lai and P. Wenston, *Bivariate splines for fluid flows*, Comput. & Fluids, 33 (2004), pp. 1047–1073.

[8] L. Evans, *Partial Differential Equations*, AMS, Providence, RI, 1998.

[9] R. Verfürth, *A Review of A Posteriori Error Estimation and Adaptive Mesh-Refinement Techniques*, Wiley, Teubner, Stuttgart, Germany, 1996.

[10] M. C. Rivara, *Algorithms for refining triangular grids suitable for adaptive and multigrid techniques*, Internat. J. Numer. Methods Engrg., 20 (1984), pp. 745–756.

[11] R. Becker and R. Rannacher, *A general concept of adaptivity in finite element methods with applications to problems in fluid and structural mechanics*, in Grid Generation and Adaptive Algorithms, IMA Vol. Math. Appl. 113, Springer-Verlag, New York, 1999, pp. 51–75.

[12] W. Dörfler, *A convergent adaptive algorithm for Poisson's equation*, SIAM J. Numer. Anal., 33 (1996), pp. 1106–1124.

[13] J. M. Melenk, *hp-Finite Element Methods for Singular Perturbations*, Lecture Notes in Math. 1796, Springer-Verlag, Berlin, 2002.

[14] P. Solín, K. Segeth, and I. Dolezel, *High-Order Finite Element Methods*, Chapman & Hall/CRC Press, Boca Raton, FL, 2003.

[15] G. Strang and F. Fix, *An Analysis of the Finite Element Method*, Prentice–Hall, Englewood Cliffs, NJ, 1973.

[16] P. G. Ciarlet, *The Finite Element Method for Elliptic Problems*, North–Holland, Amsterdam, 1978.

SIAM J. SCI. COMPUT.
Vol. 29, No. 4, pp. 1355–1374
© 2007 Society for Industrial and Applied Mathematics

NUMERICAL METHODS FOR COMPUTING NONLINEAR EIGENPAIRS: PART I. ISO-HOMOGENEOUS CASES*

XUDONG YAO† AND JIANXIN ZHOU‡

Abstract. With a Rayleigh quotient formulation, a local minimax method is developed to solve a class of (iso-homogeneous) nonlinear eigenpair problems for multiple solutions in Banach spaces following their instability order. The algorithm is implemented to compute (weighted) eigenpairs of the p-Laplacian. Numerical eigenfunctions are illustrated by their graphics. Several interesting phenomena have been observed and are open for further investigation. Mathematical analysis related to convergence and instability order of computed eigenfunctions, etc., is also presented.

Key words. nonlinear eigenpairs, multiple critical points, local minimax method

AMS subject classifications. 45R50, 35P30, 58E05, 35J65

DOI. 10.1137/060651859

1. Introduction and some preliminaries.

The semilinear Schrödinger equation

$$iw_t + \Delta w + \kappa f(|w|)w = 0, \tag{1.1}$$

where κ is a physical parameter, is one of the canonical nonlinear equations in physics, arising in various fields, such as nonlinear optics, plasma fluids, quantum physics, Bose–Einstein condensates, and surface waves. For example, the Gross–Pitaevskii equation [10, 27]

$$\begin{aligned} i\psi_t &= -\frac{1}{2}\Delta\psi + V(x)\psi + \beta|\psi|^2\psi, \quad t > 0, x \in \Omega \subseteq \mathbb{R}^d, \\ \psi(x,t) &= 0, \quad x \in \Gamma = \partial\Omega, t \geq 0, \end{aligned} \tag{1.2}$$

on a bounded open domain $\Omega \in \mathbb{R}^d$, is extensively used to model the single particle of the Bose–Einstein condensate after the first experimental realization of the Bose–Einstein condensate in dilute weakly interacting gases shows great agreement. To study stability/instability, pattern, and other solution properties, people are interested in finding the standing (solitary) wave solutions of the form $w(x,t) = u(x)e^{-i\lambda t}$, where λ is a normalized potential and u is a real function of x, and also the steady-state solutions $w(x,t) = u(x)$. Then (1.1) leads to two semilinear elliptic PDEs

$$\lambda u(x) + \Delta u(x) + \kappa f(|u|)u(x) = 0, \tag{1.3}$$

$$-\Delta u(x) = \kappa f(|u|)u(x). \tag{1.4}$$

Several different types of quasi-linear Schrodinger equations are also proposed to model "Schrodinger flows" in various fields; see [16, 14, 22] for general study and

*Received by the editors February 9, 2006; accepted for publication (in revised form) December 20, 2006; published electronically June 12, 2007. This research was supported in part by NSF grant DMS-0311905.

http://www.siam.org/journals/sisc/29-4/65185.html

†Department of Mathematics, University of Connecticut, Storrs, CT 06269 (xudong@math.uconn.edu).

‡Department of Mathematics, Texas A&M University, College Station, TX 77843 (jzhou@math.tamu.edu).

physical applications of this topic. When non-Darcian or non-Newtonian fluids are considered, Darcy's law is replaced by other relations between the specific discharge (shear stress) $\vec{q}$ and the flow head w. One of them is the Izbash formula [15, 4]: $\vec{q} = -M|\nabla w|^{p-2}\nabla w$ for some constant $M > 0$ and the (rheological) characteristic $p > 1$ of the fluid. Then after replacing the Laplacian Δu by the p-Laplacian $\Delta_p u(x) = \nabla \cdot (|\nabla u(x)|^{p-2}\nabla u(x))$, (1.3) and (1.4) are generalized to

$$\lambda u(x) + \Delta_p u(x) + \kappa f(|u(x)|)u(x) = 0, \tag{1.5}$$

$$-\Delta_p u(x) = \kappa f(|u(x)|)u(x), \tag{1.6}$$

two quasi-linear elliptic PDEs. When $p = 2, p < 2, p > 2$, the fluid is called, respectively, (Darcian) Newtonian, pseudoplastic, dilatant. The p-Laplacian Δ_p is also used to model phenomena in reaction-diffusion in porous media ($p = \frac{3}{2}$), nonlinear elasticity ($p > 2$), and glaciology ($p \in (1, \frac{4}{3})$); see [4, 24] and references there. When $p = 2$, (1.5) and (1.6) reduce to (1.3) and (1.4). When the nonlinear term $f(|u(x)|)u(x)$ is primitive, i.e., $f(|t|)t = \frac{d}{dt}F(t)$ for some function F, the problems are variational and (1.5) and (1.6) are, respectively, the Euler–Lagrange equations ($J'(\cdot) = 0$) of the (energy) functionals

$$J(u, \lambda) = \int_\Omega \left[\frac{1}{p}|\nabla u(x)|^p - \kappa F(u(x)) - \frac{\lambda}{2}(|u(x)|^2 - \alpha)\right] dx, \tag{1.7}$$

$$J(u) = \int_\Omega \left[\frac{1}{p}|\nabla u(x)|^p - \kappa F(u(x))\right] dx. \tag{1.8}$$

People are interested in knowing the values of λ and κ for which (1.5) and (1.6) have nontrivial, positive, or nodal solutions. It turns out that those values are either directly determined by or closely related to the eigenvalues of Δ_p as defined in (1.9) [23]. So we leave more general problems (1.5) and (1.6) to a subsequent paper [38] and consider, in this paper, computing eigenpairs $(\lambda, u) \in \mathbb{R} \times (B \setminus \{0\})$ of the (weighted) p-Laplacian

$$-\Delta_p u(x) = \lambda w(x)|u(x)|^{p-2}u(x) \quad \forall x \in \Omega, \quad u \in B = W_0^{1,p}(\Omega), \tag{1.9}$$

where $\Omega \subset \mathbb{R}^n$ is open and bounded, $w : \Omega \to \mathbb{R}$ is a weight function, and $|\nabla u|^2 = \sum_{i=1}^n (\frac{\partial u}{\partial x_i})^2$. Equation (1.9) is also closely related to the resonance phenomenon, spectral theory of nonlinear operators, Sobolev embedding and inequality, etc.; see, e.g., [7, 24]. When $p = 2$, (1.9) reduces to an eigenproblem of the Laplacian Δ and is well known for its importance in many applications and theoretical studies. Thus (1.9) provides a new way to study the eigenproblem of Δ by embedding it into a one-parameter family of operators Δ_p. Hence new and better understanding can be obtained; see Figure 15 and its remarks. For the existence of multiple solutions to (1.9), we refer to [18, 3].

Equation (1.9) leads us to study the following nonlinear eigenpair problem (NEP). Let B be a Banach space, B' its dual, $\langle , \rangle$ the dual relation, and $\|\cdot\|$ the norm in B. Let F' and G' be the Fréchet derivatives of functionals F and G in $C^1(B, \mathbb{R})$. For given $\alpha > 0$, find eigenpairs $(\lambda, u) \in \mathbb{R} \times (B \setminus \{0\})$ such that (s.t.)

$$F'(u) = \lambda G'(u) \text{ in } B' \qquad \text{subject to } G(u) = \alpha, \tag{1.10}$$

where λ is called an [illegible] and u is called an [illegible] associated with λ. Under certain conditions, existence of countable eigenpairs of (1.10) can be established (see Proposition 44.26 in [39]).

Recall that for $J \in C^1(B, \mathbb{R})$, a point $u^* \in B$ is a critical point of J if u^* solves its Euler–Lagrange equation $J'(u) = 0$, where $J'(u)$ is its Fréchet derivative at u. $c = J(u^*)$ is called the critical level of J at u^*. A critical point u^* is nondegenerate if $J''(u^*)$ exists and has a bounded inverse. Otherwise u^* is degenerate. The first candidates for critical points are the local extrema of J. Critical points that are not local extrema are called [illegible]. In a physical system, saddle points appear as unstable equilibria or excited states.

Eigenpairs (λ, u) of (1.9) correspond to critical points of its Lagrange functional

$$\mathcal{L}(u, \lambda) = F(u) - \lambda(G(u) - \alpha). \tag{1.11}$$

People used to be interested only in the stable solutions (ground states) to a physical system, which correspond to the local minima of its energy functional. Thus traditional existence analysis and numerical methods were focused on such solutions. Other solutions are unstable equilibria or excited states and correspond to saddle critical points of its energy functional. They are much harder for theoretical analysis and more elusive for numerical approximation. However, multiply/highly excited states have been observed in various physical systems to have a variety of configurations, instability/maneuverability. With new advanced laser (synchrotronic) technologies, scientists are able to induce and reach those excited unstable solutions and search for new applications [1, 25, 29, 28, 31]. Such advanced study and applications have changed people's view about unstable (excited) solutions. Thus a huge literature [12, 13] is available in modern critical point theory on the existence of multiple unstable solutions to various nonlinear equations including (1.3)–(1.6) and (1.10). For related numerical computations, a large literature exists in computational physics/chemistry. However most of these results are just for lowly (simply/doubly) excited states (see, e.g., a survey paper [11]). The methods are usually based upon physical intuition of a problem and are not mathematically justified. Although this type of problem did not gain enough attention in the computational mathematics community, we have witnessed the momentum building up in recent years; see [2, 5, 26], and a series of publications, e.g., [8, 9], by Garcia-Ripoll and Perez-Garcia and their associates. Even numerical results for computing highly excited states have appeared more often recently [1, 6, 17, 26, 34]. However, most of them are based on a Newton-type method. They depend heavily on a good initial guess; i.e., an initial guess must be chosen close to the unknown solution to be found, they cannot handle degenerate cases and do not assume or use the variational structure of a problem, and thus are blind to instability order of multiple solutions. However, instability of excited states is an important issue in applications. So far, the only mathematically justified numerical method for computing multiple saddle points of a functional following their instability order is the local minimax (LMM) developed in [20, 21, 40, 41, 32, 33, 35, 36]. But it does not quite fit to NEP (1.10) due to the constraint.

For NEP (1.10), the simplest case is where both F' and G' are linear, e.g., $F'(u) = Au$ and $G'(u) = u$, where A is a linear self-adjoint operator from a Hilbert space B to B. Then (1.10) is a well-studied linear eigenpair problem whose wide applications are well known with various numerical packages available. When either F' or G' is nonlinear, (1.10) is a nonlinear eigenpair problem. For example, in (1.4), F' is linear but G' is nonlinear; in (1.3), G' is linear but F' is nonlinear; and in (1.6), both F' and G' are nonlinear.

So far, people's understanding of (1.10) is still quite limited, and its exact solutions are too hard to obtain. Due to the space setting, the nonlinearity, and the

multiplicity of solutions, (1.10) is very difficult to solve numerically. The first problem one has to overcome is how to numerically search for a *new* solution, not those known solutions. A saddle-type unstable solution may be easily perturbed to decay to a solution in a lower critical level. Whether or not a proposed numerical algorithm can find eigenfunctions following their instability order becomes an important issue. First, with an order, one can avoid a search going to those known solutions, and one can discuss whether or not some eigenfunctions have been missing in a computation. Second, since all eigenfunctions are unstable solutions, except those corresponding to local minima of the energy functional, their instability index is important information in an application. Since all eigenvalues in (1.10) are real, a natural order of eigenfunctions is the order of their eigenvalues. Then two questions must be answered: (a) Can a numerical algorithm find eigenfunctions in this order? If the answer to (a) is yes, then (b) is there a relation between this natural order and the instability order of their eigenfunctions? When (1.10) is linear and in a Hilbert space (Example 1.1)), due to the orthogonality between two different eigenfunctions, the Rayleigh–Ritz method (see [39] and Theorem 2.2) uses the orthogonal complement space of the known solutions to find eigenfunctions following the order of their eigenvalues. The eigenvalues are also the critical values of the Rayleigh quotient at their eigenfunctions. Thus this order coincides with the order of critical (energy) level of the eigenfunctions. Can one do it the same way when (1.10) is nonlinear in a Banach space, and the orthogonality between two different eigenfunctions is broken? Such a numerical method is not available so far in the literature.

LMM can be modified to find critical points of (1.11) following the order of their local minimax index (LMI) (an instability index) [41]. What is the relation between LMI of critical points of (1.11) and the order of eigenvalues of (1.10)? The literature does not provide any information. It turns out that in general this is a very tough question. Can we classify eigenpair problems that may provide us with such information? In a sequence of papers we will try to answer these questions. In this paper, we assume that the eigenpair problem (1.10) satisfies the iso-homogeneous condition; i.e., there are $k = l > 0$ s.t.

$$F'(tu) = t^k F'(u) \quad \text{and} \quad G'(tu) = t^l G'(u) \qquad \forall t > 0, u \in B. \tag{1.12}$$

1.1. A linear eigenpair problem. Find $(\lambda, u) \in \mathbb{R} \times (B \setminus \{0\})$ s.t. $Fu = \lambda Gu$ where F and G are self-adjoint linear operators on a Hilbert space B and G is also positive definite, bounded. Then the iso-homogeneous condition (1.12) is satisfied with $k = l = 1$

Equation (1.9) satisfies the iso-homogeneous condition (1.12) with $k = l = p - 1$. Let $U = \{u \in B | G(u) = 0\}$. For most iso-homogeneous NEP including (1.9), $U = \{0\}$; thus, to simplify our discussion, we assume $U = \{0\}$. Define the Rayleigh quotient

$$J(u) = \frac{F(u)}{G(u)} \quad \forall u \in B \setminus U. \tag{1.13}$$

Then $J \in C^1(B \setminus U, \mathbb{R})$. Even without the iso-homogeneous condition, critical points and their critical values of J are always eigenfunctions and their eigenvalues of (1.10). The converse is not true in general. With the iso-homogeneous condition, it is proved in section 2 that the converse is also true. Thus we can focus on finding critical points of J. However, this problem has its own characters: due to the iso-homogeneous condition, α in (1.10) can be replaced by any nonzero number; i.e., if u^* is an eigenfunction, then so is tu^* for any $t > 0$; i.e., no eigenfunction is isolated, and also

$J(tu) \equiv J(u)\ \forall t > 0$ and $u \in B \setminus U$. Thus $\langle J'(u), u\rangle = 0$ and $J''(u)$ is not invertible everywhere, a severe degenerate case. We limit J on the unit sphere, and then the problem becomes constrained (still degenerate). We need to modify the LMM to solve for multiple critical points of J restricted to a unit sphere following the order of their LMI. To do so, section 2 is used to develop a corresponding LMM characterization of critical points. Its relations with other characterizations such as Rayleigh–Ritz, Courant–Fischer, and Ljusternik–Schnirelman are also presented there. Based upon this LMM characterization, a modified LMM is devised in section 3. Then in section 4, we implement the algorithm to solve our model problem (1.9). Except for the first eigenpair, which has been numerically computed in the literature [19], this is the first time that all of these eigenpairs have been numerically captured. Thus observations of those eigenfunctions on certain interesting phenomena are reported. In section 5, we present mathematical analysis related to convergence of the algorithm, an instability order of eigenfunctions, etc. In a subsequent paper [38], we solve (1.10) for nonhomogeneous cases, e.g., (1.3).

2. Characterization of eigenpairs.

2.1. Equivalence between critical points and eigenpairs.

LEMMA 2.1. (λ, u) (1.10) u J $\lambda = J(u)$

The "if" part is always true. To see the "only if" part, let $u \in B \setminus U$. Then we have

$$\int_0^1 \langle F'(tu), u\rangle dt = \int_0^1 \frac{d}{dt} F(tu) dt = F(u).$$

Similarly, $\int_0^1 \langle G'(tu), u\rangle dt = G(u)$. Thus, if (λ, u) is an eigenpair of (1.10), i.e., $F'(u) = \lambda G'(u)$, then the iso-homogeneous condition implies $F'(tu) = \lambda G'(tu)\ \forall t > 0$. This leads to $F(u) = \int_0^1 \langle F'(tu), u\rangle dt = \int_0^1 \lambda \langle G'(tu), u\rangle dt = \lambda G(u)$. Thus

$$\lambda = \frac{F'(u)}{G'(u)} = \frac{F(u)}{G(u)} \equiv J(u) \quad \text{and} \quad J'(u) = \frac{F'(u)G(u) - F(u)G'(u)}{G^2(u)} = 0. \qquad \square$$

2.1. We can focus on finding critical points of J. An important consequence of Lemma 2.1 is that if critical points are found in such a way that their critical values are in a monotone (increasing) order, then eigenvalues are obtained in the same order. Thus it is easy for us to discuss whether or not we miss any eigenfunctions.

2.2. A modified local minimax characterization. To modify the original LMM, let us introduce the following related notions. Define $L = [u_1, u_2, \ldots, u_{n-1}]$ as the space spanned by linearly independent normalized $u_1, u_2, \ldots, u_{n-1} \in B$ and $B = L \oplus L'$, where L' is the complement of L. Let $\mathcal{P} : B \to L'$ be the corresponding linear projection operator and $S_{L'}$ be the unit sphere in L'. For each $u \in S_{L'}$ denote $[L, u]_S = \{w = \sum_{k=1}^{n-1} t_k u_k + t_0 u | \sum_{k=1}^{n-1} t_k^2 + t_0^2 = 1\}$.

DEFINITION 2.2. $P : S_{L'} \to 2^B$ J *w.r.t.* L $u \in S_{L'}$ $P(u)$ J $[L, u]_S$ $w \in [L, u]_S$ $P(u)$ $\mathcal{N}(w)$ w $J(v) \le J(w)\ \forall v \in [L, u]_S \cap \mathcal{N}(w)$ $p : S_{L'} \to B$ J L $p(u) \in P(u)\ \forall u \in S_{L'}$ $u \in S_{L'}$ p local J u p u

Since J is a function continuous on the nonempty compact set $[L,u]_S$ for each $u \in S_{L'}$, $P(u)$ is always nonempty. Such a defined peak selection has some interesting properties that will be frequently used.

LEMMA 2.3. [illegible] $u_0 \in S_{L'}$ [illegible] p [illegible] J [illegible] u_0 [illegible] $\langle J'(p(u_0)), u_i\rangle = 0 \quad i = 0, 1, \ldots, n-1$

[illegible]. By the assumption, let $w = p(u_0) = \sum_{i=0}^{n-1} t_i u_i \in [L, u_0]_S$, where $\sum_{i=0}^{n-1} t_i^2 = 1$. For $0 \le i \le n-1$, if $\langle J'(w), u_i\rangle \ne 0$, let $w(s) = \frac{w+su_i}{c(s)} \in [L, u_0]_S$, where $c(s) = [(t_i+s)^2 + \sum_{k=0,k\ne i}^{n-1} t_k^2]^{\frac{1}{2}}$. Since $w(s) \to w$ as $s \to 0$, there exists $s_0 > 0$ s.t. when $0 < |s| < s_0$, we have

$$J(w(s)) - J(w) = \left\langle J'(w), \frac{s}{c(s)} u_i \right\rangle + o(\|w(s) - w\|),$$

where we have used the fact that $\langle J'(w), w\rangle = 0$. Thus when $|s|$ is small, the term $\frac{s}{c(s)}\langle J'(w), u_i\rangle$ dominates the difference of $J(w(s)) - J(w)$. Since this term can be made either positive or negative as we wish by properly selecting $s \ne 0$, it contradicts the assumption that w is a local maximum of J on $[L, u_0]_S$. Thus $\langle J'(w), u_i\rangle = 0$. ∎

Since $J'(u) \in B'$ not B, to find a descent direction, the role of the gradient in a Hilbert space is replaced by that of a pseudogradient in a Banach space.

DEFINITION 2.4. [illegible] $\bar{B} = \{u \in B : J'(u) \ne 0\}$ [illegible] $\theta \in (0, 1]$ [illegible] $u \in \bar{B}$ [illegible] $\mathcal{G}(u) \in B$ [illegible] pseudogradient [illegible] J [illegible] u [illegible] θ

$$\|\mathcal{G}(u)\| \le 1 \quad \textit{and} \quad \langle J'(u), \mathcal{G}(u)\rangle \ge \theta \|J'(u)\|.$$

[illegible] $\mathcal{G}$ [illegible] $\mathcal{G}$ [illegible] pseudogradient flow

The following lemma leads to a new LMM characterization of critical points of J and also a stepsize rule in our modified LMM method; see Step 4 of the flow chart in section 3.

LEMMA 2.5. [illegible] p [illegible] J [illegible] $\bar{u} \in S_{L'}$ [illegible] $J'(p(\bar{u})) \ne 0$ [illegible] $p(\bar{u}) \notin L$ [illegible] $s > 0$ [illegible] $\bar{u}(s) = \frac{\bar{u}+sw(\bar{u})}{\|\bar{u}+sw(\bar{u})\|} \in S_{L'}$ [illegible]

$$J(p(\bar{u}(s))) < J(p(\bar{u})) - \frac{1}{4} s\theta |t_n| \|J'(p(\bar{u}))\|, \quad \text{[illegible]} \quad p(\bar{u}) = \sum_{i=1}^{n-1} t_i u_i + t_n \bar{u} \in [L, \bar{u}]_S,$$

[illegible] $\sum_{k=1}^{n} t_k^2 = 1$ $w(\bar{u}) = -\mathrm{sign}(t_n)\mathcal{P}(\mathcal{G}(p(\bar{u})))$ [illegible] $\mathcal{P}(\mathcal{G}(p(\bar{u})))$ [illegible] [illegible] J [illegible] $p(\bar{u})$ [illegible] $\theta \in (0, 1]$

[illegible] Since $\langle J'(w), w\rangle = 0 \ \forall w \in B \setminus U$, p is continuous at $\bar{u}$, and $p(\bar{u}) \in [L, \bar{u}]_S$ when $s > 0$ is small, we have

$$\begin{aligned} J(p(\bar{u}(s))) &= J(p(\bar{u})) + \langle J'(p(\bar{u})), p(\bar{u}(s)) - p(\bar{u})\rangle + o(\|p(\bar{u}(s) - p(\bar{u}))\|) \\ &= J(p(\bar{u})) + \langle J'(p(\bar{u})), p(\bar{u}(s))\rangle + o(\|p(\bar{u}(s) - p(\bar{u}))\|). \end{aligned}$$

Since $p(\bar{u}) \notin L$ implies $t_n > 0$ and, by Lemma 2.3, as $s > 0$ is small, we have $t_n(s) > 0$ and

$$\langle J'(p(\bar{u})), p(\bar{u}(s))\rangle = -\frac{\mathrm{sign}(t_n) t_n(s) s}{\|\bar{u} + sw(\bar{u})\|} \langle J'(p(\bar{u})), \mathcal{P}(\mathcal{G}(p(\bar{u})))\rangle < -\frac{1}{2} s\theta |t_n| \|J'(p(\bar{u}))\|,$$

where $p(\bar{u}(s)) \equiv p(\frac{\bar{u}+sw(\bar{u})}{\|\bar{u}+sw(\bar{u})\|}) = t_1(s)u_1 + \cdots + t_{n-1}(s)u_{n-1} + t_n(s)\frac{\bar{u}+sw(\bar{u})}{\|\bar{u}+sw(\bar{u})\|} \in [L, \bar{u}(s)]_S$. Hence, when $s > 0$ is small,

$$J(p(\bar{u}(s))) < J(p(\bar{u})) - \frac{1}{4} s\theta |t_n| \|J'(p(\bar{u}))\|. \quad \blacksquare$$

2.2. From the last two lemmas, it is clear that the notion of a peak selection $p(\bar{u})$ can be generalized to satisfy $\langle J'(p(\bar{u})), \bar{u}\rangle = \langle J'(p(\bar{u})), u_i\rangle = 0$, $i = 1, \ldots, n-1$. As a direct consequence of Lemma 2.5, we have the following LMM characterization.

THEOREM 2.6. [illegible] p [illegible] J [illegible] $\bar{u} \in S_{L'}$ [illegible] $\bar{u} = \arg\min_{u \in S_{L'}} J(p(u))$ [illegible] $(p(\bar{u}), L) > 0$ [illegible] $p(\bar{u})$ [illegible] J. [illegible] $p(\bar{u})$ [illegible] (1.10) [illegible] $\lambda = J(p(\bar{u}))$ [illegible]

2.3. Comparison to other characterizations. Denote the generalized solution set $\mathcal{M} = \{p(u) | u \in S_{L'}\}$. Theorem 2.6 states that the nth eigenfunction u_n can be obtained through finding a local minimum of J on $\mathcal{M}$ or solving

$$\min_{u \in S_{L'}} \max_{v \in [u_1, \ldots, u_{n-1}, u]_S} J(v). \tag{2.1}$$

Next we discuss the relation between (2.1) and other well-known minimax principles.

THEOREM 2.7. [illegible] 1.1 [illegible] B [illegible] (2.1) [illegible] [39]; [illegible] $\langle u, v\rangle_G = \langle Gu, v\rangle$ [illegible] $\|u\|_G = (\langle u, u\rangle_G)^{\frac{1}{2}}$ [illegible] B $L' = L^{\perp} = \{u \in B | \langle u, u_i\rangle_G = 0, i = 1, \ldots, n-1\}$ [illegible] $S_{L^{\perp}} = \{u \in L^{\perp} : \|u\|_G = 1\}$ [illegible]

$$u_n = \arg \min_{u \in S_{L^{\perp}}} \max_{v \in [u_1, \ldots, u_{n-1}, u]_S} J(v) = \arg \min_{u \in S_{L^{\perp}}} J(u).$$

[illegible] It is known that $\langle u_i, u_j\rangle_G = 0, 1 \le i < j \le n$. Thus we only have to find $v = p(u) \in L^{\perp}$. Then (2.1) reduces to the Rayleigh–Ritz orthogonal method

$$\min_{u \in S_{L^{\perp}}} \max_{v \in [u]_S} J(v) = \min_{u \in S_{L^{\perp}}} J(u). \tag{2.2}$$

Conversely, if $u_n = \arg\min_{u \in S_{L^{\perp}}} J(u) = \arg\min_{u \in S_{L^{\perp}}} \langle Fu, u\rangle$, i.e., u_n is found by the Rayleigh–Ritz method, then there is a neighborhood $\mathcal{N}(u_n)$ of u_n s.t. $\forall u \in \mathcal{N}(u_n) \cap S_{L^{\perp}}$,

$$\max_{v \in [u_1, \ldots, u_{n-1}, u]_S} \langle Fv, v\rangle \ge \langle Fu, u\rangle \ge \langle Fu_n, u_n\rangle.$$

In addition, $\forall u \in [u_1, \ldots, u_{n-1}, u_n]_S$, we have $u = \sum_{i=1}^{n} c_i u_i$ with $\sum_{i=1}^{n} c_i^2 = 1$. Then

$$\begin{aligned} J(u) &= \langle Fu, u\rangle \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j \langle Fu_i, u_j\rangle = \sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j \lambda_i \langle Gu_i, u_j\rangle = \sum_{i=1}^{n} c_i^2 \lambda_i \le \lambda_n = J(u_n), \end{aligned}$$

where λ_i is the eigenvalue corresponding to u_i. Therefore

$$\begin{aligned} J(u_n) = \langle Fu_n, u_n\rangle &= \min_{u \in S_{L^{\perp}}} \max_{v \in [u_1, \ldots, u_{n-1}, u_n]_S} \langle Fv, v\rangle \\ &= \min_{u \in S_{L^{\perp}}} \max_{v \in [u_1, \ldots, u_{n-1}, u_n]_S} J(v). \qquad \square \end{aligned}$$

As for the NEP, since the orthogonality between two different eigenfunctions is broken, the Courant–Fischer minimax principle [39] states that u_n can be found by solving

$$\min_{W_n} \max_{v \in W_n \cap S} J(v), \tag{2.3}$$

where min is taken over all the subspaces W_n of dimension n in B, and max is global on $W_n \cap S$. The Courant–Fischer minimax principle is originally designed for linear eigenpair problems. People then found that it was also valid for nonlinear eigenpair problems where the iso-homogeneous condition is satisfied. The Ljusternik–Schnirelman minimax principle [39], which is commonly regarded as a generalization of the Courant–Fischer minimax principle, is used to characterize saddle points of a nonlinear functional J through solving

$$\inf_{K\in\mathcal{K}_n} \sup_{u\in K} J(u), \tag{2.4}$$

where $\mathcal{K}_n$ is the class of all compact subsets K of B with $\mathrm{ind}(K) \geq n$, sup is global on K, and inf is global over $\mathcal{K}_n$. When the iso-homogeneous condition is satisfied and J is the Rayleigh quotient, (2.4) coincides with (2.3). However, they are all two-level global minimax characterizations and do not help much in algorithm implementation. In contrast, our LMM method (2.1) can be implemented as the following numerical algorithm.

3. A modified LMM algorithm. Let $u_1, \ldots, u_{n-1}$ be $n-1$ previously found normalized critical points of J, $L = [u_1, \ldots, u_{n-1}]$, $B = L \oplus L'$, and $\mathcal{P} : B \to L'$ be the corresponding linear continuous projection operator.

1. Given $\theta \in (0,1)$ and $\lambda > 0$. Set $k = 1$. Choose $v_n^1 \in S_{L'}$, and solve for

$$u_n^1 \equiv p(v_n^1) \equiv \sum_{i=1}^{n-1} t_i^1 u_i + t_n^1 v_n^1 \equiv \arg\max \left\{ J(v) | v = \sum_{i=1}^{n-1} t_i u_i + t_n v_n^1, \sum_{i=1}^{n} t_i^2 = 1 \right\}.$$

2. Compute a descent direction $w_n^k = -\mathrm{sign}(t_n^k)\mathcal{P}(\mathcal{G}_n^k)$, where $\mathcal{G}_n^k$ is a pseudo-gradient of J at u_n^k with constant θ.
3. If $\|J'(p(v_n^k))\| \leq \epsilon$, then output $u_n^k = p(v_n^k)$ and stop. Otherwise, do Step 4.
4. Define $v_n^k(s) = \frac{v_n^k + s w_n^k}{\|v_n^k + s w_n^k\|}$ and

$$u_n^k(s) \equiv p(v_n^k(s)) \equiv \arg\max \left\{ J(v) | v = \sum_{i=1}^{n-1} t_i u_i + t_n v_n^k(s), \sum_{i=1}^{n} t_i^2 = 1 \right\},$$

where $(t_1^k, \ldots, t_n^k)$ is used as an initial point. Determine the stepsize

$$s_n^k = \max_{m\in N} \left\{ \frac{\lambda}{2^m} | 2^m \geq \|w_n^k\|, J\left(p\left(v_n^k\left(\frac{\lambda}{2^m}\right)\right)\right) - J(u_n^k) \ \leq -\frac{s\theta|t_n^k|}{4}\|J'(u_n^k)\| \right\}.$$

5. Set $v_n^{k+1} = v_n^k(s_n^k), u_n^{k+1} = u_n^k(s_n^k) \equiv p(v_n^{k+1}) \equiv \sum_{i=1}^{n-1} t_i^{k+1} u_i + t_n^{k+1} v_n^k(s_n^k)$. $k = k+1$ and go to Step 2.

3.1. (a) In Step 1, since L is finite-dimensional, we may either use a decomposition and normalization to choose $v_n^1 \in S_{L'}$ or use observation, among all functions "orthogonal" to L, to choose a normalized v_n^1 with the simplest nodal line structure. Our choice does not have to be exact. For example, when $L = \{0\}$, v_1^1 can be any nonzero function, but a positive one has the simplest nodal line structure. When $L = [u_1]$, where u_1 is positive, among all functions orthogonal to L we choose v_2^1 to be sign-changing and normalized with one nodal line, etc.

(b) In Step 2, theoretically we can either follow a pseudogradient flow or just find a pseudogradient. The projection is important to avoid a decay to the trivial/known

solutions, i.e., to satisfy $u^* = p(v^*) \notin L$. For a practical technique to compute the projection of a pseudogradient to a complement subspace of L in case of $B = W^{1,p}(\Omega)$, see section 4.1.

(c) In Step 4, it is important to use the initial guess $(t_1^k, \ldots, t_n^k)$ in order to consistently trace a peak selection and keep it continuous.

(d) It is easy to check that $\frac{1}{2}s^k \le s_n^k \le s^k$, where s^k is the exact stepsize defined by

$$s^k = \max\left\{ s | J(p(v_n^k(s))) - J(u_n^k) \le -\frac{1}{4} s\theta |t_n^k| \|J'(u_n^k)\|, \lambda \ge s\|w_n^k\| \ge 0 \right\} \tag{3.1}$$

and is actually used in our convergence analysis [36].

(e) The algorithm is stable in the sense $J(u_n^{k+1}) < J(u_n^k)$.

4. Some numerical results for eigenpairs of the p-Laplacian. In this section we implement the modified LMM to numerically solve our model problem (1.9) for multiple eigenpairs. The Rayleigh quotient is of the form

$$J(u) = \frac{\int_\Omega |\nabla u(x)|^p dx}{\int_\Omega w(x)|u(x)|^p dx}, \quad u \in B = W_0^{1,p}(\Omega),\ u \ne 0. \tag{4.1}$$

4.1. A practical technique for computing a pseudogradient. We develop a practical technique for computing the projection of a pseudogradient to a complement subspace of L based on the gradient of J and successfully apply it to all our numerical computations. The idea is described below.

The gradient $J'(u)$ of J at $u \in B$ is in the dual space $B' = W^{-1,q}(\Omega), (\frac{1}{p} + \frac{1}{q} = 1)$, where the smoothness is poor, and thus not qualified as a candidate for a pseudogradient in B. To increase its smoothness, we use ∇J instead of J', where $d = \nabla J(u)$ is in $W_0^{1,q}(\Omega)$ and defined by $\langle d, v\rangle_{W_0^{1,q}\times W_0^{1,p}} = \langle J'(u), v\rangle_{W^{-1,q}\times W_0^{1,p}}\ \forall v \in W_0^{1,p}(\Omega)$, or

$$\int_\Omega \nabla d(x)\cdot\nabla v(x)\,dx = -\int_\Omega \Delta d(x) v(x)\,dx = \int_\Omega J'(u)(x)v(x)\,dx \quad \forall v \in W_0^{1,p}(\Omega),$$

where $J'(u)(x) = \frac{p}{b^2}(b\Delta_p u(x) + aw(x)|u(x)|^{p-2}u(x))$ when J is given by (4.1). This leads us to solve

$$-\Delta d(x) = J'(u)(x) = \frac{p}{b^2}(b\Delta_p u(x) + aw(x)|u(x)|^{p-2}u(x)),\ x \in \Omega, \quad \text{and} \quad d|_{\partial\Omega} = 0,$$

where $a = \int_\Omega |\nabla u(x)|^p dx$ and $b = \int_\Omega w(x)|u(x)|^p dx$. Since in a numerical computation "nice" functions are usually used to approximate a solution, we check the ratio

$$\gamma = \frac{\|d\|_2^2}{\|d\|_p \cdot \|d\|_q}, \tag{4.2}$$

where $\|\cdot\|_r$ is the norm in $W_0^{1,r}(\Omega)$. By the Holder inequality, $\gamma \le 1$. If $\gamma > \alpha > 0$ at all points generated by the algorithm, then $\mathcal{G}(u) = \frac{d}{\|d\|_p} \in W_0^{1,p}(\Omega) \cap W_0^{1,q}(\Omega)$ is a pseudogradient of J at u w.r.t. α, since $\|J'(u)\| \le \|d\|_q$ [36]. All our numerical results show that $\gamma \gg 0$, although it has not been mathematically proved. As for the projection of $\mathcal{G}(p(u)) = \frac{d}{\|d\|_p}$, where $d = \nabla J(p(u))$, to a complement subspace of L, by the definition of p, we note that

$$0 = \langle J'(p(u)), v\rangle_{W^{-1,q}\times W_0^{1,p}} = \langle d, v\rangle_{W_0^{1,q}\times W_0^{1,p}} \quad \forall v \in L;$$

i.e., d is "orthogonal" to L in some sense. Thus no projection is needed. For such a pseudogradient, to get related convergence of the algorithm [36], in the flow chart, the expression of w_n^k in Step 2 should be $w_n^k = -\text{sign}(t_n^k)\nabla J(u_n^k)$, and the inequality to decide s_n^k in Step 4 and (3.1) should be $J(p(v_n^k(\frac{\lambda}{2^m}))) - J(u_n^k) \le -\frac{|t_n^k|}{4}\|\nabla J(u_n^k)\|_2^2$. Then the expression of $w(\bar{u})$ in Lemma 2.5 should be $w(\bar{u}) = -\text{sign}(t_n)\nabla J(\bar{u})$, and under the assumption $\|\nabla J(p(\bar{u}))\|_2 < +\infty$, the inequality should be $J(p(\bar{u}(s))) < J(p(\bar{u})) - \frac{1}{4}s|t_n|\|\nabla J(p(\bar{u}))\|_2^2$.

4.2. Some numerical results and observations. We select $\Omega = [0,2]\times[0,2]$, $p = 1.75, 2.5, 3.0$ for $w(x) \equiv 1$ and $p = 1.75, 2.5$, $r = 0.5, 6.0$ for $w(x) = |x - \bar{1}|^r$, where $\bar{1} = (1,\dots,1) \in \mathbb{R}^n$ and $|\cdot|$ is the Euclidean norm in $\mathbb{R}^n$. Either 800×800 or 1000×1000 linear square elements are used in the computation, and $\|\nabla J(p(v_n^k))\| \le \varepsilon = 10^{-3}$ is used to terminate the iteration. In Figures 1–14, for each case, we display the profiles of the first few numerical eigenfunctions following the order of their LMI. If two eigenfunctions have the same LMI, we list the one with a smaller eigenvalue first.

In the remaining part of this section, let us shorten the notation: (right) by (r), (middle) by (m), and (left) by (l). In Figures 1(l), 3(m), 5(r), 8(l), 9(r), 11(m), and 13(l) are the ground states (LMI = 1) computed with $L = \{0\}$. For other solutions, due to higher LMI, for computational efficiency, we utilized their symmetries. For example, for Figures 3(l), 5(l), 7(m), 9(m), 11(l), 12(r), and 14(r), we used their symmetries with $L = [u_1]$, where u_1 is their ground state, and other solutions are computed by using their symmetries with $L = \{0\}$. See [32] for details. Several interesting phenomena related to the (weighted) eigenpairs of Δ_p on the square domains have been observed from our numerical results.

(a) By comparing Figure 8(l) to Figure 9(r) and Figure 11(m) to Figure 13(l), we see that a peak-breaking phenomenon takes place for the first weighted eigenfunctions of Δ_p as r increases.

(b) By comparing eigenfunctions in each of the following groups: Figures 1(l), 3(m), and 5(l); Figures 1(m), 4(l), and 6(m); Figures 1(r), 3(r), and 6(l); Figures 2(l), 4(m), and 6(r); Figures 2(m), 5(m), and 7(r); Figures 2(r), 4(r), and 7(l); Figures 3(l), 5(l), and 7(m), we observe that for different values of p, they have the same number of peaks, similar peak locations, and peaks that become sharper when p gets larger.

(c) Keeping the order of eigenvalues in mind and grouping them by their patterns, Figures 1(m), 4(l), and 6(m) (side-to-side peak); Figures 1(r), 3(r), and 6(l) (corner-to-corner peak); Figures 2(m), 5(m), and 7(r) (3-peak); Figures 2(r), 4(r), and 7(l) (4-side-peak); Figures 3(l), 5(l), and 7(m) (5-peak), we find a very interesting pattern order switching phenomenon when p crosses 2, worth further investigation. Those phenomena also imply that when $p = 2$, the eigenvalues of the eigenfunctions with the same LMI should be the same. To see this phenomenon more clearly, we plot the first three eigenvalues of Δ_p for $1.6 \le p \le 2.4$ in Figure 15.

It is clear that the second and the third eigenfunctions of Δ_p switch their pattern order when p crosses 2. That is why the second eigenvalue of Δ is double. To our surprise, from the plot, the first eigenvalue of Δ_p seems to have a linear relation in p. This is the first time such relations have been numerically captured. They are certainly interesting and worthy of further theoretical analysis.

(d) If we pay attention to the peak locations and compare Figure 8(l) to Figure 9(r), Figure 8(m) to Figure 10(l), Figure 9(m) to Figure 11(l), Figure 11(m) to Figure 13(l), Figure 12(l) to Figure 13(r), and Figure 12(r) to Figure 14(r), we can see that the peaks prefer the corners when r increases and crosses some numbers.

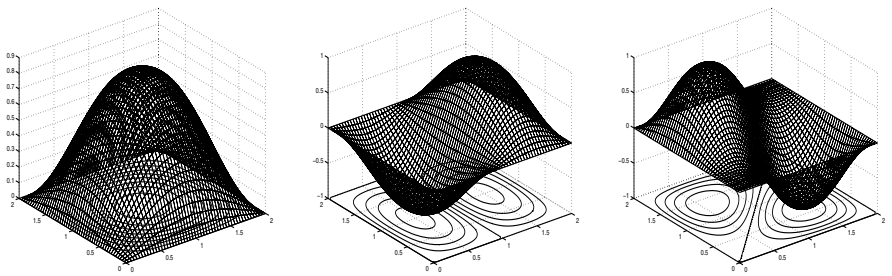


FIG. 1. *Eigenfunctions of* Δ_p, $p = 1.75$: $\lambda_1 = 4.245837$ *(left),* $\lambda_2 = 9.317313$ *(middle), and* $\lambda_3 = 9.407816$ *(right).*

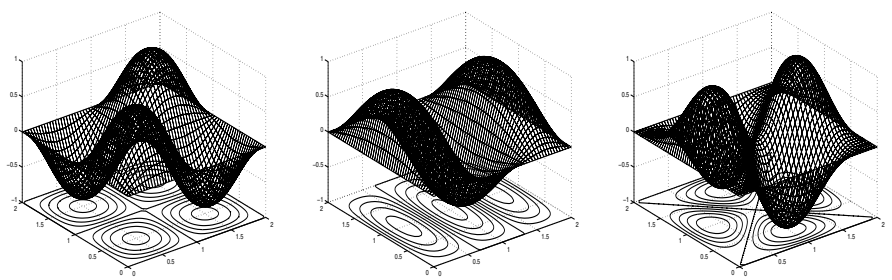


FIG. 2. *Eigenfunctions of* Δ_p, $p = 1.75$: $\lambda_4 = 14.280496$ *(left),* $\lambda_5 = 16.837822$ *(middle), and* $\lambda_6 = 17.254568$ *(right).*

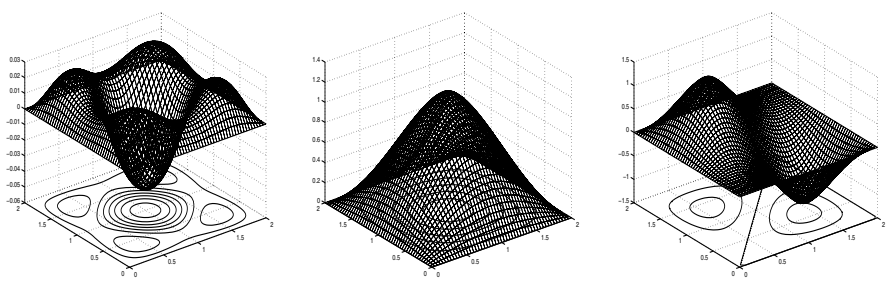


FIG. 3. *Eigenfunctions of* Δ_p, $p = 1.75$: $\lambda_7 = 23.366003$ *(left);* $p = 2.5$, $\lambda_1 = 6.354715$ *(middle); and* $\lambda_2 = 20.289627$ *(right).*

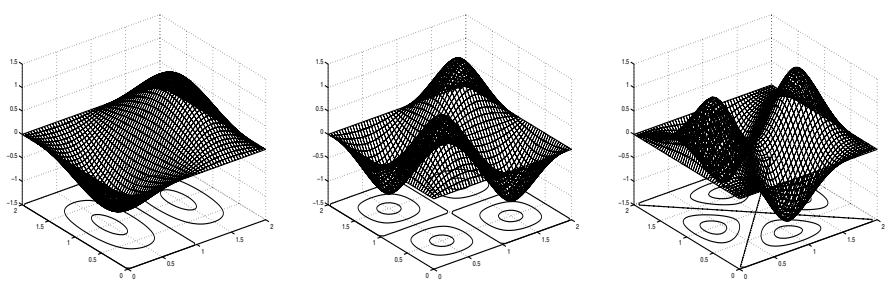


FIG. 4. *Eigenfunctions of* Δ_p, $p = 2.5$: $\lambda_3 = 20.798476$ *(right),* $\lambda_4 = 35.944786$ *(middle), and* $\lambda_5 = 48.259806$ *(right).*

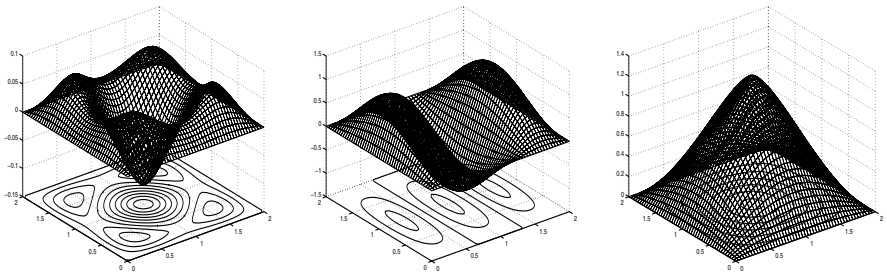


FIG. 5. *Eigenfunctions of* Δ_p, $p = 2.5$: $\lambda_6 = 49.679394$ *(left), and* $\lambda_7 = 51.104811$ *(middle);* $p = 3.0$, $\lambda_1 = 7.844420$ *(right).*

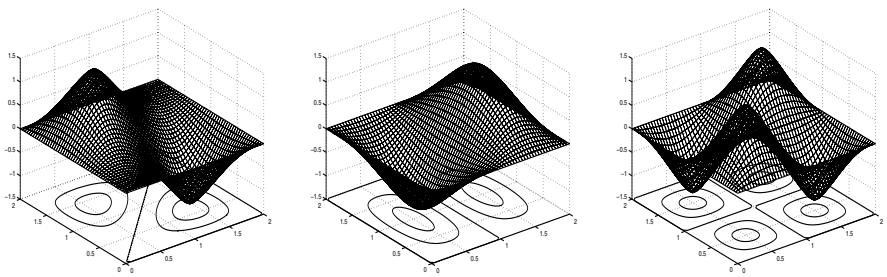


FIG. 6. *Eigenfunctions of* Δ_p, $p = 3.0$: $\lambda_2 = 32.098661$ *(left),* $\lambda_3 = 33.947805$ *(middle), and* $\lambda_4 = 62.748593$ *(right).*

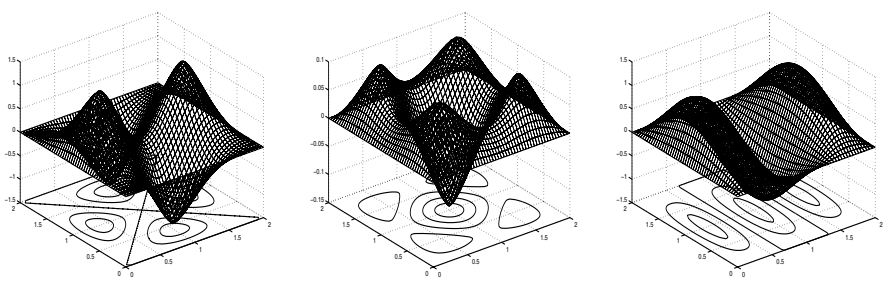


FIG. 7. *Eigenfunctions of* Δ_p, $p = 3.0$: $\lambda_5 = 90.795294$ *(left),* $\lambda_6 = 94.932100$ *(middle), and* $\lambda_7 = 102.660394$ *(right).*

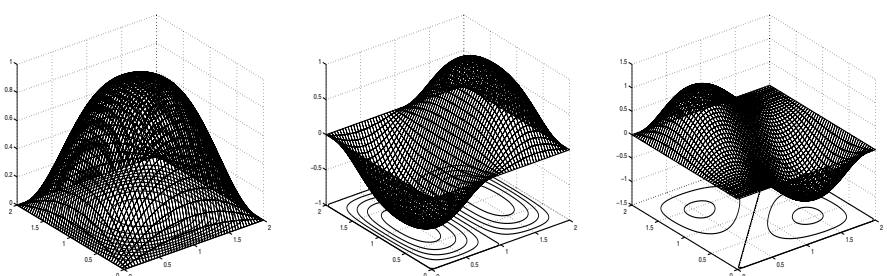


FIG. 8. *Weighted eigenfunctions of* Δ_p, $p = 1.75$, $r = 0.5$*:* $\lambda_1 = 6.088006$ *(left),* $\lambda_2 = 11.775095$ *(middle), and* $\lambda_3 = 11.938270$ *(right).*

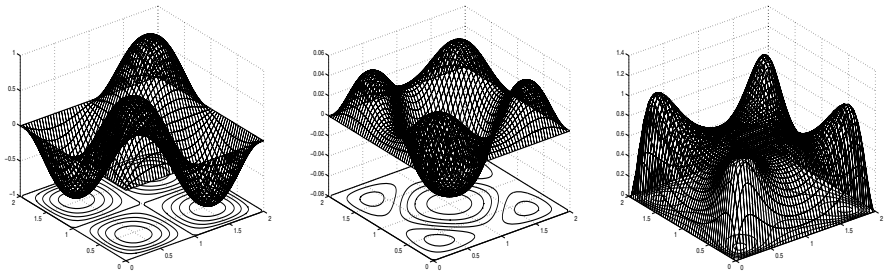


FIG. 9. *Weighted eigenfunctions of* Δ_p, $p = 1.75$, $r = 0.5$*:* $\lambda_4 = 16.633820$ *(left), and* $\lambda_5 = 23.366003$ *(middle); and* $p = 1.75$, $r = 6.0$, $\lambda_1 = 18.714875$ *(right).*

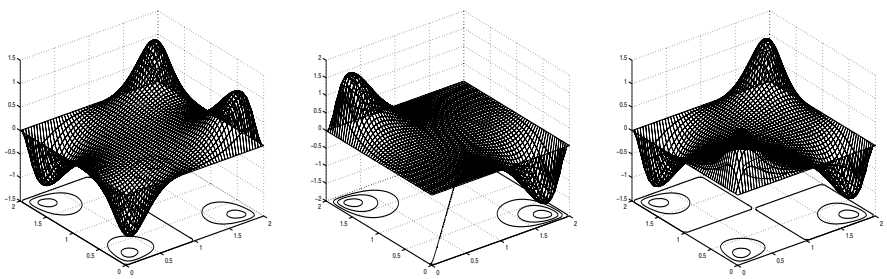


FIG. 10. *Weighted eigenfunctions of* Δ_p, $p = 1.75$, $r = 6.0$*:* $\lambda_2 = 20.312840$ *(left),* $\lambda_3 = 20.425545$ *(middle), and* $\lambda_4 = 20.738396$ *(right).*

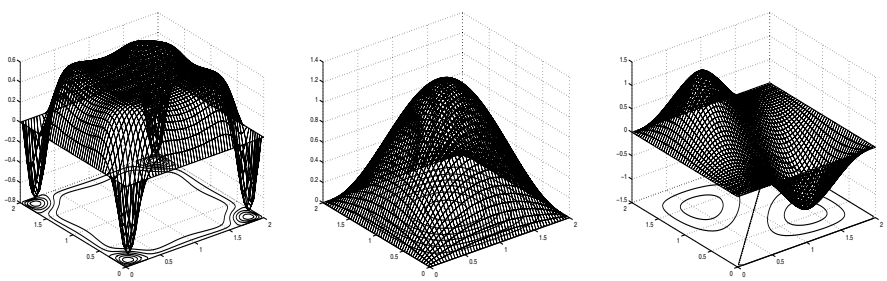


FIG. 11. *Weighted eigenfunctions of* Δ_p: $p = 1.75$, $r = 6.0$, $\lambda_5 = 34.801623$ *(left);* $p = 2.5$, $r = 0.5$, $\lambda_1 = 10.185286$ *(middle), and* $\lambda_2 = 26.174362$ *(right).*

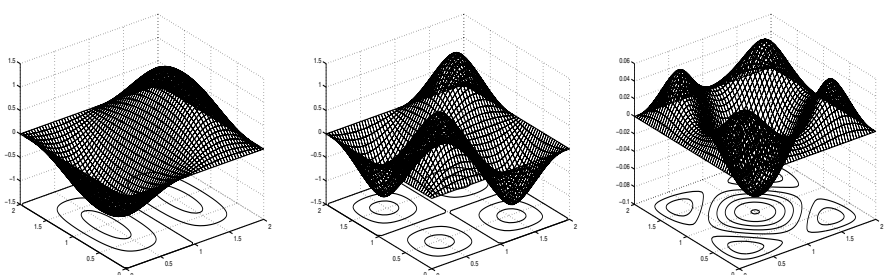


FIG. 12. *Weighted eigenfunctions of* Δ_p, $p = 2.5$, $r = 0.5$*:* $\lambda_3 = 26.991732$ *(left),* $\lambda_4 = 42.140740$ *(middle), and* $\lambda_5 = 69.931326$ *(right).*

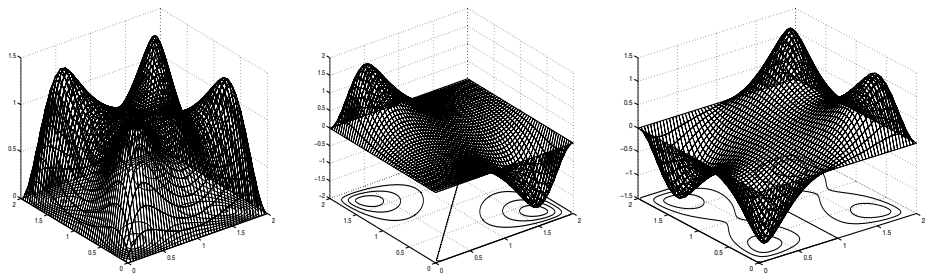


FIG. 13. *Weighted eigenfunctions of* Δ_p, $p = 2.5$, $r = 6.0$*:* $\lambda_1 = 65.223275$ *(left),* $\lambda_2 = 70.878805$ *(middle), and* $\lambda_3 = 71.815461$ *(right).*

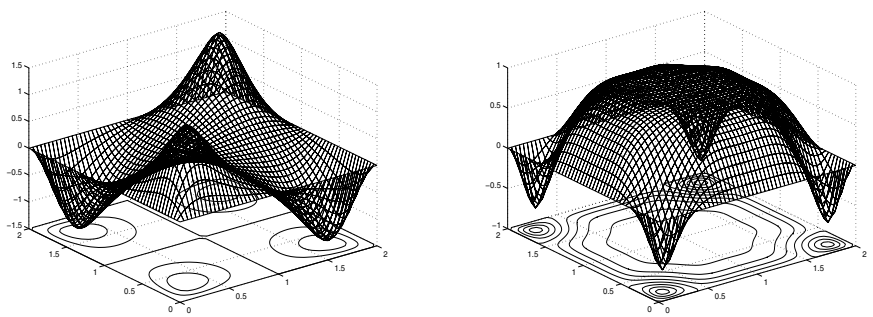


FIG. 14. *Weighted eigenfunctions of* Δ_p, $p = 2.5$, $r = 6.0$*:* $\lambda_4 = 74.271235$ *(left) and* $\lambda_5 = 161.729721$ *(right).*

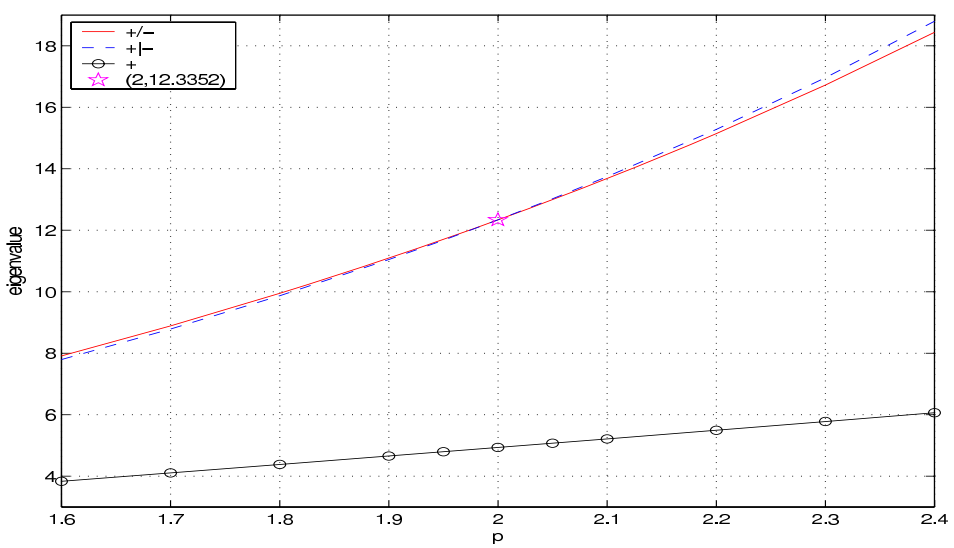


FIG. 15. *The first three eigenvalues of* Δ_p *for* $1.6 \le p \le 2.4$.

(e) To the weighted eigenpair problem of Δ_p, there is a corresponding p-Henon equation

$$-\Delta_p u + |x - \vec{1}|^r |u|^q u = 0,\ x \in \Omega, \quad u|_{\partial\Omega} = 0.$$

Usually, $q > p-2$. By the numerical results in [35], the symmetry of the ground state will be broken when r becomes large. This interesting case is called a symmetry-breaking phenomenon. But by our numerical results in this paper, it seems that for the weighted eigenpair problems, such a phenomenon never occurs. This observation motivates us to do further numerical investigations. Our findings will be reported in [37].

5. Some theoretical issues. The original LMM method was designed for finding multiple unconstrained saddle points. In contrast, our model problem in this paper is constrained and the critical points of the Rayleigh quotient J are always degenerate, which causes some theoretical issues related to the modified LMM. In this section, we present some mathematical analysis on those issues.

5.1. On convergence of the algorithm. To establish some convergence results, as suggested in Remark 2.2, we need to generalize the notion of a peak selection as follows.

DEFINITION 5.1. $P : S_{L'} \to 2^B$ $L \perp$ J $\forall v \in S_{L'}, P(v) = \{u = \sum_{i=1}^{n-1} t_i u_i + t_n v | \sum_{i=1}^{n} t_i^2 = 1, \langle \nabla J(u), u_i \rangle = 0, i = 1, \ldots, n-1\}$ $p : S_{L'} \to B$ $L \perp$ J $p(v) \in P(v)$ $\forall v \in S_{L'}$ $v \in S_{L'}$ J $L \perp$ v $\mathcal{N}(v)$ v $p : \mathcal{N}(v) \cap S_{L'} \to B$ $p(u) \in P(u)$ $\forall u \in \mathcal{N}(v) \cap S_{L'}$

The iso-homogeneous condition implies $\langle \nabla J(p(v)), p(v) \rangle = 0$ and $\langle \nabla J(p(v)), v \rangle = 0$ if $p(v) \notin L$. Then, a min-orthogonal algorithm can be designed through replacing a peak selection in the LMM approach by an L-$\perp$ selection $p(v) = t_1 u_1 + \cdots + t_{n-1} u_{n-1} + t_n v$, which is solved from

$$\langle \nabla J(t_1 u_1 + \cdots + t_{n-1} u_{n-1} + t_n v), u_i \rangle = 0, \qquad i = 1, .., n-1, \ \sum_{i=1}^{n} t_i^2 = 1. \tag{5.1}$$

We need the following condition.

DEFINITION 5.2. $u_i \in B$ $\|u_i\| = 1$ $i = 1, \ldots, n-1$ $J \in C^1(B, \mathbb{R})$ $\{w_i\} \subset B$ $w_i = \sum_{j=1}^{n-1} t_j^i u_j + t_n^i v_i$ $\{v_i\} \subset B, \|v_i\| = 1$ $\sum_{j=1}^{n} (t_j^i)^2 = 1$ $J(w_i)$ $\nabla J(w_i) \to 0$

Then with some modifications along the lines of the proofs, similar convergence results as in [36] can be established. Since the proofs are similar and very long, we omit them here.

5.2. On smoothness of L-$\perp$ selection. Since the selection of an L-$\perp$ is implicitly defined and its continuity or smoothness is important for our algorithm-design/convergence-analysis, the following implicit function theorem (IFT) approach is used to check whether p is smooth. This is one of the theoretical reasons why a peak selection has to be generalized to an L-$\perp$ selection. According to the definition of an L-$\perp$ selection, when $L = [u_1, u_2, \ldots, u_{n-1}]$, $p(v) = \sum_{i=1}^{n-1} t_i u_i + t_n v$, where $\sum_{i=1}^{n} t_i^2 = 1$ and $v \in S_{L'}$, is solved from (5.1). To apply the IFT to (5.1), we need to resolve the problem caused by the normalization $\sum_{i=1}^{n} t_i^2 = 1$, which represents the constraint of the problem and prevents us from taking derivative w.r.t. t_i. Since the system in (5.1) is iso-homogeneous, the normalization $\sum_{i=1}^{n} t_i^2 = 1$ can always be satisfied afterward through dividing each t_i by $(\sum_{j=1}^{n} t_j^2)^{\frac{1}{2}}$. Thus the normalization

condition can be released. Then for given $v \in S_{L'}$, the system (5.1) contains n unknowns $t_1, \ldots, t_n$ but $n-1$ equations. To obtain a square Hessian matrix of (5.1) and keep in mind the nondegeneracy condition $\operatorname{dis}(p(v), L) > 0$ in our local minimax characterization, Theorem 2.6, let us solve

$$\langle \nabla J(t_1 u_1 + \cdots + t_{n-1} u_{n-1} + v), u_i \rangle = 0, \quad i = 1, \ldots, n-1. \tag{5.2}$$

That is, we force $t_n = 1$ in (5.1). The IFT states that if $u = p(v) = t_1 u_1 + \cdots + t_{n-1} u_{n-1} + v$ satisfies (5.2), the $(n-1) \times (n-1)$ matrix

$$Q = \begin{bmatrix} \langle J''(u)u_1, u_1 \rangle & \cdots & \langle J''(u)u_{n-1}, u_{n-1} \rangle \\ \cdots & \cdots & \\ \langle J''(u)u_1, u_{n-1} \rangle & \cdots & \langle J''(u)u_{n-1}, u_{n-1} \rangle \end{bmatrix}$$

is invertible; i.e., $|Q| \neq 0$; then $(t_1(w), \ldots, t_{n-1}(w))$ can be solved from (5.2) around v; i.e., $p(w) = \sum_{i=1}^{n-1} t_i(w) u_i + w$ satisfies (5.2) and $(t_1(w), \ldots, t_{n-1}(w))$ is continuously differentiable around v. Thus the L-$\perp$ selection p is well defined and continuously differentiable near $v \in S_{L'}$. Then we can normalize $p(w)$ through a differentiable operation, i.e., multiplying $p(w)$ by $t_n(w) = 1/(\sum_{i=1}^{n-1} t_i(w)^2 + 1)^{\frac{1}{2}}$ to get the new $p(w) \in [L, w]_S \ \forall w$ near v in $S_{L'}$. Then (5.1) is satisfied, and the condition $|Q| \neq 0$ can be easily numerically checked.

5.3. On an instability order of eigenfunctions. In this subsection, we establish an instability order for eigenpairs of (1.10) computed by the modified LMM. Note that the role played by the iso-homogeneous condition (1.12) to define this order is double-edged, it enables the eigenvalues of (1.10) to coincide with the critical values of the Rayleigh quotient J (see Lemma 2.1), and it also makes J degenerate.

Let B be a Hilbert space and $J \in C^2(B, \mathbb{R})$. If $u^* \in B$ is a nondegenerate critical point of J, then $B = B^- \oplus B^+$, where B^- and B^+ are, respectively, the maximum negative and positive definite subspaces of $J''(u^*)$. The integer $\mathrm{MI}(u^*) = \dim(B^-)$ is called the Morse index (MI) of u^*. MI is commonly used to define an instability order for nondegenerate and unconstrained saddle points in a Hilbert space [30, 41]. But MI cannot handle degenerate cases and is not defined in a Banach space. Thus another approach allowing a Banach space setting, constrained, and degenerate cases has to be developed.

In computational chemistry and physics, the following notion of saddle order is of interest.

DEFINITION 5.3. $J \in C^1(B, \mathbb{R})$ $u^* \in B$ J $B = B^I \oplus B^D$ B^I, B^D B $u_1 \in B^I$ $u_2 \in B^D$ $\|u_1\| = 1$ $\|u_2\| = 1$ $r_1 > 0$ $r_2 > 0$

$$J(u^* + t u_1) \geq J(u^*) \qquad \forall\, 0 < |t| \leq r_1, \tag{5.3}$$

$$J(u^* + t u_2) < J(u^*) \qquad \forall\, 0 < |t| \leq r_2. \tag{5.4}$$

$\dim(B^D)$ J u^* saddle order u^*

Since (5.3) and (5.4) lack characterization and robustness, it is difficult for us to apply, in particular, in a Banach space. Thus in the definition, we replace (5.4) by

$$J(u^* + t u_2 + o(t)) < J(u^*) \qquad \forall\, 0 < |t| \leq r_2, \tag{5.5}$$

where $o(t)$ represents a higher order term. The integer $\dim(B^D)$ is then called the local saddle index (LSI) and will be used to define an instability order for critical points of J.

If u^* is a nondegenerate saddle point of J on a Hilbert space $B = B^- \oplus B^+$, then

$$J(u^* + tu_1 + o(t)) - J(u^*) = \frac{t^2}{2}\langle J''(u^*)u_1, u_1\rangle + o(t^2) > 0,$$
$$J(u^* + tu_2 + o(t)) - J(u^*) = \frac{t^2}{2}\langle J''(u^*)u_2, u_2\rangle + o(t^2) < 0$$

for any $u_1 \in B^+$, $u_2 \in B^-$ and $|t|$ small; i.e., $\mathrm{LSI}(u^*) = \mathrm{MI}(u^*)$. Since the definition of LSI by (5.3) and (5.5) does not concern the degeneracy of u^*, LSI is more general than MI. Next we assume that (5.3) and (5.5) are satisfied and prove that for a saddle point $u^* = p(v^*)$ found by our modified LMM, $\mathrm{LSI} = \dim(L) = k$, where $L = [u_1, \ldots, u_k]$ and $u_1, \ldots, u_k$ are linear independent and normalized. Due to the iso-homogeneous condition, the definition of the Rayleigh quotient J, and the condition $u^* \notin L$, as pointed out in section 5.2, our LMM characterization

$$v^* = \arg\min_{v \in S_{L'}} \max_{u \in [u_1, \ldots, u_k, v]_S} J(u), \tag{5.6}$$

i.e., $p(v) = \sum_{i=1}^{n-1} t_i u_i + t_n v, \sum_{i=1}^{n} t_i = 1$, is equivalent to solving

$$\bar{v} = \arg\min_{v \in S_{L'}} \max_{(t_1, \ldots, t_k) \in \mathbb{R}^k} J(t_1 u_1 + \cdots + t_k u_k + v), \tag{5.7}$$

i.e., $p(v) = t_1 u_1 + \cdots + t_k u_k + v$. However, (5.7) has an advantage that the local maximization at the second level is taken on a finite-dimensional space, not just on a manifold as in (5.6).

THEOREM 5.4. [illegible] B [illegible] $\|\cdot\|'$ [illegible] $B \setminus \{0\}$ [illegible] $v^* \in S_{L'}$ [illegible] J [illegible] p [illegible] L [illegible] v^* [illegible] p [illegible] v^* $u^* \equiv p(v^*) \notin L$ [illegible] $v^* = \arg\min_{v \in S_{L'}} J(p(v))$ [illegible] (5.7) [illegible] u^* [illegible] $k = \dim(L)$ [illegible]. Let B^D and B^I be defined as in (5.3) and (5.5). If $k = 0$, there is no local maximization in (5.7), or, $v^* = p(v^*)$ is a local minimum point of J. It is clear that $k = 0 = \dim(B^D)$. Now we assume that $k > 0$. Since $k > \dim(B^D)$ implies $L \cap B^I \neq \emptyset$, it contradicts our assumption that the local maximum in (5.7) is strict. Thus we have $k \leq \dim(B^D)$. To prove $k = \dim(B^D)$, recall $B = L \oplus L', v^* \in S_{L'}$. For each $v \in L'$ with $v \neq 0$, we define $E(v) = \frac{v}{\|v\|}$. We have $E'(v)w = \frac{w}{\|v\|} - \frac{v}{\|v\|}\langle \|\frac{v}{\|v\|}\|', w\rangle$. Let $L_{v^*}^{\perp} = \{w \in L' : \langle \|v^*\|', w\rangle = 0\}$. For each $w \in L'$, we can write $w = cv^* + (w - cv^*)$, where $c = \frac{d}{d\alpha}|_{\alpha=0}\|v^* + \alpha w\| = \langle \|v^*\|', w\rangle$ and $\langle \|v^*\|', v^*\rangle = \|v^*\| = 1$. Thus $w - cv^* \in L_{v^*}^{\perp}$. Since $[v^*] \cap L_{v^*}^{\perp} = \{0\}$, we have $L_{v^*}^{\perp} \oplus [v^*] = L'$. For each $w \neq 0$ in $L_{v^*}^{\perp}$, $E'(v^*)w = w$. Let $\alpha(s) \equiv \mathcal{P}(p(E(v^* + sw))) = E(v^* + sw)$, where $\mathcal{P}$ is the linear projection operator onto L'. We have $\alpha'(0) = \mathcal{P}(p'(E(v^*))E'(v^*)w) = E'(v^*)w = w$, i.e., $\mathcal{P}(p'(v^*)w) = w$. It follows that $w \in \{p'(v^*)(w), L\}$. Since $w \in L_{v^*}^{\perp} \subset L'$, we have $p'(v^*)w \notin L$. This leads to

$$p'(v^*)(L_{v^*}^{\perp}) \oplus [v^*] \oplus L = B. \tag{5.8}$$

Now we suppose $k < \dim(B^D)$. Then (5.8) implies $(p'(v^*)(L_{v^*}^{\perp}) \oplus [v^*]) \cap B^D \neq \{0\}$. Since $J(v) \equiv J(tv)\ \forall t \neq 0$ and $\forall v \neq 0$, we must have $p'(v^*)(L_{v^*}^{\perp}) \cap B^D \neq \{0\}$. Let

$w \in L_{v^*}^{\perp}$ s.t. $p'(v^*)(w) \in (B^D \setminus \{0\})$. Then $E(v^* + sw) \in \mathcal{N}(v^*) \cap S_{L'}$ for $|s|$ small. It follows that

$$\begin{aligned} p(E(v^* + sw)) &= p(E(v^*)) + sp'(E(v^*))E'(v^*)(w) + o(s) \\ &= p(v^*) + sp'(v^*)(w) + o(s) = u^* + sp'(v^*)(w) + o(s). \end{aligned}$$

Since $p'(v^*)(w) \in (B^D \setminus \{0\})$, we have, for $|s|$ small,

$$J(p(v^*(s))) = J(u^* + sp'(v^*)(w) + o(s)) < J(u^*) = J(p(v^*)).$$

On the other hand, by our assumption on v^* and (5.7), we should have

$$J(p(v^*(s))) = \max_{(t_1,\ldots,t_k)\in\mathbb{R}^k} J(t_1u_1 + \cdots + t_ku_k + v^*(s)) \geq J(p(v^*)).$$

This leads to a contradiction. We conclude that $k = \dim(B^D)$. □

The integer $k = \dim(L)$ is called the [illegible] of the constrained saddle point $u^* = p(v^*)$ of J. Theorem 5.4 states that LSI = LMI. Following our modified LMM, we start with $k = 0$ or $L = \{0\}$, to find a local minimum point u^1 of J. The LMI of u_1 is $k = 0$. Then we go to $k = 1$ or $L = [u_1]$ to find a saddle point u_2 of J of LMI $k = 1, \ldots$. Continuing in this way, we use $L = [u_1, \ldots, u_k]$ to find a saddle point u_{k+1} of J of LMI $= k$. Therefore for each $k = 0, 1, \ldots$, at least one saddle point u_{k+1} with LMI $= k$ will be found. The dimension of the maximum subspace of decreasing directions of J (energy function) at u_{k+1} is LSI = LMI $= k$. Thus the order of LMI coincides with the instability order of saddle points. In contrast, the order of the LMI can be easily followed by our modified LMM method. It is clear that $J(u_j) < J(u_k)$ for $j = 0, 1, \ldots, k-1$. Since Theorem 2.6 states $J(u_k) = \lambda_k$, we also have $\lambda_j < \lambda_k$ for $j = 0, 1, \ldots, k-1$. Since the modified LMM is local, this order may be partial and contains multiple branches. If the order contains only one branch, then no eigenfunction will be missed and the order of eigenfunctions coincides with the natural order of their eigenvalues. However, when the order contains multiple branches, i.e., more than one eigenfunction has the same LMI or a bifurcation takes place, each branch contains a sequence of eigenfunctions according to their LMI whose order coincides with the natural order of their eigenvalues. For linear eigenpair problems as in Example 1.1, all eigenfunctions with the same LMI have the same eigenvalues. For a nonlinear eigenpair problem, as we found in our numerical results, a significant difference is that eigenfunctions with the same LMI may have different eigenvalues. Eigenvalues of two eigenfunctions from two different branches cannot be compared, whether they have the same or different LMI.

Acknowledgment. The authors thank two anonymous referees for their helpful comments.

REFERENCES

[1] E. Cances, H. Galicher and M. Lewin, *Computing electronic structures: A new multiconfiguration approach for excited states*, J. Comput. Phys., 212 (2006), pp. 73–98.

[2] Y. S. Choi and P. J. McKenna, *A mountain pass method for the numerical solution of semilinear elliptic problems*, Nonlinear Anal., 20 (1993), pp. 417–437.

[3] M. Cuesta, *Eigenvalue problems for the p-Laplacian with indefinite weights*, Electronic J. Differential Equations, 2001 (2001), pp. 1–9.

[4] J. I. Diaz, *Nonlinear Partial Differential Equations and Free Boundaries, Vol.* I, *Elliptic Equations*, Res. Notes Math. 106, Pitman, Boston, 1985.

[5] Z. Ding, D. Costa, and G. Chen, *A high linking algorithm for sign-changing solutions of semilinear elliptic equations*, Nonlinear Anal., 38 (1999), pp. 151–172.
[6] R. J. Dodd, *Approximate solutions of the nonlinear Schrödinger equation for ground and excited states of Bose–Einstein condensates*, J. Res. Natl. Inst. Stand. Technol., 545 (1996), pp. 1–8.
[7] P. Drabek and S. B. Robinson, *Resonance Problems for the p-Laplacian*, J. Funct. Anal., 169 (1999), pp. 189–200.
[8] J. J. García-Ripoll and V. M. Pérez-García, *Optimizing Schrödinger functionals using Sobolev gradients: Applications to quantum mechanics and nonlinear optics*, SIAM J. Sci. Comput., 23 (2001), pp. 1316–1334.
[9] J. J. Garcia-Ripoll and V. M. Perez-Garcia, *Stable and unstable vortices in multicomponent Bose–Einstein condensates*, Phys. Rev. Lett., 84 (2000), pp. 4264–4267.
[10] E. P. Gross, *Structure of a quantized vortex in Boson systems*, Nuovo Cimento, 20 (1961), pp. 454–477.
[11] G. Henkelman, G. Johannesson, and H. Jonsson, *Methods for finding saddle points and minimum energy paths*, in Theoretical Chemistry and Physics, D. Schwartz, ed., vol. 5, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2000, pp. 269–300.
[12] http://www.ams.org/msnmain?fn=130&pg1=CC&op1=AND&s1=35B38&fmt=hl
[13] http://www.ams.org/msnmain?fn=130&pg1=CC&op1=AND&s1=58E05&fmt=hl
[14] http://www.ams.org/msnmain?fn=130&pg1=CC&op1=AND&s1=35Q55&fmt=hl
[15] S. V. Izbash, *O Filtracii V Kropnozernstom Materiale*, Izv. Nauchnoissled, Inst. Gidrotechniki (NIIG), Leningrad, USSR, 1931.
[16] C. E. Kenig, *The Cauchy Problems for the Quasilinear Schrodinger Equations*, Lecture notes to be published by American Mathematical Society.
[17] S. Komineas and N. Papanicolaou, *Static solitary waves in axisymmetric Bose–Einstein condensates*, Laser Phys., 14 (2004), pp. 571–574.
[18] P. Lindqvist, *On a Nonlinear Eigenvalue Problem*, *Fall School in Analysis Lectures*, Ber. Univ. Jyväskylä Math. Inst., 68 (1991), pp. 33–54.
[19] L. Lefton and D. Wei, *Numerical approximation of the first eigenpair of the p-Laplacian using finite elements and the penalty method*, Numer. Funct. Anal. Optim., 18 (1997), pp. 389–399.
[20] Y. Li and J. Zhou, *A minimax method for finding multiple critical points and its applications to semilinear PDEs*, SIAM J. Sci. Comput., 23 (2001), pp. 840–865.
[21] Y. Li and J. Zhou, *Convergence results of a local minimax method for finding multiple critical points*, SIAM J. Sci. Comput., 24 (2002), pp. 865–885.
[22] J.-Q. Liu, Y.-Q. Wang, and Z.-Q. Wang, *Soliton solutions for quasilinear Schrodinger equation*, II, J. Differential Equations, 187 (2003), pp. 473–493.
[23] S. Liu, *Existence of solutions to a superlinear p-Laplacian equation*, Electronic J. Differential Equations, 2001 (2001), pp. 1–6.
[24] E. Montefusco and V. Radulescu, *Nonlinear eigenvalue problems for quasilinear operators on unbounded domains*, Nonlinear Differential Equations Appl., 8 (2001), pp. 481–497.
[25] Z. H. Musslimani, M. Segev, D. N. Christodoulides, and M. Soljacic, *Composite multihump vector solitons carrying topological charge*, Phy. Rev. Lett., 84 (2000), pp. 1164–1167.
[26] J. M. Neuberger and J. Swift, *Newton's method and Morse index for semilinear elliptic PDEs*, Int. J. Bifur. Chaos, 11 (2001), pp. 802–820.
[27] L. P. Pitaevskii, *Vortex lines in an imperfect Bose gas*, Soviet Phys. JETP, 13 (1961), pp. 451–454.
[28] J. Schulz, M. Tchaplyguine, T. Rander, O. Bjrneholm, S. Svensson, R. Sankari, S. Heinsmki, H. Aksela, S. Aksela, and E. Kukk, *Shakedown in core photoelectron spectra from aligned laser-excited Na atoms*, Phys. Rev. A, 72 (2005), 010702.
[29] J. Schulz, M. Tchaplyguine, T. Rander, H. Bergersen, A. Lindblad, G. hrwall, S. Svensson, S. Heinsmki, R. Sankari, S. Osmekhin, S. Aksela, and H. Aksela, *Final state selection in the* 4*p photoemission of Rb by combining laser spectroscopy with soft-x-ray photoionization*, Phys. Rev. A, 72 (2005), 032718.
[30] J. Smoller, *Shock Waves and Reaction-Diffusion Equation*, Springer-Verlag, New York, 1983.
[31] S. I. Themelis, P. Lambropoulos, and F. J. Wuilleumier, *Laser-induced transitions between core excited states of Na*, J. Phys. B, 38 (2005), pp. 2119–2132.
[32] Z.-Q. Wang and J. Zhou, *A local minimax-Newton method for finding multiple saddle points with symmetries*, SIAM J. Numer. Anal., 42 (2004), pp. 1745–1759.
[33] Z.-Q. Wang and J. Zhou, *An efficient and stable method for computing multiple saddle points with symmetries*, SIAM J. Numer. Anal., 43 (2005), pp. 891–907.
[34] Z. Xie, C. Chen, and Y. Xu, *An improved search-extension method for computing multiple solutions of semilinear PDEs*, IMA J. Numer. Anal., 25 (2005), pp. 549–576.

[35] X. Yao and J. Zhou, *A minimax method for finding multiple critical points in Banach spaces and its application to quasi-linear elliptic PDE*, SIAM J. Sci. Comput., 26 (2005), pp. 1796–1809.
[36] X. Yao and J. Zhou, *Unified convergence results on a minimax algorithm for finding multiple critical points in Banach spaces*, SIAM J. Numer. Anal., to appear.
[37] X. Yao and J. Zhou, *Some Numerical Investigation of the p-Henon Equation*, preprint.
[38] X. Yao and J. Zhou, *Numerical Methods for Computing Nonlinear Eigenpairs: Part* II. *Non Iso-Homogeneous Cases*, submitted.
[39] E. Zeidler, *Nonlinear Functional Analysis and Its Applications* III, Springer-Verlag, New York, 1985.
[40] J. Zhou, *A local mini-orthogonal method for finding multiple saddle points*, J. Math Anal. Appl., 291 (2004), pp. 66–81.
[41] J. Zhou, *Instability analysis of saddle points by a local minimax method*, Math. Comp., 74 (2005), pp. 1391–1411.

SIAM J. SCI. COMPUT.
Vol. 29, No. 4, pp. 1375–1402
© 2007 Society for Industrial and Applied Mathematics

REGULARITY THEORY AND SUPERALGEBRAIC SOLVERS FOR WIRE ANTENNA PROBLEMS*

OSCAR P. BRUNO† AND MICHAEL C. HASLAM†

Abstract. We consider the problem of evaluating the current distribution $J(z)$ that is induced on a straight wire antenna by a time-harmonic incident electromagnetic field. The scope of this paper is twofold. One of its main contributions is a regularity proof for a straight wire occupying the interval $[-1, 1]$. In particular, for a smooth time-harmonic incident field this theorem implies that $J(z) = I(z)/\sqrt{1-z^2}$, where $I(z)$ is an *infinitely differentiable* function—the previous state of the art in this regard placed I in the Sobolev space $W^{1,p}$, $p > 1$. The second focus of this work is on numerics: we present three *superalgebraically convergent* algorithms for the solution of wire problems, two based on Hallén's integral equation and one based on the Pocklington integrodifferential equation. Both our proof and our algorithms are based on two main elements: (1) a new decomposition of the kernel of the form $G(z) = F_1(z)\ln|z| + F_2(z)$, where $F_1(z)$ and $F_2(z)$ are analytic functions on the real line; and (2) removal of the end-point square root singularities by means of a coordinate transformation. The Hallén- and Pocklington-based algorithms we propose converge superalgebraically: faster than $\mathcal{O}(N^{-m})$ and $\mathcal{O}(M^{-m})$ for any positive integer m, where N and M are the numbers of unknowns and the number of integration points required for construction of the discretized operator, respectively. In previous studies, at most the leading-order contribution to the logarithmic singular term was extracted from the kernel and treated analytically, the higher-order singular derivatives were left untreated, and the resulting integration methods for the kernel exhibit $\mathcal{O}(M^{-3})$ convergence at best. A rather comprehensive set of tests we consider shows that, in many cases, to achieve a given accuracy, the numbers N of unknowns required by our codes are up to a factor of five times smaller than those required by the best solvers previously available; the required number M of integration points, in turn, can be several orders of magnitude smaller than those required in previous methods. In particular, four-digit solutions were found in computational times of the order of four seconds and, in most cases, of the order of a fraction of a second on a contemporary personal computer; much higher accuracies result in very small additional computing times.

Key words. electromagnetic scattering, wire antenna, Pocklington, Hallén

AMS subject classifications. 65-02, 78-02

DOI. 10.1137/050648262

1. Introduction.

The problem of evaluating the current distribution that is induced on a thin straight wire antenna by a given time-harmonic incident electromagnetic field was first described by Pocklington [10] more than a century ago; its computation remains an important and difficult engineering problem to this day. Once such currents have been calculated, the fields scattered by the wire can easily be produced by integration. The current $J(z)$ induced on a thin wire of radius a by an incident electromagnetic field with wave number k and axial component $e(z)$ can be obtained as a solution of the Pocklington equation

$$\left(\frac{\partial^2}{\partial z^2} + k^2\right) \int_{-1}^{1} G(z-t)\, J(t)\, dt = -4\pi i k e(z) \tag{1.1}$$

*Received by the editors December 22, 2005; accepted for publication (in revised form) December 27, 2006; published electronically June 12, 2007. This work was supported in part by the Air Force Office of Scientific Research, the National Science Foundation, and the National Aeronautics and Space Administration.

http://www.siam.org/journals/sisc/29-4/64826.html

†Applied and Computational Mathematics, California Institute of Technology, Pasadena, CA 91125 (bruno@acm.caltech.edu, mchaslam@acm.caltech.edu). The second author was supported by the Natural Sciences and Engineering Research Council of Canada.

subject to the end-point conditions

$$J(-1) = J(1) = 0. \tag{1.2}$$

An alternative (and, in exact arithmetic, equivalent) formulation for this problem is given by the Hallén equation, which results from (1.1) by inverting the Helmholtz operator. The Hallén equation thus reads

$$\int_{-1}^{1} G(z-t)\, J(t)\, dt = \alpha_1 \cos kz + \alpha_2 \sin kz - 4\pi i \int_{-1}^{z} e(t) \sin[k(z-t)] dt, \tag{1.3}$$

where α_1 and α_2 are constants which are chosen so that the current distribution satisfies the end-point conditions (1.2). The kernel $G(z)$ in these equations, which is given by

$$G(z) = \frac{1}{\pi} \int_0^{\pi} \frac{\mathrm{e}^{ik\sqrt{z^2+4a^2\sin^2\psi}}}{\sqrt{z^2+4a^2\sin^2\psi}}\, d\psi, \tag{1.4}$$

has a logarithmic singularity at $z = 0$; see section 2.

The scope of this paper is twofold. One of its main contributions is a regularity proof: expressing the current in the form $J(z) = I(z)/\sqrt{1-z^2}$ (which has been known since the work by Jones [7] and Rynne [13]), our result relates the regularity of the function $I(z)$ to that of the incident field. In particular, for an infinitely differentiable incident field our theorem implies that the "reduced current" $I(z)$ is infinitely differentiable as well—the previous state of the art in this regard placed I in the Sobolev space $W^{1,p}$, $p > 1$. The second focus of this work is on numerics: we present three superalgebraically convergent algorithms for the solution of wire scattering problems, two based on Hallén's integral equation and one based on the Pocklington integrodifferential equation. To achieve superalgebraic convergence, these algorithms rely on a number of techniques which address specifically the difficulties posed by wire scattering problems.

Some of the concepts we present in this paper are used in both our theoretical and our numerical treatments; others are specifically designed to tackle numerical issues presented by the problem at hand. In particular, to both establish our regularity result and produce algorithms of high-order accuracy for this problem, we do the following:

1. Introduce a new decomposition of the kernel, of the form

   $$G(z) = F_1(z) \ln|z| + F_2(z), \tag{1.5}$$

   where $F_1(z)$ and $F_2(z)$ are analytic functions on the real line. Albeit analytic, the functions F_i exhibit a rather interesting "nearly singular" behavior.
2. Utilize a coordinate transformation of the form $z = \cos(\theta)$. This transformation, which is essential for our theoretical discussion (section 3), is also quite useful from a numerical standpoint, since it simultaneously regularizes the integration problem and leads to a Chebyshev representation of the reduced current I.

To obtain accurate and efficient algorithms we introduce additional numerical strategies. In particular:

3. For very thin wires we use the transformation $z = 2\cos w/(1+\cos^2 w)$ instead of the one mentioned in point 2 above. The $w \to z$ transformation results

in significantly improved performance for very thin wires (see points (a), (b), and (c) in section 5.1): instead of the quadratic z-clustering introduced at $z = \pm 1$ by the $\theta \to z$ transformation, the $w \to z$ change of variables introduces quartic point clustering around the end-points. This additional refinement resolves a near singularity that occurs in the reduced current I for a very close to zero.

4. To produce well-conditioned linear systems we discretize our integral equations by sampling them at z points corresponding to equispaced θ or w grids, depending on whether the $\theta \to z$ or the $w \to z$ transformation is used. A sampling of the integral operators at regular z grids gives rise to highly ill-conditioned linear algebra problems.
5. The integration algorithm itself, on the other hand, utilizes (i) a localized integration scheme based on use of fine and coarse Chebyshev grids for integration of highly peaked integrands that arise from corresponding (smooth) peaks in the functions F_1 and F_2, as well as (ii) a new weighted-quadrature formula for log-Chebyshev integrals (see Appendix B).
6. For efficiency, the kernel evaluations are based on a new asymptotic formula (equation (2.20)) which we use for sufficiently large values of a certain dimensionless version ρ of the spatial variable, together with numerical procedures based on expansions and integral representations for the rapid and accurate evaluation of the newly introduced functions F_1 and F_2.

Having accounted to all orders for all integrand singularities and near-singularities and in view of our regularity result, we find that the Hallén- and Pocklington-based algorithms we propose converge superalgebraically: faster than $\mathcal{O}(N^{-m})$ and $\mathcal{O}(M^{-m})$ for any positive integer m, where N and M are the numbers of unknowns and the number of integration points required for construction of the discretized operator (i.e., the linear system's matrix), respectively. Previous solution techniques for this problem, in contrast, have not taken into account the precise nature of the kernel's singularity—which, to our knowledge, had not been established before the present work. In many contributions, furthermore, the end-point singularities of the current distribution were not fully accounted for. We show, albeit not straightforward, that it is very advantageous to account for all singularities to high order: as we show in a variety of examples in section 5, in many cases, to achieve a given accuracy, the numbers N of unknowns required by our codes are as much as five times smaller than those required by the best solvers previously available. The required number M of integration points, in turn, can be several orders of magnitude smaller than those required by previous methods; see paragraph (d) below.

A few comments are in order with regards to the kernel mentioned above and its relationship to various expressions available in the literature:

(a) The normalization of the kernel (1.4) coincides with that of Jones [7] and Rynne [13]; the normalization employed by Davies, Duncan, and Funken [4], on the other hand, includes an additional multiple of $1/(4\pi)$, which, here, is incorporated on the right-hand side of (1.1) instead.

(b) Another kernel for the equations above, which results from use of approximations additional to those implicit in (1.1), (1.3), and (1.4), is the so-called reduced kernel

$$G_{red}(z) = \frac{\mathrm{e}^{ik\sqrt{z^2+a^2}}}{\sqrt{z^2+a^2}}. \tag{1.6}$$

We will not utilize this kernel: as pointed out in [4] and [5], it is not possible

to obtain reliable results using the reduced kernel $G_{red}(z)$ in place of $G(z)$ in (1.1) or (1.3).

(c) In a concise paper [16], Wang derives the expansion

$$G(z) = \frac{e^{ik\sqrt{z^2+2a^2}}}{\sqrt{z^2+2a^2}} + ik \sum_{m=1}^{\infty} \frac{(ka)^{4m}(2m-1)!!}{(2m)!(2m)!!} \frac{h_{2m}^{(1)}\left(k\sqrt{z^2+2a^2}\right)}{k^{2m}(z^2+2a^2)^m} \tag{1.7}$$

for the exact straight wire kernel, where $h_m^{(1)}(z)$ is a spherical Hankel function of the first kind [1]. This expansion is particularly useful for values of z far away from the origin, for which the series in (1.7) converges rapidly. Unfortunately, this expansion does not seem as useful for values of z close to $z = 0$, since the asymptotic behavior of the kernel in this important region cannot be easily produced from (1.7).

(d) It has been known for some time that the kernel (1.4) has the form

$$G(z) = -\frac{1}{a\pi} \ln|z| + G_1(z), \tag{1.8}$$

where the function $G_1(z)$ is continuous, it has a bounded derivative and an unbounded second derivative; see [7, p. 115], [13], or section 2.2 below. The logarithmic behavior of the function $G(z)$ can be easily grasped by consideration of the expression

$$G(z) = \frac{2}{\pi\sqrt{z^2+4a^2}} K\left(\frac{2a}{\sqrt{z^2+4a^2}}\right) - \frac{1}{\pi} \int_0^{\pi} \frac{1 - e^{ik\sqrt{z^2+4a^2\sin^2\psi}}}{\sqrt{z^2+4a^2\sin^2\psi}} d\psi, \tag{1.9}$$

where K, the complete elliptic integral of the first kind [3], contains the leading-order singularity. Equations (1.8) and (1.9) have been used as the basis for many numerical solvers for Pocklington and Hallén problems [4, 9]. Numerical integration schemes for integrands containing such singularities which, based on polynomial interpolation, do not explicitly account for the singularity of the second derivative of the integrand typically exhibit low-order convergence. Indeed, the leading singular term in $G_1(z)$ is of the form $z^2 \ln|z|$ [9]. The error that results as an nth degree Newton–Cotes scheme (e.g., Simpson's rule for $n = 2$) is used to integrate such a function is of the order of $\mathcal{O}(h^3)$ for $n \geq 2$, where h is the integration mesh size. The decomposition (1.5) we introduce extends the result embodied in (1.8) and allows for superalgebraic evaluation of the kernel and its integrals; see section 2.3 and Appendix B, respectively. In particular, efficient algorithms for the evaluation of the functions F_1 and F_2 can be found in section 2.3.

This paper is organized as follows: After introducing the new decomposition for the exact kernel in section 2, we discuss in section 3 the regularity properties of the current $J(z)$. Our approach to the discretization of the Pocklington and Hallén integral equations is then described in section 4 and Appendix B. A variety of numerical results and conclusions, finally, are presented in section 5.

2. New decomposition for the exact kernel. In this section we present a derivation of the decomposition (1.5) together with explicit expressions and algorithms for fast and accurate evaluation of the analytic functions F_1 and F_2.

2.1. Taylor expansion of the integrand's exponential. Substituting the exponential in (1.4) by its Taylor expansion and interchanging the order of summation and integration, we obtain

$$G(z) = g(\rho)/a, \quad \rho = z/(2a), \tag{2.1}$$

where, defining

$$B_m(\rho) = \int_0^\pi \left(\rho^2 + \sin^2\psi\right)^{m/2} d\psi, \tag{2.2}$$

$$\begin{aligned} g_r(\rho) &= \frac{1}{2\pi}\sum_{n=0}^{\infty} \frac{(-1)^n (2ka)^{2n}}{(2n)!} B_{2n-1}(\rho) \quad \text{and} \\ g_i(\rho) &= \frac{1}{2\pi}\sum_{n=0}^{\infty} \frac{(-1)^n (2ka)^{2n+1}}{(2n+1)!} B_{2n}(\rho), \end{aligned} \tag{2.3}$$

the dimensionless function g is given by

$$g(\rho) = g_r(\rho) + i g_i(\rho). \tag{2.4}$$

For $m = 2n$ (n a nonnegative integer), the integrals in (2.2) define a set of polynomials. In particular, calling

$$R = \sqrt{\rho^2 + 1}, \tag{2.5}$$

we have the expression

$$B_{2n}(\rho) = \pi \sum_{m=0}^{n} \frac{(-1)^m}{2^{2m}} \begin{pmatrix} n \\ m \end{pmatrix} \begin{pmatrix} 2m \\ m \end{pmatrix} R^{2n-2m}, \tag{2.6}$$

which follows from the hypergeometric function representation of the integral in (2.2) [14, eqn. IV(2)] and the series expansion for these functions [1, eqn. 15.4.1]. Using (2.6) and reversing the order of summation in the second equation of (2.3) we obtain an exponentially convergent series for the imaginary part of the kernel: denoting by $j_n(z)$ the spherical Bessel function of the first kind [1], our expansion reads

$$g_i(\rho) = ka \sum_{n=0}^{\infty} \frac{1}{2^{2n} n!} \begin{pmatrix} 2n \\ n \end{pmatrix} \left(\frac{ka}{R}\right)^n j_n(2kaR)\,. \tag{2.7}$$

A corresponding closed-form result can be obtained for the real part of the kernel as well: for the values $m = 2n - 1$ (n a nonnegative integer) relevant in this case, the integrals in (2.2) are related to the associated Legendre functions of the second kind, $Q_\nu^\mu(z)$, of degree ν and order μ. It follows directly from the integral representation for the functions $Q^\mu_{m-1/2}(z)$ [14, eqn. VI(73c)] that

$$B_{2n-1}(\rho) = \frac{2}{\sqrt{\pi}}\,\Gamma(1/2 - n)(|\rho|R)^n Q^n_{-1/2}(2\rho^2 + 1). \tag{2.8}$$

2.2. Decomposition of the real part of the kernel. To make explicit the singular nature of the kernel $g_r(\rho)$, we use (2.3) and (2.8) together with known formulae for the Legendre functions of integer order $Q^n_\nu(z)$. In particular, from [14, eqn. VI(46')] and using the notation (2.5), we obtain

(2.9)
$$Q^n_{-1/2}(2\rho^2+1) = -P^n_{-1/2}(2\rho^2+1)\ln\left(\frac{|\rho|}{R}\right) + \frac{\sqrt{\pi}}{2R}\left(\frac{|\rho|}{R}\right)^n \Gamma(n+1/2)F^n_{-1/2}(2\rho^2+1),$$

where $P^\mu_\nu(z)$ is the Legendre function of the first kind of degree ν and order μ. Using a slightly different notation from that of [14] we have set

(2.10)
$$F^n_{-1/2}(2\rho^2+1) = \frac{2}{\pi}(-1)^n\left[\sum_{m=1}^{n}\frac{1}{2m}\frac{\phi_{n-m}}{\phi_m}\left(\frac{R}{\rho}\right)^{2m} + \frac{1}{2}\sum_{m=0}^{\infty}(\beta_m+\beta_{m+n})\phi_m\phi_{m+n}\left(\frac{\rho}{R}\right)^{2m}\right],$$

where

(2.11)
$$\phi_\ell = \frac{1}{2^{2\ell}}\begin{pmatrix} 2\ell \\ \ell \end{pmatrix} = \frac{1\cdot 3\cdot 5\cdots(2\ell-1)}{2\cdot 4\cdot 6\cdots(2\ell)}, \quad \ell \geq 1$$

(with $\phi_0 = 1$), and where, using the digamma function Ψ [1], $\beta_\ell = \Psi(\ell+1)-\Psi(\ell+1/2)$.

In view of (2.9), the real part of the kernel may be expressed in the form

(2.12)
$$g_r(\rho) = -\frac{1}{\pi}\,g_{r1}(\rho)\ln\left(\frac{|\rho|}{R}\right) + g_{r2}(\rho),$$

where the functions g_{r1} and g_{r2} are given by

(2.13)
$$g_{r1}(\rho) = \frac{1}{\sqrt{\pi}}\sum_{n=0}^{\infty}\frac{(-1)^n(2ka)^{2n}}{(2n)!}\,\Gamma(1/2-n)(|\rho|R)^n P^n_{-1/2}(2\rho^2+1)$$

and

(2.14)
$$g_{r2}(\rho) = \frac{1}{2R}\sum_{n=0}^{\infty}\frac{(2ka\rho)^{2n}}{(2n)!}\,F^n_{-1/2}(2\rho^2+1);$$

as shown in what follows, these functions are analytic for real values $|\rho| < \infty$.

The expression (2.13) for g_{r1} may be simplified considerably. To do this we use the integral representation

(2.15)
$$P^n_{-1/2}(1+2\rho^2) = \frac{1}{\pi^{3/2}}(-1)^n\Gamma(n+1/2)\left(\frac{|\rho|}{R}\right)^n\int_0^\pi \frac{\sin^{2n}\psi\, d\psi}{\sqrt{1+\rho^2\cos^2\psi}},$$

which follows from a corresponding integral representation for the hypergeometric function [14, eqn. IV(2)] together with a known expression [14, eqn. VI(45)] for the Legendre functions in terms of the hypergeometric function. Substituting equation (2.15) in (2.13) and reversing the order of summation and integration yields the simple expression

(2.16)
$$g_{r1}(\rho) = \frac{1}{\pi}\int_0^\pi \frac{\cos(2ka\rho\sin\psi)d\psi}{\sqrt{1+\rho^2\cos^2\psi}}.$$

In addition to showing that $g_{r1}(\rho)$ is, indeed, an analytic function for real values of ρ, this formula gives rise to an efficient method for the evaluation of this function in a neighborhood of the origin.

To show that g_{r2} (equation (2.14)) is analytic, in turn, we establish appropriate upper bounds on the growth with respect to n of each one of the two sums in (2.10). To this end we first note from (2.11) that (i) $0 < \phi_\ell \le 1$ for $\ell = 0, 1, 2, \ldots$, and (ii) $\phi_\ell \ge 1/(2\ell)$ for $\ell = 1, 2, 3, \ldots$. Hence

$$\frac{2}{\pi}\rho^{2n}\sum_{m=1}^{n}\frac{1}{2m}\frac{\phi_{n-m}}{\phi_m}\left(\frac{R}{\rho}\right)^{2m} \le \begin{cases} 2nR^{2n}/\pi, & |\rho| \le 1, \\ 2n\rho^{2n}R^{2n}/\pi, & |\rho| > 1. \end{cases} \tag{2.17}$$

In order to obtain a bound on the infinite series in (2.10), in turn, we recall [1] that the digamma function Ψ is monotonically increasing for real $z > 0$. It follows that $\Psi(z+1) - \Psi(z+1/2) \le \Psi(z+1) - \Psi(z) = 1/z$ [1, eqn. 6.3.5], and thus $0 < \beta_m \le 1/m$ for $m \ge 1$. Since $\phi_{m+n} \le \phi_m$ and $\beta_{m+n} \le \beta_m$ for all nonnegative integers m and n, and since $\phi_0 = 1$ and $\beta_0 = 2\ln 2$ [1, eqns. 6.3.2, 6.3.3], for all real ρ we obtain the inequality

$$\sum_{m=0}^{\infty}(\beta_m + \beta_{m+n})\phi_m\phi_{m+n}\left(\frac{\rho}{R}\right)^{2m} \le 2\sum_{m=0}^{\infty}\beta_m\phi_m \le 4\ln 2 + 2\sum_{m=1}^{\infty}\frac{\phi_m}{m}. \tag{2.18}$$

It is easy to check, further, that $0 < \phi_m < C/m^{1/2}$ for some positive constant C, so that the series on the right-hand side of (2.18) is convergent, and thus the series on the left-hand side of (2.18), which is a positive quantity, is bounded above by a positive constant independent of n. (To check that $0 < \phi_m < C/m^{1/2}$ for some constant C note that $\ln(\phi_m) = \sum_{\ell=1}^{m}\ln(1 - 1/2\ell)$, use the relation $\ln(1-x) = -x + \mathcal{O}(x^2)$, and, using an integral, estimate $\sum_{\ell=1}^{m} 1/\ell \ge \ln(m)$.) The analyticity of $g_{r2}(\rho)$ for all real $|\rho| < \infty$ then follows as a result of the rapid convergence of the series (2.14).

From the dimensional and dimensionless forms considered above for the kernel function, equations (1.5), (2.4), (2.1), and (2.12), we can write $F_1(z) = f_1(\rho)/a$ and $F_2(z) = f_2(\rho)/a$, where the dimensionless functions f_1 and f_2 are given by

$$f_1(\rho) \equiv -\frac{1}{\pi}\,g_{r1}(\rho) \qquad \text{and} \qquad f_2(\rho) \equiv \frac{1}{\pi}\,g_{r1}(\rho)\ln(2aR) + g_{r2}(\rho) + ig_i(\rho). \tag{2.19}$$

In view of the rapid convergence of the series (2.7) it can be easily established that the function g_i is analytic. Thus, the analyticity of f_1 and f_2, and therefore that of F_1 and F_2, follow from the analyticity results given above for g_{r1} and g_{r2}.

2.3. Evaluation of the thin-wire kernel. As mentioned above, our algorithms for the solution of the Hallén and Pocklington problems rely on the new decomposition (1.5) of the integral kernel. More precisely, our algorithms require explicit, fast, and accurate evaluation of the analytic functions $F_1(z)$ and $F_2(z)$ for values of z close to zero (to allow for explicit treatment to arbitrarily high order of the logarithmic singularity) as well as highly accurate evaluation of the full kernel $G(z)$ for values of z away from the origin. These problems involve a degree of subtlety; the algorithms we devised for their solution are based on a number of representations for these functions and consideration of a number of subcases, as discussed in what follows.

Our numerical solvers require values of the kernel $G(z) = g(\rho)/a$ for all $z \in [-2, 2]$ ($\rho \in [-1/a, 1/a]$) and values of the functions $F_1(z)$ and $F_2(z)$ in subintervals of $[-1/a, 1/a]$ near the origin that are symmetric with respect to the origin. It is clear

that, by symmetry, however, it suffices to evaluate the functions $F_1(z)$ and $F_2(z)$ and the kernel $G(z)$ for points $\rho \geq 0$. For accuracy and efficiency we use two different algorithms within this domain: one for $\rho \in [0, \Delta]$, near the kernel's singularity, and a different one for $\rho \in (\Delta, 1/a]$, away from the singularity. The splitting parameter Δ is a suitably chosen $\mathcal{O}(1)$ constant; based on numerical experimentation we have selected the value $\Delta = 5$, which was used in finding all of the numerical results presented in this paper. In some cases we use separate treatments for evaluation of the real and imaginary parts g_r and g_i of the kernel; see (2.4). Once these functions have been obtained, the corresponding dimensional forms in (1.5) may then be produced as indicated at the end of section 2.2.

[illegible] $g_r(\rho)$ [illegible] $g_i(\rho)$ [illegible] $\rho \in (\Delta, 1/a]$. In this region our integral equation solver does not require separate evaluation of the functions $g_{r1}(\rho)$ and $g_{r2}(\rho)$. Both real and imaginary parts of the kernel are computed by means of the expansion

$$g(\rho) = \frac{\mathrm{e}^{ika\tilde{R}}}{\tilde{R}} \left\{ \sum_{n=0}^{M} \frac{(-1)^n \mathcal{A}_n[(ka)^2]}{2^{2n}(n!)^2 \tilde{R}^{2n}} + i(ka) \sum_{n=1}^{M} \frac{(-1)^n n(2n+1)\mathcal{B}_{n-1}[(ka)^2]}{2^{2n}(n!)^2 \tilde{R}^{2n+1}} + \mathcal{O}\left(\frac{1}{\tilde{R}^{2M+2}} \right) \right\} \tag{2.20}$$

($\tilde{R} = \sqrt{4\rho^2 + 2}$), which, as $M \to \infty$, converges to $g(\rho)$ for every real value of ρ. This expansion results as the finite-sum expression [1, eqn. 10.1.16] for the spherical Hankel function is substituted into Wang's expression (1.7) (thus avoiding evaluation of Hankel functions) and, subsequently, like powers of $\tilde{R}$ are collected. The coefficients $\mathcal{A}_n(x)$ and $\mathcal{B}_n(x)$ in (2.20) are polynomials of degree n; in Appendix A we present all of the polynomials necessary to evaluate the expansion (2.20) up to $M = 10$. With $M = 10$ this expansion yields at least double precision accuracy for all $ka \leq 1$ when $\rho \geq 3$—certainly sufficient for our purposes, given our choice $\Delta = 5$ of the splitting parameter.

[illegible] $g_i(\rho)$ [illegible] $\rho \in [0, \Delta]$. In this domain the imaginary part of the kernel $g_i(\rho)$ is computed using the rapidly convergent expansion (2.7).

[illegible] $g_r(\rho)$ [illegible] $\rho \in [0, \Delta]$. In the region $\rho \in [0, \Delta]$ our approach calls for separate evaluation of the analytic functions $g_{r1}(\rho)$ and $g_{r2}(\rho)$ of (2.12)—to allow for explicit treatment to arbitrarily high order of the logarithmic singularity.

(a) [illegible] $g_{r1}(\rho)$ [illegible] $\rho \in [0, \Delta]$. Since the integrand in (2.16) is analytic and periodic on the domain of integration, numerical evaluation of the integral by means of the trapezoidal rule gives rise to an exponentially convergent algorithm for evaluation of g_{r1}. Naturally, for large values of ρ this procedure would produce significant cancellation errors. For the limited set $[0, \Delta]$ of values of ρ considered here, trapezoidal-rule integration with sufficiently many (rather few!) points yields full double precision accuracy. For efficiency, the function $g_{r1}(\rho)$ is not computed by this method at every point where it is needed; instead this approach is used to precompute values of f_1 on a Chebyshev mesh within $[0, \Delta]$. These values are then used to produce a Chebyshev expansion [8, 12] of g_{r1}—which can, in turn, be utilized to produce values of g_{r1} as needed.

(b) [illegible] $g_{r2}(\rho)$ [illegible] $\rho \in [0, \Delta]$. To evaluate the function $g_{r2}(\rho)$ in this region, we introduce two subpartitions, $\rho \in [0, \delta]$ and $\rho \in (\delta, \Delta]$. The parameter δ is a suitably chosen small constant; based on our experiments we have selected the value $\delta = 1/10$, which was used to find all of the numerical

results presented in this paper. In the interval $(\delta, \Delta]$, g_{r2} can be evaluated, quite simply, by means of the expressions

$$g_{r2}(\rho) = g_r(\rho) + \frac{1}{\pi}\, g_{r1}(\rho) \ln\left(\frac{|\rho|}{R}\right) \tag{2.21}$$

and

$$g_r(\rho) = \frac{1}{2\pi} \int_0^\pi \frac{\cos(2ka\sqrt{\rho^2 + \sin^2\psi})d\psi}{\sqrt{\rho^2 + \sin^2\psi}}. \tag{2.22}$$

As in the case of the integral expression used for evaluation of the function g_{r1}, the integrand in (2.22) is analytic (for real $\rho > 0$) and periodic on the domain of integration, so that the trapezoidal rule converges exponentially fast. On the other hand, the computation of $g_{r2}(\rho)$ in the interval $\rho \in [0, \delta]$ near the singular point is accomplished by direct summation of the series (2.14). The efficiency of this process is greatly improved by a simple observation: since both $P_\nu^\mu(z)$ and $Q_\nu^\mu(z)$ satisfy the recurrence relation [14, eqn. VI(25a)], it follows that so does the second term on the right-hand side of (2.9). It thus may be shown that the function $\Xi_n(\rho) = \rho^{2n} F^n_{-1/2}(2\rho^2 + 1)$ satisfies

$$\Xi_{n+1}(\rho) = -\frac{2n}{2n+1}\,(1 + 2\rho^2)\Xi_n(\rho) - \frac{2n-1}{2n+1}\,\rho^2(1+\rho^2)\Xi_{n-1}(\rho). \tag{2.23}$$

The necessary initial values $\Xi_0(\rho)$ and $\Xi_1(\rho)$ are obtained by the direct evaluation of (2.10), whose coefficients can themselves be generated recursively using the relations $\Gamma(z+1) = z\Gamma(z)$ and $\Psi(z+1) = \Psi(z) + 1/z$ [1]. Using (2.23) the sequence of functions $\Xi_n(\rho)$ can be evaluated accurately and efficiently; clearly knowledge of these functions suffices to evaluate any truncation of the series on the right-hand side of (2.14). The function g_{r2} is precomputed on a Chebyshev mesh within $[0, \Delta]$ by the methods described in the present point (b); these values are then used to produce the Chebyshev expansion of g_{r2}, to accompany the analogous expansion for g_{r1}, discussed in point (a) above.

3. Regularity of solutions. It is well known that the solution J of (1.1) and (1.3) [illegible] (1.2) tends to zero like $\sqrt{1-z^2}$ as $z \to \pm 1$ [4, 7, 13]. Jones [7] established that if $e(z)$ has a bounded derivative, then, under the conditions (1.2), equations (1.1) and (1.3) admit unique integrable solutions $J(z)$. Rynne [13] showed that an appropriate representation of the unknown current distribution is

$$J(z) = \frac{I(z)}{\sqrt{1-z^2}}\,, \quad z \in [-1, 1], \tag{3.1}$$

where the function $I(z)$, which we refer to as the reduced current, is in the Sobolev space $W^{1,p}$ (with $p > 1$), provided the right-hand side of the Hallén equation lies in $W^{2,p}$. The end-point conditions for $I(z)$, which result from the corresponding conditions (1.2) for $J(z)$, are given by

$$I(-1) = I(1) = 0. \tag{3.2}$$

Since, as established below, $I \in C^1[-1, 1]$, provided the function e on right-hand side of (1.1) is sufficiently smooth, it follows that the current J vanishes at the end-points like a square root, as mentioned above.

In this section, defining the reduced Hallén operator by

$$\mathcal{H}\{I\}(z) = \int_{-1}^{1} \frac{G(z-t)I(t)}{\sqrt{1-t^2}}\, dt\,, \quad z \in [-1,1], \tag{3.3}$$

we expand on the work of Rynne mentioned above and further relate the smoothness of a solution of the equation

$$\mathcal{H}\{I\} = h \tag{3.4}$$

to the smoothness of the right-hand side h. Our regularity proof is made possible by the kernel decompositions presented in section 2, together with a result concerning regularity of solutions of Symm's integral equation [15]. The actual result we use on Symm's equation, which is related to a theorem that can be found in [17], does not require an assumption made in that paper on the value of the logarithmic capacity of the interval of definition. Therefore, prior to the main discussion of this section on regularity of solutions of (3.4), which concludes with Theorem 3.7, we provide a brief discussion concerning Symm's equation.

Following [17], we call $C_e^k(2\pi)$ (resp., $C_e^\infty(2\pi)$) the set of all functions v defined on the real line that are [illegible], 2π [illegible], and k times continuously differentiable (resp., infinitely differentiable). For a function $v \in C_e^\infty(2\pi)$ and $s \geq 0$, further, we define the norm $\|\cdot\|_s$ by

$$\|v\|_s^2 = |a_0|^2 + 2\sum_{m=1}^{\infty} m^{2s}|a_m|^2,$$

where a_m denotes the mth cosine coefficient of v:

$$v(\theta) = \frac{1}{2}\, a_0 + \sum_{m=1}^{\infty} a_m \cos(m\theta).$$

The Sobolev space $H_e^s(2\pi)$ of even and 2π-periodic functions, finally, is defined as the completion of $C_e^\infty(2\pi)$ with respect to the norm $\|\cdot\|_s$. The ordinary Sobolev spaces $H^s[a,b]$ will also be used in what follows (see, e.g., [2]), as will the spaces $C^k[a,b]$, $k = 0, 1, \ldots$ or $k = \infty$, of k-times continuously differentiable functions on the interval $[a,b]$.

The 2π-periodic even functions mentioned above arise in our context mainly through composition of various functions defined in the interval $[-1,1]$ with the function $\cos(\theta)$; since such compositions will occur frequently in what follows we introduce a special notation for them: for a function $g : [-1,1] \to \mathbb{R}$ we will write

$$\tilde{g}(\theta) = g(\cos(\theta)). \tag{3.5}$$

Similarly, for an operator $\mathcal{K}$ whose codomain is a space of functions defined in the interval $[-1,1]$, we will denote by $\tilde{\mathcal{K}}$ the operator defined by

$$\tilde{\mathcal{K}}\{I\}(\theta) = \mathcal{K}\{I\}(\cos(\theta)). \tag{3.6}$$

Finally, in what follows $[x]$ denotes the integer part of x. With these notations, we have the following lemma.

LEMMA 3.1. $s \geq 0$ h_0 $[-1,1]$ $\tilde{h}_0 \in H_e^{s+1}(2\pi)$ $\tilde{h}_0(\theta) = h_0(\cos(\theta))$ $I \in C^0[-1,1]$

$$h_0(z) = \int_{-1}^{1} \ln|z-t| \, \frac{I(t)}{\sqrt{1-t^2}} \, dt, \quad z \in [-1,1], \tag{3.7}$$

(a) $\tilde{I} \in H_e^s(2\pi)$ $h_0 \in C^\infty[-1,1]$ $\tilde{I} \in C_e^\infty(2\pi)$
(b) k $s > k + 1/2$ $\tilde{I} \in C^k(2\pi)$ $I \in C^{[k/2]}[-1,1]$ $h_0 \in C^\infty[-1,1]$ $I \in C^\infty[-1,1]$

. The changes of variables $t = \cos\theta$, $z = \cos\alpha$ transform (3.7) into

$$\tilde{h}_0(\alpha) = \int_0^\pi \ln|\cos\alpha - \cos\theta| \tilde{I}(\theta) d\theta \, , \quad \alpha \in \mathbb{R}. \tag{3.8}$$

Since $\tilde{I} \in H_e^0(2\pi)$ and $\tilde{h}_0 \in H_e^0(2\pi)$ we have the H_e^0-convergent cosine expansions

$$\tilde{I}(\theta) = \frac{1}{2} b_0 + \sum_{m=1}^{\infty} b_m \cos(m\theta)$$

and

$$\tilde{h}_0(\alpha) = \frac{1}{2} c_0 + \sum_{m=1}^{\infty} c_m \cos(m\alpha)$$

for these functions. As is well known, in view of (3.8) we have

$$b_m = -\frac{1}{\pi} m c_m \quad \text{for } m \geq 1;$$

see, e.g., [11, 17] or [8, p. 211]. Thus, since $\tilde{h}_0 \in H_e^{s+1}(2\pi)$, we see that $\tilde{I} \in H_e^s(2\pi)$ and point (a) follows. The fact that $\tilde{I} \in C^k(2\pi)$ for $s > k + 1/2$, on the other hand, follows directly from point (a) and Sobolev's lemma. Since $\tilde{I}$ is even and 2π-periodic, finally, it is easy to check that $\tilde{I} \in C^k(2\pi)$ implies that $I(t) = \tilde{I}(\cos^{-1}(t))$ is a $[k/2]$-times continuously differentiable function, and the proof of the lemma is thus complete. □

REMARK 3.2. $\tilde{h}_0$ h_0 $h_0(z) = \sqrt{1-z^2}$ $\tilde{h}_0 \in H_e^1(2\pi)$ $h_0 \notin H_e^1(2\pi)$ $\tilde{h}_0$ h_0 3.8

Continuing with the preliminaries for our main proof, we define the operator

$$\mathcal{T}\{I\}(z) = \int_{-1}^{1} \frac{F_1(z-t)I(t)}{\sqrt{1-t^2}} \ln|z-t| dt = \mathcal{H}\{I\}(z) - \int_{-1}^{1} \frac{F_2(z-t)I(t)}{\sqrt{1-t^2}} \, dt, \tag{3.9}$$

where F_1 and F_2 are the analytic functions in (1.5), and where z, as in many analogous situations throughout this section, lies in the interval $[-1,1]$; $z \in [-1,1]$ Since F_2 is analytic (section 2.2) and since $J = I/\sqrt{1-z^2}$ is integrable it

follows that the last integral in (3.9) is infinitely differentiable. Thus, taking into account our conventions (3.5)–(3.6), we note the following.

REMARK 3.3. [illegible] (3.4) [illegible] $\tilde{h} \in H_e^{s+1}(2\pi)$ [illegible] $\tilde{\mathcal{T}}\{I\} \in H_e^{s+1}(2\pi)$

To establish our regularity result for $I(z)$, let

$$F_{1,2m}(z) = \sum_{k=0}^{m} \frac{F_1^{(2k)}(0)}{(2k)!}\, z^{2k} \tag{3.10}$$

denote the truncated Taylor series of F_1 (it is easy to verify from (2.16) and (2.19) that all of the odd derivatives of F_1 vanish at $z = 0$). Further, define the operators

$$\mathcal{S}_{2m}\{I\}(z) = \int_{-1}^{1} \frac{F_{1,2m}(z-t)I(t)}{\sqrt{1-t^2}}\, \ln|z-t|dt \tag{3.11}$$

and

$$\mathcal{R}_{2m}\{I\}(z) = \int_{-1}^{1} \frac{[F_1(z-t) - F_{1,2m}(z-t)]\, I(t)}{\sqrt{1-t^2}}\, \ln|z-t|dt \tag{3.12}$$

so that

$$\mathcal{T}\{I\} = \mathcal{S}_{2m}\{I\} + \mathcal{R}_{2m}\{I\}. \tag{3.13}$$

For all nonnegative integers m, $\mathcal{R}_{2m}\{I\} \in C^{2m+2}[-1,1]$, so that in view of Remark 3.3 we have the following.

REMARK 3.4. [illegible] (3.4) [illegible] $\tilde{h} \in H_e^{s+1}(2\pi)$ [illegible] s $p_m = \min(s+1,\, 2m+2)$ [illegible] m [illegible] $\mathcal{S}_{2m}\{I\} \in H_e^{p_m}(2\pi)$

In view of (3.10) we have

$$\mathcal{S}_{2m}\{I\} = \sum_{k=0}^{m} \frac{F_1^{(2k)}(0)}{(2k)!}\, \mathcal{V}_{2k}\{I\}, \tag{3.14}$$

where

$$\mathcal{V}_\ell\{I\}(z) = \int_{-1}^{1} \frac{(z-t)^\ell I(t)}{\sqrt{1-t^2}}\, \ln|z-t|dt. \tag{3.15}$$

The following lemma follows directly by differentiation of (3.15).

LEMMA 3.5. [illegible] $I \in C^0[-1,1]$ [illegible] $k \geq 0$ $\mathcal{V}_k\{I\} \in C^k[-1,1]$ [illegible]

$$\mathcal{V}_k^{(k)}\{I\} = k!\,(\mathcal{V}_0\{I\} - \mathcal{W}_k\{I\}), \tag{3.16}$$

[illegible] $\mathcal{W}_k\{I\}$ [illegible]

$$\mathcal{W}_k\{I\} = \left[\sum_{m=1}^{k} \binom{k}{m} \frac{(-1)^m}{m}\right] \cdot \int_{-1}^{1} \frac{I(t)}{\sqrt{1-t^2}}\, dt. \tag{3.17}$$

The connection between Symm's integral equation and Hallén's integral equation stems from the (inductive) proof of the following lemma. (Note that $\mathcal{V}_0$ coincides with the Symm's integral operator on the right-hand side of (3.7).)

LEMMA 3.6. $s \geq 0$ (3.4) $\tilde{h} \in H_e^{s+1}(2\pi)$ $\tilde{\mathcal{V}}_0\{I\} \in H_e^{s+1}(2\pi)$

. By induction in $m = 0, 1 \ldots$ we establish that $\tilde{\mathcal{V}}_0\{I\} \in H_e^{p_m}(2\pi)$ for all integers m, where $p_m = \min(s+1, 2m+2)$ (see Remark 3.4). Taking m such that $2m+2 \geq s+1$ then shows that $\tilde{\mathcal{V}}_0\{I\} \in H_e^{s+1}(2\pi)$, as desired.

For $m = 0$, we note from (3.14) that

$$\mathcal{S}_0\{I\} = F_1(0)\mathcal{V}_0\{I\}.$$

Since $F_1(0) \neq 0$ (see (2.1), (2.16), and (2.19)) and since, by Remark 3.4, $\tilde{\mathcal{S}}_0\{I\} \in H_e^{p_0}(2\pi)$, it follows that $\tilde{\mathcal{V}}_0\{I\} \in H_e^{p_0}(2\pi)$, as needed.

Assume now that $\tilde{\mathcal{V}}_0\{I\} \in H_e^{p_m}(2\pi)$. In the case $2m+2 \geq s+1$ we have $p_{m+1} = p_m$ and the inductive step is complete. Let us consider, then, the case $2m+2 < s+1$. We know that $\tilde{\mathcal{V}}_0\{I\} \in H_e^{p_m}(2\pi)$ and thus, from Lemma 3.5, that $\tilde{\mathcal{V}}_{2k}\{I\} \in H_e^{p_m+2k}(2\pi)$. In particular, for all $k \geq 1$, $\tilde{\mathcal{V}}_{2k}\{I\} \in H_e^{p_m+2}(2\pi)$. In view of the relation (3.14) we have

$$\tilde{\mathcal{S}}_{2m+2}\{I\} = \sum_{k=0}^{m+1} \frac{F_1^{(2k)}(0)}{(2k)!}\, \tilde{\mathcal{V}}_{2k}\{I\}.$$

Since as noted above $F_1(0) \neq 0$, and since by Remark 3.4 $\tilde{S}_{2m+2} \in H_e^{p_{m+1}}(2\pi)$, it follows that $\tilde{\mathcal{V}}_0\{I\} \in H_e^r(2\pi)$, where $r = \min\{p_{m+1}, p_m + 2\} = p_{m+1}$, and thus the inductive step and the proof of the lemma are complete. □

In view of Lemmas 3.1, 3.6, and the Sobolev lemma, the main result of this section now follows directly.

THEOREM 3.7. I (3.4) $\tilde{h} \in H_e^{s+1}(2\pi)$ $\tilde{I} \in H_e^s(2\pi)$ $h \in C^k[-1,1]$ $\tilde{I} \in C_e^{k-2}(2\pi)$ $I \in C^{[(k-2)/2)]}[-1,1]$ $h \in C^\infty[-1,1]$ $\tilde{I} \in C^\infty[0,\pi]$ $I \in C^\infty[-1,1]$

REMARK 3.8. $s = 1$ 4.2 [13] $s \geq 0$ h I $s = 1$ 3.7 $s = 1$ I $W^{1,p}$ $p > 1$ h $W^{2,p}$ $p = 2$ 3.2 $s = 1$ $\tilde{h} \in H_e^2(2\pi)$ $h \in H^2[-1,1] = W^{2,2}[-1,1]$ $s = 1$ $\tilde{I}$ I H^1

4. Numerical treatment of the Hallén and Pocklington equations. In this section we present our solvers for the Hallén and Pocklington problems. The theoretical equivalence of the Hallén and Pocklington equations is well established [7, 13]; a comparison of the performance of our numerical solvers for these equations is provided in section 5. In section 4.1 we introduce our Hallén solver, which we base on a Chebyshev series representation of the reduced current function (equation (3.1)), the use of which allows us to incorporate explicitly the current's end-point singularities within a high-order treatment. Our Pocklington solver, which results as a natural extension of the methods of section 4.1, is then presented in section 4.2. Finally, in section 4.3, we introduce an alternative formulation of the Hallén problem, which is suitable for use in problems involving very thin wires.

4.1. Hallén solver. Since the singular weight function $(1-t^2)^{-1/2}$ in the integrand of (3.3) can be eliminated by means of the Chebyshev change of variables $t = \cos\theta$ [8, eqn. (1.1)], the set of Chebyshev polynomials $T_n(z)$ is a natural choice of basis functions to represent $I(z)$ for $z \in [-1,1]$. We thus use the discretization

$$I(z) \approx \frac{1}{2}\, b_0 + \sum_{n=1}^{N} b_n T_n(z), \tag{4.1}$$

where the Chebyshev coefficients b_n are the unknowns of the problem. The use of such an expansion is highly advantageous: for the case of an incident plane wave, for example, the solution $I \in C^\infty[-1,1]$ (see section 3), and the Chebyshev series (4.1) thus converges faster than $\mathcal{O}(N^{-m})$ for any positive integer m—i.e., it achieves superalgebraic convergence.

From (4.1), the action of the Hallén operator $\mathcal{H}$ in (3.3) can be approximated by

$$\mathcal{H}\{I\}(z) \approx \frac{1}{2}\, b_0 A_0(z) + \sum_{n=1}^{N} b_n A_n(z), \tag{4.2}$$

where $A_n = \mathcal{H}\{T_n\}$. In Appendix B we tackle the difficult and important problem of evaluating the functions A_n accurately and efficiently. The problems of evaluating both the kernel of the operator $\mathcal{H}$ (section 2.3) and the functions A_n, which present similar challenges, constitute two of the main contributions of this paper.

The discrete form (4.2) of the operator $\mathcal{H}$ is the basis of our collocation method to solve the Hallén problem. Each column of the resulting system matrix corresponds to a particular index n of the function A_n, while the rows are generated by testing (4.2) at $N+1$ values of $z \in [-1,1]$.

REMARK 4.1.

$$z_j = \cos\left(\frac{j\pi}{N}\right), \qquad j = 0,1,2,\ldots,N, \tag{4.3}$$

5

Jones [7] has shown that the solution to the Hallén problem may be obtained by linear superposition. We implement Jones's idea in our context by writing the reduced current as the sum

$$I(z) = \alpha_1 I^{(1)}(z) + \alpha_2 I^{(2)}(z) + I^{(3)}(z), \tag{4.4}$$

where $\mathcal{H}\{I^{(1)}\}(z) = \cos kz$, $\mathcal{H}\{I^{(2)}\}(z) = \sin kz$, and

$$\mathcal{H}\{I^{(3)}\}(z) = -4\pi i \int_{-1}^{z} e(t)\sin[k(z-t)]dt; \tag{4.5}$$

the values of the constants α_1 and α_2 follow from the end-point conditions (3.2). Since, as is easy to check, we have $I^{(1)}(1) = I^{(1)}(-1)$ and $I^{(2)}(1) = -I^{(2)}(-1)$, it follows that

$$\alpha_1 = -\frac{1}{2}\left(\frac{I^{(3)}(1) + I^{(3)}(-1)}{I^{(1)}(1)}\right) \quad \text{and} \quad \alpha_2 = -\frac{1}{2}\left(\frac{I^{(3)}(1) - I^{(3)}(-1)}{I^{(2)}(1)}\right). \tag{4.6}$$

To obtain I numerically we use a representation of the form (4.1) for each one of the quantities $I^{(1)}$, $I^{(2)}$, and $I^{(3)}$. The corresponding Chebyshev coefficients $b_n^{(j)}$ of $I^{(j)}(z)$ $(n = 0, 1, \dots)$ are obtained from the equations for the $I^{(j)}$'s (equation (4.5) and immediately above) and repeated use of the LU decomposition of the $(N+1)\times(N+1)$ matrix that discretizes the operator $\mathcal{H}$ (as described earlier in this section), together with back- and forward substitution for $j = 1, 2, 3$. Once these Chebyshev coefficients have been obtained, the constants α_1 and α_2 are calculated by means of (4.6), using the Chebyshev series to produce $I^{(j)}(\pm 1)$. The reduced current I is then found by summation of its Chebyshev series, whose coefficients are given by

$$b_n = \alpha_1 b_n^{(1)} + \alpha_2 b_n^{(2)} + b_n^{(3)}, \qquad n = 0, 1, \dots, N. \tag{4.7}$$

4.2. Pocklington solver. A method for solving the Pocklington problem embodied in (1.1) and (1.2), which uses, again, a representation of the form (4.1) and the coefficients b_n as unknowns, is readily derived from results presented in section 4.1. To obtain such a method we first represent the functions $A_n(z)$ by truncated Chebyshev expansions

$$A_n(z) \approx \frac{1}{2}\, a_0^{(n)} + \sum_{m=1}^{N} a_m^{(n)} T_m(z). \tag{4.8}$$

The coefficients in the series (4.8) can be obtained directly from an (FFT-based) Chebyshev transform of the columns of the matrix for the Hallén problem—which, by construction, contain the point values of the functions A_n at the Chebyshev points (4.3).

Using the coefficients in (4.8) we can then obtain the Chebyshev series coefficients of $(\partial^2/\partial z^2 + k^2)A_n(z)$,

$$c_m^{(n)} = \frac{2}{\pi}\int_{-1}^{1} \frac{T_m(z)}{\sqrt{1-z^2}} \left(\frac{\partial^2}{\partial z^2} + k^2\right) A_n(z)\, dz = k^2 a_m^{(n)} + 4 \sum_{\ell=1}^{\left[\frac{N-m}{2}\right]} \ell(\ell+m)(2\ell+m) a_{2\ell+m}^{(n)}, \tag{4.9}$$

which is valid for $m = 0, 1, 2, \dots$. The second derivative of the Chebyshev polynomial in (4.9) is evaluated by means of the relation [8, p. 38].

The Pocklington system matrix contains the quantity $c_m^{(n)}$ in the mth row and nth column. Since it is necessary to enforce vanishing of the reduced current at the end-points $z = \pm 1$ we only use the first $N-1$ of these equations to allow for the two additional equations enforcing the end-point conditions (3.2):

$$\frac{1}{2}\, b_0 + \sum_{n=1}^{N} b_n = 0 \quad \text{and} \quad \frac{1}{2}\, b_0 + \sum_{n=1}^{N} (-1)^n b_n = 0. \tag{4.10}$$

The right-hand side of the system consists of the first $N-1$ Chebyshev coefficients of $-4\pi ike(z)$ and zeros in the last two positions. The system is solved directly by an LU decomposition.

4.3. Alternative Hallén method for very thin wires. For particular physical cases involving very thin wires ($a \ll$ wire-length, independently of the value of ka), the rate of convergence to the solution may be substantially improved by means of a distribution of discretization points within the wire different from those used

in the previous sections. We speculate that the slower convergence in the absence of such change of variables must be the result of a small-weight singularity in the thin-wire solutions located near the origin on the imaginary axis (associated with a corresponding singularity, the kernel itself has at $(z - z') = 2ai$; see (1.4)). While the methods outlined in sections 4.1 and 4.2 exhibit superalgebraic convergence even for such cases, use of the point-distributions described in what follows, which are more concentrated on the end-points than those considered in previous sections, result in significant additional convergence acceleration. For the sake of brevity, here we restrict attention to the Hallén equation; the development of this method proceeds in analogy to that presented in section 4.1.

In contrast with the Chebyshev (cosine) change of variables used earlier in this text, here we introduce the coordinate transformation

$$z = \frac{2\cos w}{1 + \cos^2 w}, \tag{4.11}$$

which maps $w \in [0, \pi]$ to $z \in [-1, 1]$ and which, like the cosine change of variables used earlier, eliminates the end-point singularities in the integrand of (3.3). The cosine change of variables results in a locally quadratic clustering of points near the ends of the wire; the clustering of points near the ends of the wire induced by the transformation (4.11), on the other hand, is quartic. It is precisely this property which gives rise to the improved convergence rates of the solution for very thin wires. The reduced current is an even, periodic function of the new variables, and thus it may be approximated by the cosine series

$$I\left(\frac{2\cos w}{1 + \cos^2 w}\right) \approx \frac{1}{2} b_0 + \sum_{n=1}^{N} b_n \cos(nw), \tag{4.12}$$

where the coefficients b_n are the unknowns of the problem. For $I(z) \in C^{m+1}[-1, 1]$, the error of the cosine series (4.12) is $\mathcal{O}(N^{-m})$ for all $w \in [-\pi, \pi]$. In particular, for $I(z) \in C^{\infty}[-1, 1]$, the cosine series (4.12) converges superalgebraically to $I(z)$. As discussed in section 5.1, for thin wires, the representation (4.12) converges much more quickly than the expansion (4.1).

In the new variables, the Hallén operator may be approximated by the discrete form

$$\mathcal{H}\{I\}\left(\frac{2\cos w}{1 + \cos^2 w}\right) \approx \frac{1}{2}\, b_0 B_0(w) + \sum_{n=1}^{N} b_n B_n(w). \tag{4.13}$$

The integral functions resulting from the method may be written as

$$B_n(w) = -2 \int_0^{\pi} \frac{\sin u \cos(nu)}{1 + \cos^2 u}\, G\left(\frac{2\cos w}{1 + \cos^2 w} - \frac{2\cos u}{1 + \cos^2 u}\right) du. \tag{4.14}$$

Much in the same way as in section 4.1, the discrete form of the operator (4.13) is the basis for a collocation method to solve the Hallén problem. The collocation points are chosen to be the $N + 1$ uniformly spaced values of $w \in [0, \pi]$, which, as will be demonstrated in section 5, results in a well-conditioned system. The total current is found by our adaptation of Jones's method, as outlined in (4.4) and below.

TABLE 5.1
Correspondence of parameters between [4] and the present text. The wire occupies the interval $[0,1]$ *in that work, while it occupies the interval* $[-1,1]$ *here.*

Present	Reference [4]
a	$2a$
k	$\omega/2$
ka	ωa

5. Results and discussion. In addition to several efficient novel algorithms for wire problems, the contribution [4] presents an excellent survey of modern methods in the area; we therefore use the results in that reference to demonstrate the improvements that can be offered by our algorithms. We note that only the Pocklington equation is treated in [4], and, further, that our methods differ substantially from those presented in that contribution. In fact, several methods are discussed in [4], some which exhibit $\mathcal{O}(N^{-1})$ or $\mathcal{O}(N^{-2})$ convergence in the solution, and some methods which exhibit much faster convergence. In particular, the hp-geometric Galerkin scheme introduced in [4] appears to exhibit superalgebraic convergence and is, quite definitely, the most efficient and accurate of the methods available previously. In what follows, we therefore compare our results to hp-geometric Galerkin data in every instance for which such data was made available in [4]; our results compare much more favorably to all others in the literature.

The identification of parameters and unknowns between [4] and the present text is as described in Table 5.1. In that reference, the right-hand side of the Pocklington equation is taken to equal unity in all cases. To account for this prescription, as well as the different kernel normalization and wire-length, we set $\mathcal{H}\{I^{(3)}(z)\} = \pi/k^2$ for the Hallén problem, and $e(z) = i/(4k)$ for the Pocklington problem; with these definitions the current J we consider matches that of [4].

In this section the maximum relative errors e_{max} are defined by

$$e_{max} = \frac{\max_j |J(z_j) - J_{ref}(z_j)|}{\max_j |J_{ref}(z_j)|}. \tag{5.1}$$

The reference data sets $J_{ref}(z_j)$ we used for evaluation of e_{max} were obtained using large numbers $N+1$ of unknowns, to ensure high accuracies. The numerical values of e_{max} displayed in the tables, in turn, were obtained by evaluating J and J_{ref} at sufficiently fine Chebyshev meshes $\{z_j\}$, so that the quantity $|J(z) - J_{ref}(z)|$, which experiences rapid oscillations near the wire end-points, is fully resolved, and thus allows for accurate computation of e_{max}. See the captions of Tables 5.2 to 5.6 for full details on the meshes used for evaluation of J_{ref} and e_{max}. Details concerning the accurate evaluation of the total current $J(z) = I(z)/\sqrt{1-z^2}$ from the reduced current $I(z)$ are presented in Appendix C; here we simply point out that a straightforward division does give rise to significant error increases—unbounded, in fact, as the evaluation points tend toward the end-points. The strategies described in Appendix C completely resolve this difficulty and result in currents J that are as accurate as the corresponding reduced currents I.

In what follows we first examine the convergence rates of our algorithms, with attention to the relative advantages offered by our Hallén, Pocklington, or alternative Hallén approaches. All computations presented here were performed on an AMD 2600+ computer system. Solutions of all linear systems were obtained by means of the LU-based direct solvers provided in the LAPACK linear algebra package

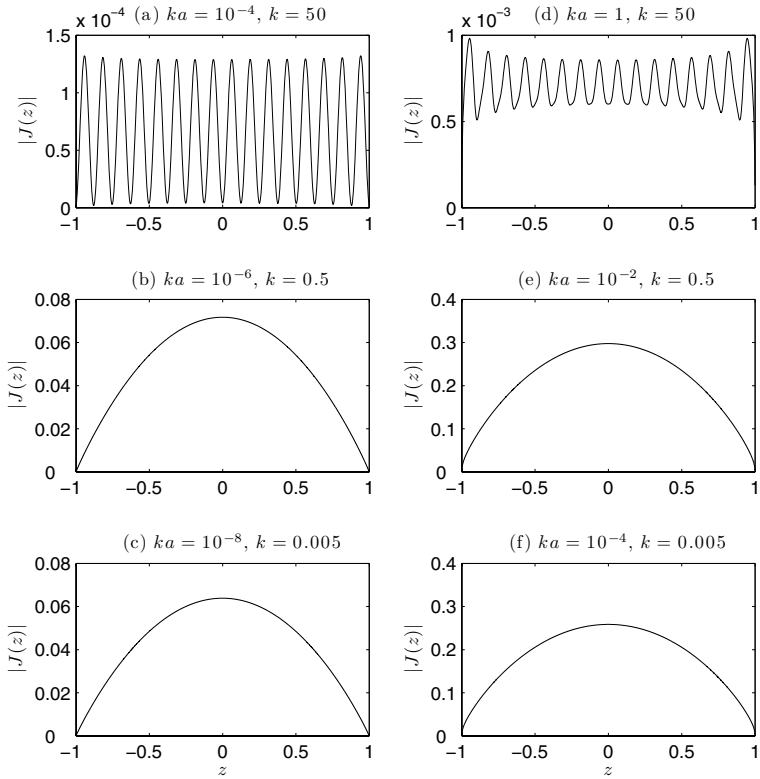


FIG. 5.1. *Profiles of the current function $J(z)$ studied by Davies, Duncan, and Funken* [4].

(www.netlib.org/lapack/). In a rather comprehensive set of tests that we consider, four-digit solutions, for example, were found in a execution time of at most four seconds; in most cases computing time for a four-digit solve was a fraction of a second. As shown below, further, in all cases much higher accuracies result in very small additional computing times: see Tables 5.2 to 5.6 in section 5.1; in all tables the execution time is denoted by T.

5.1. Numerical results. To demonstrate the superalgebraic convergence of our methods we consider a set of six test cases put forward in [4], which span a broad range of physical configurations. The profiles of the total currents J for these cases are displayed in Figure 5.1; note that all cases in the left-hand column of that figure have radii $a = 0.5 \cdot 10^{-6}$, while those in the right-hand column have radii $a = 0.5 \cdot 10^{-2}$. The tables presented in this section display, among other quantities, the computational times required to obtain the Chebyshev series of the reduced current I; in all cases the table captions list the (very small) corresponding additional times required to evaluate the current J on a fine Chebyshev grid (of a size also listed on the table captions); see also Appendix C. Note, however, that in the context of our algorithm, it is possible (and most efficient) to use the reduced current I, not the physical current J, to evaluate the scattered fields in space. Thus, if the quantity of interest is the field scattered by the wire, the calculation of the current J is unnecessary.

TABLE 5.2

Alternative Hallén solver for case (a)*:* $k = 50$ *and* $ka = 10^{-4}$ *(very thin wires). The current profile is shown in Figure* 5.1(a)*. An* $N + 1$*-term cosine series is used, and the algorithm is run with integration parameters* M_1 *and* M_2 *and splitting parameter* $\Delta = 5$ *(see Appendix* C*). The quantity* C_N *is the estimate of the reciprocal condition number provided by the linear algebra package LAPACK. The error oscillations are fully resolved: the maximum relative error* e_{max} *(see* (5.1)*) was obtained by evaluation of* J *and* J_{ref} *on a* $2^{13} + 1$*-point Chebyshev grid; the reference solution* J_{ref} *was obtained with* $N = 240$*,* $M_1 = 2^{12}$*, and* $M_2 = 2^9$*. The evaluation of* J *at the* $2^{13} + 1$*-point Chebyshev grid required approximately* 0.07 *s of computational time in addition to the tabulated computing times. Note from Figure* 5.1(a) *that the maximum value of the solution is of the order of* 10^{-4} *so that the maximum absolute errors decrease up to order of* 10^{-12}*—consistent, for the* 16*-digit accuracy used, with a condition number of the order of* 10^{-4} *listed above.*

N	M_1	M_2	T (s)	e_{max}	C_N
110	2^7	2^2	0.28	$3.3 \cdot 10^{-1}$	$7.8 \cdot 10^{-4}$
120	2^8	2^2	0.63	$2.1 \cdot 10^{-2}$	$5.8 \cdot 10^{-4}$
130	2^9	2^3	1.50	$2.0 \cdot 10^{-3}$	$4.2 \cdot 10^{-4}$
140	2^{10}	2^3	3.43	$6.1 \cdot 10^{-5}$	$3.8 \cdot 10^{-4}$
150	2^{11}	2^4	8.11	$3.2 \cdot 10^{-6}$	$3.6 \cdot 10^{-4}$
160	2^{11}	2^4	9.18	$7.3 \cdot 10^{-8}$	$3.4 \cdot 10^{-4}$
170	2^{11}	2^4	10.29	$2.0 \cdot 10^{-8}$	$3.2 \cdot 10^{-4}$

(a) $ka = 10^{-4}$, $k = 50$. The current profile is shown in Figure 5.1(a). The relative error and the reciprocal matrix condition number (estimated by LAPACK) are presented in Table 5.2 for increasing values of the index N of truncation of the solution's cosine expansion. For this case, a very thin wire problem, we used the alternative Hallén method described in section 4.3. We see from Table 5.2 that, for example, full single precision accuracy results from this method in an $N = 160$, 9-second computation. (Use of the regular Hallén and Pocklington methods of sections 4.1 and 4.2 requires much larger numbers of unknowns, of the order of $N = 800$, to achieve comparable accuracies.) Note that the condition number remains virtually constant throughout the table. A graphical solution J, consistent with our Figure 5.1(a), was given in [4] (case $a = 1e - 6$, $\omega = 100$ in that work), but errors were not reported there for this case. Through comparison with cases (c) and (d) below we speculate that the hp-geometric Galerkin method of [4] in the present case (a) requires a number of the order of at least $N = 350$ unknowns to achieve errors of the order of 10^{-5}—since, in the present case, the discretization needs to resolve both the extreme end-point singularities arising from the very thin wire (for which hp-geometric Galerkin requires $N = 50$ in case (c)) as well as the higher frequency $k = 50$ (which for hp-geometric Galerkin requires $N = 350$ in case (d)). Taking into account, further, the superalgebraic convergence (in M_1 and M_2) of the integration methods we use versus the third-order or lower convergence inherent in previous methods (see paragraph (b) in the introduction), we expect the approaches in the present paper should prove significantly advantageous over most solvers otherwise available.

(b) $ka = 10^{-6}$, $k = 0.5$. The current profile is shown in Figure 5.1(b). Statistics for this case are virtually identical to those of case (c). A graphical solution J, consistent with our Figure 5.1(b), was given in [4] (case $a = 1e - 6$, $\omega = 1$ in that work). In that reference, errors were reported for slower numerical methods (which require approximately 100 unknowns to produce errors of order 10^{-3}), but not for the hp-geometric Galerkin approach, which is the main subject of our comparison.

(c) $ka = 10^{-8}$, $k = 0.005$. The current profile is shown in Figure 5.1(c). The relative error and the reciprocal matrix condition number are presented

TABLE 5.3

Same as Table 5.2*, with the following exceptions:* $k = 0.005$*,* $ka = 10^{-8}$*; the current profile is shown in Figure* 5.1(c)*; the reference solution* J_{ref} *was obtained with* $N = 140$*,* $M_1 = 2^{12}$*, and* $M_2 = 2^9$*; in this case, in which the maximum value of the solution is of order* 0.08*, the condition number is not consistent with stagnation at the level* 10^{-8} *—which, in fact, does not occur: a certain oscillatory behavior exhibited by the cosine series coefficients, shown in Figure* 5.2*, gives rise to nonmonotone convergence.*

N	M_1	M_2	T (s)	e_{max}	C_N
10	2^9	2^2	0.02	$1.9 \cdot 10^{-1}$	$4.4 \cdot 10^{-3}$
20	2^{10}	2^2	0.11	$2.4 \cdot 10^{-4}$	$2.5 \cdot 10^{-3}$
30	2^{11}	2^3	0.41	$4.5 \cdot 10^{-6}$	$2.0 \cdot 10^{-3}$
40	2^{11}	2^4	0.69	$5.2 \cdot 10^{-7}$	$1.3 \cdot 10^{-3}$
50	2^{11}	2^4	1.02	$5.2 \cdot 10^{-7}$	$1.1 \cdot 10^{-3}$
60	2^{11}	2^4	1.41	$3.5 \cdot 10^{-7}$	$9.0 \cdot 10^{-4}$
70	2^{11}	2^4	1.87	$2.5 \cdot 10^{-7}$	$7.9 \cdot 10^{-4}$
80	2^{11}	2^4	2.44	$9.8 \cdot 10^{-8}$	$6.9 \cdot 10^{-4}$
90	2^{11}	2^4	3.02	$7.4 \cdot 10^{-8}$	$6.0 \cdot 10^{-4}$
100	2^{11}	2^4	3.64	$3.1 \cdot 10^{-8}$	$5.4 \cdot 10^{-4}$

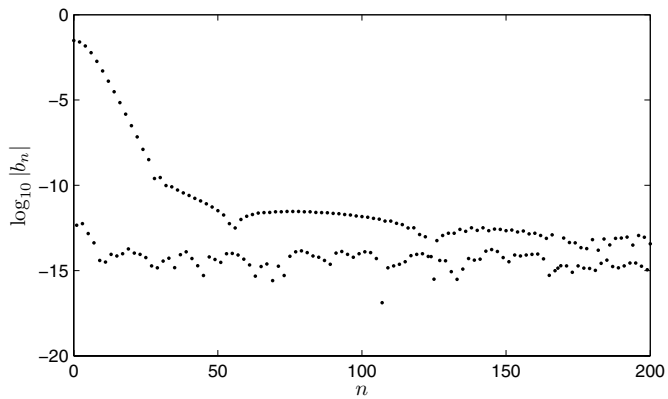


FIG. 5.2. *Cosine series coefficients of the alternative Hallén formulation for case* (c) *with* $k = 0.005$*,* $ka = 10^{-8}$*; a thin wire case (*$a \ll 1$*). Virtually identical behavior is seen for case* (b)*. The coefficients are divided into two sets: the even-index coefficients are clustered in the upper set, and the odd-index coefficients in the lower set. Note that the even set demonstrates extremely rapid convergence for* $0 \leq n \leq 50$*, followed by a regime of slower convergence. We speculate this must be the result of a small-weight singularity in the thin-wire solutions located near the origin on the imaginary axis (associated with a corresponding singularity the kernel itself has at* $(z - z') = 2ai$*), which results in a decrease of the convergence rates for* $n > 50$*; see Table* 5.3*.*

in Table 5.3. For this case, yet again a very thin wire problem, we used the alternative Hallén method described in section 4.3. We see from Table 5.3 that, for example, single precision accuracy results from this method for $N = 40$; use of the regular Hallén and Pocklington methods of sections 4.1 and 4.2 requires much larger numbers of unknowns, of the order of $N = 500$, to achieve comparable accuracies. Note that the condition number remains virtually constant throughout the table. A graphical solution J, consistent with our Figure 5.1(c), was given in [4] (case $a = 1e{-}6$, $\omega = 0.01$ in that work). Errors of the order of 10^{-6} were reported in that reference from use of the hp-geometric Galerkin approach with 50 unknowns; Table 5.3 shows an accuracy of the order of 10^{-6} for $N = 30$ unknowns.

TABLE 5.4

Hallén and Pocklington solvers for case (d)*: $k = 50$, $ka = 1$. The current profile is shown in Figure* 5.1(d). *An $N + 1$-term Chebyshev is used, and the algorithm is run with integration parameters M_1 and M_2 and splitting parameter $\Delta = 5$. The quantity C_N is the estimate of the reciprocal condition number provided by the linear algebra package LAPACK. The error oscillations are fully resolved: the maximum relative error e_{max} was obtained by evaluation of J and J_{ref} on a $2^{10} + 1$-point Chebyshev grid; the reference solution J_{ref} was obtained by means of the Hallén solver with $N = 140$, $M_1 = 2^{12}$, and $M_2 = 2^9$. The evaluation of J at the $2^{10} + 1$-point Chebyshev grid required approximately* 0.004 *s of computational time in addition to the tabulated computing times. Note that in this case, in which the maximum value of the solution is of order 10^{-3}, the solution errors are smaller than what might have been expected for the condition numbers reported in the table.*

N	M_1	M_2	Hallén			Pocklington		
			e_{max}	T (s)	C_N	e_{max}	T (s)	C_N
50	2^6	2^4	$8.9 \cdot 10^{-2}$	0.04	$8.8 \cdot 10^{-3}$	$7.9 \cdot 10^{-1}$	0.05	$9.7 \cdot 10^{-7}$
60	2^6	2^5	$3.4 \cdot 10^{-4}$	0.08	$6.1 \cdot 10^{-3}$	$3.3 \cdot 10^{-2}$	0.09	$5.4 \cdot 10^{-7}$
70	2^7	2^5	$1.7 \cdot 10^{-7}$	0.15	$4.8 \cdot 10^{-3}$	$2.3 \cdot 10^{-5}$	0.16	$3.7 \cdot 10^{-7}$
80	2^7	2^5	$4.9 \cdot 10^{-9}$	0.19	$3.6 \cdot 10^{-3}$	$2.9 \cdot 10^{-9}$	0.20	$2.5 \cdot 10^{-7}$
90	2^7	2^6	$3.5 \cdot 10^{-10}$	0.31	$2.8 \cdot 10^{-3}$	$3.5 \cdot 10^{-10}$	0.32	$1.8 \cdot 10^{-7}$
100	2^7	2^6	$9.5 \cdot 10^{-12}$	0.38	$2.2 \cdot 10^{-3}$	$9.5 \cdot 10^{-12}$	0.39	$1.4 \cdot 10^{-7}$
110	2^7	2^6	$5.6 \cdot 10^{-12}$	0.45	$1.8 \cdot 10^{-3}$	$6.5 \cdot 10^{-12}$	0.46	$1.0 \cdot 10^{-7}$
120	2^7	2^6	$3.9 \cdot 10^{-13}$	0.54	$1.5 \cdot 10^{-3}$	$4.5 \cdot 10^{-13}$	0.55	$7.9 \cdot 10^{-8}$

TABLE 5.5

Same as Table 5.4*, with the following exceptions: $k = 0.5$, $ka = 10^{-2}$; the current profile is shown in Figure* 5.1(e)*; the maximum value of the solution is of order* 0.3*—the errors are consistent with the condition numbers reported, although in the Pocklington case they are smaller than what might have been expected in view of the tabulated condition numbers.*

N	M_1	M_2	Hallén			Pocklington		
			e_{max}	T (s)	C_N	e_{max}	T (s)	C_N
10	2^3	2^2	$3.6 \cdot 10^{-3}$	0.003	$4.6 \cdot 10^{-2}$	$3.6 \cdot 10^{-3}$	0.004	$3.2 \cdot 10^{-4}$
20	2^3	2^3	$2.3 \cdot 10^{-4}$	0.005	$1.2 \cdot 10^{-2}$	$2.3 \cdot 10^{-4}$	0.006	$3.8 \cdot 10^{-5}$
30	2^4	2^4	$2.3 \cdot 10^{-5}$	0.010	$5.7 \cdot 10^{-3}$	$2.3 \cdot 10^{-5}$	0.012	$1.1 \cdot 10^{-5}$
40	2^5	2^4	$8.7 \cdot 10^{-7}$	0.020	$5.1 \cdot 10^{-3}$	$8.7 \cdot 10^{-7}$	0.022	$4.8 \cdot 10^{-6}$
50	2^6	2^4	$1.9 \cdot 10^{-7}$	0.042	$3.5 \cdot 10^{-3}$	$1.9 \cdot 10^{-7}$	0.045	$2.5 \cdot 10^{-6}$
60	2^6	2^5	$2.6 \cdot 10^{-8}$	0.076	$2.4 \cdot 10^{-3}$	$2.6 \cdot 10^{-8}$	0.079	$1.5 \cdot 10^{-6}$
70	2^7	2^5	$7.2 \cdot 10^{-10}$	0.149	$1.8 \cdot 10^{-3}$	$7.2 \cdot 10^{-10}$	0.154	$9.3 \cdot 10^{-7}$
80	2^7	2^5	$3.0 \cdot 10^{-10}$	0.191	$1.4 \cdot 10^{-3}$	$3.0 \cdot 10^{-10}$	0.197	$6.3 \cdot 10^{-7}$
90	2^7	2^5	$3.4 \cdot 10^{-11}$	0.237	$1.1 \cdot 10^{-3}$	$3.4 \cdot 10^{-11}$	0.246	$4.4 \cdot 10^{-7}$
100	2^7	2^6	$1.5 \cdot 10^{-12}$	0.379	$8.8 \cdot 10^{-4}$	$1.5 \cdot 10^{-12}$	0.389	$3.2 \cdot 10^{-7}$

(d) $ka = 1$ $k = 50$. The current profile is shown in Figure 5.1(d), and numerical results for the regular Hallén and Pocklington methods of sections 4.1 and 4.2 are presented in Table 5.4. Both methods produce errors of the order of 10^{-7} for $N = 70$, and yield double precision accuracy for $N = 120$. The hp-geometric Galerkin approach of [4] (case $a = 0.01$, $\omega = 100$ in that work) yields results with an accuracy of 10^{-5} using a number of the order $N = 350$ unknowns.

(e) $ka = 10^{-2}$ $k = 0.5$. The current profile is shown in Figure 5.1(e), and numerical results for the regular Hallén and Pocklington methods of sections 4.1 and 4.2 are presented in Table 5.5. Both methods produce virtually identical results in this case, with rapid convergence in N to nearly double precision accuracy. In particular, Table 5.5 shows results with errors of 10^{-5} and 10^{-6} for $N = 30$ and

TABLE 5.6

Same as Table 5.4*, with the following exceptions:* $k = 0.005$*,* $ka = 10^{-4}$*; the current profile is shown in Figure* 5.1(f)*; the reference solution* J_{ref} *was obtained by means of the Pocklington solver with* $N = 140$*,* $M_1 = 2^{12}$*, and* $M_2 = 2^9$*; the maximum value of the solution is of order* 0.3*—the detailed discussion of conditioning and errors is given in the text.*

N	M_1	M_2	Hallén			Pocklington		
			e_{max}	T (s)	C_N	e_{max}	T (s)	C_N
10	2^3	2^2	$3.6 \cdot 10^{-3}$	0.003	$4.7 \cdot 10^{-2}$	$3.6 \cdot 10^{-3}$	0.004	$3.5 \cdot 10^{-4}$
20	2^3	2^3	$2.3 \cdot 10^{-4}$	0.005	$1.2 \cdot 10^{-2}$	$2.3 \cdot 10^{-4}$	0.006	$4.3 \cdot 10^{-5}$
30	2^4	2^4	$2.3 \cdot 10^{-5}$	0.010	$8.7 \cdot 10^{-3}$	$2.3 \cdot 10^{-5}$	0.012	$1.3 \cdot 10^{-5}$
40	2^5	2^4	$8.9 \cdot 10^{-7}$	0.020	$5.1 \cdot 10^{-3}$	$8.9 \cdot 10^{-7}$	0.022	$5.4 \cdot 10^{-6}$
50	2^6	2^4	$1.9 \cdot 10^{-7}$	0.042	$3.5 \cdot 10^{-3}$	$1.9 \cdot 10^{-7}$	0.045	$2.8 \cdot 10^{-6}$
60	2^6	2^5	$2.8 \cdot 10^{-8}$	0.076	$2.5 \cdot 10^{-3}$	$2.7 \cdot 10^{-8}$	0.079	$1.6 \cdot 10^{-6}$
70	2^7	2^5	$8.5 \cdot 10^{-10}$	0.149	$1.8 \cdot 10^{-3}$	$7.2 \cdot 10^{-10}$	0.154	$1.0 \cdot 10^{-6}$
80	2^7	2^5	$5.2 \cdot 10^{-10}$	0.191	$1.4 \cdot 10^{-3}$	$3.1 \cdot 10^{-10}$	0.197	$6.9 \cdot 10^{-7}$
90	2^7	2^5	$1.5 \cdot 10^{-9}$	0.237	$1.1 \cdot 10^{-3}$	$4.8 \cdot 10^{-11}$	0.246	$4.9 \cdot 10^{-7}$
100	2^7	2^6	$1.4 \cdot 10^{-9}$	0.379	$8.9 \cdot 10^{-4}$	$1.2 \cdot 10^{-11}$	0.389	$3.6 \cdot 10^{-7}$

$N = 40$, respectively. The hp-geometric Galerkin approach of [4] (case $a = 0.01$, $\omega = 1$ in that work) yields results with an accuracy of 10^{-4} using a number of the order $N = 40$ unknowns.

(f) $ka = 10^{-4}$ $k = 0.005$. The current profile is shown in Figure 5.1(f), and numerical results for both methods are presented in Table 5.6. This is the only case in which we have found that the regular Pocklington method is clearly superior to the Hallén method. While both methods exhibit rapid convergence, the Hallén approach appears to break down beyond a relative error $\mathcal{O}(10^{-9})$, while the relative error of the Pocklington method continues to decrease to $\mathcal{O}(10^{-11})$. The reason for this is subtle: the current generated by the Hallén method relies on a superposition of solutions, shown in (4.4). We have noted that, under some circumstances, a subtractive cancellation occurs here that limits the accuracy obtainable using finite precision arithmetic. In view of these considerations, in this table, unlike all others, we use the results provided by the Pocklington method to produce the reference solution J_{ref}. A graphical solution J, consistent with our Figure 5.1(f), was given in [4] (case $a = 0.01$, $\omega = 0.01$ in that work), but errors were not reported in that reference for this case.

5.2. Conclusions. In contrast to published reports, we have found our Hallén and Pocklington approaches numerically equivalent in many situations. For very thin wires, the alternative method developed in section 4.3 (which gives rise to higher mesh-densities at the end-points than those implicit in the Chebyshev method of sections 4.1 and 4.2) results in significant efficiency improvements. The numerical results we presented show that, in many cases, to achieve a given accuracy, the numbers N of unknowns required by our codes are up to five times smaller than those required by the best solvers previously available. The short computing times demonstrated in section 5.1 arise not only from use of small numbers of unknowns but also from the small numbers M_1 and M_2 of integration points required for construction of the discretized operator. Unfortunately, timings have not been reported previously in the literature. In view of the superalgebraic convergence in the numbers M_1 and M_2 (see point (d) in the introduction) and the small numbers N of unknowns required by the present solvers, we suggest that these methods should be significantly faster than others available previously.

Appendix A. Asymptotic expansion of the kernel. We present the exact polynomials required to evaluate the asymptotic form of the thin-wire kernel given in (2.20) to order $M = 10$.

$$\mathcal{A}_0(x) = 1, \qquad \mathcal{A}_1(x) = x, \qquad \mathcal{A}_2(x) = x^2 + 48, \qquad \mathcal{A}_3(x) = x(x^2 + 1620), \tag{A.1}$$

$$\mathcal{A}_4(x) = x^4 + 13440x^2 + 241920, \qquad \mathcal{A}_5(x) = x(x^4 + 63000x^2 + 30240000), \tag{A.2}$$

$$\mathcal{A}_6(x) = x^6 + 213840x^4 + 748440000x^2 + 9580032000, \tag{A.3}$$

$$\mathcal{A}_7(x) = x(x^6 + 588588x^4 + 8899450560x^2 + 2669835168000), \tag{A.4}$$

$$\begin{aligned}\mathcal{A}_8(x) &= x^8 + 1397760x^6 + 67805337600x^4 \\ &\quad + 136695560601600x^2 + 1464595292160000,\end{aligned} \tag{A.5}$$

$$\begin{aligned}\mathcal{A}_9(x) &= x(x^8 + 2974320x^6 + 381093672960x^4 \\ &\quad + 3137163115806720x^2 + 726029178229555200),\end{aligned} \tag{A.6}$$

$$\begin{aligned}\mathcal{A}_{10}(x) &= x^{10} + 5814000x^8 + 1714199760000x^6 + 434446787174400000x^4 \\ &\quad + 63863677714636800000x^2 + 613091306060513280000,\end{aligned} \tag{A.7}$$

$$\mathcal{B}_0(x) = 1, \qquad \mathcal{B}_1(x) = x, \qquad \mathcal{B}_2(x) = x^2 + 180, \qquad \mathcal{B}_3(x) = x(x^2 + 2240), \tag{A.8}$$

$$\mathcal{B}_4(x) = x^4 + 12600x^2 + 1209600, \qquad \mathcal{B}_5(x) = x(x^4 + 47520x^2 + 49896000), \tag{A.9}$$

$$\mathcal{B}_6(x) = x^6 + 140140x^4 + 762810048x^2 + 54486432000, \tag{A.10}$$

$$\mathcal{B}_7(x) = x(x^6 + 349440x^4 + 6780533760x^2 + 4881984307200), \tag{A.11}$$

$$\begin{aligned}\mathcal{B}_8(x) &= x^8 + 771120x^6 + 42343741440x^4 \\ &\quad + 149388719800320x^2 + 8963323188019200,\end{aligned} \tag{A.12}$$

$$\begin{aligned}\mathcal{B}_9(x) &= x(x^8 + 1550400x^6 + 205703971200x^4 \\ &\quad + 2482553069568000x^2 + 1419192838103040000).\end{aligned} \tag{A.13}$$

Appendix B. Numerical evaluation of the A_n and B_n functions. In this section we discuss the efficient computation of the quantities A_n and B_n in (4.2) and (4.14), respectively. The numerical evaluation of these functions is complicated not only by the logarithmic singularity of the kernel at $z = t$ (resp., $\rho = 0$) but also by an interesting near-singular behavior of the kernel functions F_1 and F_2 in a neighborhood of the singularity. As the wire radius a decreases, the real part of each of the kernel functions F_1 and F_2 on the domain $[-1/a, 1/a]$ contains increasingly rapid variations, developing a sharp peak in a neighborhood of $z - t = \mathcal{O}(a)$. This behavior, which is closely related to the complex singularities mentioned in section 4.3, is depicted in Figure B.1 for the function $g_{r1}(\rho)$—which is related to the dimensional kernel function $F_1(z - t)$ by (2.19). Qualitatively similar behavior is exhibited by the function F_2. While these functions remain analytic, their resolution with an evenly spaced FFT may require prohibitively large numbers of sampling points. In what follows we outline our procedure for simultaneously treating the unusual combination in the kernel of a logarithmic singularity and the sharp peaks discussed above.

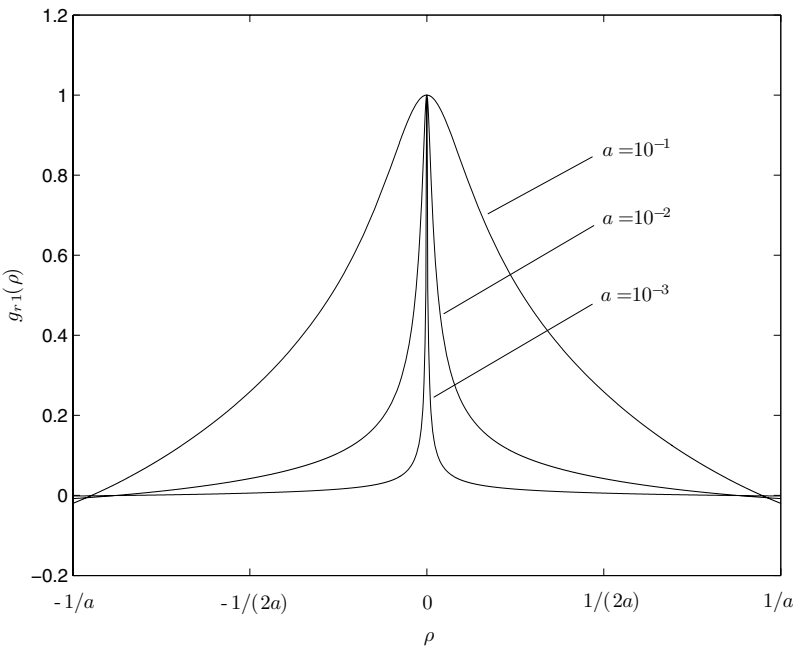


FIG. B.1. *The nondimensional kernel function $g_{r1}(\rho)$ for $k = 1$ and $a = 10^{-1}$, $a = 10^{-2}$ and $a = 10^{-3}$. Both kernel functions $g_{r1}(\rho)$ and $g_{r2}(\rho)$ exhibit a sharp peak in the neighborhood $\rho = \mathcal{O}(1)$.*

B.1. Evaluation of the functions A_n. In what follows we are concerned with the evaluation of the integrals

$$A_n(z) = \mathcal{H}\{T_n\}(z) = \int_{-1}^{1} [F_1(z-t)\ln|z-t| + F_2(z-t)]\,\frac{T_n(t)}{\sqrt{1-t^2}}\,dt. \tag{B.1}$$

Since $G(z)$ is an even function of z, it easy to verify that the functions A_n satisfy

$$A_n(-z) = (-1)^n A_n(z) \tag{B.2}$$

so that, although the values of the A_n's in the full interval $z \in [-1, 1]$ are required by our method, actual integrations to produce these functions numerically need only be performed for $z \in [0, 1]$.

To evaluate the functions A_n on a Chebyshev grid (see Remark 4.1) and to eliminate the end-point singularities in (B.1) we introduce the substitutions $z = \cos\alpha$ and $t = \cos\theta$, which reduce the corresponding integrals to

$$A_n(\cos\alpha) = \int_0^{\pi} G(\cos\alpha - \cos\theta)\cos n\theta d\theta. \tag{B.3}$$

The irregular features in the integrand of (B.3), which arise from corresponding features of F_1 and F_2, can be resolved with a reasonable expense by means of a local integration scheme using a fine grid on the domain $|z-t| \le 2a\Delta$, where Δ is the splitting parameter defined in section 2.3. Outside this domain, integration on a much coarser grid is sufficient to achieve the desired accuracy.

In detail, we partition the integration domain $[0,\pi]$ into two regions, the fine- and coarse-discretization domains, that correspond to regions of fast and slow variations of the integrand in (B.1) and (B.3). To define the fine-discretization region, in most cases it is sufficient to restrict the angular coordinate in (B.3) to the domain $\theta \in [\theta_1, \theta_2]$, with $\theta_1 = \cos^{-1}(z + 2a\Delta)$ and $\theta_2 = \cos^{-1}(z - 2a\Delta)$. One notable exception occurs when $z + 2a\Delta > 1$; this special case is treated by setting $\theta_1 = 0$. The analogous case, when $z - 2a\Delta < 1$, does not occur in our method, since values of (B.3) for $z \in [-1, 0)$ are obtained by the symmetry relation (B.2). Setting $\varepsilon_1 = \alpha - \theta_1$ and $\varepsilon_2 = \theta_2 - \alpha$, the domain of integration is thus partitioned as

$$[0,\pi] \equiv [0, \alpha - \varepsilon_1] \cup [\alpha - \varepsilon_1, \alpha] \cup [\alpha, \alpha + \varepsilon_2] \cup [\alpha + \varepsilon_2, \pi]\,. \tag{B.4}$$

Owing to (B.2), we have the restriction $\alpha \in [0, \pi/2]$. In the fine-discretization region $\theta \in [\alpha - \varepsilon_1, \alpha + \varepsilon_2]$ we develop a special integration method which treats the logarithmic singularity of G. Outside of this region, different methods are applied on a coarse grid. [illegible] $\theta \in [0, \alpha - \varepsilon_1] \cup [\alpha + \varepsilon_2, \pi]$. In this region, which lies away from the $\theta = \alpha$ singularity, the kernel G is smooth and contributions to the A_n functions are computed by means of an FFT-based $M_1 + 1$-point Clenshaw–Curtis integration rule on each of the two relevant subintervals.

[illegible] $\theta \in [\alpha - \varepsilon_1,\ \alpha + \varepsilon_2]$. We illustrate the method of computation of contributions to $A_n(\cos\alpha)$ for the left-hand interval $\theta \in [\alpha - \varepsilon_1, \alpha]$; results for the right-hand interval $\theta \in [\alpha, \alpha + \varepsilon_2]$ are obtained in an analogous fashion. We map $x \in [-1, 1]$ to $\theta \in [\alpha - \varepsilon_1, \alpha]$ with the change of variables $\theta = \varepsilon_1(x - 1)/2 + \alpha$, and for each n, we introduce the approximation

$$F_1(\cos\alpha - \cos\theta)\cos n\theta \approx \frac{1}{2}\, e_0^{(n)}(\alpha) + \sum_{m=1}^{M_2} e_m^{(n)}(\alpha) T_m(x); \quad \theta = \theta(x). \tag{B.5}$$

A similar expansion of length $M_2 + 1$ for $F_2(\cos\alpha - \cos\theta)\cos n\theta$ is introduced. The quantity $M_2 + 1$, which is independent of M_1, is the length of the fast cosine transform used to compute the series coefficients in this region. The integrals involving F_2 are computed using an FFT-based $M_2 + 1$-point Clenshaw–Curtis integration rule. In view of (1.5), the integral involving F_1 possesses a logarithmic singularity at $x = 1$ which, for high-order integration, requires special treatment. The method we use to resolve this singularity integrates the logarithmic terms exactly: For $\alpha \neq 0$ and $\alpha = 0$ we use the identities

$$\ln|\cos\alpha - \cos\theta| = \ln\left|\frac{\cos\alpha - \cos\theta}{x - 1}\right| + \ln|x - 1| \tag{B.6}$$

and

$$\ln|1 - \cos\theta| = \ln\left|\frac{1 - \cos\theta}{(x-1)^2}\right| + 2\ln|x - 1|\,, \tag{B.7}$$

respectively. The Taylor expansion

$$\cos\alpha - \cos\theta = \frac{\varepsilon_1}{2}\,(x-1)\sin\alpha + \frac{\varepsilon_1^2}{8}\,(x-1)^2\cos\alpha + \mathcal{O}\big(\varepsilon_1^3(x-1)^3\big) \tag{B.8}$$

then shows that the first logarithmic term on the right-hand side of each of (B.6) and (B.7) is regular, so that the integrals involving those terms may be computed with

high-order accuracy by means of the $M_2 + 1$-point Clenshaw–Curtis integration rule. The singular terms are integrated using the rule

$$\int_{-1}^{1} F_1(\cos\alpha - \cos\theta)\cos n\theta \ln|x-1|\,dx \approx \frac{1}{2}\,w_0 e_0^{(n)}(\alpha) + \sum_{m=1}^{M_2} w_m e_m^{(n)}(\alpha), \tag{B.9}$$

where w_m are the integration weights

$$w_m = \int_{-1}^{1} T_m(t)\ln|1-t|\,dt. \tag{B.10}$$

A closed-form expression for the weights w_m may be derived by making the substitution $t = \cos 2\psi$ in (B.10), expanding the terms $\cos m\psi$ in powers of $\cos\psi$, and applying the identity [6, 4.387] on a term-by-term basis. This leads to the results $w_0 = 2\ln 2 - 2$, $w_1 = -1$, and, for $m > 1$,

$$\begin{aligned} w_m =& \frac{[1+(-1)^m]\ln 2}{1-m^2} \\ &- 2m\sum_{n=0}^{m} \frac{(-1)^n 2^{2m-2n}}{(m-n+1)(2m-n)} \left(\begin{array}{c} 2m-n \\ n \end{array} \right) [\gamma + \Psi(m-n+2)]\,, \end{aligned} \tag{B.11}$$

where $\gamma = 0.57721566\ldots$ is Euler's constant. In our implementation, the weights (B.11) have been precomputed to machine precision for all values of m up to $m = 2^9$ and saved in a data file which is read into memory every time the code is executed. The logarithmic integrals over the subinterval $\theta \in [\alpha, \alpha + \varepsilon_2]$ are computed in a similar manner, with integration weights that can be obtained from the data set above, namely,

$$\int_{-1}^{1} T_m(t)\ln|1+t|\,dt = (-1)^m w_m. \tag{B.12}$$

B.2. Integration for the alternative Hallén method. We now consider the efficient evaluation of the B_n functions defined in (4.14); these must obey the symmetry relations

$$B_n(w + \pi/2) = (-1)^n B_n(w), \qquad w \in [0, \pi/2]; \tag{B.13}$$

compare with (B.2). Thus, numerical integration of the B_n functions need only be performed for uniformly spaced values of $w \in [0, \pi/2]$. The integration of the B_n functions follows in close analogy to the treatment of the A_n functions, outlined in Appendix B.1.

Appendix C. Computation of the total current. As mentioned in section 5, the accurate evaluation of the total current $J(z) = I(z)/\sqrt{1-z^2}$ from the reduced current $I(z)$ requires special consideration: clearly the error in the current obtained by straightforward division tends to infinity as the evaluation points tend toward the end-points. To resolve this difficulty it is convenient to rewrite $J(z)$ in terms of a trigonometric series, and avoid the explicit division by the quantity $\sqrt{1-z^2}$. In what follows, we derive a representation for the total current which is used to evaluate currents arising from all three methods of solution presented in this paper.

With the substitution $z = \cos\theta$ in (3.1) and (4.1), an approximation for the total current is readily derived by exploiting the orthogonality of the trigonometric functions:

$$J(\cos\theta) \approx \sum_{n=1}^{N+1} c_n \sin(n\theta). \tag{C.1}$$

The coefficients in (C.1) are given in terms of the b_n coefficients in (4.1) by $c_1 = b_0$, $c_2 = 2b_1$, and

$$c_{n+1} = 2b_n + c_{n-1}, \qquad 2 \le n \le N. \tag{C.2}$$

The coefficients c_n for $n \ge 2$ are easily found by forward recursion using (C.2). Accurate values of the total current on a , Chebyshev grid (which is necessary to obtain maximum relative errors owing to fast error oscillations) are obtained by means of a zero-padded fast-sine-evaluation of the series (C.1).

In the context of the very thin wire algorithm, in turn, it is possible to derive a formula analogous to (C.1) for the current J in the w variables to produce this quantity everywhere in the wire. However, the current J in the w variables vanishes quadratically at the end-points, while it only vanishes linearly at those points in the θ variables. The vanishing order at the end-points is related to ill-conditioning in evaluation of the J function: evaluation of quadratically vanishing quantities on the basis of cosine coefficients of order 1 is clearly more ill-conditioned than evaluation of corresponding quantities that only vanish linearly at the end-points. Thus, higher accuracy can be extracted for J if the reduced current function I in w-space is first transformed to θ-space and then a sine series of J in θ is used as earlier in this section. In detail, given the N-term cosine series for the reduced current in (4.12), values of I are computed at the set of N_2 Chebyshev points (see (4.3)) for $N_2 \gg N$—which can be accomplished by an appropriately zero-padded FFT, together with low-order polynomial interpolation. For the results presented in this work, we have used $N_2 = 2^{10} + 1$. Using this data the b_n coefficients of the Chebyshev series (4.1) can be obtained, and can then be used to compute the c_n coefficients of the sine-series representation for the total current in (C.1). This method produces uniformly accurate solutions throughout the wire for the very thin wire problem—including around the wire end-points. Note that the supersampling procedure described here is computationally inexpensive (as supported by data in the relevant table captions), since it can be accomplished by means of an FFT.

REFERENCES

[1] M. Abramowitz and I. A. Stegun, eds., *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, Dover, New York, 1992. Reprint of the 1972 edition.

[2] R. A. Adams, *Sobolev Spaces*, Pure Appl. Math. 65, Academic Press, New York, 1975.

[3] P. F. Byrd and M. D. Friedman, *Handbook of Elliptic Integrals for Engineers and Scientists*, 2nd ed., revised, Grundlehren Math. Wiss. 67, Springer-Verlag, New York, 1971.

[4] P. J. Davies, D. B. Duncan, and S. A. Funken, *Accurate and efficient algorithms for frequency domain scattering from a thin wire*, J. Comput. Phys., 168 (2001), pp. 155–183.

[5] G. Fikioris, *The approximate integral equation for a cylindrical scatterer has no solution*, J. Electromagn. Waves Appl., 15 (2001), pp. 1153–1159.

[6] I. S. Gradshteyn and I. M. Ryzhik, *Table of Integrals, Series, and Products*, 6th ed., Academic Press, San Diego, CA, 2000. Translated from the Russian; translation edited and with a preface by A. Jeffrey and D. Zwillinger.

[7] D. S. Jones, *Note on the integral equation for a straight wire antenna*, Proc. IEE-H, 128 (1981), pp. 114–116.
[8] J. C. Mason and D. C. Handscomb, *Chebyshev Polynomials*, Chapman & Hall/CRC, Boca Raton, FL, 2003.
[9] L. W. Pearson, *A separation of the logarithmic singularity in the exact kernel of the cylindrical antenna integral equation*, IEEE Trans. Antennas Propagat., AP-23 (1975), pp. 256–258.
[10] H. C. Pocklington, *Electrical oscillations in wires*, Proc. Cambridge Phil. Soc., 9 (1897), pp. 324–332.
[11] J. B. Reade, *Asymptotic behaviour of eigenvalues of certain integral equations*, Proc. Edinburgh Math. Soc. (2), 22 (1979), pp. 137–144.
[12] T. J. Rivlin, *Chebyshev Polynomials: From Approximation Theory to Algebra and Number Theory*, 2nd ed., Pure Appl. Math., John Wiley & Sons, New York, 1990.
[13] B. P. Rynne, *The well-posedness of the integral equations for thin wire antennas*, IMA J. Appl. Math., 49 (1992), pp. 35–44.
[14] C. Snow, *Hypergeometric and Legendre Functions with Applications to Integral Equations of Potential Theory*, National Bureau of Standards Appl. Math. Ser. 19, U.S. Government Printing Office, Washington, D.C., 1952.
[15] G. T. Symm, *An integral equation method in conformal mapping*, Numer. Math., 9 (1966), pp. 250–258.
[16] W.-X. Wang, *The exact kernel for cylindrical antenna*, IEEE Trans. Antennas Propagat., AP-39 (1991), pp. 434–435.
[17] Y. Yan and I. H. Sloan, *On integral equations of the first kind with logarithmic kernels*, J. Integral Equations Appl., 1 (1988), pp. 549–579.

SIAM J. SCI. COMPUT.
Vol. 29, No. 4, pp. 1403–1419
© 2007 Society for Industrial and Applied Mathematics

STABILITY RESULTS FOR SCATTERED DATA INTERPOLATION BY TRIGONOMETRIC POLYNOMIALS*

STEFAN KUNIS† AND DANIEL POTTS†

Abstract. A fast and reliable algorithm for the optimal interpolation of scattered data on the torus $\mathbb{T}^d$ by multivariate trigonometric polynomials is presented. The algorithm is based on a variant of the conjugate gradient method in combination with the fast Fourier transforms for nonequispaced nodes. The main result is that under mild assumptions the total complexity for solving the interpolation problem at M arbitrary nodes is of order $\mathcal{O}(M \log M)$. This result is obtained by the use of localized trigonometric kernels where the localization is chosen in accordance with the spatial dimension d. Numerical examples show the efficiency of the new algorithm.

Key words. trigonometric approximation, scattered data interpolation, localization, iterative methods, fast Fourier transforms for nonequispaced nodes, FFT

AMS subject classifications. 65F10, 65T40, 65F15

DOI. 10.1137/060665075

1. Introduction.

We discuss the approximation of scattered data by d-variate periodic functions $f : \mathbb{T}^d \to \mathbb{C}$, where $\mathbb{T} := [-\frac{1}{2}, \frac{1}{2})$ denotes the torus. In practical applications we are often confronted with the situation that experimental data or measured values of a function are known only at a finite sampling set $\mathcal{X} := \{\boldsymbol{x}_j \in \mathbb{T}^d : j = 0, \dots, M-1\}$. Especially, nonuniform sampling sets appear in more and more applications in recent years. Given a notion of the distance of two points by $\operatorname{dist}(\boldsymbol{x}, \boldsymbol{x}_0) := \min_{\boldsymbol{j} \in \mathbb{Z}^d} \|(\boldsymbol{x} + \boldsymbol{j}) - \boldsymbol{x}_0\|_\infty$, we measure the "nonuniformity" of $\mathcal{X}$ by the mesh norm and the separation distance, defined by

$$\delta := 2 \max_{\boldsymbol{x} \in \mathbb{T}^d} \min_{j=0,\dots,M-1} \operatorname{dist}(\boldsymbol{x}_j, \boldsymbol{x}), \qquad q := \min_{j,l=0,\dots,M-1; j \neq l} \operatorname{dist}(\boldsymbol{x}_j, \boldsymbol{x}_l),$$

respectively. Obviously, the relation $q \leq M^{-1/d} \leq \delta$ is fulfilled.

For given samples $(\boldsymbol{x}_j, y_j) \in \mathbb{T}^d \times \mathbb{C}$, $j = 0, \dots, M-1$, a polynomial degree $N \in 2\mathbb{N}$, and the index set $I_N := \left\{-\frac{N}{2}, \dots, \frac{N}{2} - 1\right\}^d$ of frequencies, we construct a d-variate trigonometric polynomial

$$f(\boldsymbol{x}) := \sum_{\boldsymbol{k} \in I_N} \hat{f}_{\boldsymbol{k}} \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{k} \boldsymbol{x}}$$

such that $f(\boldsymbol{x}_j) \approx y_j$, $j = 0, \dots, M-1$. Turning this into matrix vector notation, we aim to solve the system of linear equations

$$\boldsymbol{A} \hat{\boldsymbol{f}} \approx \boldsymbol{y} \tag{1.1}$$

for the unknown vector of Fourier coefficients $\hat{\boldsymbol{f}} := (\hat{f}_{\boldsymbol{k}})_{\boldsymbol{k} \in I_N} \in \mathbb{C}^{N^d}$. Throughout the paper, we denote the vector of the given sample values by $\boldsymbol{y} := (y_j)_{j=0,\dots,M-1} \in \mathbb{C}^M$

*Received by the editors July 14, 2006; accepted for publication (in revised form) February 2, 2007; published electronically June 12, 2007.
http://www.siam.org/journals/sisc/29-4/66507.html

†Department of Mathematics, Chemnitz University of Technology, 09107 Chemnitz, Germany (kunis@mathematik.tu-chemnitz.de, potts@mathematik.tu-chemnitz.de). The first author is grateful for partial support of this work by the German Academic Exchange Service (DAAD) and the warm hospitality during his stay at the Numerical Harmonic Analysis Group, University of Vienna.

and the nonequispaced Fourier matrix by

$$\boldsymbol{A} = \boldsymbol{A}_{\mathcal{X}} := \left(\mathrm{e}^{2\pi \mathrm{i} \boldsymbol{k} \boldsymbol{x}_j}\right)_{j=0,\ldots,M-1;\boldsymbol{k}\in I_N} \in \mathbb{C}^{M\times N^d}.$$

In contrast to the widely used nonequispaced FFT for the fast matrix vector multiplication with $\boldsymbol{A}$ (see [16] and its references), the [illegible] solution of (1.1) is still a challenging goal. Besides recently developed sparse reconstruction techniques (see, e.g., [5, 13] and their references), a standard method to determine $\hat{\boldsymbol{f}}$ is to solve the general linear least squares problem $\|\hat{\boldsymbol{f}}\|_2 \to \min$ subject to $\|\boldsymbol{y} - \boldsymbol{A}\hat{\boldsymbol{f}}\|_2 = \min$ (see, e.g., [3, p. 15]). This can be done by means of the singular value decomposition, which is not practical in the present situation for large problems due to its time and memory requirements. Direct solvers for the univariate case $d = 1$ in [6, 18] obtain a solution in $\mathcal{O}(NM)$ floating point operations.

For $N^d < M$, the linear system (1.1) is overdetermined, so that, in general, the given data $\boldsymbol{y}$ will be approximated only up to a [illegible] $\boldsymbol{r} := \boldsymbol{y} - \boldsymbol{A}\hat{\boldsymbol{f}}$. In order to compensate for clusters in the sampling set $\mathcal{X}$, it is useful to incorporate weights $w_j > 0$ and to consider the weighted approximation problem

$$\|\boldsymbol{y} - \boldsymbol{A}\hat{\boldsymbol{f}}\|_{\boldsymbol{W}}^2 = \sum_{j=0}^{M-1} w_j |y_j - f(\boldsymbol{x}_j)|^2 \xrightarrow{\hat{\boldsymbol{f}}} \min, \tag{1.2}$$

where $\boldsymbol{W} := \operatorname{diag}(w_j)_{j=0,\ldots,M-1}$. In [10] it has been proven that this problem has a unique solution if $N < (\frac{\pi}{\log 2}\, d\, \delta)^{-1}$. Its solution is computed iteratively by means of the conjugate gradient method in [7, 2, 9], where the multilevel Toeplitz structure of $\boldsymbol{A}^{\mathsf{H}}\boldsymbol{W}\boldsymbol{A}$ is used for fast matrix vector multiplications. Slightly more stable with respect to rounding errors is the Conjugate gradient method, Normal equation, Residual minimization (CGNR) method (cf. [3, p. 288]), which iterates the original residual $\boldsymbol{r}_l = \boldsymbol{y} - \boldsymbol{A}\hat{\boldsymbol{f}}_l$ instead of the residual $\boldsymbol{A}^{\mathsf{H}}\boldsymbol{W}\boldsymbol{r}_l$ of the normal equations. Note, furthermore, that it has been suggested in [17] to incorporate some "knowledge on the decay of the Fourier coefficients" $\hat{\boldsymbol{W}} := \operatorname{diag}(\hat{w}_{\boldsymbol{k}})_{\boldsymbol{k}\in I_N}$, $\hat{w}_{\boldsymbol{k}} > 0$. Their approach is based on the weighted least squares problem

$$\|\boldsymbol{A}^{\mathsf{H}}\boldsymbol{W}(\boldsymbol{y} - \boldsymbol{A}\hat{\boldsymbol{f}})\|_{\hat{\boldsymbol{W}}^{-1}} \xrightarrow{\hat{\boldsymbol{f}}} \min.$$

In contrast, we focus on the underdetermined and consistent linear system $\boldsymbol{A}\hat{\boldsymbol{f}} = \boldsymbol{y}$; i.e., we expect to interpolate the given data $y_j \in \mathbb{C}$, $j = 0, \ldots, M-1$, exactly. We show that the nonequispaced Fourier matrix $\boldsymbol{A}$ has full rank M for every polynomial degree $N > 2\, d\, q^{-1}$. In particular, we incorporate [illegible] $\hat{w}_{\boldsymbol{k}} > 0$, $\boldsymbol{k} \in I_N$, and consider the [illegible]

$$\|\hat{\boldsymbol{f}}\|_{\hat{\boldsymbol{W}}^{-1}}^2 = \sum_{\boldsymbol{k}\in I_N} \frac{|\hat{f}_{\boldsymbol{k}}|^2}{\hat{w}_{\boldsymbol{k}}} \xrightarrow{\hat{\boldsymbol{f}}} \min \quad \text{subject to} \quad \boldsymbol{A}\hat{\boldsymbol{f}} = \boldsymbol{y}, \tag{1.3}$$

where $\hat{\boldsymbol{W}} := \operatorname{diag}(\hat{w}_{\boldsymbol{k}})_{\boldsymbol{k}\in I_N}$. We prove that for a large class of "smooth" damping factors $\hat{w}_{\boldsymbol{k}}$ problem (1.3) is well-conditioned, where the "smoothness" has to be chosen with respect to the spatial dimension d. We propose to solve problem (1.3) by a version of the conjugate gradient method in combination with the nonequispaced FFT [19, 16, 12] to efficiently perform each iteration step.

The outline of this paper is as follows: In section 2 we set up the basic notation and relate the optimal interpolation problem (1.3) to a particular trigonometric kernel.

Furthermore, we propose Algorithm 1 for computing the solution of the interpolation problem efficiently. For the sake of analyzing the convergence of this algorithm, we then present our theory on localized trigonometric kernels in section 3. Our first result in Theorem 3.3 is a version of the typical smoothness-decay principle in Fourier analysis and relates the smoothness of the weights in (1.3) to the localization of the corresponding trigonometric kernel. We use this decay in section 4 to prove that well-separated sampling nodes yield a stable interpolation problem (1.3). The eigenvalue estimates are given for the univariate setting in Theorem 4.1 and for the multivariate setting in Theorem 4.6. Subsequently, Corollary 4.7 applies the general result to a particular class of damping factors and concludes with conditions sufficient for the full rank of $\boldsymbol{A}$. As the equidistant case in Theorem 4.10 and Corollary 4.11 reveals, the assumption on the separation distance is of optimal order. We provide numerical examples in section 5 and draw our conclusion in section 6. The software and all numerical examples are available from our NFFT homepage [12].

2. Optimal interpolation and its iterative solution. After setting up our notation in Definition 2.1, we prove in Lemma 2.2 that the optimal interpolation problem (1.3) can be stated as normal equations and that the matrix in these equations obeys special structure. Furthermore, we propose Algorithm 1 for the iterative solution of the interpolation problem and state a basic convergence result for this scheme.

DEFINITION 2.1. $d \in \mathbb{N}$, $N \in 2\mathbb{N}$, $I_N = \{-\frac{N}{2}, \dots, \frac{N}{2}-1\}^d$, $\hat{w}_{\boldsymbol{k}} > 0$, $\boldsymbol{k} \in I_N$, $\sum_{\boldsymbol{k}\in I_N} \hat{w}_{\boldsymbol{k}} = 1$, $\boldsymbol{x} \in \mathbb{T}^d$, kernel

$$K_N(\boldsymbol{x}) := \sum_{\boldsymbol{k}\in I_N} \hat{w}_{\boldsymbol{k}} \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{k}\boldsymbol{x}}.$$

tensor product kernels

$$K_N(\boldsymbol{x}) = \prod_{t=0}^{d-1} \tilde{K}_N(x_t),$$

$\tilde{K}_N$, $\boldsymbol{x} = (x_0, \dots, x_{d-1})^\top$, $\mathcal{X} \subset \mathbb{T}^d$, kernel matrix

$$\boldsymbol{K}_N := (K_N(\boldsymbol{x}_j - \boldsymbol{x}_l))_{j,l=0,\dots,M-1} \in \mathbb{C}^{M\times M}. \tag{2.1}$$

$\Lambda = \Lambda(\boldsymbol{K}_N)$, $\lambda = \lambda(\boldsymbol{K}_N)$, $\boldsymbol{K}_N$, $\operatorname{cond}(\boldsymbol{K}_N) = \frac{\Lambda}{\lambda}$

Note that from the definition immediately follows $K_N(\boldsymbol{0}) = \max_{\boldsymbol{x}\in\mathbb{T}^d} |K_N(\boldsymbol{x})| = 1$ and $(\boldsymbol{K}_N)_{j,j} = 1$, $j = 0, \dots, M-1$. The following theorem collects some basic facts.

LEMMA 2.2. $M \in \mathbb{N}$, $\mathcal{X} \subset \mathbb{T}^d$, $N \in 2\mathbb{N}$, $\hat{w}_{\boldsymbol{k}} > 0$, $\boldsymbol{k} \in I_N$, (1.3), damped normal equations of the second kind

$$\boldsymbol{K}_N \tilde{\boldsymbol{f}} = \boldsymbol{y}, \qquad \hat{\boldsymbol{f}} = \hat{\boldsymbol{W}} \boldsymbol{A}^{\mathsf{H}} \tilde{\boldsymbol{f}}, \tag{2.2}$$

$\boldsymbol{K}_N \in \mathbb{C}^{M\times M}$

$$\boldsymbol{K}_N = \boldsymbol{A} \hat{\boldsymbol{W}} \boldsymbol{A}^{\mathsf{H}} \tag{2.3}$$

. The second assertion follows from $(\boldsymbol{A}\hat{\boldsymbol{W}}\boldsymbol{A}^{\mathrm{H}})_{j,l} = \sum_{\boldsymbol{k}\in I_N} \mathrm{e}^{2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}_j}\hat{w}_{\boldsymbol{k}}\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}_l}$ and Definition 2.1. Furthermore, a solution $\hat{\boldsymbol{f}}$ of $\boldsymbol{A}\hat{\boldsymbol{f}} = \boldsymbol{y}$ has minimal weighted norm if and only if it is perpendicular with respect to the weights to the null-space of $\boldsymbol{A}$, i.e., $\hat{\boldsymbol{W}}^{-1/2}\hat{\boldsymbol{f}} \perp \mathcal{N}(\boldsymbol{A}\hat{\boldsymbol{W}}^{1/2})$. We conclude (2.2) by the fact that the orthogonal complement of the null-space of a matrix is just the range of its adjoint. □

Denoted in Algorithm 1 by Conjugate gradient method, Normal equation, Error minimization (CGNE) (cf. [3, p. 288]), we solve the normal equations (2.2) by the conjugate gradient method, minimizing in each iteration the error.

Algorithm 1. CGNE.

Input: dimension $d \in \mathbb{N}$, number of samples $M \in \mathbb{N}$, polynomial degree $N \in 2\mathbb{N}$; sampling set $\mathcal{X} \subset \mathbb{T}^d$, samples $\boldsymbol{y} \in \mathbb{C}^M$, and initial vector $\hat{\boldsymbol{f}}_0 \in \mathbb{C}^{N^d}$

$\boldsymbol{r}_0 = \boldsymbol{y} - \boldsymbol{A}\hat{\boldsymbol{f}}_0$

$\hat{\boldsymbol{p}}_0 = \boldsymbol{A}^{\mathrm{H}}\boldsymbol{r}_0$

for $l = 0, \ldots$ **do**

$\alpha_l = \boldsymbol{r}_l^{\mathrm{H}}\boldsymbol{r}_l \,/\, \hat{\boldsymbol{p}}_l^{\mathrm{H}}\hat{\boldsymbol{W}}\hat{\boldsymbol{p}}_l$

$\hat{\boldsymbol{f}}_{l+1} = \hat{\boldsymbol{f}}_l + \alpha_l\hat{\boldsymbol{W}}\hat{\boldsymbol{p}}_l$

$\boldsymbol{r}_{l+1} = \boldsymbol{r}_l - \alpha_l\boldsymbol{A}\hat{\boldsymbol{W}}\hat{\boldsymbol{p}}_l$

$\beta_l = \boldsymbol{r}_{l+1}^{\mathrm{H}}\boldsymbol{r}_{l+1} \,/\, \boldsymbol{r}_l^{\mathrm{H}}\boldsymbol{r}_l$

$\hat{\boldsymbol{p}}_{l+1} = \beta_l\hat{\boldsymbol{p}}_l + \boldsymbol{A}^{\mathrm{H}}\boldsymbol{r}_{l+1}$

end for

Output: the lth iterate $\hat{\boldsymbol{f}}_l$

The proposed method finds approximations from a Krylov space closely related to the one of the CGNR method for (1.2) but with minimal error instead of minimal residual. Note that we exploit the factorization in (2.3) to iterate the original vector $\hat{\boldsymbol{f}}$ instead of the vector $\tilde{\boldsymbol{f}}$; cf. (2.2). Hence, we use fast matrix vector multiplications for $\boldsymbol{A}$ and $\boldsymbol{A}^{\mathrm{H}}$ by means of the fast Fourier transforms at nonequispaced nodes (NFFT) having an arithmetical complexity of $\mathcal{O}(N^d\log(N^d) + M|\log\epsilon|^d)$ in each iteration, where ϵ is the prescribed accuracy. Details concerning NFFT algorithms can be found, for example, in [19, 16] and a corresponding software package in [12]. Applying the standard estimate for the convergence of the conjugate gradient method, we obtain the following lemma.

LEMMA 2.3. $\boldsymbol{K}_N$ (2.1) $\hat{\boldsymbol{e}}_l := \hat{\boldsymbol{f}}_l - \hat{\boldsymbol{W}}\boldsymbol{A}^{\mathrm{H}}\boldsymbol{K}_N^{-1}\boldsymbol{y}$ l 1

$$\|\hat{\boldsymbol{e}}_l\|_{\hat{\boldsymbol{W}}^{-1}} \leq 2\left(\frac{\sqrt{\Lambda}-\sqrt{\lambda}}{\sqrt{\Lambda}+\sqrt{\lambda}}\right)^l \|\hat{\boldsymbol{e}}_0\|_{\hat{\boldsymbol{W}}^{-1}}$$

. We note that $\|\hat{\boldsymbol{e}}_l\|_{\hat{\boldsymbol{W}}^{-1}} = \|\tilde{\boldsymbol{f}}_l - \boldsymbol{K}_N^{-1}\boldsymbol{y}\|_{\boldsymbol{K}_N}$, where $\tilde{\boldsymbol{f}}_l$ denotes the lth iterate of the conjugate gradient method applied to the equation $\boldsymbol{K}_N\tilde{\boldsymbol{f}} = \boldsymbol{y}$ (cf. Lemma

2.2), and apply the standard estimate for the conjugate gradient method; see also [3, p. 288]. □

This result includes the special case of $M = N^d$ equispaced nodes and no damping, i.e., $\hat{w}_{\boldsymbol{k}} = 1$, $\boldsymbol{k} \in I_N$, where the first iterate of our algorithm is already the solution to (2.2). We present estimates for the extremal eigenvalues λ and Λ dependent only on the quantities N and q and the damping factors $\hat{w}_{\boldsymbol{k}}$, $\boldsymbol{k} \in I_N$. Analogous results for the stability of the interpolation by radial and zonal functions are obtained in [15, 20]. Section 3 prepares our estimates by constructing localized kernels.

. . . . 2.4. Before that, we would like to comment on the following:

The weighted norm in (1.3) is induced by the inner product $\hat{\boldsymbol{g}}^{\mathsf{H}} \hat{\boldsymbol{W}}^{-1} \hat{\boldsymbol{f}}$. In particular, the definition $\langle f, g \rangle_{\hat{\boldsymbol{W}}^{-1}} := \hat{\boldsymbol{g}}^{\mathsf{H}} \hat{\boldsymbol{W}}^{-1} \hat{\boldsymbol{f}}$ makes the space of trigonometric polynomials $T_N := \operatorname{span} \left\{ \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{k} \cdot} : \boldsymbol{k} \in I_N \right\}$ to a reproducing kernel Hilbert space. Its reproducing kernel is given by K_N; i.e., the point evaluations obey $f(\boldsymbol{x}) = \langle f, K_N(\cdot - \boldsymbol{x}) \rangle_{\hat{\boldsymbol{W}}^{-1}}$.

Moreover, the solution $f(\boldsymbol{x}) = \sum_{\boldsymbol{k} \in I_N} \hat{f}_{\boldsymbol{k}} \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{k} \boldsymbol{x}}$ of the normal equations (2.2) has a comparable norm to the given samples, i.e.,

$$\Lambda^{-1} \|\boldsymbol{y}\|_2^2 \leq \langle f, g \rangle \leq_{\hat{\boldsymbol{W}}^{-1}} \leq \lambda^{-1} \|\boldsymbol{y}\|_2^2 .$$

This norm equivalence is due to the fact that the field of values of the matrix $\boldsymbol{K}_N^{-1}$ is bounded by its extremal eigenvalues and $\boldsymbol{y}^{\mathsf{H}} \boldsymbol{K}_N^{-1} \boldsymbol{y} = \tilde{\boldsymbol{f}}^{\mathsf{H}} \boldsymbol{K}_N \tilde{\boldsymbol{f}} = \hat{\boldsymbol{f}}^{\mathsf{H}} \hat{\boldsymbol{W}}^{-1} \hat{\boldsymbol{f}}$.

3. Localized kernels. Starting from a class of admissible weight functions in Definition 3.1, we construct localized trigonometric kernels in Theorem 3.3, where Lemma 3.2 serves as an intermediate step. Following the smoothness-decay principle in Fourier analysis, we relate the smoothness of the weight function to the decay of the kernel K_N built upon the sampled weights. A related approach is taken in [14, Thm. 2.2] for the detection of singularities. The particular class of B-spline kernels (cf. Definition 3.4) is considered in Corollary 3.5. While we present our results on the connection between smooth weight functions and localized kernels for the univariate case, we give its generalization to the class of tensor product kernels in Corollary 3.7.

Definition 3.1. . . $\beta \in \mathbb{N}$ $\beta \geq 2$ . . . . . $g : \mathbb{R} \to \mathbb{R}$ . . admissible weight function of order β . . . . . $(\beta - 1)$ . . . . . $g^{(\beta-1)}$ . . . . . .

$$|g^{(\beta-1)}|_V := \int_{\mathbb{R}} |\mathrm{d} g^{(\beta-1)}(z)| = \sup \sum_{j=0}^{n-1} |g^{(\beta-1)}(z_{j+1}) - g^{(\beta-1)}(z_j)| < \infty,$$

. . . . . $\{z_j\}_{j \in \mathbb{N}_0}$ . . . . . $\operatorname{supp} g = [-\frac{1}{2}, \frac{1}{2}]$ $g^{(\gamma)}(\pm\frac{1}{2}) = 0$ . . $\gamma = 0, \ldots, \beta - 1$ $g(z) > 0$, $|z| < \frac{1}{2}$ . . . . . $\|g\|_{L^1} = 1$ . . . . . $BV_0^{\beta-1}$ . . . . . β . . . . . zeta function $\zeta(\beta) := \sum_{r=1}^{\infty} r^{-\beta}$ $\beta > 1$ . . . $g \in BV_0^{\beta-1}$ . . norm of the samples

$$\|g\|_{1,N} := \sum_{k=-\frac{N}{2}}^{\frac{N}{2}} g\left(\frac{k}{N}\right) .$$

The following lemma prepares Theorem 3.3.

LEMMA 3.2. [illegible] $\beta \in \mathbb{N}$, $\beta \geq 2$ [illegible] $g \in BV_0^{\beta-1}$ [illegible] $N \in 2\mathbb{N}$ $N \geq 2\beta$ [illegible] $x \in \left[-\frac{1}{2}, \frac{1}{2}\right] \setminus \{0\}$ [illegible]

$$\left| \sum_{k=-\frac{N}{2}}^{\frac{N}{2}} g\left(\frac{k}{N}\right) \mathrm{e}^{2\pi \mathrm{i} k x} \right| \leq \frac{\left(2^{\beta}-1\right) \zeta\left(\beta\right) \left| g^{(\beta-1)} \right|_V}{(2N)^{\beta-1} \left|2\pi x\right|^{\beta}},$$

$$\|g\|_{1,N} \geq N \left(1 - 2\zeta\left(\beta\right) \left(4\pi\beta\right)^{-\beta} |g^{(\beta-1)}|_V\right).$$

[illegible] First, we define for $x, z \in \left[-\frac{1}{2}, \frac{1}{2}\right]$ the function $h_x(z) := g(z) \mathrm{e}^{2\pi \mathrm{i} N x z}$. Thus, the Poisson summation formula yields

$$\frac{1}{N} \sum_{k=-\frac{N}{2}}^{\frac{N}{2}} g\left(\frac{k}{N}\right) \mathrm{e}^{2\pi \mathrm{i} k x} = \frac{1}{N} \sum_{k=-\frac{N}{2}}^{\frac{N}{2}} h_x\left(\frac{k}{N}\right) = \sum_{r \in \mathbb{Z}} \int_{-\frac{1}{2}}^{\frac{1}{2}} h_x(z) \, \mathrm{e}^{-2\pi \mathrm{i} N r z} \mathrm{d}z,$$

and, by applying integration by parts and the fact that $g^{(\gamma)}\left(\pm\frac{1}{2}\right) = 0$ for $\gamma = 0, \ldots, \beta - 2$ further,

$$\begin{aligned}
\left| \sum_{k=-\frac{N}{2}}^{\frac{N}{2}} h_x\left(\frac{k}{N}\right) \right| &= \left| N \sum_{r \in \mathbb{Z}} \left(2\pi \mathrm{i} N (r-x)\right)^{-(\beta-1)} \int_{-\frac{1}{2}}^{\frac{1}{2}} g^{(\beta-1)}(z) \, \mathrm{e}^{2\pi \mathrm{i} N z (x-r)} \mathrm{d}z \right| \\
&= \left| \frac{N}{(2\pi \mathrm{i} N)^{\beta}} \sum_{r \in \mathbb{Z}} (r-x)^{-\beta} \int_{-\frac{1}{2}}^{\frac{1}{2}} g^{(\beta-1)}(z) \left(\frac{\mathrm{d}}{\mathrm{d}z} \mathrm{e}^{2\pi \mathrm{i} N z (x-r)} \right) \mathrm{d}z \right| \\
&\leq \frac{1 + |x|^{\beta} \sum_{r \in \mathbb{Z} \setminus \{0\}} |r-x|^{-\beta}}{(2\pi)^{\beta} N^{\beta-1} |x|^{\beta}} \sup_{r_0 \in \mathbb{Z}} \left| \int_{-\frac{1}{2}}^{\frac{1}{2}} g^{(\beta-1)}(z) \left(\frac{\mathrm{d}}{\mathrm{d}z} \mathrm{e}^{2\pi \mathrm{i} N z (x-r_0)} \right) \mathrm{d}z \right|.
\end{aligned}$$

Using $1 + |x|^{\beta} \sum_{r \in \mathbb{Z} \setminus \{0\}} |r-x|^{-\beta} \leq (2^{\beta}-1) 2^{1-\beta} \zeta(\beta)$ for $|x| \leq \frac{1}{2}$ and

$$\left| \int_{-\frac{1}{2}}^{\frac{1}{2}} g^{(\beta-1)}(z) \left(\frac{\mathrm{d}}{\mathrm{d}z} \mathrm{e}^{2\pi \mathrm{i} N z (x-r_0)} \right) \mathrm{d}z \right| \leq \left| g^{(\beta-1)} \right|_V$$

yields the assertion.

By the Poisson summation formula, we note furthermore that

$$\frac{1}{N} \|g\|_{1,N} \geq 1 - \left| \sum_{r \in \mathbb{Z} \setminus \{0\}} \int_{-\frac{1}{2}}^{\frac{1}{2}} g(z) \, \mathrm{e}^{-2\pi \mathrm{i} N r z} \mathrm{d}z \right|$$

and proceed analogously in order to prove the second assertion, where we use $N \geq 2\beta$ to obtain an estimate independent of N. □

THEOREM 3.3. [illegible] $\beta \in \mathbb{N}$, $\beta \geq 2$ [illegible] $g \in BV_0^{\beta-1}$ [illegible] $N \in 2\mathbb{N}$ $N \geq 2\beta$ [illegible]

$$\hat{w}_k = \frac{g\left(\frac{k}{N}\right) + g\left(\frac{k+1}{N}\right)}{2 \|g\|_{1,N}}, \qquad k = -\frac{N}{2}, \ldots, \frac{N}{2} - 1,$$

[illegible] K_N [illegible] 2.1 [illegible]

$$|K_N(x)| \leq \frac{\left(2^{\beta}-1\right) \zeta\left(\beta\right) \left| g^{(\beta-1)} \right|_V}{2^{\beta-1} \left(2\pi\right)^{\beta} - \zeta\left(\beta\right) \beta^{-\beta} \left| g^{(\beta-1)} \right|_V} \frac{1}{N^{\beta} |x|^{\beta}}$$

[illegible] $x \in \left[-\frac{1}{2}, \frac{1}{2}\right] \setminus \{0\}$

. Note first that

$$K_N(x) = \frac{1 + \mathrm{e}^{-2\pi \mathrm{i} x}}{2 \|g\|_{1,N}} \sum_{k=-\frac{N}{2}}^{\frac{N}{2}} g\left(\frac{k}{N}\right) \mathrm{e}^{2\pi \mathrm{i} k x}.$$

Thus, we obtain $K_N(0) = 1$ and by applying Lemma 3.2 also the decay property. □

We apply Theorem 3.3 in the following to the particular class of B-spline kernels.

DEFINITION 3.4. $\beta \in \mathbb{N}$ normalized B-spline

$$g_\beta(z) := \beta N_\beta\left(\beta z + \frac{\beta}{2}\right),$$

N_β β $N_1(z) = 1$ $z \in (0,1)$ $N_1(z) = 0$ $N_{\beta+1}(z) = \int_{z-1}^{z} N_\beta(\tau)\mathrm{d}\tau$ [4]

$\beta \in \mathbb{N}$ $N \in 2\mathbb{N}$ B-spline kernel

$$B_{\beta,N}(x) := \frac{1 + \mathrm{e}^{-2\pi \mathrm{i} x}}{2 \|g_\beta\|_{1,N}} \sum_{k=-\frac{N}{2}}^{\frac{N}{2}} g_\beta\left(\frac{k}{N}\right) \mathrm{e}^{2\pi \mathrm{i} k x}.$$

COROLLARY 3.5. $\beta \in \mathbb{N}$, $\beta \geq 2$ $N \in 2\mathbb{N}$ $N \geq 2\beta$ $B_{\beta,N}(x)$ 3.4

$$|B_{\beta,N}(x)| \leq \frac{\left(2^\beta - 1\right)\zeta(\beta)\,\beta^\beta}{2^{\beta-1}\pi^\beta - \zeta(\beta)} |Nx|^{-\beta}$$

$x \in \left[-\frac{1}{2}, \frac{1}{2}\right] \setminus \{0\}$ $B_{\beta,N}(0) = 1$

. Note that $g_\beta \in BV_0^{\beta-1}$. Using $N_\beta'(z) = N_{\beta-1}(z) - N_{\beta-1}(z-1)$, we conclude

$$\left|g_\beta^{(\beta-1)}\right|_V = \beta^\beta \left|N_\beta^{(\beta-1)}\right|_V = \beta^\beta \left|\sum_{\tau=0}^{\beta-1} (-1)^\tau \binom{\beta-1}{\tau} N_1(\cdot - \tau)\right|_V = (2\beta)^\beta$$

and apply Theorem 3.3. □

Note that in contrast to [15] the order β of the B-spline and the degree $N \in 2\mathbb{N}$ of the kernel $B_{\beta,N}$ are independent of each other. The special case $\beta = 1$, i.e., the "top-hat function" $g_1(z) = 1$ for $|z| < \frac{1}{2}$ and $g_1(z) = 0$ elsewhere, leads to the well-known $B_{1,N}(x) = \frac{1}{N}\sum_{k \in I_N} \mathrm{e}^{2\pi \mathrm{i} k x}$. Analogously, $\beta = 2$, i.e., the "hat function" $g_2(z) = 2 - 4|z|$ for $|z| \leq \frac{1}{2}$ and $g_2(z) = 0$ elsewhere, leads to the . The increasing localization of the B-spline kernels is illustrated in Figure 3.1.

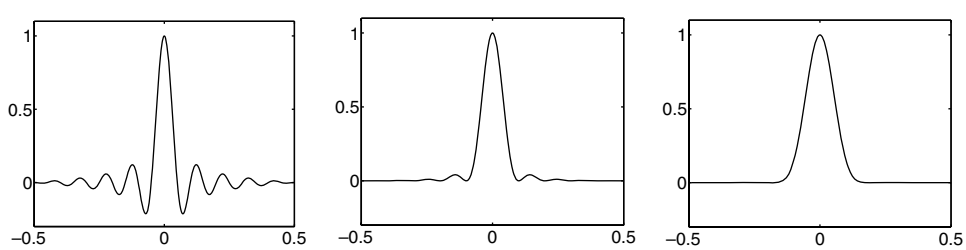


FIGURE 3.1. *From left to right: Real part of the Dirichlet kernel $B_{1,20}$, the Fejér kernel $B_{2,20}$, and the B-spline kernel $B_{4,20}$.*

3.6. If we assume in Corollary 3.5, furthermore, that $N = \beta\sigma$, $\sigma \in \mathbb{N}$, then the stronger estimate $|B_{\beta,N}(x)| \leq 2\zeta(\beta)(1-2^{-\beta})(\frac{\beta}{\pi})^{\beta}|Nx|^{-\beta}$ holds true. This improvement is due to $\|g_\beta\|_{1,N} = N$ in Lemma 3.2 and follows from the partition of unity of the cardinal B-spline N_β and the refinement equation $N_\beta(z) = \sum_{\tau\in\mathbb{Z}} a_\tau^{(\beta,\sigma)} N_\beta(\sigma z - \tau)$ for some finitely supported coefficients $a_\tau^{(\beta,\sigma)} > 0$; see, e.g., [4, p. 8].

In particular, the Fejér kernel fulfills $|B_{2,N}(x)| \leq |Nx|^{-2}$, $N \in 2\mathbb{N}$, which also follows from the estimate $|\sin(\pi x)| \geq 2|x|$ for $|x| \leq 1/2$ and the representation

$$B_{2,N}(x) = \frac{2\left(1+\mathrm{e}^{-2\pi \mathrm{i} x}\right)}{N^2}\left(\frac{\sin\left(\frac{N}{2}\pi x\right)}{\sin(\pi x)}\right)^2.$$

Along the same line follows the localization property $B_{1,N}(x) \leq |Nx|^{-1}$ and $B_{1,N}(0) = 1$ for the Dirichlet kernel.

We complete this section by an extension of our result to the multivariate case $d > 1$. Indeed, tensor products of the kernels constructed in Theorem 3.3 also yield localized multivariate kernels as shown in the following corollary.

COROLLARY 3.7. [illegible] $\tilde{K}_N$ [illegible] 2.1 [illegible] $\beta \in \mathbb{N}$ [illegible] $C_\beta > 1$ [illegible] $x \in \left[-\frac{1}{2},\frac{1}{2}\right] \setminus \{0\}$ [illegible] $|\tilde{K}_N(x)| \leq C_\beta |Nx|^{-\beta}$ [illegible] $K_N(\boldsymbol{x}) = \prod_{t=0}^{d-1} \tilde{K}_N(x_t)$ [illegible] $\boldsymbol{x} \in \left[-\frac{1}{2},\frac{1}{2}\right]^d \setminus \{\boldsymbol{0}\}$ [illegible]

$$|K_N(\boldsymbol{x})| \leq \frac{C_\beta}{N^\beta \|\boldsymbol{x}\|_\infty^\beta}.$$

[illegible] The assertion follows simply from the estimate $|K_N(\boldsymbol{x})| \leq |\tilde{K}_N(\|\boldsymbol{x}\|_\infty)|$. ∎

4. Stability of the interpolation problem. The nonequispaced Fourier matrix $\boldsymbol{A}$ has full rank M for $d = 1$ and $N \geq M$. Unfortunately and due to the famous result of Mairhuber–Curtis (see, e.g., [20, Thm. 2.3]), we cannot expect full rank of the matrix $\boldsymbol{A}$ for $N \geq M^{\frac{1}{d}}$ in the multivariate case $d > 1$.

However, we prove stability results for the trigonometric interpolation problem at q-separated nodes in the univariate case (cf. Theorem 4.1) and the multivariate case (cf. Theorem 4.6). Basically, a localized kernel K_N yields a diagonal-dominated kernel matrix $\boldsymbol{K}_N$ and thus full rank of the nonequispaced Fourier matrix $\boldsymbol{A}$ for q-separated nodes. Furthermore, we prove stability results for a slightly generalized interpolation problem at equispaced nodes and subsets of equispaced nodes in Theorem 4.10. These results are applied to the B-spline kernels from section 3 in Corollaries 4.7 and 4.11.

The univariate setting. The following theorem gives estimates for the extremal eigenvalues of the matrix $\boldsymbol{K}_N$ under reasonable assumptions on the kernel K_N.

THEOREM 4.1. [illegible] $N \in 2\mathbb{N}$ [illegible] K_N [illegible] 2.1 [illegible] $\beta > 1$ [illegible] $x \in \left[-\frac{1}{2},\frac{1}{2}\right] \setminus \{0\}$ [illegible]

$$|K_N(x)| \leq \frac{C_\beta}{N^\beta |x|^\beta}.$$

[illegible] $\mathcal{X}$ [illegible] $q > 0$ [illegible] $\boldsymbol{K}_N$ [illegible]

$$1 - \frac{2\,\zeta(\beta)\,C_\beta}{N^\beta q^\beta} \leq \lambda \leq 1 \leq \Lambda \leq 1 + \frac{2\,\zeta(\beta)\,C_\beta}{N^\beta q^\beta}.$$

. As usual, let M denote the number of nodes in $\mathcal{X}$. Due to $K_N(0) = 1$ (cf. Definition 2.1) we obtain $\operatorname{trace}(\boldsymbol{K}_N) := \sum_{j=0}^{M-1} K_N(0) = M$. Since the trace is invariant under similarity transforms, all eigenvalues sum up to M, and, thus, the inequality $\lambda \le 1 \le \Lambda$ is fulfilled.

Now let $\lambda_\star$ be an arbitrary eigenvalue of $\boldsymbol{K}_N$; then for some index $0 \le j < M$ Gershgorin's circle theorem yields

$$|\lambda_\star - 1| \le \sum_{l=0; l\neq j}^{M-1} |K_N(x_j - x_l)|.$$

By using that the separation distance of the sampling set is q and by the localization of the kernel K_N, we obtain

$$|\lambda_\star - 1| \le \frac{C_\beta}{N^\beta} \sum_{l=0; l\neq j}^{M-1} \frac{1}{|x_j - x_l|^\beta} \le \frac{2\,C_\beta}{N^\beta q^\beta} \sum_{l=1}^{\lfloor M/2 \rfloor} l^{-\beta} < \frac{2\,\zeta(\beta) C_\beta}{N^\beta q^\beta}. \qquad \square$$

Indeed, the kernels constructed in Theorem 3.3 yield well-conditioned matrices $\boldsymbol{K}_N$. We also note that the decay of the Dirichlet kernel allows only for a weaker result.

. 4.2. Let a sampling set $\mathcal{X} \in \mathbb{T}$ with separation distance $q > 0$ be given. Then the application of Theorem 4.1, where the last step of its proof is replaced by the estimate $\sum_{l=1}^{\lfloor M/2 \rfloor} l^{-1} \le 1 + \ln \frac{1}{2q}$, yields: The matrix $(B_{1,N}(x_j - x_l))_{j,l=0,\dots,M-1} = \frac{1}{N} \boldsymbol{A}\boldsymbol{A}^{\mathsf{H}}$ is nonsingular for $N > (1 + |\log(2q)|)\, q^{-1}$. The logarithmic term in this condition is clearly suboptimal.

As an immediate consequence of Theorem 4.1 we state a stability result for an equispaced grid disturbed by jitter.

Corollary 4.3. 4.1 $x_j = -\frac{1}{2} + \frac{j - \varepsilon_j}{M}$, $j = 0, \dots, M-1$, $0 \le \varepsilon_j \le \varepsilon < 1$ $\boldsymbol{K}_N$

$$1 - \frac{2\,\zeta(\beta)\, C_\beta M^\beta}{N^\beta (1-\varepsilon)^\beta} \le \lambda \le 1 \le \Lambda \le 1 + \frac{2\,\zeta(\beta)\, C_\beta M^\beta}{N^\beta (1-\varepsilon)^\beta}.$$

. Since the separation distance is bounded by $q \ge M^{-1}(1-\varepsilon)$, the result follows by Theorem 4.1. $\square$

The multivariate setting. First, we borrow a packing argument on the sphere from [15] and refine it in Lemma 4.5 for the present setting; i.e., we show how many q-separated nodes can be placed in a certain distance to a reference node; see also Figure 4.1.

Definition 4.4. $d \in \mathbb{N}$ q $0 < q \le \frac{1}{2}$

$$R_{q,m} := \left\{ \boldsymbol{x} \in \mathbb{T}^d : mq \le \operatorname{dist}(\boldsymbol{x}, \boldsymbol{0}) < (m+1)\, q \right\}$$

$m = 0, \dots, \lfloor q^{-1}/2 \rfloor - 1$

$$R_{q,\lfloor q^{-1}/2 \rfloor} := \left\{ \boldsymbol{x} \in \mathbb{T}^d : \lfloor q^{-1}/2 \rfloor\, q \le \operatorname{dist}(\boldsymbol{x}, \boldsymbol{0}) \le 1/2 \right\}.$$

$\mathcal{X}$ $R_{\mathcal{X},q,m} := R_{q,m} \cap \mathcal{X}$

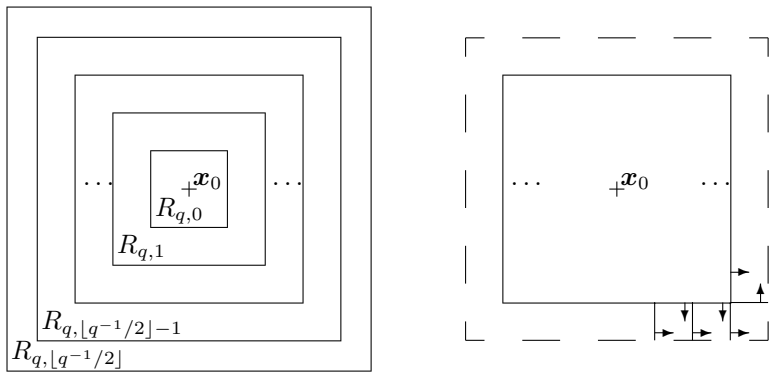


FIGURE 4.1. *Partitioning of the torus $\mathbb{T}^2$ into the rings $R_{q,m}$, $m = 0, \ldots, \lfloor q^{-1}/2 \rfloor$ (left). Further subdivision into shifted and rotated versions of the cube $[0, q)^d$, where arrows indicate the "ownership" of the faces to a particular cube (right).*

LEMMA 4.5. [illegible] $d \in \mathbb{N}$ [illegible] q [illegible] $\mathcal{X}$ [illegible] $0 < q \leq \frac{1}{2}$ [illegible] $R_{\mathcal{X},q,m}$ [illegible]

$$|R_{\mathcal{X},q,m}| \leq 2^d \left(2^d - 1\right) m^{d-1}, \qquad m = 1, \ldots, \left\lfloor q^{-1}/2 \right\rfloor.$$

[illegible]. We use a packing argument for the partition $\{R_{q,m}, m = 0, \ldots, \lfloor q^{-1}/2 \rfloor\}$ of the torus $\mathbb{T}^d$. Each ring $R_{q,m}$ is subdivided into shifted and rotated versions of the cube $[0, q)^d$; cf. Figure 4.1 (right). This is done such that each point in $R_{q,m}$ is contained in at least one of these boxes and the boxes share no interior points with each other. Every box contains at most one node of the sampling set, and, hence, the estimate

$$|R_{\mathcal{X},q,m}| \leq \frac{1}{q^d} \int_{R_{q,m}} \mathrm{d}\boldsymbol{x} \leq 2^d \left((m+1)^d - m^d\right) = 2^d \sum_{t=1}^{d} \binom{d}{t} m^{d-t} \leq 2^d m^{d-1} \sum_{t=1}^{d} \binom{d}{t}.$$

is valid. □

Using localized kernels in conjunction with a separated sampling set and Lemma 4.5, we state the following theorem on the stability of the interpolation problem.

THEOREM 4.6. [illegible] $d \in \mathbb{N}$ $N \in 2\mathbb{N}$ [illegible] K_N [illegible] 2.1 [illegible] $\beta > d$ [illegible] $\boldsymbol{x} \in \left[-\frac{1}{2}, \frac{1}{2}\right]^d \setminus \{\mathbf{0}\}$ [illegible]

$$|K_N(\boldsymbol{x})| \leq \frac{C_\beta}{N^\beta \|\boldsymbol{x}\|_\infty^\beta}.$$

[illegible] $\mathcal{X}$ [illegible] $0 < q \leq \frac{1}{2}$ [illegible] $\boldsymbol{K}_N$ [illegible]

$$1 - \frac{2^d \left(2^d - 1\right) \zeta\left(\beta - d + 1\right) C_\beta}{N^\beta q^\beta} \leq \lambda \leq 1 \leq \Lambda \leq 1 + \frac{2^d \left(2^d - 1\right) \zeta\left(\beta - d + 1\right) C_\beta}{N^\beta q^\beta}.$$

[illegible]. Let $\lambda_\star$ be an arbitrary eigenvalue of $\boldsymbol{K}_N$. Without loss of generality, let the diagonal element of the matrix $\boldsymbol{K}_N$ used in Gershgorin's circle theorem correspond to $\boldsymbol{x}_0 = \mathbf{0}$. Then we conclude by $K_N(\mathbf{0}) = 1$ (cf. Definition 2.1) that

$$|\lambda_\star - 1| \leq \sum_{l=1}^{M-1} |K_N(\mathbf{0} - \boldsymbol{x}_l)|.$$

Using the partition from Definition 4.4, Lemma 4.5, and the localization of the kernel K_N, we get

$$\begin{aligned}|\lambda_\star - 1| &\leq \sum_{m=1}^{\lfloor q^{-1}/2 \rfloor} \sum_{\boldsymbol{x}_l \in R_{\mathcal{X},q,m}} |K_N(-\boldsymbol{x}_l)| \\ &\leq \frac{2^d\left(2^d-1\right) C_\beta}{N^\beta} \sum_{m=1}^{\lfloor q^{-1}/2 \rfloor} m^{d-1} \max_{\boldsymbol{x}\in R_{q,m}} \|\boldsymbol{x}\|_\infty^{-\beta} \\ &\leq \frac{2^d\left(2^d-1\right)\zeta(\beta-d+1)\, C_\beta}{N^\beta q^\beta}. \qquad \square\end{aligned}$$

Particularly, this result includes Theorem 4.1 if we set $d = 1$.

COROLLARY 4.7. [illegible] $d \in \mathbb{N}$ [illegible] $\mathcal{X} \in \mathbb{T}^d$ [illegible] $0 < q \leq \frac{1}{2}$ [illegible] $N \in 2\mathbb{N}$ $N > 2dq^{-1}$ [illegible] $\boldsymbol{A}$ [illegible] $\boldsymbol{K}_N = \boldsymbol{A}\hat{\boldsymbol{W}}\boldsymbol{A}^{\mathsf{H}}$ [illegible] $\beta = d+1$ [illegible]

$$0 < 1 - \left(\frac{2d}{Nq}\right)^{d+1} \leq \lambda \leq 1 \leq \Lambda \leq 1 + \left(\frac{2d}{Nq}\right)^{d+1}.$$

[illegible]. Note first that $N \geq 2\beta$. We apply Theorem 4.6 where we use the estimates for C_β given in Corollary 3.5 and simplify the involved constant. Hence, the full rank of $\boldsymbol{A}$ follows. $\square$

Thus, we have shown that the optimal trigonometric interpolation problem at q-separated nodes in d dimensions obeys a uniformly bounded condition number for a polynomial degree $N > 2dq^{-1}$ and appropriate damping factors. The dependence on q^{-1} is optimal as the subsequent analysis of the equispaced case shows. However, the constant $2d$ is not optimal for high spatial dimensions. As pointed out in [2] for the related approximation problem (1.2), it is an open problem to improve on this.

In summary, Lemma 2.3 and Corollary 4.7 assure in our situation a prescribed reduction of the error $\|\hat{\boldsymbol{e}}_l\|_{\hat{\boldsymbol{W}}^{-1}}$ in a constant number of iterations. Hence, if we assume an additional uniformity condition $q = cM^{-\frac{1}{d}}$ for the sampling set $\mathcal{X}$, the total arithmetical complexity of Algorithm 1 for solving (1.3) up to a prescribed error is bounded by $\mathcal{O}(M \log M)$.

[illegible] 4.8. Other frequently applied kernels also possess specific localization properties and, thus, yield stable interpolation at q-separated nodes by means of Theorem 4.6 as follows:

We define for $\beta \in 2\mathbb{N}$, $\sigma \in \mathbb{N}$, and $N = \beta(\sigma-1)+2$ the Jackson kernel by

$$J_{\beta,N}(x) := \frac{1+\mathrm{e}^{-2\pi \mathrm{i} x}}{2\sigma^\beta}\left(\frac{\sin(\sigma\pi x)}{\sin(\pi x)}\right)^\beta.$$

Being a normalized power of the Fejér kernel $B_{2,2\sigma}$, the coefficients $\hat{w}_k$ of the Jackson kernel can be obtained by an iterated discrete convolution of the coefficients of the Fejér kernel; see [1] for details. In contrast, the B-spline kernel relies on a continuous convolution. The Jackson kernel is localized as $|J_{\beta,N}(x)| \leq (\frac{\beta}{2})^\beta |Nx|^{-\beta}$ for $x \in \left[-\frac{1}{2}, \frac{1}{2}\right] \setminus \{0\}$ and fulfills $J_{\beta,N}(0) = 1$. Hence, the tensor product Jackson kernel of appropriate order yields for $N \in 2\mathbb{N}$, $N > 2.1dq^{-1}$, the nonsingular kernel matrix

$$\left(J_{2\lceil \frac{d+1}{2} \rceil, N}(\boldsymbol{x}_j - \boldsymbol{x}_l)\right)_{j,l=0,\ldots,M-1}.$$

Second, it is well known that the weight $1+(2\pi k)^{2\alpha}$ is associated to the squared Sobolev norm $\|f\|_2^2+\|f^{(\alpha)}\|_2^2$. For $\beta\in\mathbb{N}$ and $\alpha,\gamma>0$, a regularized and slightly generalized weight is given by

$$g_{\alpha,\beta,\gamma}(z):=c_{\alpha,\beta,\gamma}\frac{\left(\frac{1}{4}-z^2\right)^{\beta}}{\gamma+|z|^{2\alpha}}$$

for $|z|\le\frac{1}{2}$ and $g_{\alpha,\beta,\gamma}(z)=0$ elsewhere, where the constant $c_{\alpha,\beta,\gamma}$ is chosen such that $\|g_{\alpha,\beta,\gamma}\|_{L_1}=1$. Here, the denominator generalizes the weight $1+(2\pi k)^{2\alpha}$, and the nominator ensures $g_{\alpha,\beta,\gamma}\in BV_0^{\beta-1}$. We define for $N\in 2\mathbb{N}$ the Sobolev kernel by

$$S_{\alpha,\beta,\gamma,N}(x):=\frac{1+\mathrm{e}^{-2\pi\mathrm{i}x}}{2\,\|g_{\alpha,\beta,\gamma}\|_{1,N}}\sum_{k=-\frac{N}{2}}^{\frac{N}{2}}g_{\alpha,\beta,\gamma}\left(\frac{k}{N}\right)\mathrm{e}^{2\pi\mathrm{i}kx}.$$

The kernel is localized as $|S_{\alpha,\beta,\gamma,N}(x)|\le\tilde{c}_{\alpha,\beta,\gamma}|Nx|^{-\beta}$ for $x\in\left[-\frac{1}{2},\frac{1}{2}\right]\setminus\{0\}$ and some constant $\tilde{c}_{\alpha,\beta,\gamma}>0$ and fulfills $S_{\alpha,\beta,\gamma,N}(0)=1$.

Results for equispaced nodes. In the case of equispaced nodes we employ the fact that the matrix $\boldsymbol{K}_N$ is circulant. We present a slightly generalized result in Theorem 4.10.

DEFINITION 4.9. [illegible] $d,n\in\mathbb{N}$ [illegible] $\hat{w}_{\boldsymbol{k}}\in\mathbb{R}$, $\boldsymbol{k}\in\mathbb{Z}^d$ [illegible] $\sum_{\boldsymbol{k}\in\mathbb{Z}^d}|\hat{w}_{\boldsymbol{k}}|<\infty$ [illegible]

$$K(\boldsymbol{x}):=\sum_{\boldsymbol{k}\in\mathbb{Z}^d}\hat{w}_{\boldsymbol{k}}\mathrm{e}^{2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}}$$

[illegible] $\boldsymbol{j}=(j_0,\ldots,j_{d-1})^{\top}\in I_n$ [illegible]

$$\boldsymbol{K}:=\left(K\left(\frac{\boldsymbol{j}-\boldsymbol{l}}{n}\right)\right)_{\boldsymbol{j},\boldsymbol{l}\in I_n}\in\mathbb{C}^{n^d\times n^d}.$$

THEOREM 4.10. [illegible] $\boldsymbol{K}$ [illegible] 4.9 [illegible]

$$\lambda_{\boldsymbol{s}}(\boldsymbol{K})=n^d\sum_{\boldsymbol{r}\in\mathbb{Z}^d}\hat{w}_{\boldsymbol{s}+n\boldsymbol{r}}$$

[illegible] $\boldsymbol{s}\in I_n$ [illegible] $\hat{w}_{\boldsymbol{k}}=\prod_{t=0}^{d-1}\hat{w}_{k_t}$ $\sum_{k\in\mathbb{Z}}|\hat{w}_k|<\infty$ [illegible]

$$\lambda_{\boldsymbol{s}}(\boldsymbol{K})=n^d\prod_{t=0}^{d-1}\sum_{r_t\in\mathbb{Z}}\hat{w}_{s_t+nr_t}$$

[illegible] $\boldsymbol{s}\in I_n$ [illegible]

$$\boldsymbol{K}_{\Gamma}:=\left(K\left(\frac{\boldsymbol{j}-\boldsymbol{l}}{n}\right)\right)_{\boldsymbol{j},\boldsymbol{l}\in\Gamma}$$

[illegible] $\boldsymbol{K}$ [illegible] $\Gamma\subset I_n$

. The matrix $\boldsymbol{K}$ is multilevel circulant and thus diagonalized by the Fourier matrix $\boldsymbol{F}_n = (\mathrm{e}^{2\pi \mathrm{i} \boldsymbol{k}\boldsymbol{j}/n})_{\boldsymbol{j},\boldsymbol{k}\in I_n}$. We calculate

$$\begin{aligned}\left(\boldsymbol{F}_n^{\mathsf{H}} \boldsymbol{K} \boldsymbol{F}_n\right)_{\boldsymbol{s},\boldsymbol{t}} &= \sum_{\boldsymbol{j},\boldsymbol{l}\in I_n} \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{s}\boldsymbol{j}/n} K\left(\frac{\boldsymbol{j}-\boldsymbol{l}}{n}\right) \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{t}\boldsymbol{l}/n} \\ &= \sum_{\boldsymbol{k}\in\mathbb{Z}^d} \hat{w}_{\boldsymbol{k}} \sum_{\boldsymbol{j}\in I_n} \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{j}(\boldsymbol{s}-\boldsymbol{k})/n} \sum_{\boldsymbol{l}\in I_n} \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{l}(\boldsymbol{t}-\boldsymbol{k})/n}\end{aligned}$$

for $\boldsymbol{s}, \boldsymbol{t} \in I_n$ and use

$$\sum_{\boldsymbol{j}\in I_n} \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{j}(\boldsymbol{s}-\boldsymbol{k})/n} = \begin{cases} n^d & \text{if } \ \frac{\boldsymbol{s}-\boldsymbol{k}}{n} \in \mathbb{Z}^d, \\ 0 & \text{otherwise.} \end{cases}$$

See also [15, Cor. 3.10, Thm. 3.11] for the univariate case.

The second assertion is due to the Kronecker product structure of the matrix $\boldsymbol{K}$ in the case of tensor product kernels. The last assertion follows from the fact that removing a node is nothing else than removing its corresponding row and column in $\boldsymbol{K}$ and from the interlacing property for eigenvalues; see [11, p. 185]. ∎

Now let $d, n \in \mathbb{N}$, $N \in 2\mathbb{N}$, and the equispaced sampling set $\mathcal{X} = \frac{1}{n} I_n \subset \mathbb{T}^d$ be given. A simple consequence of Theorem 4.10 is the fact that the kernel matrix $\boldsymbol{K}_N = \frac{1}{N} \boldsymbol{A}\boldsymbol{A}^{\mathsf{H}}$ is singular whenever $N < q^{-1} = n$. Hence, the condition $N > 2dq^{-1}$ for the full rank of $\boldsymbol{A}$ is optimal with respect to q. Nevertheless, we also apply Theorem 4.10 to obtain positive results in the equispaced setting for the Dirichlet and the Fejér kernels.

COROLLARY 4.11. $d \in \mathbb{N}$ $\mathcal{X}$ $M \geq 2$ $N \in 2\mathbb{N}$ $N > q^{-1} = M^{\frac{1}{d}}$ $B_{1,N}$ $d > 1$ $\boldsymbol{K}_N$

$$\left(\frac{\lfloor Nq \rfloor}{Nq}\right)^d = \lambda \leq 1 \leq \Lambda = \left(\frac{\lceil Nq \rceil}{Nq}\right)^d.$$

$B_{2,N}$

$$\left(1 - \frac{1}{N^2 q^2}\right)^d \leq \lambda \leq 1 \leq \Lambda \leq \left(1 + \frac{1}{N^2 q^2}\right)^d,$$

$N = (2\sigma+1)\, q^{-1}$, $\sigma \in \mathbb{N}$ K_N

. Throughout this proof, let $n = q^{-1}$ and the damping factors be extended by $\hat{w}_{\boldsymbol{k}} = 0$ for $\boldsymbol{k} \notin I_N$. We apply for $d = 1$ the first statement of Theorem 4.10 to the weights $\hat{w}_k = \frac{1}{N}$, $k \in I_N$, of the Dirichlet kernel for $N \in 2\mathbb{N}$ and to the weights $\hat{w}_k = \frac{2}{N}(1 - \frac{|2k+1|}{N})$, $k \in I_N$, of the Fejér kernel, if $N = (2\sigma+1)\, q^{-1}$, $\sigma \in \mathbb{N}$, respectively.

The assertion is a little more delicate for the univariate Fejér kernel and $N \neq (2\sigma+1)\, q^{-1}$. We use the representation

$$|B_{2,N}(x)| = \frac{4}{N^2} \sum_{r=0}^{\frac{N}{2}-1} \sum_{k=-r}^{r} \mathrm{e}^{2\pi \mathrm{i} k x}.$$

Now let $\lambda_\star$ be an arbitrary eigenvalue of the kernel matrix $\boldsymbol{K}_N$; then Gershgorin's circle theorem yields

$$|\lambda_\star - 1| \le \sum_{l=1}^{n-1} \left| B_{2,N}\left(\frac{l}{n}\right)\right| = \frac{4n}{N^2} \sum_{r=0}^{\frac{N}{2}-1} \left(2\left\lfloor \frac{r}{n}\right\rfloor + 1\right) - 1.$$

Since for $Q := \left\lfloor \frac{N-2}{2n}\right\rfloor$ and $R := \frac{N}{2} - 1 - nQ$ the identity

$$\sum_{r=0}^{\frac{N}{2}-1} \left\lfloor \frac{r}{n}\right\rfloor = \sum_{s=0}^{Q-1} \sum_{r=sn}^{(s+1)n-1} s + \sum_{r=nQ}^{nQ+R} Q = \frac{(N-n)^2 - (2(R+1)-n)^2}{8n}$$

holds, we proceed:

$$\begin{aligned}\frac{4n}{N^2} \sum_{r=0}^{\frac{N}{2}-1} \left(2\left\lfloor \frac{r}{n}\right\rfloor + 1\right) - 1 &= \frac{4n}{N^2}\left(2\frac{(n-N)^2 - (2(R+1)-n)^2}{8n} + \frac{N}{2}\right) - 1 \\ &= \frac{n^2}{N^2} - \left(\frac{2(R+1)-n}{N}\right)^2 \\ &\le \frac{n^2}{N^2}.\end{aligned}$$

The case $d > 1$ is due to the second statement in Theorem 4.10. ∎

5. Numerical results. In this section, we exemplify our findings on the stability of the optimal interpolation problem (1.3) and its iterative solution by Algorithm 1.

The estimates for the condition number of the kernel matrix $\boldsymbol{K}_N$ for equispaced nodes (cf. Corollary 4.11) are shown in Figure 5.1 (left). For $Nq \in \mathbb{N}$ and the Dirichlet kernel $B_{1,N}$ the matrix $\boldsymbol{K}_N$ is just the identity. However, using the better localized Fejér kernel $B_{2,N}$ improves the condition number already for $N > \sqrt{3}q^{-1}$ when $Nq \notin \mathbb{N}$.

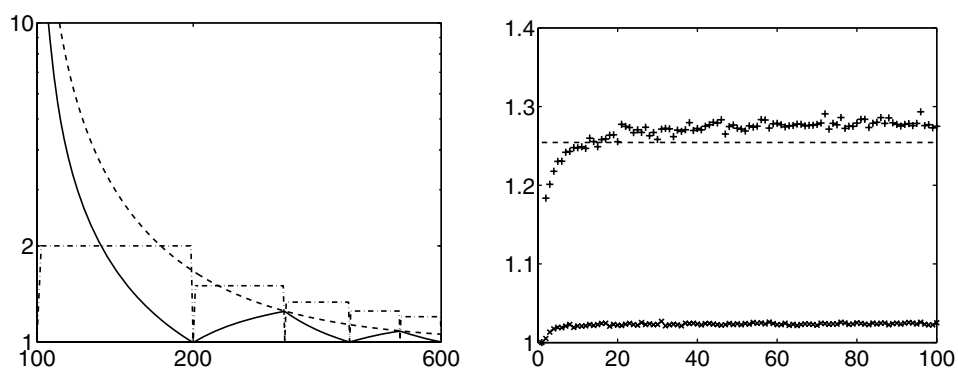


FIGURE 5.1. *Condition number of the kernel matrix $\boldsymbol{K}_N$. Left: Condition number with respect to polynomial degree $N = 100, \dots, 600$, no weights, i.e., Dirichlet kernel (dashed-dotted line); weight function g_2, i.e., Fejér kernel (solid line), and the estimate of Corollary* 4.11 (dashed line)*; here the number of equispaced nodes is $M = 100$. Right: Condition number with respect to the number of nodes $M = 1, \dots, 100$; the nodes are equispaced perturbed by $\varepsilon_{rel.} = 0.1$ jitter error; and the polynomial degree is $N = 6M$; no weights, i.e., Dirichlet kernel ($+$), weight function g_2, i.e., Fejér kernel ($\times$), and its estimate by Corollary* 4.3 *(dashed line).*

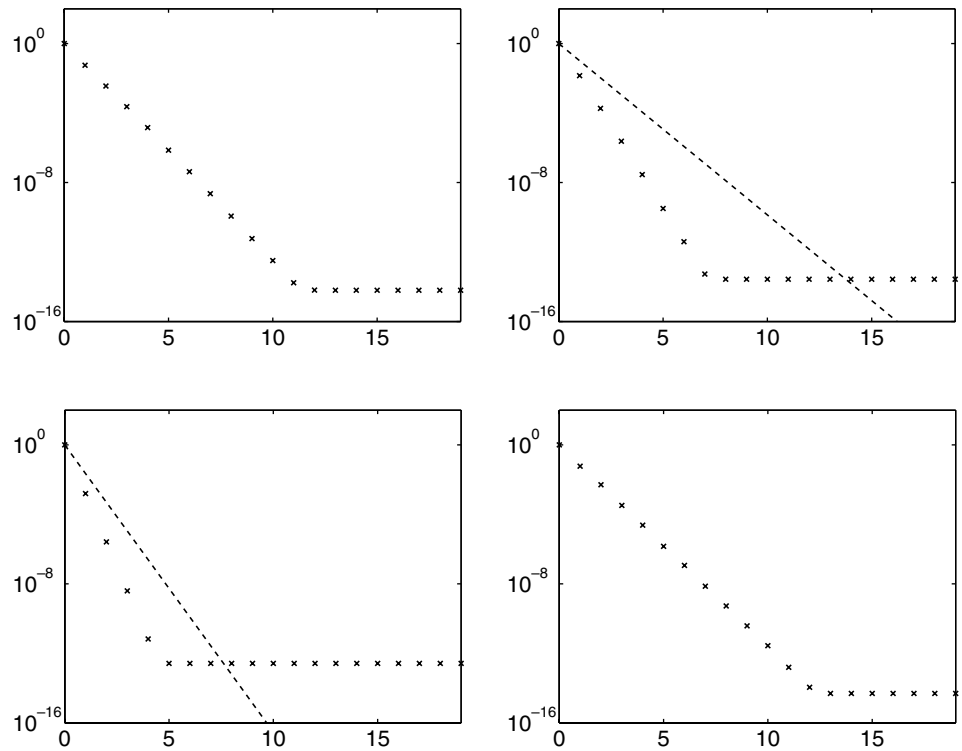


FIGURE 5.2. *Native error* $\|\hat{\boldsymbol{f}}_l - \hat{\boldsymbol{W}}\boldsymbol{A}^{\mathsf{H}}\boldsymbol{K}_N^{-1}\boldsymbol{y}\|_{\hat{W}^{-1}}$ *for the univariate interpolation problem with respect to the current iteration* l*. The number of samples is* $M = 100$*, the number of computed Fourier coefficients is* $N = 1000$*, and the separation distance of the nodes is* $q = 4 \times 10^{-3}$*. Top left: no weights, i.e., Dirichlet kernel. Top right: weight function* g_2*, i.e., Fejér kernel, and the predicted decay rate (dashed line). Bottom left: weight function* g_4*, i.e., B-spline kernel, and the predicted decay rate (dashed line). Bottom right: weight function* $g_{1,2,10^{-2}}$*, i.e., Sobolev kernel.*

We present the effect on the stability of the interpolation problem when the equispaced nodes are perturbed by jitter error (cf. Corollary 4.3) in Figure 5.1 (right). We choose different sampling sets of size $M = 1, \ldots, 100$ with equispaced nodes disturbed by 10% jitter error and evaluate the maximum condition number over 100 reruns for the Dirichlet kernel $B_{1,6M}$ and the Fejér kernel $B_{2,6M}$, respectively. The Fejér kernel produces a lower condition number, which is also validated by the shown upper bound. These results confirm the theoretical results of Corollaries 4.11 and 4.3.

Furthermore, we apply Algorithm 1 using the NFFT software package [12] to reconstruct a univariate signal from randomly scattered data in Figure 5.2 and show in Figure 5.3 the reconstruction of a bivariate signal from a glacier data set [8]. The main tool in our iterative algorithms is the NFFT, i.e., the fast matrix times vector multiplication with $\boldsymbol{A}$ and $\boldsymbol{A}^{\mathsf{H}}$, respectively. Details concerning NFFT algorithms can be found, for example, in [16], and a software package can be found in [12].

The reconstruction of the randomly sampled univariate signal shows the decay rates of our iterative scheme. The sampling set consists of $M = 100$ nodes separated by $q = 4 \times 10^{-3}$, and we reconstruct with a polynomial degree $N = 1000$ and the Dirichlet, Fejér, B-spline, and Sobolev kernels. All schemes converge within 15 iterations where this is justified only for the Fejér and the B-spline kernels.

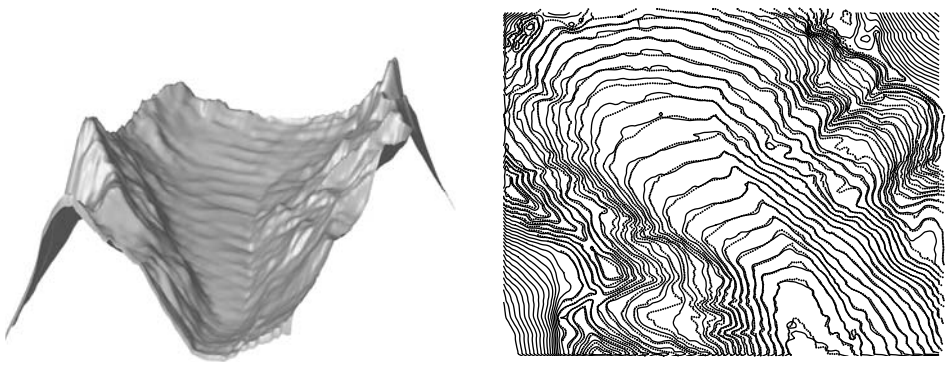


FIGURE 5.3. *Reconstruction of the glacier data set* `vol87.dat` *from* [8], $M = 8345$ *nodes,* $N = 256$, 40 *iterations, tensor product damping factors* $\hat{w}_{\mathrm{k}}$ *to the weight function* $g_{\frac{1}{2},3,10^{-3}}$; *see* `glacier` *in* [12]. *Left: Surface plot. Right: Contour plot and sampling set* $(\cdot)$.

TABLE 5.1

Cross-validation of the reconstructions. The parameters of the CGNE scheme are as before. Moreover, we show the residuals of the CGNR scheme for the least squares problem (1.2) *with* $N = 256$ *(underdetermined) and* $N = 64$ *(overdetermined).*

	CGNE		CGNR ($N = 256$)		CGNR ($N = 64$)	
$\tilde{M}$	r	$\tilde{r}$	r	$\tilde{r}$	r	$\tilde{r}$
200	$6.9e-04$	$1.7e-02$	$5.0e-04$	$1.4e-01$	$8.3e-03$	$1.7e-02$
400	$4.7e-04$	$2.3e-02$	$5.0e-04$	$2.0e-01$	$8.3e-03$	$2.3e-02$
600	$5.7e-04$	$2.9e-02$	$5.1e-04$	$2.5e-01$	$8.1e-03$	$2.9e-02$
800	$4.7e-04$	$3.4e-02$	$5.0e-04$	$2.8e-01$	$8.0e-03$	$3.4e-02$
1000	$4.6e-04$	$3.8e-02$	$4.7e-04$	$3.2e-01$	$8.0e-03$	$3.8e-02$

The last example shows a typical test case known in radial basis function methods. We reconstruct from a data set of $M = 8345$ samples on level curves of a glacier a total number of $2^8 \times 2^8 \approx 8M$ Fourier coefficients. Note, however, that the sampling set is highly nonuniform in the sense that the separation distance is very small compared to the mesh norm. The assumptions of Theorem 4.6 are not fulfilled. Nevertheless, the proposed method yields a very good approximation to the given data after 40 iterations, which is also supported by the cross validation test in Table 5.1. Here we exclude $\tilde{M}$ randomly chosen samples $\tilde{\mathcal{X}} \subset \mathcal{X}$, $\boldsymbol{y}_{\tilde{\mathcal{X}}} \in \mathbb{C}^{\tilde{M}}$ from the reconstruction process and compare our approximations at these left-out nodes. The comparison is done by means of the relative data residual and the relative validation residual after 40 iterations:

$$r := \frac{\|\boldsymbol{y}_{\mathcal{X}\setminus\tilde{\mathcal{X}}} - \boldsymbol{A}_{\mathcal{X}\setminus\tilde{\mathcal{X}}}\hat{\boldsymbol{f}}_{40}\|_2}{\|\boldsymbol{y}\|_2}, \qquad \tilde{r} := \frac{\|\boldsymbol{y}_{\tilde{\mathcal{X}}} - \boldsymbol{A}_{\tilde{\mathcal{X}}}\hat{\boldsymbol{f}}_{40}\|_2}{\|\boldsymbol{y}\|_2}.$$

As can be readily seen, the CGNE scheme achieves both a small data residual r and a small validation residual $\tilde{r}$. The proposed CGNE method combines the good data fit of the CGNR scheme ($N = 256$) with the smooth approximation of the CGNR scheme ($N = 64$).

6. Conclusion. We have shown that the optimal trigonometric interpolation problem at q-separated nodes in d dimensions is well conditioned for a polynomial

degree $N > 2dq^{-1}$. However, in our further extensive numerical examples we observe that for $N \sim M^{1/d}$ one can expect fast convergence of Algorithm 1. If we assume furthermore a uniformity condition $q = cM^{-\frac{1}{d}}$ for the sampling set $\mathcal{X}$ of cardinality M, then the total arithmetical complexity for solving the interpolation problem (1.3) up to a prescribed error is of order $\mathcal{O}(M \log M)$.

We remark that, dependent on the application, one solves the weighted approximation problem (1.2) or the optimal interpolation problem (1.3). Under some further mild conditions on the sampling set, both problems are solved efficiently by means of the conjugate gradient method in conjunction with the nonequispaced FFT.

Acknowledgment. We thank the referees for their valuable suggestions.

REFERENCES

[1] V. G. Alekseev, *Jackson– and Jackson–Vallee Poussin-type kernels and their probability applications*, Theory Probab. Appl., 41 (1993), pp. 137–143.
[2] R. F. Bass and K. Gröchenig, *Random sampling of multivariate trigonometric polynomials*, SIAM J. Math. Anal., 36 (2004), pp. 773–795.
[3] A. Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, 1996.
[4] C. K. Chui, *Multivariate Splines*, SIAM, Philadelphia, 1988.
[5] I. Daubechies, M. Defrise, and C. D. Mol, *An iterative thresholding algorithm for linear inverse problems with a sparsity constraint*, Comm. Pure Appl. Math., 57 (2004), pp. 1413–1457.
[6] H. Fassbender, *On numerical methods for discrete least-squares approximation by trigonometric polynomials*, Math. Comp., 66 (1997), pp. 719–741.
[7] H. G. Feichtinger, K. Gröchenig, and T. Strohmer, *Efficient numerical methods in non-uniform sampling theory*, Numer. Math., 69 (1995), pp. 423–440.
[8] R. Franke, http://www.math.nps.navy.mil/∼rfranke/README.
[9] D. Grishin and T. Strohmer, *Fast multi-dimensional scattered data approximation with Neumann boundary conditions*, Linear Algebra Appl., 391 (2004), pp. 99–123.
[10] K. Gröchenig, *Reconstruction algorithms in irregular sampling*, Math. Comp., 59 (1992), pp. 181–194.
[11] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, 1985.
[12] S. Kunis and D. Potts, *NFFT, Softwarepackage, C subroutine library*, http://www.tu-chemnitz.de/∼potts/nfft, 2002–2006.
[13] S. Kunis and H. Rauhut, *Random Sampling of Sparse Trigonometric Polynomials* II - *Orthogonal Matching Pursuit Versus Basis Pursuit*, Preprint 06-06, TU-Chemnitz, 2006.
[14] H. N. Mhaskar and J. Prestin, *On the detection of singularities of a periodic function*, Adv. Comput. Math., 12 (2000), pp. 95–131.
[15] F. J. Narcowich, N. Sivakumar, and J. D. Ward, *Stability results for scattered-data interpolation on Euclidean spheres*, Adv. Comput. Math., 8 (1998), pp. 137–163.
[16] D. Potts, G. Steidl, and M. Tasche, *Fast Fourier transforms for nonequispaced data: A tutorial*, in Modern Sampling Theory: Mathematics and Applications, J. J. Benedetto and P. J. S. G. Ferreira, eds., Birkhäuser, Boston, 2001, pp. 247–270,
[17] M. Rauth and T. Strohmer, *Smooth approximation of potential fields from noisy scattered data*, Geophysics, 63 (1998), pp. 85–94.
[18] L. Reichel, G. S. Ammar, and W. B. Gragg, *Discrete least squares approximation by trigonometric polynomials*, Math. Comp., 57 (1991), pp. 273–289.
[19] G. Steidl, *A note on fast Fourier transforms for nonequispaced grids*, Adv. Comput. Math., 9 (1998), pp. 337–353.
[20] H. Wendland, *Scattered Data Approximation*, Cambridge Monogr. Appl. Comput. Math., Cambridge University Press, Cambridge, 2005.

SIAM J. SCI. COMPUT.
Vol. 29, No. 4, pp. 1420–1438
© 2007 Society for Industrial and Applied Mathematics

A FAST ALGORITHM FOR THE CALCULATION OF THE ROOTS OF SPECIAL FUNCTIONS*

ANDREAS GLASER†, XIANGTAO LIU†, AND VLADIMIR ROKHLIN‡

Abstract. We describe a procedure for the determination of the roots of functions satisfying second-order ordinary differential equations, including the classical special functions. The scheme is based on a combination of the Prüfer transform with the classical Taylor series method for the solution of ordinary differential equations and requires $O(1)$ operations for the determination of each root. When the functions in question are classical orthogonal polynomials (Legendre, Hermite, etc.), the techniques presented here also provide tools for the evaluation of the weights for the associated Gaussian quadratures. The performance of the scheme for several classical special functions (prolate spheroidal wave functions, Bessel functions, and Legendre, Hermite, and Laguerre polynomials) is illustrated with numerical examples.

Key words. roots, special functions, Gaussian quadratures, Sturm–Liouville

AMS subject classifications. 65D20, 65D30

DOI. 10.1137/06067016X

1. Introduction.

Zeros of classical special functions play an important role in the mathematical sciences, being related to Gaussian quadratures (both classical and generalized), resonances in mechanical and electrical systems, quantum mechanical calculations, etc. As a result, algorithms for the determination of roots of functions of this type are quite well developed; most of them are based on the existence of three-term recurrence relations for the classical special functions and the associated connection to the eigenvalues of certain tridiagonal matrices (see, for example, [6, 9, 2]). The latter are dealt with via well-understood and reliable techniques, such as the QR iteration.

While such methods are quite efficient for small-scale problems, they require order n^2 operations, where n is the number of roots to be computed. Relatively recently, algorithms for the diagonalization of tridiagonal matrices were introduced with CPU time requirements proportional to $n \cdot \log(n)$ (see [7]); a detailed description of the resulting fast root-finding scheme is given in [13]. Another relatively recent development involves straightforward root-finding algorithms applied to the orthogonal polynomials (see [15, 11]); this leads to more efficient schemes producing nodes and weights to higher precision. However, the cost of the resulting procedures is still of the order n^2, albeit with a smaller constant. The scheme of this paper is a modification of this latter approach using the solution of the underlying ordinary differential equation (ODE) as a tool for the evaluation of the special function whose roots are to be found.

The subject of this paper is the observation that the roots of classical special functions (and of many others) can be found for a cost proportional to n, using the

*Received by the editors September 18, 2006; accepted for publication (in revised form) March 15, 2007; published electronically June 12, 2007. This work was supported in part by the U.S. Department of Defense under DARPA grant FA9550-06-1-0239 and AFOSR grant FA9550-06-1-0197. This work was released previously as Technical report YALEU/DCS/TR-1367, Yale University, New Haven, CT, 2006.

http://www.siam.org/journals/sisc/29-4/67016.html

†Program of Applied Mathematics, Yale University, New Haven, CT 06511 (andreas.glaser@yale.edu, xiangtao.liu@yale.edu).

‡Departments of Computer Science and Mathematics, Yale University, New Haven, CT 06511 (vladimir.rokhlin@yale.edu).

fact that such functions satisfy ordinary differential equations of the form

$$p(x)\,\frac{d^2u}{dx^2}(x) + q(x)\,\frac{du}{dx}(x) + r(x)\,u(x) = 0, \tag{1}$$

where $p(x)$, $q(x)$, and $r(x)$ are polynomials of degree 2. Using classical tools such as Taylor series and Prüfer transformation, our approach leads to remarkably simple and efficient schemes for the construction of classical Gaussian quadratures, roots of Bessel and prolate spheroidal wave functions, and roots of other functions satisfying a second-order differential equation with polynomial coefficients.

The numerical procedure of this paper uses two steps to find each root (except the first one, which has to be determined by some other means). In the first step, an approximation to the root is found via the (numerical) solution of the Cauchy problem for an auxiliary differential equation; the equation is obtained via the so-called Prüfer transform (see, for example, [3]), and the initial condition is provided by the preceding root. In the second step, the root is found via the Newton method, coupled with a Taylor series-based scheme for the solution of the original equation (1).

It is worth noting that each of the two approaches [illegible] leads to a scheme with an asymptotically optimal CPU time estimate (i.e., requiring order n operations to determine n roots); we have implemented and tested such schemes. However, each of the two schemes has its drawbacks. The Taylor-based solution of the ODE (1) is extremely accurate and efficient but requires a local search (Newton, bisection, etc.) to find the root, and local search algorithms require initial values. Constructing such initial values is not very expensive, and the subject is quite well understood; on the other hand, schemes of this type tend to be problem-dependent or to require highly adaptive search algorithms.

In contrast, the Prüfer-based approach finds the root as a solution of a Cauchy problem for an ODE, obviating the need for a search. However, the Prüfer ODE has to be solved numerically and is not susceptible to high-order Taylor-based techniques. Thus, obtaining a high-accuracy solution tends to be relatively expensive (though the asymptotic cost is $O(n)$). In this paper we combine the two approaches, solving the Prüfer ODE to a low accuracy (2 to 4 digits) via a cheap low-order scheme and using the obtained solution as a starting point for a Taylor-based Newton method. However, in the case of classical Gaussian quadratures we eschew the use of the Prüfer equation, taking advantage of a well-known analytical approximation (see, for example, [15]).

The structure of the paper is as follows. In section 2, we summarize the mathematical and numerical tools to be used in the paper, while section 3 contains a description of the algorithm, and in section 4 we demonstrate numerical results for a number of special functions.

2. Mathematical and numerical preliminaries. In this section we summarize several well-known facts to be used in this paper; all of these can be found in [1, 3, 4, 12, 5]. Throughout this paper, the derivative of a function $f : \mathbb{R} \to \mathbb{R}$ will be denoted by f', and p, q, r will denote the second-order polynomials introduced in (1). All functions are assumed to be sufficiently smooth.

2.1. Taylor series for special functions. As is well known, any function $u : \mathbb{R} \to \mathbb{R}$ that is sufficiently smooth in the neighborhood of the point $x_0 \in \mathbb{R}$ can be approximated by its Taylor series

$$u(x_0 + h) = \sum_{k=0}^{m} \frac{u^{(k)}(x_0)}{k!} h^k + \varepsilon, \tag{2}$$

with

$$|\varepsilon| \leq \sup_{|x-x_0|\leq h} \left\{ \frac{u^{m+1}(x)}{(m+1)!} h^{m+1} \right\}, \tag{3}$$

where $u^{(k)}$ denotes the kth derivative of u; the Taylor series for u' is

$$u'(x_0+h) = \sum_{k=1}^{m} \frac{u^{(k)}(x_0)}{(k-1)!} h^{k-1} + \tilde{\varepsilon}, \tag{4}$$

with

$$|\tilde{\varepsilon}| \leq \sup_{|x-x_0|\leq h} \left\{ \frac{u^{m+1}(x)}{m!} h^{m} \right\}. \tag{5}$$

If u satisfies (1), and p, q, r are polynomials of order 2, then the consecutive derivatives of u can be obtained via a simple recursion. Indeed, differentiating (1) k times and carrying out elementary manipulations, we have

$$\begin{aligned} pu^{(k+2)} = &- (kp' + q)\, u^{(k+1)} - \left(\frac{k(k-1)}{2} p'' + kq' + r \right) u^{(k)} \\ &- \left(\frac{k(k-1)}{2} q'' + kr' \right) u^{(k-1)} - \frac{k(k-1)}{2} r'' u^{(k-2)} \end{aligned} \tag{6}$$

for any $k \geq 0$.

2.2. The Prüfer transform. Given (1) and a differentiable positive function $\gamma : \mathbb{R} \to \mathbb{R}$, we define the function $\theta : \mathbb{R} \to \mathbb{R}$ as the solution of the differential equation

$$\theta' = -\frac{\gamma}{p} \sin^2(\theta) - \frac{r}{\gamma} \cos^2(\theta) - \left(\frac{\gamma'}{\gamma} + \frac{q-p'}{p} \right) \frac{\sin(2\theta)}{2}, \tag{7}$$

where θ, γ, p, q, r are functions of x; integrating both sides of (7) with respect to x and carrying out elementary manipulations, we obtain

$$\theta(x) = \arctan\left(\frac{1}{\gamma(x)} \frac{p(x)u'(x)}{u(x)} \right) + k\,\pi, \tag{8}$$

where u is the function defined by (1), and k is an arbitrary integer; we observe that, for any $\tilde{x}$ such that $u(\tilde{x}) = 0$,

$$\theta(\tilde{x}) = (n + 1/2)\pi, \tag{9}$$

with n an integer. Similarly, for any $\tilde{x}$ such that $u'(\tilde{x})$=0,

$$\theta(\tilde{x}) = n\pi, \tag{10}$$

where again n is some integer.

The choice of the function γ in (8) influences the behavior of the function θ, without changing the solutions of (9) and (10). For example, if $\gamma \equiv 1$ and (1) is self-adjoint (i.e., $q = p'$), then

$$\theta' = -\frac{1}{p} \sin^2(\theta) - r \cos^2(\theta), \tag{11}$$

and the function θ is strictly decreasing if p and r are everywhere positive.

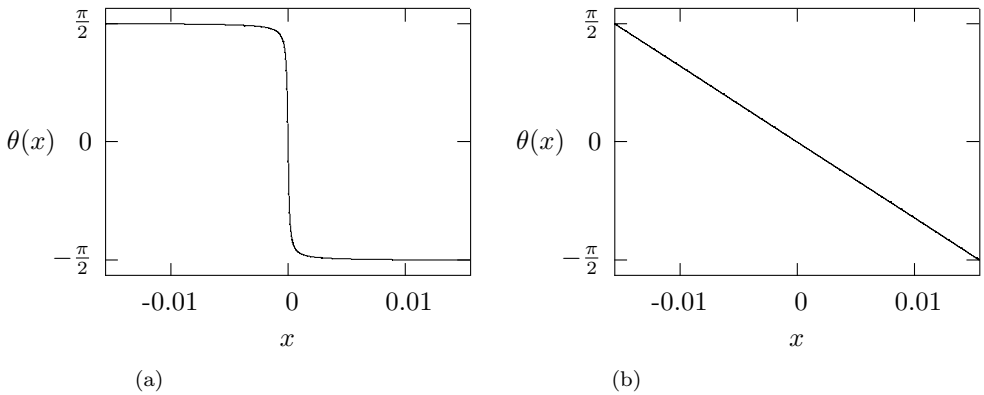


FIG. 1. (a) *shows* θ *as defined in* (7) *and* (8) *for a Legendre polynomial of order* 100 *with* $\gamma \equiv 1$. (b) *shows* θ *for the same polynomial with* $\gamma \equiv \sqrt{rp}$.

However, it often happens that $1/p$ and r are of different orders of magnitude; this causes the value of θ' to vary dramatically on the interval $\theta = [-\pi/2, \pi/2]$, making it inconvenient when used as a numerical tool. Figure 1(a) illustrates the last point. It shows θ for a Legendre polynomial of order $l = 100$, with $p = 1 - x^2$, $q = p'$, and $r = l(l+1)$, in the interval between the polynomial's greatest negative root and its least positive root.

Such steplike behavior complicates many numerical operations involving (7) and can be controlled via an appropriate choice of γ. Indeed, choosing

$$\gamma \equiv \sqrt{rp} \tag{12}$$

converts (8) into

$$\theta(x) = \arctan\left(\frac{1}{\sqrt{r(x)p(x)}}\frac{p(x)u'(x)}{u(x)}\right), \tag{13}$$

and substituting (12) into (7) yields

$$\theta' = -\sqrt{\frac{r}{p}} - \frac{r'p - p'r + 2rq}{2rp} \cdot \frac{\sin(2\theta)}{2}. \tag{14}$$

In this case, the large values of the product rp do not cause excessive variations in the magnitude of θ', which is illustrated by Figure 1(b), which shows θ for the same Legendre polynomial as in Figure 1(a) but with γ defined in (12).

Whenever $\frac{d\theta}{dx} < 0$, the ODE (14) can be rewritten in the form

$$\frac{dx}{d\theta} = -\left(\sqrt{\frac{r}{p}} + \frac{r'p - p'r + 2rq}{2rp} \cdot \frac{\sin(2\theta)}{2}\right)^{-1}, \tag{15}$$

where θ, γ, r, q, p are functions of x, and (15) can be viewed as a differential equation defining x as a function of θ. Clearly, θ' is guaranteed to be negative whenever

$$\left|\frac{r'p - p'r + 2rq}{4rp}\right| < \sqrt{\frac{r}{p}}, \tag{16}$$

and the latter condition is easy to test for and is satisfied by all standard special functions. We also observe that, given a solution x of (15) and an integer n, the function $\bar{x}$ defined by the formula

$$\bar{x}(\theta) = x(\theta + n\,\pi) \tag{17}$$

is also a solution of (15).

Historically, the Prüfer transform has been used as an instrument of analysis for differential equations in the vicinity of the WKB regime. Indeed, whenever

$$\left| \frac{r'p - p'r + 2rq}{2rp} \right| << \sqrt{\frac{r}{p}}, \tag{18}$$

the solution of the differential equation (14) is close to the solution of

$$\theta' = -\sqrt{\frac{r}{p}}, \tag{19}$$

and combining the latter with (13) yields the classical WKB approximation to the solution u of (1); higher-order terms in the WKB expansion of u are obtained via more detailed analysis of (14).

Observation 1. The combination of (9) and (10) with (15) suggests the use of (15) as a tool for finding roots of the function u defined by (1), as well as roots of the derivative u' of u.

Indeed, suppose that x_0 is a root of u. Defining the function x via the ODE (15) with the initial condition

$$x\left(\frac{\pi}{2}\right) = x_0, \tag{20}$$

we observe that, for each root $\tilde{x}$ of u, there exists an integer n such that

$$\tilde{x} = x\left(\frac{\pi}{2} + n\,\pi\right); \tag{21}$$

by the same token, for each root $\tilde{y}$ of u', there exists an integer n such that

$$\tilde{x} = x(n\,\pi). \tag{22}$$

In other words, given a single root x_0 of u, all other roots (and all roots of the derivative) of u can be determined by solving the Cauchy problem (15) with the initial condition (20). A trivial modification of the above procedure permits one to start from a root y_0 of the derivative of u; in this case, one has to solve (15) with the initial condition

$$x(0) = y_0. \tag{23}$$

An interesting characteristic of the above construction is the fact that it finds roots of the differential equation (1) by solving the Cauchy problem for an auxiliary equation, eliminating the need for a "solver" (bisection, Newton, etc.) as such.

Observation 2. Observation 1 is behind the use of the Prüfer transform as an analytical tool for the construction of asymptotic formulas for the roots of special functions (and also of their derivatives, etc.) in the vicinity of the WKB regime. In this paper, we use the Prüfer transform as a numerical tool applicable in a much less

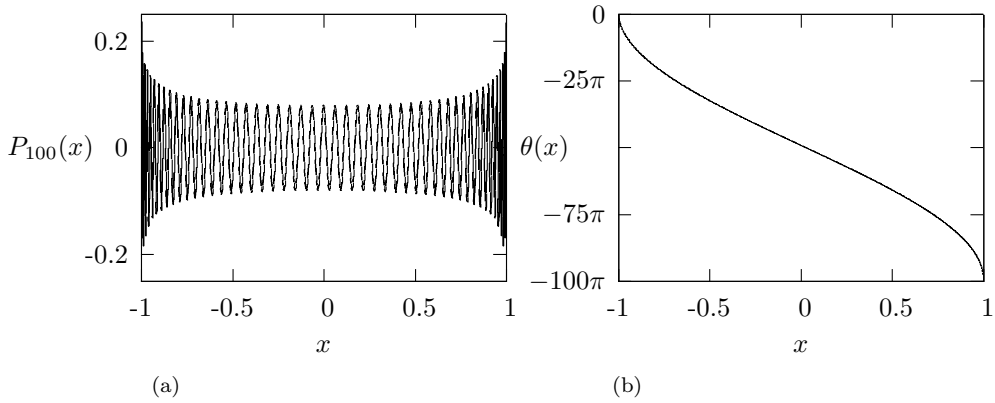


FIG. 2. (a) *shows the Legendre polynomial of order* 100. (b) *shows* θ *as defined in* (7) *and* (8) *for the same polynomial.*

restrictive environment. We solve (15) to low accuracy, and use the result as an initial approximation for the Newton procedure; the latter is based on a high-order Taylor method (see section 2.3.2 below).

In the WKB regime, our numerical solution of (15) is close to the WKB approximation and can be replaced by the latter. However, this involves detailed asymptotic analysis of each particular differential equation and results in only moderate reduction in the CPU time requirements of the method. Thus we use this latter approach only for the case of the Legendre nodes, where it is both trivial and well-known (see section 4.1 below).

. . . . 3. In our codes, we solve ODE (15) via a second-order Runge–Kutta scheme; it takes about 5 steps to march from one root of u to the next while maintaining three-digit accuracy. Replacing a second-order Runge–Kutta scheme with a second-order leapfrog scheme produces essentially identical results; replacing it with the Euler method leads to a minor increase in the CPU time requirements of the method. The Runge–Kutta scheme was chosen because of its simplicity.

We further illustrate the behavior of θ (as opposed to that of u) in Figures 2(a) and 2(b), containing graphs on the interval $[-1, 1]$ of the Legendre polynomial P_{100} of order 100 and of the corresponding θ defined via (7), with γ defined in (12). The difference in the behaviors of the functions θ and u implicit in Figures 2(a) and 2(b) is the reason for the popularity of the Prüfer transform as a tool of analysis; we use it as a numerical tool by converting (7) into (15), solving the latter to low precision on appropriately chosen intervals, and using the results as initial values for the Newton process.

2.3. Numerical tools. In this section we summarize several numerical techniques to be used in the paper. All of them can be found, for example, in [4].

2.3.1. A second-order Runge–Kutta method. The second-order Runge–Kutta method solves the initial-value problem

$$\begin{aligned} y(x_0) &= y_0, \\ y'(x) &= f(x, y) \end{aligned} \tag{24}$$

on the interval $[x_0, x_0 + L]$ by taking a sequence of n steps:

$$\begin{aligned} x_{i+1} &= x_i + h, \\ k_{i+1} &= h\, f(x_i + h, y + k_i), \\ y_{i+1} &= y_i + \frac{1}{2}(k_i + k_{i+1}), \end{aligned} \tag{25}$$

where $h = L/n$ and $k_0 = hf(x_0, y_0)$. The cost of this algorithm is of order n, where in this version of the scheme exactly $n + 1$ evaluations of the function f are required, producing the global truncation error of order h^2.

2.3.2. The Taylor series method for the solution of ODEs. The Taylor series method is a classical numerical scheme for the solution of ODEs. Given (1), the values of $u(x_k)$ and $u'(x_k)$ at the point x_k, and an appropriately chosen integer $m \geq 2$, the Taylor series method calls for repeated differentiation of the differential equation (1), resulting in the values of $u'''(x_k), u^{(4)}(x_k), \ldots, u^{(m-1)}(x_k), u^{(m)}(x_k)$. The latter are used to evaluate approximations to $u(x_{k+1}), u'(x_{k+1})$ via the expansions (2) and (4), with $h = x_{k+1} - x_k$. Obviously, the truncation error of one step of the Taylor series method is of the order h^m, and its cost is determined by the cost of finding the derivatives $u'''(x_k), \ldots, u^{(m-1)}(x_k), u^{(m)}(x_k)$; in the case of (1), the recursion (6) permits $u'''(x_k), \ldots, u^{(m-1)}(x_k), u^{(m)}(x_k)$ to be evaluated in $O(m)$ operations.

By choosing the order m of the expansions (2) and (4), one can control the accuracy of the Taylor series method, and often it makes sense to choose very high orders, obtaining essentially machine precision.

In our experience, choosing $m = 30$ results in slightly better than 16-digit accuracy, and $m = 60$ produces about 34 digits; we use the first choice in our double precision codes and the second choice in extended precision calculations (see section 4 below). Figure 3 depicts stability regions (in the usual λh-plane) in the two cases. Examination of Figure 3 indicates that the 30-term scheme is stable as long as the

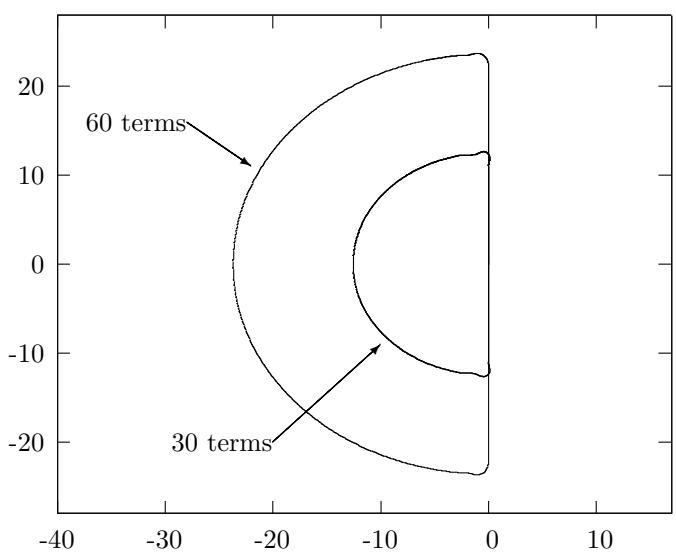


FIG. 3. *Stability domain of the Taylor series method for* 30 *terms and* 60 *terms.*

step is less than roughly 2 wavelengths (1 wavelength being 2π), and the 60-term scheme stays stable for steps that are roughly 4 wavelengths. Our algorithm makes steps from one root of the special function function to the next; if the function in question were $\sin(\lambda x)$ or $\cos(\lambda x)$, this would be exactly 1/2 of a wavelength, keeping us well away from stability problems. This crude analysis indicates that the algorithm should be stable in our environment and is supported by our numerical experience.

2.3.3. Newton's method. Given an initial approximation x_0, Newton's method solves the equation $f(x) = 0$ iteratively, with the nth iteration defined by the formula

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}. \tag{26}$$

If $\tilde{x}$ is a simple root of f and x_0 has been chosen sufficiently close to $\tilde{x}$, then the convergence is quadratic; i.e., the number of correct digits of the approximation x_n will double after each iteration.

2.4. Orthogonal polynomials, Bessel functions, and prolate spheroidal wave functions. This section summarizes certain well-known properties of several classes of special functions, used in section 4 to test the performance of the algorithm of this paper; all of the facts in this section can be found in [1, 12, 5, 8], where the reader is referred for a more exhaustive discussion.

2.4.1. Orthogonal polynomials. Given a (possibly infinite) interval (a, b) and a positive function w such that the product of w with any polynomial is integrable on (a, b), we say that $p_0, p_1, \ldots, p_n$ is an orthogonal family of polynomials on (a, b) with weight w, if for each $k = 1, 2, 3, \ldots$, the polynomial p_k is of degree k and

$$\int_a^b p_i(x)\, p_j(x)\, w(x)\, dx = \begin{cases} 0, & i \neq j, \\ a_i, & i = j, \end{cases} \tag{27}$$

where $a_i > 0$.

In this subsection we briefly discuss properties of Legendre, Hermite, and Laguerre polynomials (denoted P_n, H_n, and L_n, respectively), which are examples of classical orthogonal families of polynomials. Polynomials within each of these families are orthogonal on an interval $(a, b) \in R^1$ with a certain weight function ω, and each polynomial p_k satisfies a homogeneous second-order ordinary differential equation of the form (1); the interval (a, b), the function ω, and coefficients $p(x), q(x), r(x)$ in (1) are specific for each family and can be found in columns 2–6 in Table 1. Each family satisfies a three-term recurrence relation of the form

$$a_m\, p_{m+1}(x) = (b_m + c_m x)\, p_m(x) - d_m\, p_{m-1}(x), \tag{28}$$

with $c_0, c_1, c_2, c_3, \ldots$, $d_0, d_1, d_2, d_3, \ldots$ real sequences specific to the families of orthogonal polynomials. Thus, given $x \in \mathbb{R}$, an integer $n > 2$, and the values $p_0(x)$ and $p_1(x)$, it follows from (28) that the values $p_2(x), p_3(x)$, $p_4(x), \ldots, p_n(x)$ can be evaluated in $O(n)$ operations.

Differentiating (28), we obtain the recurrence

$$a_m\, p'_{m+1}(x) = (b_m + c_m x)\, p'_m(x) - d_m\, p'_{m-1}(x) + c_m p_m(x), \tag{29}$$

which can be used for the efficient evaluation of the derivatives of $p_2, p_3, \ldots$. For the Legendre, Hermite, and Laguerre polynomials, the coefficients a_m, b_m, c_m, d_m can be found in columns 3–6 in Table 2.

TABLE 1
Intervals of orthogonality, weight functions, and coefficients of the ODEs.

Case	$[a, b]$	ω	p	q	r
P_n	$[-1, 1]$	1	$1 - x^2$	$-2\,x$	$n\,(n+1)$
H_n	$[-\infty, \infty]$	e^{-x^2}	1	$-2x$	$2n$
L_n	$[0, \infty]$	e^{-x}	x	$1 - x$	n

TABLE 2
Quadrature weights and coefficients of three-term recurrences.

Case	w_n^k	a_n	b_n	c_n	d_n
P_n	$2(1 - x_k^2)^{-1}\, P_n'^{-2}(x_k)$	$n + 1$	0	$2n + 1$	n
H_n	$\sqrt{\pi} 2^{n+1} n!\, H_n'^{-2}(x_k)$	1	0	2	$2n$
L_n	$[x_k L_n'^2(x_k)]^{-1}$	$n + 1$	$2n + 1$	-1	n

The Legendre and Hermite polynomials of even degrees are even and have a local extremum at $x = 0$; for odd degrees they are odd and have a root at $x = 0$. We will also make use of the fact that the first (smallest) root x_1^n of the nth Laguerre polynomial satisfies the inequality

$$x_1^n > \frac{2}{4n + 2} \tag{30}$$

(see, for example, formula 22.16.8 in [1]).

Orthogonal polynomials are a classical tool for the design of numerical integration rules known as Gaussian quadratures. Given the roots $x_1, \ldots, x_n$ of the nth degree polynomial p_n of an orthogonal family on the interval (a, b) with weight function w, there exist real numbers $w_1^n, \ldots, w_n^n$ such that the formula

$$\int_a^b f(x)\, w(x)\, dx \approx \sum_{k=1}^{n} w_k^n\, f(x_k) \tag{31}$$

is exact for any f that is a polynomial of degree $2n - 1$ or less.

Given the roots of the appropriate polynomials, the quadrature weights for the classical Gaussian quadratures are available analytically, and the formulas for the Legendre, Hermite, and Laguerre cases can be found in column 2 of Table 2.

4. As is well known, numerical evaluation of the polynomials on the interval $[-1, 1]$ cannot cause overflow problems, since $|P_n(x)| \leq 1$ for all $x \in [-1, 1]$. On the other hand, the polynomials H_n and L_n grow rapidly with x, even for moderate values of x; for all practical purposes, they cannot be evaluated in the standard floating point arithmetic.

This problem is easily avoided by replacing the polynomials H_n and L_n with the so-called normalized Hermite and Laguerre functions, defined by the formulas

$$\widetilde{H}_n(x) = \frac{e^{-x^2/2}}{\pi^{\frac{1}{4}} 2^{n/2} \sqrt{n!}} H_n(x), \tag{32}$$

$$\widetilde{L}_n(x) = e^{-x/2} L_n(x), \tag{33}$$

respectively. Obviously, the functions $\widetilde{H}_n(x)$ and $\widetilde{L}_n(x)$ have the same roots as $H_n(x)$ and $L_n(x)$, and $|\widetilde{H}_n(x)| < 1$, $|\widetilde{L}_n(x)| \leq 1$ for all $x \geq 0$, since

$$|H_n(x)| < \pi^{\frac{1}{4}} 2^{n/2} \sqrt{n!}\, e^{x^2/2}, \tag{34}$$

$$|L_n(x)| \leq e^{x/2} \tag{35}$$

(see, for example, section 22.14 in [1]).

Simple algebraic manipulations show that

$$\widetilde{H}_n''(x) + (2n + 1 - x^2)\widetilde{H}_n(x) = 0, \tag{36}$$

$$x^2 \widetilde{L}_n''(x) + x\, \widetilde{L}_n'(x) - \left(\frac{1}{4}x^2 - \left(n + \frac{1}{2}\right) x\right) \widetilde{L}_n(x) = 0, \tag{37}$$

and

$$\widetilde{H}_{n+1}(x) = \sqrt{\frac{2}{n+1}}\, x\, \widetilde{H}_n(x) - \sqrt{\frac{n}{n+1}}\, \widetilde{H}_{n-1}(x), \tag{38}$$

with $\widetilde{H}_{-1} = 0$, $\widetilde{H}_0 = \pi^{-\frac{1}{4}} e^{-x^2/2}$. Furthermore, if x is a root of H_n, then $\widetilde{H}_n'(x)$ can be evaluated via the recurrence

$$\widetilde{H}_{n+1}'(x) = \sqrt{\frac{2}{n+1}} \left(x\, \widetilde{H}_n'(x) + \widetilde{H}_n(x)\right) - \sqrt{\frac{n}{n+1}}\, \widetilde{H}_{n-1}'(x), \tag{39}$$

with $\widetilde{H}_{-1} = 0$, $\widetilde{H}_0 = 0$. Combining (32) with column 2 in Table 2, we observe that

$$w_k^n = \frac{2\, e^{-x_k^2}}{\widetilde{H}_n'^2(x_k)}. \tag{40}$$

The functions $\widetilde{L}_n$ can be evaluated via the recurrence

$$\widetilde{L}_{n+1}(x) = \frac{2n + 1 - x}{n+1} \widetilde{L}_n(x) - \frac{n}{n+1} \widetilde{L}_{n-1}(x), \tag{41}$$

the same recurrence relation as for L_n, with $\widetilde{L}_{-1}(x) = 0$, $\widetilde{L}_0(x) = e^{-x/2}$. If x is a root of $\widetilde{L}_n$, then $\widetilde{L}_n(x)$ can be evaluated via the recurrence

$$\widetilde{L}_{n+1}'(x) = \frac{2n + 1 - x}{n+1} \widetilde{L}_n'(x) - \frac{1}{n+1} \widetilde{L}_n(x) - \frac{n}{n+1} \widetilde{L}_{n-1}'(x), \tag{42}$$

with $\widetilde{L}_{-1}(x) = 0$, $\widetilde{L}_0(x) = 0$, $\widetilde{L}_{-1}'(x) = 0$, $\widetilde{L}_0'(x) = 0$. Finally, combining (33) with column 2 in Table 2, we obtain

$$w_k^n = \frac{e^{-x_k}}{x_k \widetilde{L}_n'^2(x_k)}. \tag{43}$$

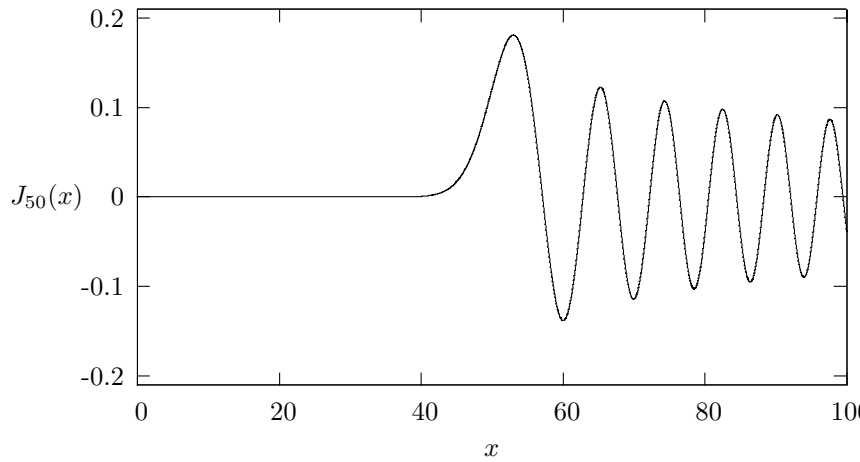


FIG. 4. *Plot of* J_{50}.

2.4.2. Bessel functions. The Bessel function J_n of order n is analytic in the whole plane and satisfies the differential equation

$$x^2 J_n''(x) + x\, J_n'(x) + (x^2 - n^2) J_n(x) = 0. \tag{44}$$

The behavior of J_n on the real axis is illustrated in Figure 4, where the graph of J_{50} is depicted. The Bessel function of order n is small and monotone for $|x| < n + n^{1/3}$ and starts to oscillate after that; for any $n \geq 1$

$$x_1 > n + n^{\frac{1}{3}}, \tag{45}$$

with x_1 the smallest positive root of J_n. For large values of x, the asymptotic behavior of J_n is given by the formula

$$J_n(x) \approx \sqrt{\frac{2}{\pi\, x}} \cos\left(x - \left(n + \frac{1}{2}\right)\frac{\pi}{2}\right) + \quad \left(\frac{1}{x^{1.5}}\right). \tag{46}$$

Numerical evaluation of J_n is a well-studied subject; there exist several classical schemes for the evaluation of $J_n(x)$ and its derivative for a cost proportional to n; several are described in [1].

2.4.3. Prolate spheroidal wave functions. Prolate spheroidal wave functions corresponding to the band limit c are eigenvectors of the integral operator G_c defined via the formula

$$F_c(\psi)(x) = \int_{-1}^{1} e^{icxt} \psi(t)\, dt; \tag{47}$$

they are also the eigenvectors of the differential operator

$$G_c(\psi)(x) = (1 - x^2)\psi''(x) - 2x\, \psi'(x) - c^2 x^2 \psi(x). \tag{48}$$

The nth eigenvalue of the operator (48) is normally denoted by χ_n, so that the nth eigenvector ψ_n of (48) satisfies the ODE

$$(1 - x^2)\psi_n''(x) - 2x\, \psi_n'(x) + (\chi_n - c^2 x^2)\psi_n(x) = 0, \tag{49}$$

which provides standard tools for the evaluation of the functions ψ_n, eigenvalues χ_n, and related quantities.

The function ψ_n has n roots on the interval $[-1, 1]$; for even n the function ψ_n is even and has a local extreme value at $x = 0$; for odd n the function ψ_n is odd and has a root at $x = 0$. We refer the reader to [14, 10] for more detailed information about prolate spheroidal wave functions.

3. Description of the algorithm. In this section we describe an algorithm for computing the roots of a function u satisfying (1). We start with an informal outline in the section below, followed by a more detailed description in section 3.2.

3.1. Informal description. The algorithm of this paper finds all roots of a function u satisfying (1) recursively, in that the values of u and its derivatives at the preceding root are used in finding the next one.

Given a starting value x_s, the algorithm calculates the smallest root x_1 that is larger or equal to x_s in two steps. First, it finds an approximation $\tilde{x_1}$ to the initial root by calculating $\theta_0 = \theta(x_s)$ via (13) and solving the differential equation (15) with the initial condition $x(\theta_0) = x_s$ in the interval $\theta \in (\theta_0, -\pi/2)$ via the Runge–Kutta method (25). Due to (9), the value of x at $-\pi/2$ is a root of u.

5. In our experience, four steps of the Runge–Kutta method are sufficient to yield x_1 with two to three digits of accuracy. Using a higher-order scheme for the solution of the differential equation (15) and increasing the number of nodes in the discretization of the interval $(\pi/2, -\pi/2)$, one could obtain the root x_1 with arbitrarily high precision. While this scheme could be used to find all roots $x_1, x_2, \ldots, x_n$ for a cost proportional to n, its actual cost would be much greater than the cost of the scheme of this paper, where we solve (15) to low precision and refine the solution via the Newton procedure described below.

The second step uses the Newton procedure (26) to obtain the root x_1, starting with the approximation $\tilde{x}_1$. Since the Newton process is quadratically convergent, it takes about three steps per root to obtain 16-digit accuracy, and each of the steps requires the evaluation of u and u' at a point in the vicinity of x_{i+1}.

The evaluation of u and u' is performed by a scheme specific to the function u. If the cost of the evaluation is k operations, then the cost for calculating the first root is $O(k)$. For orthogonal polynomials, u and u' can be evaluated via the recurrence relations (28) and (29) for a cost proportional to the degree of the polynomial.

6. In many cases (such as Legendre and Hermite polynomials, prolate spheroidal wave functions, etc.), the initial root or extreme value are available analytically due to symmetry considerations, since an odd function always has a root at 0, and the derivative of an even function is odd. In such cases, the determination of the initial root is trivial. We discuss some of the specific details in sections 3.2 and 4 below.

Given a root x_i of u, the algorithm finds the next bigger root x_{i+1} of u in two steps. The first step finds an approximation $\tilde{x}_{i+1}$ by solving the differential equation (15) via the Runge–Kutta scheme (25) on the interval $\theta \in (\pi/2, -\pi/2)$ with the initial condition $x(\pi/2) = x_i$. Due to (9), $\tilde{x}_{i+1}$ is an approximation to the root x_{i+1}.

The second step uses the Newton procedure (26) to increase the accuracy of the approximation $\tilde{x}_{i+1}$. The values of u and u' required by the Newton method are evaluated via the expansions (2) and (4), and the derivatives of u are computed via the recurrence (6).

. . . . 7. While inequalities (3) and (5) provide an analytical tool for estimating the required number m of terms in the Taylor expansion (2), we have found it convenient to determine the appropriate values of m numerically. For all examples in this paper, 30 terms are sufficient for double precision (16-digit) accuracy, and 60 terms produce 34 digits.

The cost of evaluating expansion (2) is more or less independent of the coefficients p, q, r of (1) and is reasonably small. Adding up the costs for all n roots, we observe that the total cost of the algorithm is of the order $n\,m + k$ operations, with n the number of roots to be found, m the number of terms in the Taylor series, and k the (problem-specific) cost of a single evaluation of u and u'. The repeated evaluation of u via its Taylor expansion can be seen as a version of the Taylor series method described in section 2.3.2.

. . . . 8. In the algorithm described above, Taylor expansions are used to evaluate u close to its roots. Needless to say, Taylor expansions can also be used to evaluate u between roots, thus providing a scheme for the rapid evaluation of u. Specifically, given a set of M points $\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_M$, where $\tilde{x}_{i-1} \leq \tilde{x}_i$, and n roots $x_1, x_2, \ldots, x_n$ of u, where $x_{i-1} \leq x_i$, the scheme finds the closest root to each point $\tilde{x}_i$ and evaluates u at $\tilde{x}_i$ via a Taylor expansion at that root. Including the cost of computing the n roots of u, the total cost of this scheme is of the order of $(M+n)\,m+k$ operations.

3.2. Detailed description. In this section we describe the numerical scheme of this paper in some detail. Algorithm 1 describes the core procedure. As input, it requires the smallest root of u and the number N of roots to be determined. It returns the roots $x_1 \ldots x_N$ and the derivatives $u'(x_1) \ldots u'(x_N)$ for a cost of $O(N)$ operations. Algorithms 2 and 3 are auxiliary schemes for finding the initial root.

Algorithm 1
Comment[The input parameters are the polynomial coefficients of p, q, and r in (1), an initial root x_1 of u, the number of roots N which are greater than or equal to x_1, and the derivative $u'(x_1)$ at the initial root x_1. Algorithm 1 returns in the array *roots* the N roots $x_1, x_2, \ldots, x_N$ of u such that $x_i \geq x_1$ for all $i = 1, 2, \ldots, N$. Furthermore, it returns in the array *ders* the values $u'(x_i)$ for $i = 1, 2, \ldots, N$.]

Set $roots(1) = x_1$ and $ders(1) = u'(x_1)$.

do $i = 1,\ N - 1$

1. Use the Runge–Kutta method (25) to solve (15) for $\theta = \pi/2$ to $-\pi/2$ with initial conditions $\theta_0 = \pi/2$ and $x(\theta_0) = roots(i)$. The resulting value for $x(-\pi/2)$ is the initial approximation to $roots(i+1)$.
2. Use recurrence relation (6) to calculate the first m derivatives of u at $x = roots(i)$, starting with $u(x) = 0$, $u'(x) = ders(i)$.
3. Improve the precision of $roots(i+1) = x(-\pi/2)$ via Newton's method (26), where u and u' are evaluated via the first m terms of the Taylor expansion around $roots(i)$. Set $ders(i+1) = u'(roots(i+1))$.

end do

If a local extremum instead of a root of u is given, an analogous procedure to steps 1–3 of Algorithm 1 can be used to calculate the next biggest root. This procedure is described in Algorithm 2.

Algorithm 2
Comment [The input parameters are the polynomial coefficients of p, q, and r in (1), an extreme value x_e of u, and $u(x_e)$. Algorithm 2 returns the smallest root x_1 of u such that $x_1 > x_e$; it also returns the derivative $u'(x_1)$ of u at x_1.]

1. Use the Runge–Kutta method (25) to solve (15) for $\theta = 0$ to $-\pi/2$ with initial conditions $\theta_0 = 0$ and $x(\theta_0) = x_e$. The resulting value for $x(-\pi/2)$ is the initial approximation to x_1.
2. Use recurrence relation (6) to calculate the first m derivatives of u at x_e, starting with $u(x_e)$, $u'(x_e) = 0$.
3. Improve the precision of $x_1 = x(-\pi/2)$ via Newton's method (26), where u and u' are evaluated via the first m terms of the Taylor expansion around x_e. Return x_1 and $u'(x_1)$.

If the first roots of u lie in a regime where the Taylor expansion converges poorly, the following algorithm can be used.

Algorithm 3
Comment [Algorithm 3 requires a routine for evaluating u and u'. Its input parameters are such a routine, the polynomial coefficients of p, q, and r in (1), and a starting value x_s such that $x_s < x_1$, with x_1 the smallest root of u. Algorithm 3 returns the smallest root x_1 of u and the derivative $u'(x_1)$ of u at x_1.]

1. Compute $\theta(x_s)$ via (13).
2. Use the Runge–Kutta method (25) to solve (15) for $\theta = \theta(x_s)$ to $-\pi/2$ with initial conditions $\theta_0 = \theta(x_s)$ and $x(\theta_0) = x_s$. The resulting value for $x(-\pi/2)$ is the first approximation to x_1.
3. Improve the precision of $x_1 = x(-\pi/2)$ via Newton iterations, where u and u' are evaluated by the given routine, which is specific to u. Return x_1 and $u'(x_1)$.

9. Once the algorithms described above were implemented and profiled, it turned out that, in most cases, about 30% of the CPU time was spent solving (15) via the Runge–Kutta method, and most of that time was expended evaluating the *sine* function in (15). Observing that the algorithm uses the solution of (15) to compute initial approximations to the roots with approximately 3-digit accuracy, we replaced the standard FORTRAN evaluation of the *sine* function with a table lookup using a table with 1000 entries. This modification reduced the CPU time requirements of the scheme by about 20%.

4. Implementation and numerical results. The algorithm of this paper has been implemented in FORTRAN 77 and tested on several classical families of special functions. We used the Lahey–Fujitsu FORTRAN 95 compiler with the optimization flag set to –o2, run on Intel Pentium 4, 3.6 GHz with 3 GB RAM. The algorithm has been applied to Legendre, Hermite, and Laguerre polynomials, Bessel functions, and prolate spheroidal wave functions.

The results of our tests involving Legendre, Hermite, and Laguerre polynomials are summarized in Tables 3, 4, and 5, respectively; since one of the principal uses of roots of orthogonal polynomials is in numerical integration, we included the evaluation of the weights of the corresponding quadratures via (40), (43), and Table 2. In each case, the roots of the corresponding polynomial and the associated weights were evaluated in double precision with the number of terms m in the expansion (2) set to 30 and in extended precision with $m = 60$; the difference between values of a root (or a weight) obtained via these two calculations was viewed as the error of the double precision calculation.

TABLE 3
Legendre polynomials.

Number of nodes	$\varepsilon_{mr}(\{x_i\})$	$\varepsilon_{rm}(\{w_i\})$	$\varepsilon_{mr}(\{w_i\})$	$\varepsilon_{quad}(\{(x_i, w_i)\})$	CPU time (seconds)
1 000	3.98×10^{-15}	7.45×10^{-14}	2.61×10^{-11}	7.34×10^{-13}	0.0006
10 000	5.07×10^{-15}	8.49×10^{-13}	1.27×10^{-9}	6.50×10^{-11}	0.0057
100 000	6.76×10^{-15}	1.44×10^{-11}	7.39×10^{-8}	8.08×10^{-11}	0.057
1 000 000	2.69×10^{-14}	1.08×10^{-10}	4.13×10^{-6}	2.47×10^{-9}	0.57
1 000 000	Extended (32-digit) precision			3.95×10^{-28}	260.5

TABLE 4
Hermite polynomials.

Number of nodes	$\varepsilon_{mr}(\{x_i\})$	$\varepsilon_{rm}(\{w_i e^{x_i^2}\})$	$\varepsilon_{mr}(\{w_i e^{x_i^2}\})$	$\varepsilon_{quad}(\{(x_i, w_i)\}$	CPU time (seconds)
1 000	4.78×10^{-15}	1.76×10^{-12}	1.76×10^{-12}	1.89×10^{-13}	0.0009
10 000	1.04×10^{-14}	6.79×10^{-12}	6.79×10^{-12}	2.22×10^{-12}	0.0078
100 000	3.68×10^{-14}	5.62×10^{-11}	5.62×10^{-11}	1.31×10^{-11}	0.076
1 000 000	9.37×10^{-13}	3.75×10^{-9}	3.57×10^{-9}	1.32×10^{-10}	0.76
1 000 000	Extended (32-digit) precision			3.00×10^{-28}	297.1

TABLE 5
Laguerre polynomials.

Number of nodes	$\varepsilon_{mr}(\{x_i\})$	$\varepsilon_{rm}(\{w_i e^{x_i}\})$	$\varepsilon_{mr}(\{w_i e^{x_i}\})$	$\varepsilon_{quad}(\{(x_i, w_i)\}$	CPU time (seconds)
1 000	8.15×10^{-14}	4.89×10^{-13}	1.08×10^{-12}	1.03×10^{-12}	0.003
10 000	1.08×10^{-11}	3.57×10^{-12}	1.36×10^{-11}	8.04×10^{-12}	0.027
100 000	1.26×10^{-9}	2.59×10^{-11}	7.20×10^{-10}	7.27×10^{-11}	0.27
1 000 000	2.16×10^{-9}	4.39×10^{-10}	8.40×10^{-10}	8.41×10^{-10}	2.69
1 000 000	Extended (32-digit) precision			9.46×10^{-27}	1161.3

Throughout this section, given the roots

$$x_1, x_2, \ldots, x_n \tag{50}$$

of a special function, $x_1^{(d)}, x_2^{(d)}, \ldots, x_n^{(d)}$ will denote the approximation to (50) obtained via our double precision calculation, and $x_1^{(q)}, x_2^{(q)}, \ldots, x_n^{(q)}$ will denote the approximation to (50) obtained via our extended precision calculation. Similarly, given the weights

$$w_1, w_2, \ldots, w_n \tag{51}$$

of a quadrature formula, $w_1^{(d)}, w_2^{(d)}, \ldots, w_n^{(d)}$ will denote the approximation to (51) obtained via our double precision calculation, and $w_1^{(q)}, w_2^{(q)}, \ldots, w_n^{(q)}$ will denote the approximation to (51) obtained via our extended precision calculation.

Several measures of error were calculated.

1. The maximum relative error $\varepsilon_{mr}(\{x_i\})$ of the nodes $x_1, x_2, \ldots, x_N$ defined by the formula

$$\varepsilon_{mr}(\{x_i\}) = \max_i \left\{ \frac{\left| x_i^{(d)} - x_i^{(q)} \right|}{|x_i|} \right\} \tag{52}$$

TABLE 6
Bessel functions.

Number of evaluated roots	Order n of Bessel function	$\varepsilon_{mr}(\{x_i\})$	CPU time (seconds)
2 000	100	5.00×10^{-15}	0.004
20 000	1 000	6.73×10^{-15}	0.036
200 000	10 000	2.87×10^{-14}	0.36
2 000 000	100 000	8.62×10^{-14}	3.60
1 000 000	50 000	Extended (32-digit) precision	792.9

TABLE 7
Prolate spheroidal wave functions.

Number of nodes	Band limit c	$\varepsilon_{mr}(\{x_i\})$	CPU time (seconds)
2 000	1 000	4.10×10^{-15}	0.002
20 000	10 000	6.28×10^{-15}	0.018
200 000	100 000	2.08×10^{-14}	0.18
2 000 000	1 000 000	4.13×10^{-14}	1.83
2 000 000	1 000 000	Extended (32-digit) precision	386.6

was evaluated in all cases and can be found in column 2 of each of Tables 3, 4, and 5 and in column 3 of Tables 6 and 7.

2. The maximum relative error $\varepsilon_{mr}(\{w_i\})$ of the weights $w_1, w_2, \ldots, w_N$ defined by the formula

$$\varepsilon_{mr}(\{w_i\}) = \max_i \left\{ \frac{\left| w_i^{(d)} - w_i^{(q)} \right|}{|w_i|} \right\} \tag{53}$$

was evaluated for each quadrature (Legendre, Hermite, Laguerre) and can be found in column 3 of each of Tables 3, 4, and 5.

3. The relative maximum error $\varepsilon_{rm}(\{w_i\})$ of the weights $w_1, w_2, \ldots, w_N$ defined by the formula

$$\varepsilon_{mr}(\{w_i\}) = \left\{ \frac{\max_i \left| w_i^{(d)} - w_i^{(q)} \right|}{\max_i |w_i|} \right\} \tag{54}$$

was evaluated for each quadrature and can be found in column 4 of each of Tables 3, 4, and 5.

4. Measuring the accuracy of the quadratures (as opposed to the accuracy of nodes or weights separately) does not involve comparison between results of double precision calculations and those obtained in extended precision. Indeed, the quadrature is expected to evaluate exactly polynomials of orders up to $2N-1$ with the appropriate weight function, and some of these integrals are known analytically, in particular,

$$\int_a^b \omega(x)\, p_j(x)\, p_l(x)\, dx = \sum_{i=1}^{N} w_i\, p_j(x_i)\, p_l(x_i) = \delta_{jl}, \tag{55}$$

for any j, l such that $j, l \leq 2N-1$, and δ_{jl} denoting the Kronecker δ-function.

We use the identity (55) to test the accuracy of our quadratures by choosing at random K indices $i_1, i_2, \ldots, i_K$ between 0 and $N-1$ and viewing the maximum

$$\varepsilon_{quad}(\{(x_i, w_i)\}) = \max_{j,l \in [1,K]} \left| \delta_{jl} - \sum_{k=1}^{n} w_k \, p_{i_j}(x_k) p_{i_l}(x_k) \right| \tag{56}$$

as a measure of the accuracy of the scheme. The polynomials p_{i_j} in (56) are evaluated via the procedure described in Remark 8. In order to avoid additional accuracy loss from this computation, we perform the evaluation of p_{i_j} in extended precision. Furthermore, K in (56) above is chosen to be sufficiently large to provide a reasonable estimate and sufficiently small to keep the calculation feasible. For the calculations in extended precision, we set $K = 20$; the values of $\varepsilon_{quad}(\{(x_i, w_i)\})$ can be found in column 5 in each of Tables 3, 4, and 5.

To conclude the description of Tables 3–7, we note that the last column in each of them contains the CPU time taken by the corresponding calculation in seconds; for the polynomials (Legendre, Hermite, and Laguerre), this includes the time used to calculate the weights of the corresponding quadrature weights. Finally, the last row describes an extended precision calculation involving 1,000,000 roots; in the latter case, we report $\varepsilon_{quad}(\{(x_i, w_i)\})$ for the Gaussian quadratures (when applicable) and the CPU time taken by the calculation.

4.1. Legendre polynomials. Since for every positive root of a Legendre polynomial there exists a negative root of the same absolute value, it is sufficient to compute only the positive roots. If n is even, P_n has an extreme value at $x = 0$, and Algorithm 2 is used to find the first positive root, followed by Algorithm 1 to find the $n/2 - 1$ subsequent roots. If n is odd, P_n has a root at $x = 0$, and Algorithm 1 is used to find the $(n-1)/2$ positive roots. The values for $P_n(0)$ or $P_n'(0)$ which are required as input for Algorithm 2 or 1 are calculated via the recurrence relations (28) and (29). Finally, the Legendre quadrature weights are calculated via the appropriate formula in column 2 of Table 2.

. . . . 10. In this case, we replaced the initial low-accuracy approximation of each root via the numerical solution of the Prüfer equation (step 1 in Algorithm 1) with a well-known asymptotic formula

$$x_k = \left\{ 1 - \frac{(n-1)}{8n^3} - \frac{1}{384n^4} \left(39 - \frac{28}{\sin^2(\theta_k)} \right) \right\} \cos(\theta_k) + O(n^{-5}), \tag{57}$$

with

$$\theta_k = \left(\frac{4(\lceil \frac{n}{2} \rceil - k) - 1}{4n + 2} \right) \pi, \tag{58}$$

and $\lceil \frac{n}{2} \rceil$ denoting the integer part of $\frac{n}{2}$ (see, for example, [15]); this change resulted in a reduction in the CPU time requirements of the scheme of about 25%. While asymptotic formulas of various types are available for roots of all of the special functions considered in this section, in other cases we used the generic step 1 in Algorithm 1, in order to demonstrate the general nature of the approach.

The results of our numerical experiments with Legendre polynomials are summarized in Table 3.

4.2. Hermite polynomials. To avoid numerical overflow, we applied our algorithm to the normalized Hermite functions $\widetilde{H}_n$ instead of the polynomials (see Remark 4).

Since Hermite polynomials have the same symmetry as Legendre polynomials, the computation of the first root and the application of the main algorithm is analogous. The values $\widetilde{H}_n(0)$ and $\widetilde{H}'_n(0)$ are calculated via the recurrence relations (38) and (39). Formula (40) is used to calculate the Hermite weights from the obtained roots and derivatives of the Hermite functions. Instead of weights w_k^n defined in (40), we calculate the "scaled" version $w_k^n e^{x_k^2}$, in order to avoid underflow problems. Results of our experiments with Hermite polynomials are summarized in Table 4.

4.3. Laguerre polynomials. To avoid overrflow problems, we used the differential equation (37) for the normalized Laguerre functions, instead of the standard Laguerre equation (see Remark 4). Inequality (30) motivates the starting value $x_s = \frac{2}{4n+2}$, and recurrence relations (41) and (42) are used for the evaluation of the function and its derivative. The quadrature weights are calculated via formula (43) and returned in the form $w_k^n e^{x_k}$. Results of these experiments are summarized in Table 5.

[illegible] 11. This is the only case discussed in this paper where the procedure of section 3 had to be altered slightly. Indeed, the differential equation (37) has a singularity at $x = 0$, causing the roots to concentrate near that point. Thus, the Taylor series (2) with the origin at x_i does not converge at x_{i+1} for the first several values of i and converges slowly for several more. Without analyzing the problem carefully, we changed the procedure, as follows. We started with the procedure described in section 3 but omitting the Newton step altogether in other words, we used the Prüfer ODE to find all of the roots to low precision. After that, we refined the obtained approximations using the Taylor series exactly as described in section 3 but starting with the node x_n (i.e., the [illegible] node) and marching in the direction of decreasing roots. Since this simple expedient eliminated the problem completely, we did not investigate the matter further.

4.4. Bessel functions. The function J_n satisfies the differential equation (44). The first root is found via Algorithm 3 with the starting value $x_s = n + n^{\frac{1}{3}}$ motivated by (45). The values for $J_n(x_s)$ and $J'_n(x_s)$ are calculated by means of a standard scheme found, for example, in sections 9.12 and 9.1.27 of [1].

In each case, we calculated the first $20\,n$ roots of J_n, and results of these numerical experiments are summarized in Table 6.

4.5. Prolate spheroidal wave functions. The function ψ_n satisfies the ODE (49); the reader is referred to [14, 10] for the procedure we used to to evaluate the eigenvalue χ_n in each case. The determination of the first root is analogous to that for the Legendre polynomials, since the symmetry structures of these two classes of functions are similar. Results of these numerical experiments are summarized in Table 7.

4.6. Observations. Several observations can be made from Tables 3–7 and from our more detailed numerical experiments.

1. The cost of the algorithm increases linearly with n (the number of roots), as expected, and is not very sensitive to the choice of the special function.

2. While the procedure for the determination of the first root to be found depends somewhat on the class of special functions to which the algorithm is applied, the principal part of the calculation ("marching" from one root to the other) is performed by a single "black-box" subroutine. The Laguerre polynomials are the single (fairly minor) exception (see Remark 11).

3. The algorithm determines the roots to (essentially) machine precision, more or less independently of the value of n or the type of special function, with the

exception of the Laguerre polynomials. In the latter case, the [illegible] precision is virtually independent of n, while the [illegible] error is proportional to n, due to the concentration of nodes near 0.

4. The absolute error in the evaluation of the weights is proportional to the number of nodes n, while the relative error is proportional to n in the Hermite and Laguerre cases and $n^{3/2}$ in the Legendre case. This phenomenon is well known, and [15, 11] describe two different procedures for the reduction of the rate of growth of the error to n in the case of classical Gaussian quadrature (Legendre case).

5. The obtained Gaussian quadratures evaluate the appropriate integrals with the precision proportional to n.

6. When actually applied on a modern computer, the algorithm is extremely fast: Finding 1,000,000 roots in double precision takes from 0.5 to 3 seconds on a low-end desk-top PC. So, while extended precision calculations tend to be almost 500 times slower than the double precision ones, it might be attractive to precompute the necessary nodes and weights in extended precision, perhaps in advance. Indeed, even constructing 1,000,000 nodes in extended precision costs only a few minutes of CPU time.

Acknowledgments. We thank Dr. Mark W. Tygert for many useful discussions and suggestions. Furthermore, we thank an anonymous referee for several suggestions that improved the paper significantly.

REFERENCES

[1] M. Abramovitz and I. A. Stegun, eds., *Handbook of Mathematical Functions*, Dover, Mineola, NY, 1972.
[2] J. S. Ball, *Automatic computation of zeros of Bessel functions and other special functions*, SIAM J. Sci. Comput., 21 (1999), pp. 1458–1464.
[3] G. Birkhoff and G.-C. Rota, *Ordinary Differential Equations*, Ginn, Boston, 1962, Chap. 10.5.
[4] G. Dahlquist and A. Björk, *Numerical Methods*, Dover, Mineola, NY, 2003.
[5] W. Gautschi, *Orthogonal Polynomials: Computation and Approximation*, Oxford University Press, London, 2004.
[6] G. H. Golub and J. H. Welsch, *Calculation of Gauss quadrature rules*, Math. Comp., 23 (1969), pp. 221–230.
[7] M. Gu and S. Eisenstat, *A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 172–191.
[8] I. Ryzhik and I. Gradstein, *Table of Integrals, Series, and Products*, Academic Press, NY, 1980.
[9] Y. Ikebe, Y. Kikuchi, and I. Fujishiro, *Computing zeros and orders of bessel functions*, J. Comput. Appl. Math., 38 (1991), pp. 169–184.
[10] V. Rokhlin and H. Xiao, *Approximation formulae for certain prolate spheroidal wave functions valid for large values of both order and band-limit*, Applied and Computational Harmonic Analysis, 22 (2007), pp. 105–123.
[11] P. N. Swarztrauber, *On computing the points and weights for Gauss-Legendre quadrature*, SIAM J. Sci. Comput., 24 (2002), pp. 945–954.
[12] G. Szegö, *Orthogonal Polynomials*, 4th ed., AMS, Providence, RI, 1978.
[13] M. Tygert, *Recurrence Relations and Fast Algorithms*, arXiv:cs/0609081, 2006. Technical report YALEU/DCS/TR-1343, Yale University, New Haven, CT, 2006.
[14] H. Xiao, V. Rokhlin, and N. Yarvin, *Prolate spheroidal wavefunctions, quadrature and interpolation*, Inverse Problems, 17 (2001), pp. 805–838.
[15] E. Yakimiw, *Accurate computation of weights in classical Gauss-Christoffel quadrature rules*, J. Comput. Phy., 129 (1996), pp. 406–430.

SIAM J. SCI. COMPUT.
Vol. 29, No. 4, pp. 1439–1459
© 2007 Society for Industrial and Applied Mathematics

A NEW APPROACH TO SIMULATING FLOW IN DISCRETE FRACTURE NETWORKS WITH AN OPTIMIZED MESH*

HUSSEIN MUSTAPHA† AND KASSEM MUSTAPHA‡

Abstract. Natural fractured media are highly unpredictable because of existing complex structures at the fracture and at the network levels. Fractures are by themselves heterogeneous objects of broadly distributed sizes, shapes, and orientations that are interconnected in large correlated networks. With little field data and evidence, numerical modeling can provide important information on the underground hydraulic phenomena. However, it must overcome several barriers. First, the complex network structure produces a structure difficult to mesh. Second, the absence of a priori homogenization scale, along with the double fracture and network heterogeneity levels, requires the calculation of large but finely resolved fracture networks resulting in very large simulation domains. To tackle these two related issues, we reduce the highly complex geometry of the fractures by applying a local transformation that suppresses the cumbersome meshing configurations while keeping the networks fundamental, geological, and geometrical characteristics. We show that the flow properties are marginally affected while the problem complexity (i.e., memory capacity and resolution time) can be divided by orders of magnitude. The goal of this article is to propose a method of resolution which takes into account the geometrical complexity met in the networks and which makes it possible to treat a few thousand fractures. The principal aim of this article is to present a tool to slowly modify the structures of the fracture networks to have a good quality mesh with a marginal loss in precision.

Key words. fractured medium, Darcy flow, stochastic discrete fracture network, mixed hybrid finite element method, projection method

AMS subject classifications. 74S05, 76S05, 86A05, 74S30

DOI. 10.1137/060653482

1. Introduction.

The underground waste repository projects and exploitation of hot, dry, rock geothermal energy have spurred studies in fracture networks transport properties. Many site-specific studies have flourished around various projects [7], which are generally based on a careful characterization of the structure of the fractured rock mass, with the classical difficulty of deducing three-dimensional information from one- or two-dimensional field data [3]. Then the hydraulic properties are computed by using reconstructed model networks, based on the experimental geometrical characteristics, and various flow models. The flow fluid modeling in underground media requires taking into account the very high geological heterogeneity. The complexity of the geological mediums comes from metamorphic and sedimentary processes, and mechanics which create structures having hydraulic properties varying on several orders of magnitude and correlated on a broad range of scales. A general overview of stochastic generation of fractured media problems is given in [11]. Numerically, the challenge is to integrate this range of heterogeneity in models of great extension and good resolution outcome, with linear systems from 10^6 to 10^8 unknown factors. In the fractured medium with a matrix of very low permeability, the fluid flow is focused in the highly heterogeneous fractures [21]. The objective of numerical modeling consists of simulating hydraulic phenomena in a large number of networks containing a high

*Received by the editors March 2, 2006; accepted for publication (in revised form) December 13, 2006; published electronically June 19, 2007.
http://www.siam.org/journals/sisc/29-4/65348.html
†Reservoir Engineering Research Institute (RERI), Palo Alto, CA 94306 (hmusta@rerinst.org).
‡Department of Mathematical Sciences, King Fahd University of Petroleum and Minerals, Dhahran, 31261, Saudi Arabia (kassem@kfupm.edu.sa).

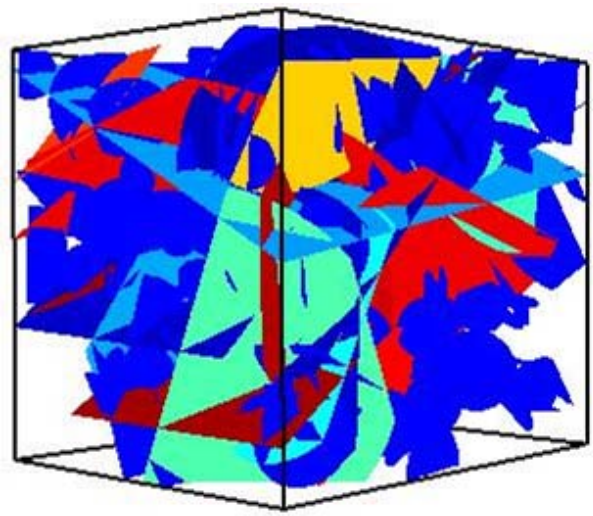


FIG. 1. *Discrete fracture network: FN. FN contains* 500 *fractures generated in a cube of size* $8 * lmin$*, where lmin is the minimal size of fracture.*

number of fractures, e.g., from 10^3 to 10^5. In the second phase we seek an optimum between the quality of the hydraulic simulation approximation on one hand and the costs of calculation (CPU time and memory capacity) on the other.

The major purpose of the present paper is to develop a full solution of the steady flow problem in a general three-dimensional complex network made of two-dimensional ellipse fractures as illustrated in Figure 1. A literature survey shows that, in most cases, the description is two-dimensional [25] or that the three-dimensional network is replaced by a capillary model. The intersection of two fractures is schematized by a channel that joins their centers with an effective hydraulic conductivity that results from simple geometric arguments; this model is improved in [7, 3, 6, 2] by introducing flow channels in the fractures, which results in complex capillary networks. In this model, the authors assume that the flow occurs through bonds joining the center of each fracture to the center of adjacent fractures, provided that the fractures are connected (Figure 2). Nordqvist et al. [22] went one step further by building the network with spatially variable apertures for which flow was solved beforehand. However, again, the final representation is a tube model. These capillary models suffer two main flaws. First, the assignment of the bond conductivities is somewhat arbitrary and the assumption that the fluid flows from center to center is unrealistic. Second, only pair intersections are taken into account. Another method is the boundary element method [3], in which the solution is correctly computed in the planes of the fractures. This method is limited to the homogeneous fractures, and the treated fracture networks contain a very limited number of fractures [10, 16]. Maryška, Severýn, and Vohralík [17] and Vohralík [27] have developed a new method based in the simplification of complex configurations. The authors have showed an extension of the mixed hybrid finite element (MHFE) method to a geometrical domain composed by a network of planar circle disks placed in three-dimensional space. In their works, the MHFE method is validated on a simple geometrical case. But the simplification method of the fracture network complexities is not validated. The authors showed only an example of computed solution in a fracture network without discussion about

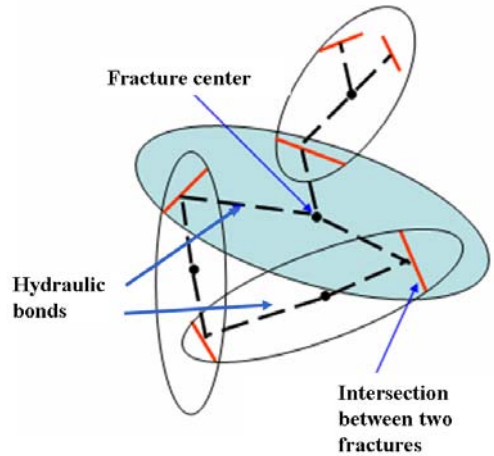


FIG. 2. *Flow fluid simulation in capillary network model* [7].

the error in precision, or a comparison with a solution computed without simplification of geometry. To conclude, the general limits of these methods are obtained from the limited number of fractures and by taking into account only of a part of natural networks complexity.

In Figure 1, we present a very complex discrete fracture network (FN) [5, 4, 9]. This network contains difficult configurations that complicate the generation of the fractures' mesh (to simplify the notation, a mesh of a network means a triangulation). Their number and their difficulty increase with the number of the generated fractures. These configurations are found mainly in the stochastic generation and in the natural environment's fracture networks. It is possible to discretize these configurations provided that we have a sufficient number of elements. However, for very large fracture networks, these configurations are, generally, not essential for the hydraulic properties but very penalizing for the number of elements of the mesh. A mesh of good properties contains a minimum number of triangles for a given precision of resolution.

To realize our object, we have developed and validated a new method of simplification different from the method of Maryška, Severýn, and Vohralík [17] and Vohralík [27]. The principal aim of this article is to present a tool to slowly modify the structures of the fracture networks to have a good quality mesh with a marginal loss in precision. This is the originality and the news of this paper. The developed method is not based on work of other authors; it is original. Then, the goal of this article is to propose a method of resolution which takes into account the geometrical complexity met in the networks and which makes it possible to treat a few thousand fractures. The achievement of this objective is done in two parts: (1) establishment of the mesh and (2) resolution of the flow equation. The principles of the proposed method are based on two essential points: (1) the penalizing configurations are replaced by very close configurations for which we can easily generate a good quality mesh without adaptation; (2) the resolution method of the flow equation is efficient because it is based on a good quality mesh.

The MHFE methods are known to accurately approximate the velocity field and locally conserve the mass on each element. Thus, we have decided to use the lowest

order Raviart–Thomas mixed finite element method in order to find an approximate solution of the local second-order elliptic problem with a discontinuous permeability tensor. We can easily note that an essential property of the fracture networks is the existence of interelement edges (i.e., intersection edges) which belong to three or more triangular elements. We note that the MHFE method is used in this work as a tool to give an idea about the computed solution, and it is not developed by the author.

This paper is organized as follows: First, we present the differential and algebraic equations describing the problem of stationary Darcy flow in a fracture network. Different aspects of the numerical approach and the discretization of Darcy's law by the MHFE method are discussed in detail in section 3. To apply the MHFE method, we need to have a complete mesh of the network. Then, in section 4 we present the difficulties that accompany the generation of the mesh of fractures, and then we explain in detail the proposed approach (i.e., the principal aim of this article) to analyze these difficulties. We present numerical examples in order to validate the developed method and show the originality and efficiency of our approach in section 5. A discussion about the optimality of the mesh is detailed in section 6. Finally, a summary and a preview of future work are presented.

2. Mathematical model. The original three-dimensional fracture network can be approximated by planar elliptic whose frequency, size, assigned aperture, and orientation are statistically derived from field measurements and considers two-dimensional Darcy flow in such a network. However, due to the high computer requirements, it is only possible to solve local problems by using stochastic discrete fracture network models. We refer the reader to [10] for more details. The network is included in a cube of size L; the fractures are modeled by ellipses. The fractures' length is modeled by a random power-law distribution. We define the fracture network R by $R = \bigcup_{i=1,NF} \bar{Fi} \backslash \partial R$. R is the bounded domain with boundary $\partial R = \Gamma^D \cup \Gamma^N$, and Γ^D (resp., Γ^N) corresponds to the Dirichlet (resp., Neumann) boundary conditions. Fi is the fracture i, and NF is the total number of fractures. We seek the fracture flow velocity u (a two-dimensional vector in each Fi), which is the solution of the problem

$$u = -K(p + z) \quad \text{in } R, \tag{2.1}$$

$$\nabla . u = f \quad \text{in } R, \tag{2.2}$$

$$p = p^D \quad \text{in } \Gamma^D, \tag{2.3}$$

$$u.n = u_N \quad \text{in } \Gamma^N, \tag{2.4}$$

where all the variables are expressed in the local coordinates of appropriate Fi and the differentiation is always done with respect to these local coordinates. Equation (2.1) is Darcy's law, (2.2) is the mass balance equation, and (2.3)–(2.4) describe Dirichlet and Neumann boundary conditions. The variable p denotes the piezometric head, $p = \bar{p}/\rho g$, where $\bar{p}$ is the fluid pressure, g is the gravitational acceleration constant, ρ is the fluid density, f represents stationary sources or sinks density, and z is the elevation, i.e., the upward vertical three-dimensional coordinate. The second-rank tensor K of hydraulic conductivity is a function of the original three-dimensional fracture aperture, wall roughness, and filling. We suppose that K is symmetric and uniformly positive definite on each Fi.

3. Numerical method in fracture network. The numerical approach used to solve the system of equations (2.1)–(2.4) is based on the MHFE method. For

more details about the theoretical and validation of the MHFE method in a three-dimensional domain composed by a simple set of rectangles, we refer the reader to [17]. We consider a spatial triangulation of the computational domain R consisting of triangles. Throughout the paper we use the following notation: b is the discretized triangle, E is the edges of the triangle b, NT is the number of triangles in the mesh, and NE is the number of edges in the mesh not belonging to Γ^D.

3.1. Discretization of Darcy's law. The essential idea of the mixed methods is to individually approximate Darcy's law and the flow equation, and we get additionally the Darcy velocity as an unknown function. Thus, the variation formulations of the given PDE systems are chosen in such a way as to have the pressure and its gradient in the basic formulation. The Raviart–Thomas space of lowest order ($RT0$) is used to approximate the velocity [23, 26, 13, 14]. The main idea is to express the velocity over each grid cell with respect to the fluxes across the cell edges. Based on Raviart–Thomas approximation space, the vectors u in (2.1) can be expressed as

$$u = \sum_{b} \sum_{E \in \partial b} q_{b,E} W_{b,E}, \tag{3.1}$$

where $W_{b,E}$ is an $RT0$ basis function and $q_{b,E}$ is the total flux across an edge E. These vectors are determined by their normal fluxes across the cell edges. By inverting K, Darcy's velocity equation becomes

$$K^{-1}u = -(\nabla p + \nabla z). \tag{3.2}$$

We denote by E' (resp., b' or b'_i) an edge (resp., a triangle) of the triangulation mesh. Multiplying (3.2) by the test function and integrating by parts, the total flux is suppressed through each edge E as a function of the cell-pressure average p_b and the edge-pressure averages $tp_{b,E}$ for each triangle b, i.e.,

$$q_{b,E} = \alpha_{b,E} p_b - \sum_{E' \in \partial b} (B_b^{-1})_{E,E'} tp_{b,E'}, \quad E \in \partial b, \tag{3.3}$$

where B_b is a $3 * 3$ symmetric positive definite matrix whose elements are

$$B_b = \int_b W_{b,E}^T K^{-1} W_{b,E'} \quad \text{and} \quad \alpha_{b,E} = \sum_{E' \in \partial b} (B_b^{-1})_{E,E'}.$$

The continuity of the fluxes across the interelement boundaries provides

$$q_{b,E} = \left\{ \begin{array}{ll} \sum\limits_{E', E' \in \partial b'} q_{b',E} & \text{if } E \notin \Gamma^N,\ E \in \bigcap\limits_{i=1,nbE} b'_i, \\ q^{N,E} & \text{if } E \notin \Gamma^N \end{array} \right| , \tag{3.4}$$

where nbE is the number of triangles which contain the edge E. We note that $nbE > 2$ if E is an intersection edge between fractures. Equations (3.3) and (3.4) lead to the following algebraic linear system with main unknowns the pressure-cell averages in P and the pressure-edge averages in Tp:

$$S^T P - MTp - V = 0, \tag{3.5}$$

where S is a sparse matrix of dimension $NT * NE$ with nonzero elements given by $S_{b,E} = \alpha_{b,E}$, $E \in \partial b$, M is an $NE * NE$ sparse matrix with nonzero entries defined as $M_{E,E'} = \sum_{b,E,E' \in \partial b} (B_b^{-1})_{E,E'}$ for $E \notin \Gamma^D$, and V is an NE-dimensional vector corresponding to the Dirichlet and Neumann boundary conditions. For more details, we refer the reader to [13, 19, 20].

3.2. Discretization of the mass conservation equation. By integrating the mass conservation equation where the test functions are successively replaced by the basis functions of M^0 (M^0 is the space of discrete functions constant by triangles), we get

$$\sum_{E\in\partial b} q_{b,E} = f^b, \tag{3.6}$$

where f^b is the approximation of the sink/source term over b.

Therefore, using (3.3) to replace the sum of fluxes in (3.6), we obtain an ordinary differential system given in its matrix form

$$DP - STp = F, \tag{3.7}$$

where D is an $NT * NT$ diagonal matrix whose coefficients are $(D_{b,b}) = \alpha_b = \sum_{E\in\partial b} \alpha_{b,E}$ and F is a vector of dimension NT; it corresponds to the source/sink function as well as to the imposed pressure given by the Dirichlet boundary conditions.

3.3. The derived algebraic systems. The spatial discretization of the governing equations obtained by applying the mixed hybrid formulation leads to two systems. The first one, given in (3.5), is an algebraic system of unknowns P and Tp, and the second is an algebraic equation (3.7). They can be written in a matrix form given by

$$\begin{bmatrix} D & -S \\ S^T & M \end{bmatrix} * \begin{bmatrix} P \\ Tp \end{bmatrix} = \begin{bmatrix} F \\ V \end{bmatrix}. \tag{3.8}$$

4. Fracture network triangular meshes. The discretization of fracture networks into triangular meshes has presented a crucial problem. First, each fracture has to be discretized into a triangular mesh respecting the intersections with other fractures. In order to decrease the number of elements and to avoid ill-conditioned matrices resulting from the cases where there exist elements with very small angles, the algorithm should in addition simplify the geometrical situation.

The fracture F illustrated in Figure 3 contains complex configurations like the small distance between the points. To have a mesh of good quality, we must use an adaptation method to be able to remove the flattened triangles. In Figure 4, we present the distribution of the values of network mesh triangles angles. The results obtained show that 28% of the angles are less than 40°, and 9% of the angles are more than 100°. In conclusion, the resulting mesh is not of good quality. To solve this problem, we use an adaptation method to improve the quality of the mesh. Such an approach requires a high capacity memory cost because of the large number of elements (triangles) obtained. Moreover, the resulting linear system is of large size and leads to a significant CPU time. Because our aim is to simulate a very large fracture network, the refinement of the mesh is an ineffective solution. A more powerful solution is required. The principle of the suggested solution consists in very slightly modifying the positions of the intersections to transform the complex configurations to very close configurations. This facilitates the mesh generation. In other words, the geometry is modified slightly to contribute to the generation of a good quality mesh. The suggested method consists in discretizing the boundaries of the fractures and the intersections between the fractures on a three-dimensional regular grid ($3DRG$).

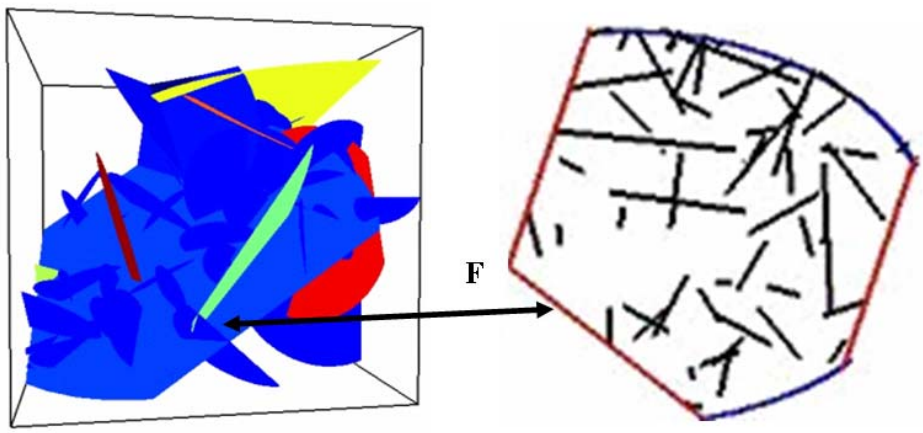


FIG. 3. *Left: The fracture F of radius $6 * lmin$ is extracted from FN. Only the 56 fractures intersecting the fracture F are represented in the figure. Right: Plane representation of the fracture F.*

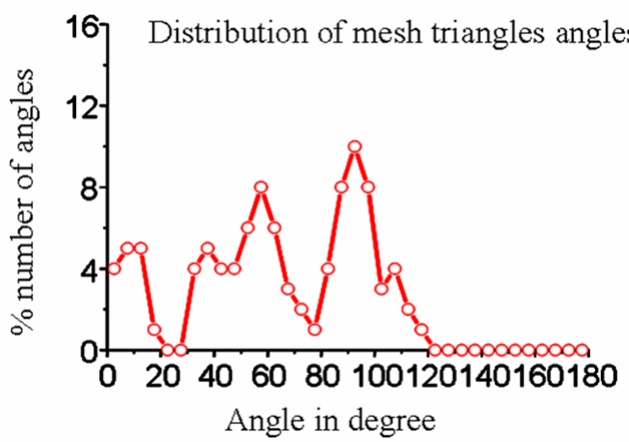


FIG. 4. *Distribution of mesh triangles angles of FN. The angles are grouped in packages of 5°.*

4.1. Discretization of fractures on a 3*DRG*. To have a complete conforming mesh of the fracture networks, the intersections between fractures must be treated in the same manner. During the considered modification, the following constraints must be respected: (1) connectivity between the fractures must be preserved; (2) the discretization of an intersection in three dimensions is the same in all the fractures that belong to it. The suggested approach respects the constraints mentioned above and consists in projecting the intersections between the fractures of the network and the boundaries of fractures, on the cubes of a regular grid of step equal to the uniform step of the mesh (h). The principal characteristics of this method are that (1) each intersection of fracture or limit of fracture is approached by the whole of the crossed cubes; (2) two intersections of fractures are intersected if and only if they have a joint cube. These characteristics make it possible to respect the connectivity of the

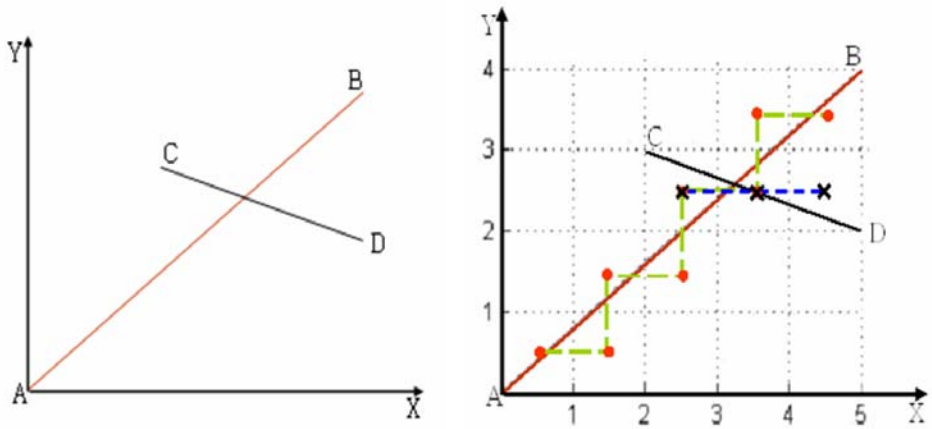


FIG. 5. *Left: Two segments AB and CD to be discretized. Right: Projected AB and CD on a 2DRG. Each segment is approached by the whole of the centers of the squares crossed by the segment.*

network and to ensure a minimal distance and angles of 90° or 180° between two successive cubes belonging to the discretized object. To simplify the ideas, we explain the principle in two dimensions and then generalize the results in three dimensions. Let us consider the two segments of straight lines AB and CB, which represent the intersections of a given fracture (Figure 5, left). The segments AB and CD are approached by the set of squares centers crossed by these segments (Figure 5, right). The segment AB is approached by the set of points ($\bullet$) and CD is replaced by the crosses ($\times$). The angle between the approximation of AB and CD is equal to 90° and outdistances it between the projected points is equal to the regular grid step (h). To generalize this method in three dimensions, the boundaries of the fractures and the intersections between the fractures of the network are approached by the centers of the cubes crossed by these segments (Figure 6, right).

4.2. Fractures' mesh. After discretizing the boundaries and the intersections of the fractures in a $3DRG$, we need to make a passage of the three-dimensional reference mark toward the plane of the fractures. This phase is obligatory in order to be able to generate the mesh of the fractures. At the time of the discretization in the $3DRG$, each intersection or boundary is represented by a succession of cubes which are very close to the fracture. The centers of these cubes are projected in the plane of the fractures and are connected to each other to form a succession of edges corresponding to the intersection of the projection (Figure 7). The regularization of the complexity of fractures is illustrated in Figure 8. The geometry of the fracture F contains penalizing configurations. Using the suggested method, we have obtained a rather regular geometry which does not present any problem of mesh, and where the stresses on the geometry are relieved. In conclusion, the developed method offers a regular base to generate a mesh of good quality (Figures 9 and 10). The mesh quality is measured with respect to the number of angles equal to 60°. Therefore, we do not need to adapt the mesh to improve the quality anymore. We can generate the mesh for all the fracture networks presenting complex configurations. As an example, we illustrate in Figure 11 (left) the mesh of the network FN.

The grid mesh of each fracture is stored in three vectors. The first vector is of

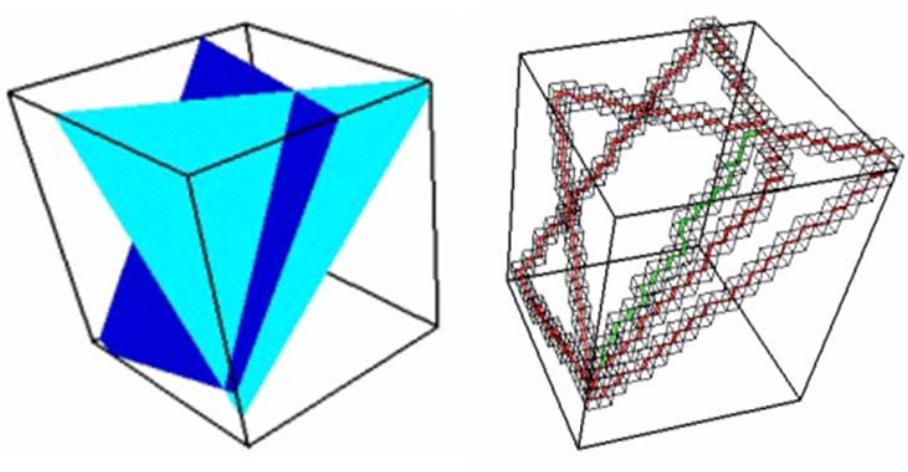


FIG. 6. *Discretization of the intersection and the edges of the two fractures represented in the left. Right: The projection of the boundaries of the fractures and their intersections on the centers of the cubes of a 3DRG.*

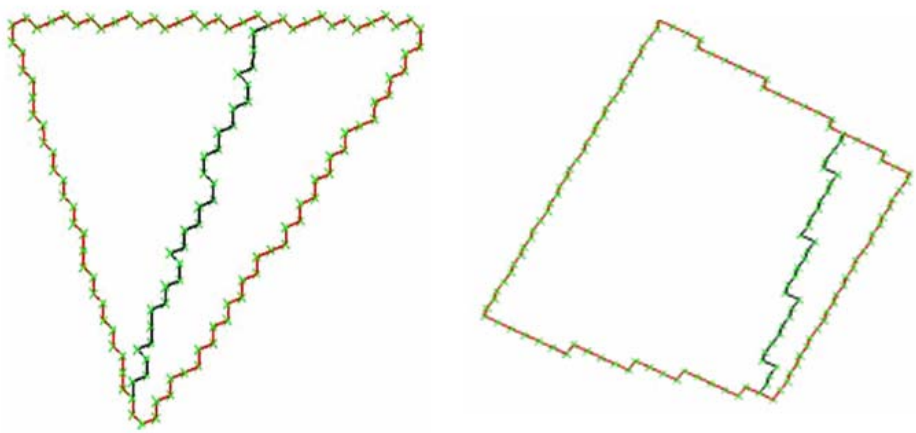


FIG. 7. *Projection of the discretization of the two fractures, presented in Figure* 6 *(right), in their respective planes. The intersection between the two fractures is projected differently in the planes of the fractures.*

type [illegible] to store the coordinates of the vertices, and it is of size Nv (i.e., the number of mesh vertices). The two other vectors are of type [illegible], and of sizes NE and NT, respectively. For each vertex of the mesh, there are two coordinates to store. For each edge (resp., triangle), we have seven (resp., six) values to store. We suppose that the number of triangles or the number of vertices is equal to half of the number of edges; see [12] for more details. Therefore, we must store $2 * Nv$ [illegible] $+ 7 * NE$ [illegible] $+ 6 * NT$ [illegible], or in total the memory mesh $MemMesh = 6 * NE$ [illegible]. We note that $NE = SL$.

We note by $nz(A)$ the number of nonzero elements of the linear system matrix A. On each line of A we have five nonzero elements, except for the lines corresponding

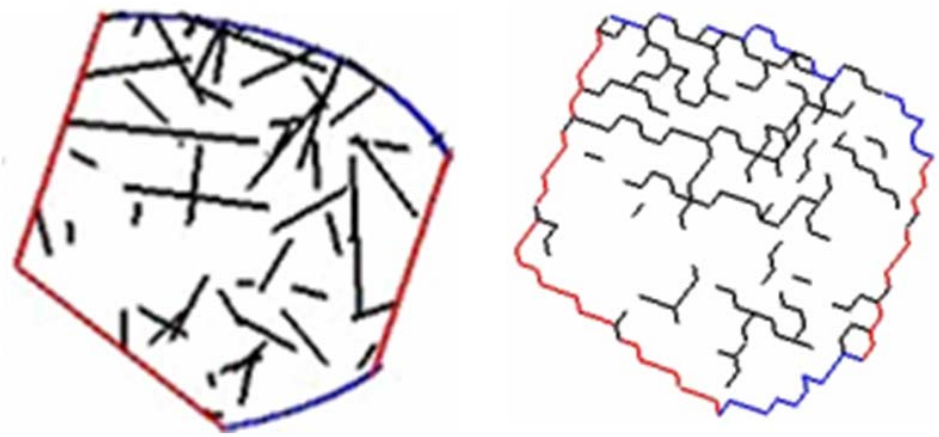


FIG. 8. *Projection method. Left: The fracture F of FN. Right: The obtained geometry of F by the projection method.*

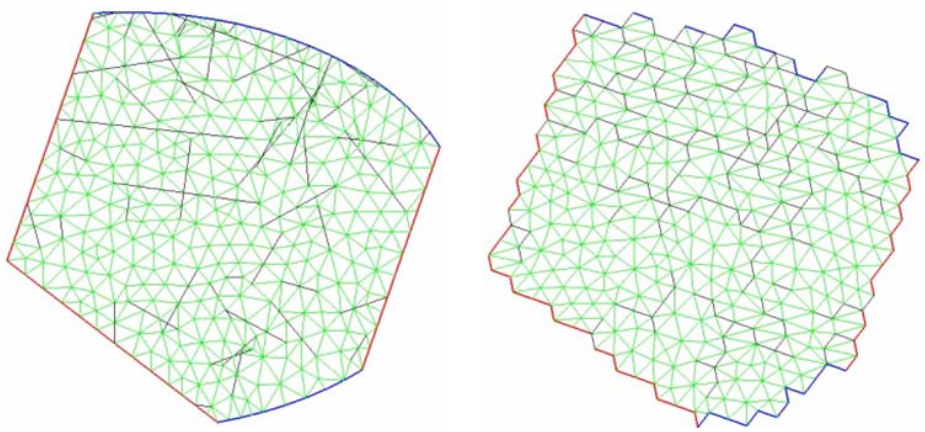


FIG. 9. *Left: Mesh of the fracture F without projection on the $3DRG$. Right: Mesh of the same fracture but after projection.*

to the edges' intersections which have more than five elements. Therefore, $nz(A)$ is equal to about $5 * SL$.

5. Validation tests and results. In this section, we develop three types of tests. In the first test, we validate the projection method by comparison to a reference solution computed in finest mesh. We give an idea about the error in precision in the second test. To look further into the study of the error in precision, we present a Monte Carlo simulation in the third test. In the following, we present the details of each one of tests. We note that all the fracture networks used are generated stochastically. The definition of fractures' principal characteristics is based on observations and experiments. The treated examples represent real tests of natural fracture networks.

A. [illegible] 1: [illegible]. Projection on a $3DRG$ of the intersections and the boundaries of fractures is an intermediate step to generate the mesh and to solve the flow equations. To validate such an approach, we have compared

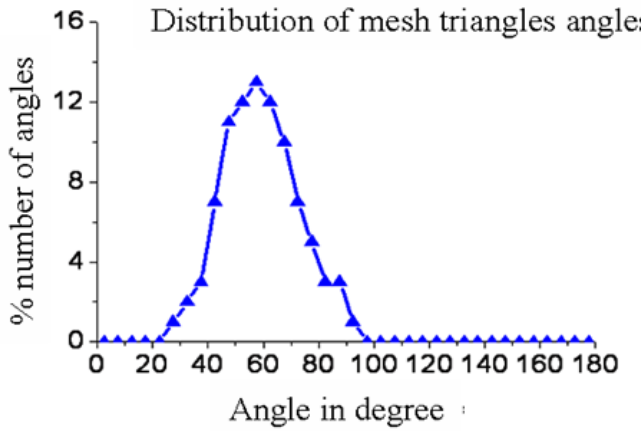


FIG. 10. *Distribution of the mesh triangles angles of FN. All the angles that are not between* $20°$ *and* $100°$ *have disappeared.*

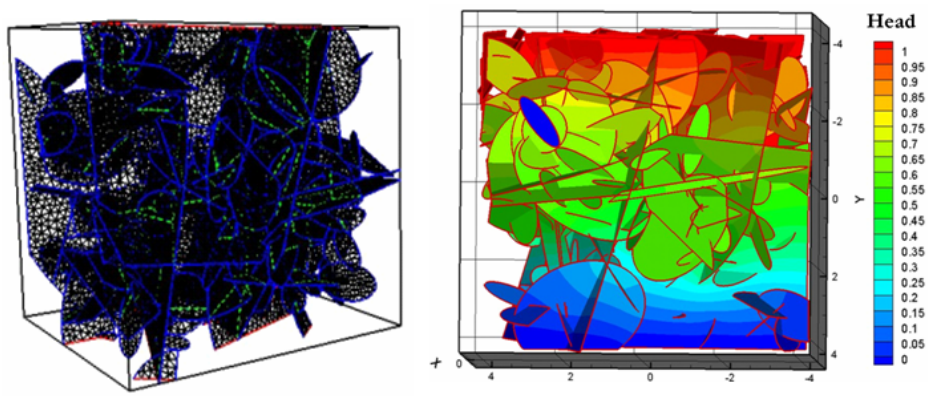


FIG. 11. *Example of mesh and flow computation in the network FN (left picture: mesh and right picture: flow computation).*

the approximate solution with a reference solution obtained from a very fine mesh (e.g., using a very small step of mesh h) without using the projection method. We chose the fracture network (FN) because the mesh of this network is possible without and with projection. We note that the developed method is validated with more examples. To compare the mesh in terms of quantity and quality, we compared the number of triangles and the size of the linear system corresponding to each mesh and we measured the quality of the mesh compared to the values of the angles of the triangles (Table 1 and Figure 12). An exhaustive study of other more localized measurements consists in calculating the differences between the two methods: the flows crossed by the fractures and the flows crossed by the intersections (Table 2). The projection method makes it possible to obtain a solution close to the solution obtained without projection. The error made while projecting on the grid is small. In general, it is about 10^{-5}.

In this test, we showed that by working with a finest mesh, the proposed approach

TABLE 1

Comparison of two meshes of the network FN with and without the projection method. NT is the number of the triangulation mesh.

Mesh	Without projection	With projection
h	0.08	0.08
NT	207032	206696
SL	315627	313776
$Time$	5.5	5.1

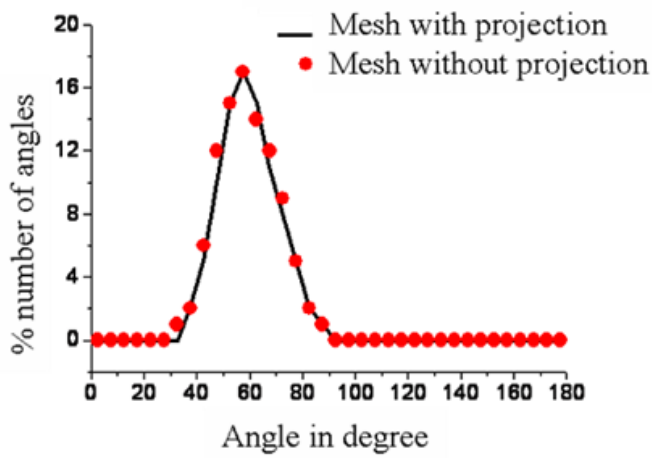


FIG. 12. *The quality of the mesh of FN. The black and grey curves correspond to the meshes that are obtained with and without projecting on the regular grid. The angles are classified in packages of $5°$. $h = 0.08$.*

TABLE 2

The difference between measurements corresponding to the reference solutions and approximate solutions of FN.

NF	500
h	0.08
$Kref$	45.9017
$Kapp$	45.9015
EK	$4 * 10^{-6}$
EF	$1.8 * 10^{-5}$
EI	$2 * 10^{-5}$

gives a very precise solution. Then, this very interesting result helps us to study the very complex fracture networks. In fact, because of complex configurations of the fracture network, the triangulation of this network is in most cases impossible. To conclude, the projected method can give a reference solution (working in a finest mesh) for all complex fracture networks. Finally, Figure 11 (right) shows the distribution of the piezometric head in the whole simulated fracture network FN. To fix the idea, in the following we consider the approximate solution obtained using the projection method in finest regular grid as a reference solution.

B. [illegible] 2: [illegible] FN [illegible]. In this section, we measure the precision error by varying h. First, we evaluate the errors

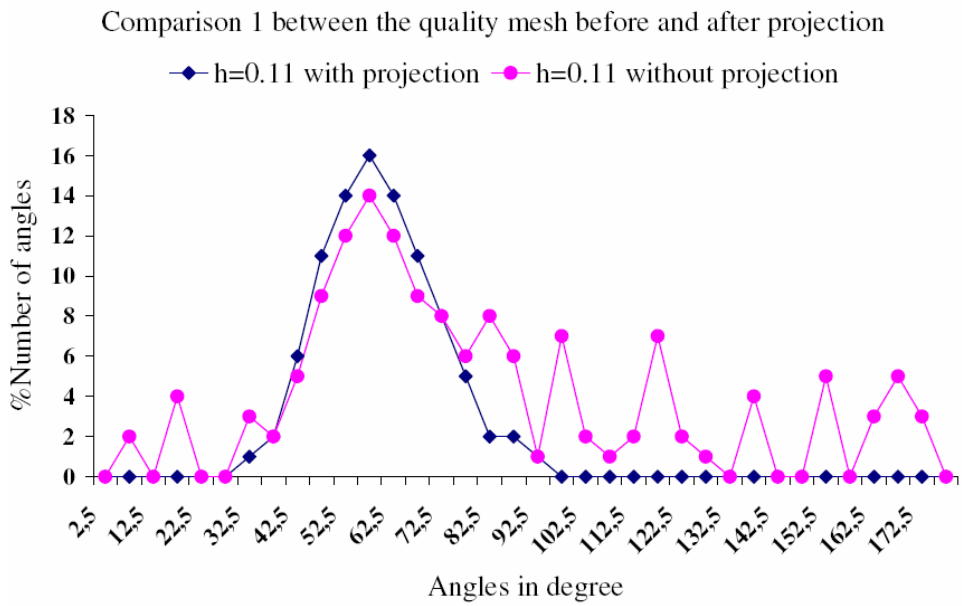


FIG. 13. *Comparison* 1 *between the quality mesh before and after projection.* $h = 0.11$.

in global equivalent permeability, the flow across each fracture and each intersection. Then we observe the mesh quality obtained; we compute the linear system size and the memory capacity for each step h. In this test, we simulate the flow in the fracture network FN. For this network we compute some solutions for different values of h. The aim is to observe the evolution of precision error, the mesh quality, and the improvement in CPU time and memory capacity with respect to h.

1. [illegible]. In Figures 13, 14, and 15, we present the curves that represent the mesh quality of the network (FN) for each mesh step using the projection method and without it. These figures show that the majority of mesh triangle angles is close to 60° for the projection method and the quality is bad without projection. Then, we deduce that the mesh quality remains very good by varying the mesh step for the projection method. This result is interesting because it is necessary to have a good quality mesh to correctly compute all solutions by reducing the problem size at the same time.

2. [illegible]. For each step h, we calculate the differences between the approximate solution and the reference solution calculated on a finest grid (Figure 16). The difference between the approximate solution and the reference solution always remains weak when we use the projection method. The errors increase quickly if we do not project. For example, for the mesh step $h = 0.21$ we have a loss in precision equal to 33% for the classical method and 2 to 4% for the projection method. We can conclude that the projection solution calculated by increasing the step mesh remains precise. We carry out a linear regression to consider the error corresponding to the equivalent permeability. According to what precedes, the straight regression lines represent more than 98% of the distribution of the points and the data are well estimated using the projection method. This result is coherent with the numerical estimate error of the MHFE method which is $O(h)$ (method of order 1). We checked in the two preceding parts that the approximate solution obtained with the projection

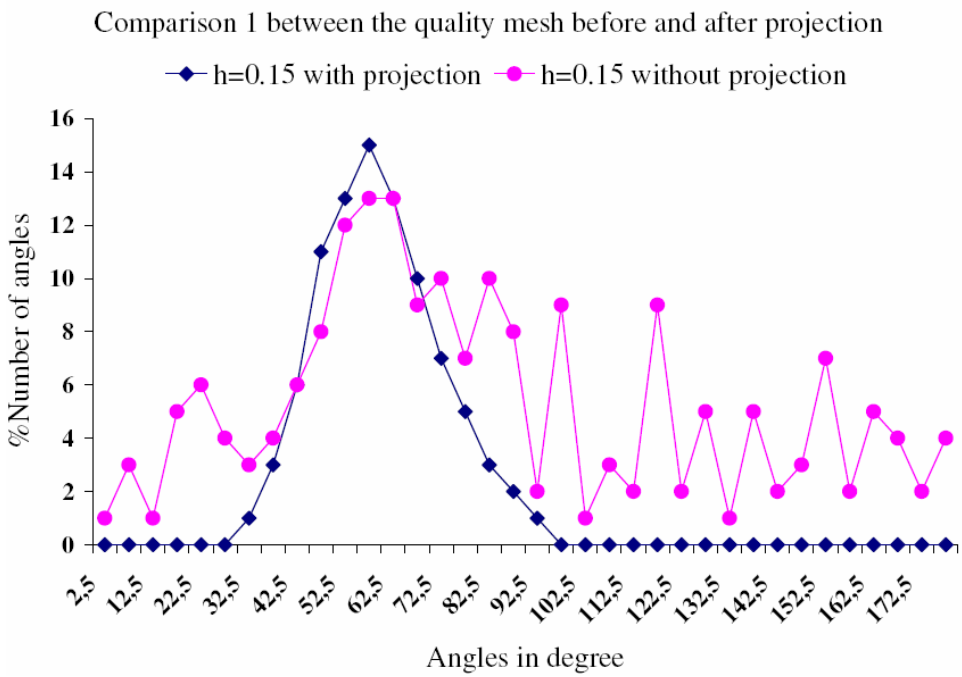


FIG. 14. *Comparison* 1 *between the quality mesh before and after projection.* $h = 0.15$.

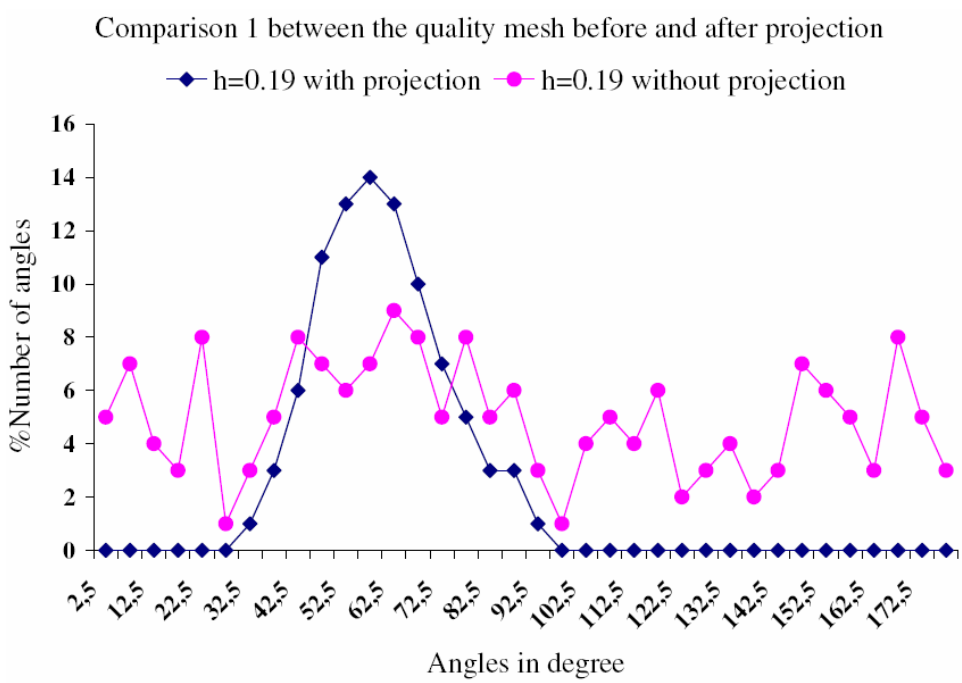


FIG. 15. *Comparison* 1 *between the quality mesh before and after projection.* $h = 0.19$.

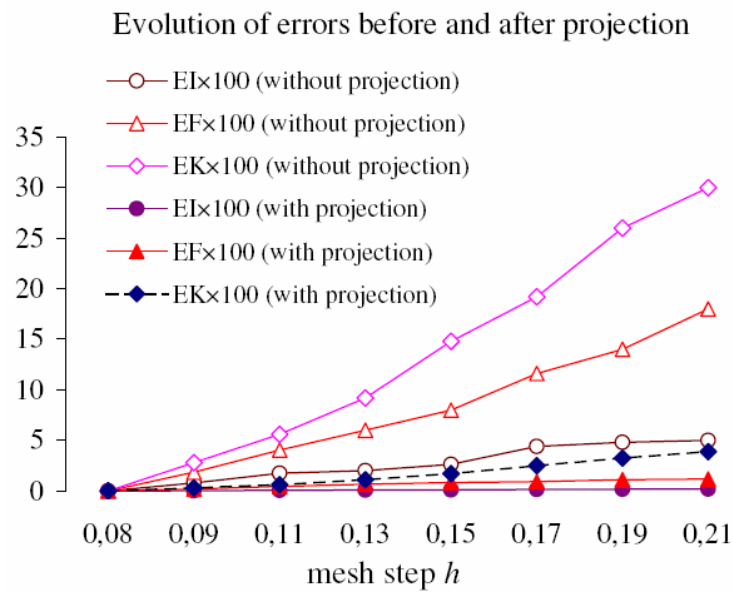


FIG. 16. *Evolution of the difference expressed as a percentage between the approximate solutions obtained with projection and without projection.*

TABLE 3
Linear system size, memory capacity, and CPU time for various mesh steps.

h	SL	$MemMesh$	$nz(A)$	$Time$
0.08	313776	1882656	$1.58 * 10^6$	27.65
0.21	50588	303528	$0.26 * 10^6$	2.36

method makes it possible to consider rather large steps of mesh.

3. [illegible]. The projection method provides an improvement in CPU time and memory capacity by decreasing the size of the problem. The larger the step of the mesh is, the better the improvements of CPU time and memory capacity are (Table 3). For one variation of h between 0.08 and 0.21 the linear system size (resp., the memory mesh and the nonzero elements of A) decreases from 313776 to 50588 (resp., from 1882656 to 303528 and from $1.58{*}10^6$ to $0.26 * 10^6$ real values to storage). To conclude, the decrease of memory capacity needed is of 85%. However, the decrease of the linear system size reduces the time of the linear system resolution. The obtained result (Table 3) shows that the CPU time decreases from 28 seconds to 2.5 seconds, using the direct linear solver [illegible] [8].

To conclude, the improvement in CPU time is equal to 92%. In this part, we show that the improvement in CPU time can be realized because the loss in precision is small with respect to the mesh step. We do not need to speak about the improvement in CPU when we do not use the projection method because the loss in precision becomes significant. To conclude, in these tests we want to realize a compromise between the precision of the solution and the CPU time.

C. [illegible] 3: [illegible]. The characteristics of fractures are generated stochastically. The orientation, the length, the position, and the eccentricity of the fractures are generated according to statistical laws. To generate a network of fractures, we fix the type of distributions and the seeds. For the same parameter dis-

TABLE 4
Parameter distribution type to generate fracture networks.

Parameter	Distribution type
Length of the fractures	Power law with exponent a = 3
Orientation	Uniform
Position	Uniform
Eccentricity of ellipses	Uniform

TABLE 5
Results obtained by simulating the flow in GN with a small value of h.

h	0.02
$SLmref$	1213100
$\sigma SLref$	350120
$Tmref$	175.3
$Tref$	55.2
$Kref$	36.48
$Kref$	12.23

TABLE 6
Results of simulations in set of 700 *networks GN; Sim: Simulation.*

GN	$Sim1$	$Sim2$	$Sim3$	$Sim4$
h	0.02	0.1	0.18	0.22
SLm	$1.2 * 10^6$	$0.4 * 10^6$	$0.17 * 10^6$	$0.1 * 10^6$
σN	$3.5 * 10^5$	$1.3 * 10^5$	$0.5 * 10^5$	$0.3 * 10^5$
Tm	175.3	43.45	13.16	6.9
σT	55.2	22.1	6.3	3.4
$\langle K \rangle$	36.89	35.83	34.89	34.23
σK	10.1	10.2	12.23	10.45

tributions, we can generate a high number of networks by changing the seeds. In this part, we choose a set of parameter distributions (SD) (Table 4). We generate 700 networks for SD by changing the seeds and fixing the exponent of the power law, the cube size, the minimal length of a fracture, and the density of the fracture (a threshold density). To begin, we note by GN the set of NN (NN: number of networks) networks generated with the distributions SD. Afterward, we compute the reference solutions represented by the average equivalent permeability reference ($Kref$) (e.g., average of the permeability computed, on a finest mesh, in the set of networks GN) and the standard deviation ($\sigma kref$) of this solution. We also seek the average size ($SLmref$) of the linear systems associated to the packages of networks, the average time ($Tmref$) of resolution of these linear systems, and finally, the variations of these values ($\sigma SLref$ and $\sigma Tref$). The results are summed up in Table 5. After computing the reference solution, we vary the mesh step h. The average numerical error for a given value of h is defined as the difference between the average permeability obtained for this step on one hand and the weakest step on the other hand. In this test, we vary h between 0.02 and 0.2 to observe the variations of the parameters $SLmref$, $Tmref$, and $Kref$ and of their variations ($\sigma kref$, $\sigma SLref$, $\sigma Tref$) in the sets of networks GN. We note by Tm (resp., Tm and $\langle K \rangle$) the average time (resp., the average size and the average of equivalent permeability) obtained for each step of mesh and by σTm, σSLm, and σk the standard deviations. The results (Table 6) show that the error in equivalent permeability, by varying h from 0.02 to 0.2, is about 7% on one hand and the improvement in CPU time of calculation and in memory capacity are,

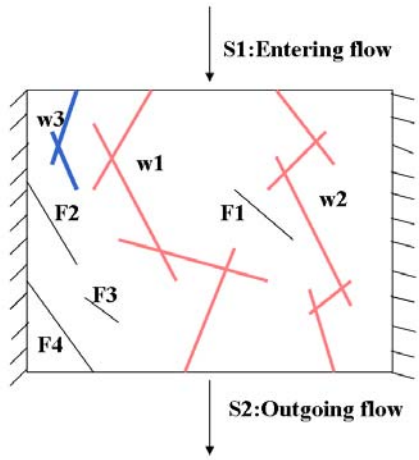


FIG. 17. *Two-dimensional fracture network. The fractures are modeled by lines. $w1$ and $w2$ cross the domain from $S1$ to $S2$. The other fractures are isolated and do not have an effect on the flow.*

respectively, about 96% and 92% on the other hand. These results are dependent on the value of the parameters and probably on deterministic information (e.g., position of the greatest fractures) available on the site. It is foreseeable that the more the quantity of data is significant, the better the quality of numerical simulation should be.

6. Discussion about the optimized mesh. The fractures of the network are linked by lines in three-dimensional space and by points in two dimensions (Figure 17). In Figure 17, the fractures are modeled by lines. The intersections between fractures are points. Suppose that the flow in the domain is from the side $S1$ to the side $S2$ and the other boundaries of the domain are impervious. We will study different cases that can be presented as follows:

A. [illegible]. The generation mesh of complex fracture networks using automatic generators is a challenge. Each generator mesh has a control parameter ε. This parameter can be the small distance or angle between two neighbor points or the small size acceptable of an edge mesh. If one of these cases is available, the generator cannot work. For example; in Figure 18 (left), we present a two-dimensional fracture network. The distances and angles between certain fractures are shown by the zoom (Figure 18, right) on Figure 18 (left). In general, the structures of the fracture network depend on the distribution of fractures. Because the fractures are distributed randomly, it is possible to have a distance or angle between tow fractures small than ε. The same conclusion can be obtained regarding the length of the fracture F (Figure 18, left) which can be smaller than ε. In this case, we can say that the generation mesh is impossible using an automatic generator.

Now, suppose that we do not have these cases. Then the small fracture's length or the small distance or angle between the fractures is greater than or equal to ε. In this case we have two possibilities: (1) The generation mesh is possible but perhaps the quality of the mesh is bad regarding the angle distribution (Figure 4). The quality of this case depends on the geometrical structure of the fracture networks. For example, if the number of fractures is small we can have a simple geometry without complexities, and then the generation of a good quality mesh is possible. In [24, 1], the authors simulate multiphase flow in fractured porous media. The studied fracture

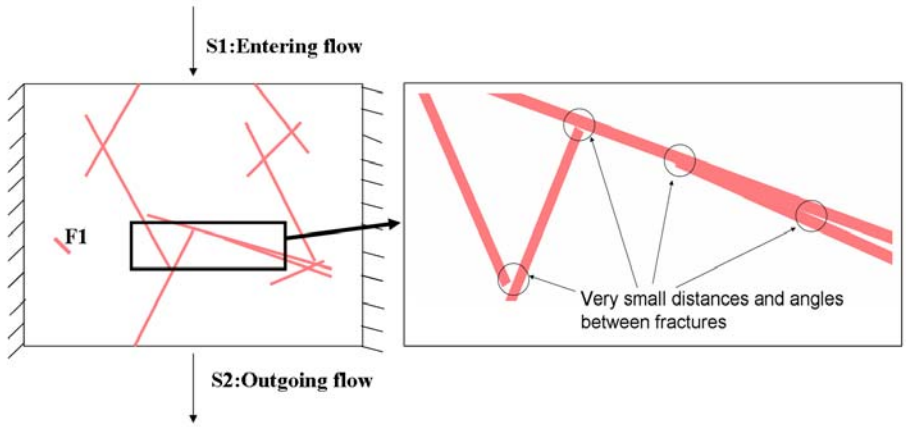


FIG. 18. *Two-dimensional fracture network. Left: Example of a distribution of fractures. Right: Zoom of the left figure to show the small distances and angles between fractures.*

media contain a small number of fractures, and then the geometrical configurations can be simple to generate the mesh. We note that, in this paper, we are interested in the highly heterogeneous fractured media. (2) The improvement in the quality of the mesh can be done by refinement, and then the size of the problem becomes very large. In this case, we will be interested in the precision of the solution. This idea is not effective because of the geological inaccuracy and the large size of the studied fracture networks.

B. . To cure the above difficulties, the projection method is developed. This method can work with the existence of a small distance or angle between fractures because the fractures are projected on a regular grid (Figure 5).

Now, to make a prediction about the mesh step that can be used to project the fractures, we perform the following steps. First, we determine all the ways that fractures cross the domain from $S1$ to $S2$ (Figure 17). We find $w1$ and $w2$. In the second step, the other fractures $F1$, $F2$, $F3$, and $F4$ and way $w3$ are identified. We note that the size of a fracture is equal to the length of the line. In our method we are interested in the treatment of the fractures that form the ways $w1$ and $w2$. The mesh step h needs to be greater than the smallest size of these fractures. This condition makes it possible to guarantee the connection of the essential ways of the flows. We note that this problem is the same in the case of resolution without projection.

C. . The mesh step h can be a function of the small fracture size, of the density of fractures, or of other characteristics of the fracture networks. When h increases, the error in precision increases for two reasons. The first is the numerical approximation of the solution. The second reason is that some of the essential ways can disappear. A way is removed from the network if h is bigger than one of the fractures that form this way, and then the quantity of outgoing flux decreases.

Because the geometry is very complex, we do not have an idea about the form of the generated structures. This is because of the random generation and stochastic properties of the fractures. The determination of the fracture networks is based on observations and experiences. Then, we have a geological inaccuracy.

In the fractured medium with a matrix of very low permeability, the fluid flow is focused in the highly heterogeneous fractures [21]. In our study, we are interested in these fractures. For each fracture network, we define the ways of flow as explained

above. For us, these ways are the base of the fluid flow. The rest of the fractures do not have an effect on the flow. Because the number of found ways depends on the mesh step h, we cannot exactly define the error in precision. This is because the quantity of outgoing flux depends on the number of effective ways. In the case of very dense fracture networks [15], the variation of the mesh step is more relevant and the quantity of capture flow remains important with respect to h.

The results evolve according to the mesh and converge toward the exact solution, but how do we determine an optimal mesh density? We note that an analytical solution does not exist, and then the convergence is shown with respect to the mesh step. The optimality of the mesh depends on the application and then can have different definitions according to the phenomenon in question. The precision of a solution calculated in a fracture network cannot compensate for the problem of geological inaccuracy of the synthetic fractured media. Now, a good quality mesh helps to have a well-approximated solution. This idea is clearly checked in several examples above. In Figures 13, 14, and 15, we show that the quality of the mesh obtained without projection is bad when h increases and the error in precision increases significantly, too (Figure 16). Then to have a better precision a refinement of the mesh must be used. But we remark that the quality of the mesh using the projection method remains good with the increase of h, and then the errors in precision are small (Figure 16). We have shown that for all mesh steps, the quality of the mesh is better when using the projection method. For each mesh step, the best compromise between the quality of the mesh, the precision of the approximated solution, the memory capacity, and the CPU time is obtained using the projection method. Here, we say that the obtained mesh using the projection method is optimal because it realizes this compromise as detailed above.

7. Conclusion. In this paper, we have proposed a simplified method of the complex geometries of fracture networks. The goal of this method is to remove the singularities in the geometry using a method of projection on a $3DRG$. For more details, see [18, 19, 20]. These singularities are represented by the small distances between the points and the small angles between the intersections of fractures. The obtained geometry is very regular and does not have penalizing configurations. The small angles are completely removed, and the obtained mesh is of good quality. Therefore, the main goal is to carry out a better compromise between the size of the problem and the quality of the mesh on one hand, and the precision of the approximate solution on the other hand. In this context, the regularization of the geometry should not induce a significant error in the approximate solution. Because the principal goal of our study is to search the laws of scaling, it is very interesting to study a very large number of fracture networks that contain varying numbers of fractures. The proposed method in this article aims to facilitate the simulation in heterogeneous fracture networks that have complex geometries.

We note that it is very difficult (section 4.2) to generate the mesh of most of these networks using an automatic generator mesh. The proposed method offers regular and similar geometries of the complex networks for which we can easily generate a good quality mesh. The validation of the method is carried out in several steps. We initially checked the precision of the approximate solution. In the first part, we showed the good quality of the mesh obtained using the projection method. We checked the precision of the solution by comparing it with a reference solution calculated on a finest mesh without projection in the second part. Afterward, we varied the step of mesh. We showed that the good quality of the mesh is preserved. This property

is a significant advantage compared to the mesh obtained without projection. We measured the error in precision made during this change of h and observed that the algorithm remains of order 1. Finally, we measured the improvement in CPU time and memory capacity and we found that results are significant and show the efficiency of the numerical approach developed. To conclude, in this paper we showed that the relative error is lower than 7% when h varies from 0.08 to 0.21, and the linear system becomes six to ten times smaller. This reduction in the linear system size induced a reduction of the needed memory and a significant decrease in CPU times. For example, with a loss of precision lower than 5%, the CPU time and the memory capacity are divided by 10.

The refinement of the mesh to improve the quality is not a good solution in networks containing large numbers of fractures. The definition of the error in precision is a problem to solve. We need to do further tests on various types of networks that contain various numbers of fractures. In forthcoming work, we will focus in detail on the three following essential tasks: (1) perform stochastic tests on various types of networks to precisely determine the error; (2) study other hydraulic phenomena: transient, transport, etc.; (3) take into account the matrix (for oil studies).

Nomenclature.

FN: Fracture network of Figure 1.
ref: Reference.
app: Approached.
NF: Number of fractures.
Ni: Number of intersections.
$Kref, Kapp$: Global equivalent permeability.
$Iiref, Iiapp$: Flow through the intersection number i.
$Firef, Fiapp$: Flow across the fracture i.
$Iref, Iapp$: $(Iiref), (Iiapp)$, $i = 1, Ni$.
$Fref, Fapp$: $(Firef), (Fiapp)$, $i = 1, NF$.
EK: $(Kref - Kapp)/Kref$.
EF: $norm1(Fref - Fapp)/norm1(Fref)$.
EI: $norm1(Iref - Iapp)/norm1(Iref)$.
h: Mesh step.
NT: Number of mesh triangles.
SL: Linear system size.
, : Time of generation mesh in seconds.
$2DRG$ and $3DRG$: Two- and three-dimensional regular grid.

REFERENCES

[1] P. Bastian, Z. Chen, R. E. Ewing, R. Helmig, H. Jakobs, and V. Reichenberger, *Numerical simulation of multiphase flow in fractured porous media*, in Numerical Treatment of Multiphase Flows in Porous Media, Lecture Notes in Phys. 552, Springer-Verlag, New York, 2000, pp. 50–68.

[2] D. Billaux, *Hydrogéologie des Milieux Fracturés. Géométrie, Connectivité et Comportement Hydraulique*, Thèse de doctorat, Université d'Orléans, Orléans, France, 1990.

[3] D. Billaux, J. P. Chiles, K. Hestir, and J. Long, *Three-dimensional statistical modelling of a fractured rock mass—an example from the Fanay-Augères mine*, Int. J. Rock Mech. Min. Sci., 26 (1989), pp. 281–299.

[4] E. Bonnet, O. Bour, N. E. Odling, P. Davy, I. Main, P. Cowie, and B. Berkowitz, *Scaling of fracture systems in geological media*, Reviews of Geophysics, 39 (2001), pp. 347–384.

[5] O. Bour and P. Davy, *One the connectivity of three dimensional fault networks*, Water Resources Research, 34 (1998), pp. 2611–2622.
[6] D. Bruel and F. H. Cornet, *Forced fluid flow through fractured reservoirs modelling*, in Fractured and Jointed Rock Masses, L. R. Myer, N. G. W. Cook, R. E. Goodman, and C. F. Tsang, eds., Balkema, Rotterdam, 1995, pp. 503–510.
[7] M. C. Cacas, E. Ledoux, G. DeMarsily, B. Tillie, A. Barbreau, E. Durand, B. Feuga, and P. Peaudecerf, *Modeling fracture flow with a stochastic discrete fracture network: Calibration and validation.* 1. *The flow model*, Water Resources Research, 26 (1990), pp. 479–489.
[8] T. A. Davis, *Algorithm* 832*: UMFPACK V*4.3*, an unsymmetric-pattern multifrontal method with a column pre-ordering strategy*, ACM Trans. Math. Software, 30 (2004), pp. 196–199.
[9] J. R. De Dreuzy, P. Davy, and O. Bour, *Percolation threshold of* 3*D random ellipses with widely-scattered distributions of eccentricity and size*, Phys. Rev. E (3), 62 (2000), pp. 5948–5952.
[10] W. S. Dershowitz and C. Fidelibus, *Derivation of equivalent pipe networks analogues for three-dimensional discrete fracture networks by the boundary element method*, Water Resources Research, 35 (1999), p. 2685.
[11] P. Dietrich, R. Helmig, M. Sauter, H. Hötzl, J. Köngeter, and G. Teutsch, *Flow and Transport in Fractured Porous Media*, XVIII, Springer-Verlag, Berlin, 2005.
[12] P. J. Frey and P. L. George, *Maillages, Applications aux Éléments Finis*, HERMES Science Publications, Paris, 1999.
[13] H. Hoteit, J. Erhel, R. Mosé, B. Philippe, and Ph. Ackerer, *Numerical reliability for mixed methods applied to flow problems in porous media*, Comput. Geosci., 6 (2002), pp. 161–194.
[14] H. Hoteit and A. Firoozabadi, *Multicomponent fluid flow by discontinuous Galerkin and mixed methods in unfractured and fractured media*, Water Resources Research, 41 (2005), pp. 1–15.
[15] S. Kirkpatrick, *Percolation and conduction*, Rev. Modern Phys., 45 (1973), pp. 574–588.
[16] V. Lenti and C. Fidelibus, *A BEM solution of steady-state flow problems in discrete fracture networks with minimization of core storage*$*$1, Computers and Geosciences, 29 (2003), pp. 1183–1190.
[17] J. Maryška, O. Severýn, and M. Vohralík, *Numerical simulation of fracture flow with a mixed-hybrid FEM stochastic discrete fracture network model*, Comput. Geosci., 8 (2004), pp. 217–234.
[18] H. Mustapha, *Simulation Numérique de L'écoulement dans des Milieux Fracturés Tridimensionnels*, Thèse de Doctorat, Université de Rennes 1, Rennes, France, 2005.
[19] H. Mustapha, *An adaptive method for flow simulation in three-dimensional heterogeneous discrete fracture networks*, in the International Conference on Scientific Computing, Las Vegas, NV, 2006.
[20] H. Mustapha, *Monte Carlo simulation of flow in three-dimensional discrete fracture networks*, in the 2006 World Congress in Computer Science, Computer Engineering, and Applied Computing, Las Vegas, NV, 2006.
[21] I. Neretnieks, *Transport in fractured rocks*, Proceedings of Hydrogeology of Rock of Low Permeability, Mem. Intern. Assoc. Hydrogeol., 17 (1985), pp. 301–318.
[22] A. W. Nordqvist, Y. W. Tsang, C. F. Tsang, B. Dverstorp, and J. Andersson, *A variable aperture fracture network model for flow and transport in fractured rocks*, Water Resources Research, 28 (1992), pp. 1703–1713.
[23] P. A. Raviart and J. M. Thomas, *A mixed hybrid finite element method for* 2*nd order elliptic problems*, in Mathematical Aspects of Finite Element Methods, Lecture Notes in Math. 606, Springer-Verlag, New York, 1977, pp. 292–315.
[24] V. Reichenberger, H. Jakobs, P. Bastian, and R. Helmig, *A mixed-dimensional finite volume method for two-phase flow in fractured porous media*, Advances in Water Resources, 29 (2006), pp. 1020–1036.
[25] A. Rouleau and J. E. Gale, *Stochastic discrete fracture simulation of groundwater flow into an underground excavation in granite*, Int. J. Rock Mech. Min. Sci., 24 (1987), pp. 99–112.
[26] J. Thomas, *Sur l'Analyse Numérique des Méthodes d'Elément Finis Hybrides et Mixtes*, Thèse de Doctorat d'Etat, Université de Pierre et Marie Curie, Paris, France, 1977.
[27] M. Vohralik, *Méthodes Numériques pour des Équations Elliptiques et Paraboliques non Linéaires: Application à des Problèmes D'écoulement en Milieux Poreux et Fracturés*, Thèse de Doctorat, Université Paris XI Orsay, Orsay, France, 2004.

SIAM J. SCI. COMPUT.
Vol. 29, No. 4, pp. 1460–1475
© 2007 Society for Industrial and Applied Mathematics

ON GLOBAL ERROR ESTIMATION AND CONTROL FOR INITIAL VALUE PROBLEMS*

J. LANG† AND J. G. VERWER‡

Abstract. This paper addresses global error estimation and control for initial value problems for ordinary differential equations. The focus lies on a comparison between a novel approach based on the adjoint method combined with a small sample statistical initialization and the classical approach based on the first variational equation. Control is achieved through tolerance proportionality. Both approaches are found to work well and to enable estimation and control in a reliable manner.

Key words. numerical integration for ODEs, global error estimation, global error control, defects and local errors, tolerance proportionality, adjoint method, small sample statistical initialization

AMS subject classifications. 65L05, 65L06, 65L20, 65L70, 65M20

DOI. 10.1137/050646950

1. Introduction.

Suppose one is given in $\mathbb{R}^m$ the initial value problem for the system of ODEs

$$w' = F(t, w)\,, \quad w(0) = w_0\,, \qquad 0 < t \le T\,, \tag{1.1}$$

and a sequence of approximations w_n to its exact solution values $w(t_n)$ computed by a numerical integration method at a certain time grid

$$0 = t_0 < t_1 < \cdots < t_n < \cdots < t_{N-1} < t_N = T\,. \tag{1.2}$$

Hereby the major research concerns are efficiency—how to get the w_n at minimal CPU costs (if $m \gg 1$)—and reliability—how large are the global errors

$$\varepsilon_n = w(t_n) - w_n\,, \qquad n = 0\,, \ldots\,, N\,. \tag{1.3}$$

In the past, numerical ODE research has focused on the efficiency question. The reliability question has received much less attention in spite of the by now twenty years old survey paper [23]. Existing popular codes focus on efficiency by adaptively optimizing time grids (1.2) in accordance with local error control. Such a control makes sense if solutions exhibit sharp changes at local intervals much smaller than the total interval $[0, T]$ and are smooth elsewhere. However, local errors (errors made within a single integration step) may substantially differ from the global ones (1.3). This largely depends on the conditioning (stability) of the system (1.1) at hand (sensitivity to growth in time of perturbations of w_0 and $F(t, w)$). If a system is well-conditioned, a well-designed local error control [20, 21] will work out reliably. But if the conditioning is bad, even the best-designed local error control should not be trusted.

For global error control it is necessary to take into account the conditioning of system (1.1), similar to the matrix condition number in numerical linear algebra.

*Received by the editors December 5, 2005; accepted for publication (in revised form) February 13, 2007; published electronically June 19, 2007.
http://www.siam.org/journals/sisc/29-4/64695.html
†Darmstadt University of Technology, Schlossgartenstrasze 7, 64289 Darmstadt, Germany (lang@mathematik.tu-darmstadt.de).
‡Center for Mathematics and Computer Science, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands (Jan.Verwer@cwi.nl).

Herewith it is desirable to avoid strict a priori error bounds as these can be overly pessimistic, e.g., when fortunate cancellation effects occur. Taking into account the conditioning of system (1.1) is best done during actual computation. This requires at least two full integrations over $[0,T]$, in both the classical (forward-forward) and the adjoint (forward-backward) approaches. In recent years the adjoint approach has gained popularity for a posteriori error estimation purposes [2, 6, 13, 17], in particular for PDEs where it is used for goal-oriented adaptive control, that is, control of relevant physical quantities [1, 5, 7].

Our interest in this paper lies in estimation and control of the global errors (1.3), i.e., for the ODE initial value problem (1.1). Specifically, regarding estimation we will compare a novel approach based on the adjoint method combined with a small sample statistical initialization proposed by Cao and Petzold [2] to the classical approach based on the first variational equation. For both, global control is achieved by exploiting the property of tolerance proportionality derived from local control [20]. As a typical integration method we will use the existing Runge–Kutta–Rosenbrock method ROS3P [16]. Both approaches are found to work well and to enable estimation and control in a reliable manner.

2. The classical (forward-forward) approach.

2.1. The perturbed system. In the spirit of backward error analysis and following an approach proposed in [26] (see also [23, 24]), let us suppose that there exists a nearby solution $v(t) \approx w(t)$, that is the exact solution of a perturbed system

$$v' = F(t,v) + r(t)\,, \quad v(0) = v_0\,, \qquad 0 < t \le T\,, \tag{2.1}$$

with perturbation $v_0 - w_0$ at $t = 0$ and perturbation $r(t)$ for $0 < t \le T$. Assuming F to be continuously differentiable, so that the mean-value theorem for vector functions is applicable, the error function

$$e(t) = v(t) - w(t)\,, \qquad 0 \le t \le T\,, \tag{2.2}$$

then satisfies

$$e' = A(t)e + r(t)\,, \quad e(0) = v_0 - w_0\,, \qquad 0 < t \le T\,, \tag{2.3}$$

where

$$A(t) = \int_0^1 F'\big(t, v(t) + (s-1)e(t)\big)ds = \int_0^1 F'\big(t, w(t) + se(t)\big)ds\,. \tag{2.4}$$

Here F' denotes the Jacobian matrix with respect to the dependent variable. As we speak of perturbations we tacitly assume that $e(0)$ is significantly smaller than w_0 and likewise that $r(t)$ is significantly smaller than $F(t, v(t))$. The error function $e(t)$ can be expressed as

$$e(t) = \Phi^{-1}(t)\Phi(0)\, e(0) + \Phi^{-1}(t) \int_0^t \Phi(s) r(s)\, ds\,, \tag{2.5}$$

where the $m \times m$ fundamental matrix solution $\Phi(t)$ solves $\Phi'(t) = -\Phi(t)A(t)$. Note that $\Phi(t) = \Phi(0)\exp(-At)$ for constant A, where $\Phi(0)$ is an arbitrary nonsingular matrix. Apparently, the product $\Phi^{-1}(t)\Phi(s), 0 \le s \le t$, governs growth or decay of $e(t)$ in time and thus determines the conditioning of (1.1).

2.1. For practical purposes the fundamental matrix solution concept is clearly of little use. An a priori bound which sometimes can be used in practice is based on the logarithmic matrix norm. Let $\mu(A(t))$ denote the logarithmic matrix norm of $A(t)$, and suppose $\mu(A(t)) \leq \omega$, with ω a constant valid for all $t \in [0,T]$. Then, with $\|\cdot\|$ denoting the associated vector norm in $\mathbb{R}^m$, there holds [3, 12]

$$\|e(t)\| \leq e^{\omega t}\|e(0)\| + \int_0^t e^{\omega(t-s)}\|r(s)\|ds\,, \qquad 0 \leq t \leq T\,. \tag{2.6}$$

This a priori bound is useful for strictly negative ω. Let for simplicity $e(0) = 0$, and suppose that $\|r(t)\|$ is uniformly bounded by a small number δ_r. Then

$$\|e(t)\| \leq \frac{1}{\omega}\left(e^{\omega t} - 1\right)\delta_r\,, \tag{2.7}$$

uniformly bounding $e(t)$ in norm by $\delta_r/|\omega|$ for infinite time if $\omega < 0$. Due to the negative logarithmic norm this is an instance of a well-conditioned ODE system (1.1). On the other hand, with a positive logarithmic norm the bound predicts exponential growth in time. Then we speak of an ill-conditioned ODE system, provided the bound is realistic or even sharp. However, any result of this sort depends on the chosen vector norm since $\mu(A(t))$ depends on the norm. More precisely, it can happen that $\mu(A(t)) \gg 0$, even if in a global sense the problem is well-conditioned; see, e.g., [3] and Example I.2.5 in [12]. Also, for hard problems from practice, finding estimates of ω may be very cumbersome.

For properly using the perturbed system in the classical (forward-forward) approach for pth order consistent one-step integration methods, it is helpful to associate (2.3) with the well-known first variational equation for the global error ε_n of such a one-step method. For this purpose we adopt for the one-step method the general Henrici notation [9]

$$w_{n+1} = w_n + \tau_n\Psi(t_n, w_n; \tau_n)\,, \qquad \tau_n = t_{n+1} - t_n\,, \tag{2.8}$$

and introduce its local error

$$\delta_n = w(t_{n+1}) - w(t_n) - \tau_n\Psi(t_n, w(t_n); \tau_n)\,. \tag{2.9}$$

Assuming now a constant step size τ (for simplicity only), pth order consistency, and a sufficiently smooth solution, the local error defined at $(t, w(t))$

$$\delta(t) = w(t+\tau) - w(t) - \tau\Psi(t, w(t); \tau) \tag{2.10}$$

possesses an expansion of the form $\delta(t) = \tau\rho(t) + \mathcal{O}(\tau^{p+2})$. Hence in this notation the principal error function $\rho(t) = \mathcal{O}(\tau^p)$. Next, let $G(t) = F'(t, w(t))$. With this local expansion at hand the global error ε_n is then known to possess an expansion $\varepsilon_n = \eta(t_n) + \mathcal{O}(\tau^{p+1})$, where $\eta(t)$ is the solution of the first variational equation

$$\eta' = G(t)\eta + \rho(t)\,, \quad \eta(0) = 0\,, \qquad 0 < t \leq T\,. \tag{2.11}$$

We refer to [9, section 3.3] and [8, section II.8] for further details. One can now connect (2.3) to (2.11). To that end put $e(0) = 0$ (this bears no restriction) and approximate $A(t) = G(t) + \mathcal{O}(e(t))$ by $G(t)$, so that (2.3) is replaced by

$$e' = G(t)e + r(t)\,, \quad e(0) = 0\,, \qquad 0 < t \leq T\,. \tag{2.12}$$

Apparently, by implementing a proper choice of the defect $r(t)$, solving (2.3) or likewise (2.12) will in leading order amount to solving the first variational equation (2.11) for the true global error. We will illustrate this next.

Remark 2.2. In spite of the fact that the asymptotic theory behind (2.11) is classical in the sense that F is assumed Lipschitz without taking into account stiffness ($\tau\|F'(w)\| \gg 1$), we will use it successfully for stiff problems in section 4. For success it is important that the stiffness does not enter the defect computation. We prevent this by filtering; see (2.23). Equally important is validity of the asymptotics, that is, the local error is well approximated by its leading term. For smooth, sufficiently differentiable solutions this always holds for linear multistep methods. For Runge–Kutta-type methods applied to semidiscrete PDEs, this generally holds with periodic boundary conditions. With prescribed boundary conditions, such as Dirichlet conditions, it also holds for Runge–Kutta methods, except that temporal error coefficients can get large on fine grids due to order reduction. In such cases smaller step sizes are needed to catch the leading local error term with sufficient accuracy. As is well known, this may also be needed with stiff problems not originating from PDEs.

2.2. Interpolation and defect computation. We define the nearby solution $v(t)$ by piecewise cubic Hermite interpolation of the given approximation sequence. Hence at every subinterval $[t_n, t_{n+1}]$, $n = 0, 1, \ldots, N-1$, we form

$$P(t) = w_n + (t-t_n)\,\mathcal{A}_n + (t-t_n)^2\,\mathcal{B}_n + (t-t_n)^3\,\mathcal{C}_n\,, \qquad t_n \le t \le t_{n+1}\,, \tag{2.13}$$

and choose the coefficients such that $P(t_n) = w_n, P(t_{n+1}) = w_{n+1}$ and $P'(t_n) = F_n$, $P'(t_{n+1}) = F_{n+1}$, where $F_n = F(t_n, w_n)$ and $F_{n+1} = F(t_{n+1}, w_{n+1})$. This gives

$$\begin{aligned} \mathcal{A}_n &= F_n\,,\\ \mathcal{B}_n &= \big(3w_{n+1} - 3w_n - \tau F_{n+1} - 2\tau F_n\big)/\tau^2\,,\\ \mathcal{C}_n &= \big(2w_n - 2w_{n+1} + \tau F_n + \tau F_{n+1}\big)/\tau^3\,. \end{aligned} \tag{2.14}$$

Were one to interpolate (smooth) exact solution values $w(t), w'(t)$ at $t = t_n$ and $t = t_{n+1}$, then

$$\mathcal{A} = w^{(1)}\,, \quad \mathcal{B} = \tfrac{1}{2}w^{(2)} - \tfrac{1}{24}\tau^2 w^{(4)} + \mathcal{O}(\tau^3)\,, \quad \mathcal{C} = \tfrac{1}{6}w^{(3)} + \tfrac{1}{12}\tau w^{(4)} + \mathcal{O}(\tau^2)\,, \tag{2.15}$$

where the expressions are evaluated at $t = t_n$. Writing $t = t_n + s\tau$, $0 \le s \le 1$, yields

$$\begin{aligned} P(t) &= w(t) + \tfrac{1}{24}\big(2s^3 - s^2 - s^4\big)\tau^4 w^{(4)}(t_n) + \mathcal{O}(\tau^5)\,,\\ P'(t) &= w'(t) + \tfrac{1}{24}\big(6s^2 - 2s - 4s^3\big)\tau^3 w^{(4)}(t_n) + \mathcal{O}(\tau^4)\,, \end{aligned} \tag{2.16}$$

so that at every subinterval $[t_n, t_{n+1}]$ the defect function $d(t) = P'(t) - F(t, P(t))$ satisfies

$$d(t) = P'(t) - w'(t) + F\big(t, w(t)\big) - F\big(t, P(t)\big) = \begin{cases} \mathcal{O}(\tau^3)\,, & s \ne \frac{1}{2}\,,\\ \mathcal{O}(\tau^4)\,, & s = \frac{1}{2}\,. \end{cases} \tag{2.17}$$

Here we have assumed that F is Lipschitz and have used that $6s^2 - 2s - 4s^3 = 0$ for $s = \frac{1}{2}$.

In actual application the interpolation is based on numerical approximations of order p. By assuming exact local solution values at $t = t_n$ and corresponding local Taylor expansions of w_{n+1}, F_{n+1}, it then follows in the same way as above that $d(t) =$

$\mathcal{O}(\tau^q)$, where $q = \min(p, 3)$, and with the special value $s = \frac{1}{2}$ we have $q = \min(p, 4)$. Hence the cubic Hermite polynomial can be used up to consistency order $p = 3$, and when using only $s = \frac{1}{2}$ even up to order $p = 4$. In the remainder we will employ the defect $d(t)$ halfway to the step intervals, that is,

(2.18)
$$d(t_{n+1/2}) = \frac{3w_{n+1} - 3w_n}{2\tau} - \frac{F_n + F_{n+1}}{4} - F\Big(t_{n+1/2}, \frac{w_n + w_{n+1}}{2} + \frac{\tau}{8}\left(F_n - F_{n+1}\right)\Big)\,.$$

Now let $w_n = w(t_n)$, and consider local expansions

$$w_{n+1} = w + \tau w' + \sum_{k=2}^{p} \frac{\tau^k}{k!} w^{(k)} + \frac{\tau^{p+1}}{(p+1)!} C_{p+1} + \mathcal{O}(\tau^{p+2})\,, \tag{2.19}$$

with the right-hand side evaluated at $w = w(t_n)$ and an empty sum for $p = 1$. Hence the ρ-term in the local error expansion at $t = t_n$ then reads

$$\rho_n = \frac{\tau^p}{(p+1)!}\Big(w^{(p+1)}(t_n) - C_{p+1}\Big)\,, \tag{2.20}$$

and inserting the expansion for w_{n+1} into $d(t_{n+1/2})$ will reveal

$$d(t_{n+1/2}) = -\tfrac{3}{2}\,\rho_n + \mathcal{O}(\tau^{p+1})\,, \qquad 1 \le p \le 3\,. \tag{2.21}$$

The cubic Hermite defect halfway to the step interval thus can be used to retrieve in leading order the local error of any one-step method of order $1 \le p \le 3$.

Finally we connect (2.12) and (2.11) by putting $r(t_{n+1/2}) = \frac{2}{3}\,d(t_{n+1/2})$ in the stepwise frozen version of (2.12), i.e.,

$$e' = F'(t_n, w_n)e + r(t_{n+1/2})\,, \quad t_n < t \le t_{n+1}\,, \qquad n = 0, \ldots, N-1\,, \tag{2.22}$$

which will be integrated for the global error estimation. In this manner we actually work in leading order with the stepwise frozen version of the first variational equation (2.11) for the true global error (both $G(t)$ and $\rho(t)$ are frozen at $t = t_n$) multiplied by -1, i.e., $e = -\eta$ in first-order approximation. Within the more general setting of continuous Runge–Kutta methods, this use of defects and relations like (2.21) was discussed earlier; see, e.g., [4, 10, 11, 21, 22]. Trivially, multiplying $r(t_{n+1/2})$ by a certain constant multiplies the solution by the same constant (if $e(0) = 0$). This simple property forms the basis for tolerance proportionality, which we shall use for attempting control over the global errors (in both the classical and the novel approaches).

2.3. The example integration formulas. As the example integrator for the comparison between the adjoint and the classical approaches, we have used an A-stable scheme, viz., the third-order Runge–Kutta–Rosenbrock scheme ROS3P [16]. To save space we refer to [15, 16] for details.

The implemented step size strategy by which the time grid (1.2) is generated with ROS3P is standard, except that, different from [16], the defect $r(t_{n+1/2})$ is used. So for local control we work with

$$Est = (I - \gamma\tau_n A)^{-1} r(t_{n+1/2})\,, \qquad A = F'(t_n, w_n)\,, \tag{2.23}$$

where γ is a ROS3P coefficient. The common filter $(I - \gamma\tau_n A)^{-1}$ serves to damp spurious stiff components which would otherwise be amplified through the F-evaluations

within $r(t_{n+1/2})$.[1] Note that, while the local error is $\mathcal{O}(\tau_n^4)$, this estimate is $\mathcal{O}(\tau_n^3)$ by which we asymptotically have tolerance proportionality [20].

Let $D_n = \|Est\|$, with $\|\cdot\|$ the L_2-norm.[2] The step is accepted if $D_n \le Tol_n$, where $Tol_n = Tol_A + Tol_R\,\|w_n\|$, with Tol_A and Tol_R given tolerances. Otherwise the step is rejected and redone. In both cases the new step size is determined by the standard rule $\tau_{new} = \min(1.5, \max(2/3, 0.9\,r))\tau_n$, where $r = (Tol_n/D_n)^{1/3}$. After each step size change we adjust τ_{new} to $\tau_{n+1} = (T - t_n)/\lfloor(1 + (T - t_n)/\tau_{new})\rfloor$ so as to guarantee to reach the end point T with a step of averaged normal length. The initial step size τ_0 is prescribed and is adjusted similarly.

Simultaneously, (2.22) is integrated by means of the implicit midpoint rule

$$e_{n+1} = e_n + \tau_n A\Big(\frac{e_{n+1} + e_n}{2}\Big) + \tau_n r(t_{n+1/2})\,, \qquad A = F'(t_n, w_n)\,, \tag{2.24}$$

implemented in the equivalent form

$$\begin{aligned} \tilde{e}_{n+1} &= 2e_n + \tfrac{1}{2}\tau_n A\tilde{e}_{n+1} + \tau_n r(t_{n+1/2})\,,\\ e_{n+1} &= \tilde{e}_{n+1} - e_n\,. \end{aligned} \tag{2.25}$$

The main additional costs for (2.25) come from an extra decomposition since $\gamma \neq \frac{1}{2}$ (assuming a direct solve). Due to freezing $G(t)$ and $\rho(t)$ at $t = t_n$ as discussed above, the second-order midpoint rule (2.24) is a first-order method when interpreted for solving the first variational equation (2.11). As a result, the associated local error takes the form $C(t_n)\tau_n^2 + \text{h.o.t.}$, where the leading error constant $C(t_n) = \mathcal{O}(\tau_{max}^3)$ as it is proportional to $\eta(t)$, which itself is $\mathcal{O}(\tau_{max}^3)$. Consequently, the global error approximations e_n satisfy $e_n = -\epsilon_n + \mathcal{O}(\tau_{max}^4)$.

2.4. The control rule. Suppose ROS3P and (2.25) have delivered a numerical solution w_N and a global error estimate e_N at time $t_N = T$. We then verify whether

$$\|e_N\| \le C_{control}\,Tol_N\,, \qquad Tol_N = Tol_A + Tol_R\,\|w_N\|\,, \tag{2.26}$$

where $C_{control} \approx 1$, typically > 1. If (2.26) holds, the true global error is considered small enough relative to the chosen tolerance, and w_N is accepted. Otherwise, the computation with ROS3P and (2.25) is redone over $[0, T]$ with the same (small) τ_0 and the adjusted tolerances

$$Tol_A = Tol_A \times fac\,, \quad Tol_R = Tol_R \times fac\,, \qquad fac = Tol_N/\|e_N\|\,. \tag{2.27}$$

The primary aim of global error estimation is to provide an additional check on accuracy. Especially when the problem at hand is ill-conditioned (unstable), this is useful since local control will not detect instability so that ε_N might be substantially larger than the imposed tolerance. The second (control) computation with ROS3P (and (2.25) for an additional check) serves to reduce ε_N to the imposed tolerance level. If all is going well, with (2.26) we thus count on the quality of the global error estimation and with (2.27) on tolerance proportionality, thus expecting that reducing the local error estimates with the factor fac will reduce ε_N by fac [20].

[1] The defect contains a new F-evaluation, which in turn contains $F(t_n, w_n)$ and $F(t_{n+1}, w_{n+1})$, indicating that two filter steps would be needed. Our practical experience is that one filter step is sufficient, although we know of at least one hypothetical problem (the Kaps problem given at page 215 of [3]) for which two filter steps are appropriate.

[2] By L_2-norm we mean the weighted inner product norm, i.e., $\|v\|^2 = v^T v/m$, $v \in \mathbb{R}^m$. This norm will also be used for the adjoint approach.

2.3. It is possible to avoid (2.27) by storing the whole step size sequence (1.2) from the first run over $[0,T]$ and to carry out the second computation on a new step size sequence obtained by dividing all intervals $[t_n, t_{n+1}]$ into $\lceil 1/fac^{1/3} \rceil$ equal subintervals. We then count only on the quality of (2.26) and even would have the possibility to also use global Richardson extrapolation for global error estimation for an additional check. However, we then also give up local control. This renders no problem if all is going well, but it might result in numerical instability which otherwise would have been detected by local control.

3. The adjoint (forward-backward) approach. Like for the classical approach the analysis of the adjoint approach starts from the perturbed system derived in section 2.1.

3.1. Error representation for scalar-derived functions. The error representation formula (2.5) reveals that an approximation of $\Phi^{-1}(t)\,\Phi(s), \quad 0 \le s \le t$, would be desirable to estimate the sensitivity of (1.1) with respect to perturbations $v_0 - w_0$ and $r(t)$. Instead of computing these matrix products, which is in general far too expensive or even unattainable, we first derive error estimates for a scalar-derived function. The analysis is based on the adjoint method, which has been used successfully to obtain goal-oriented a posteriori error estimation for an adaptive error control for PDEs (see [1, 5, 7]) and ODEs (see [2, 6, 13, 17]).

Let $M(w(t))$ be the scalar quantity of interest. Then one has for the error in M

$$\triangle M(t) := M\big(v(t)\big) - M\big(w(t)\big) = \overline{M}(t)\, e(t), \tag{3.1}$$

with row vector

$$\overline{M}(t) = \int_0^1 M'\big(v(t) + (s-1)e(t)\big)\, ds = \int_0^1 M'\big(w(t) + se(t)\big)\, ds\,. \tag{3.2}$$

Hence, using (2.5),

$$\triangle M(t) = \overline{M}(t)\Phi^{-1}(t)\Phi(0)e(0) + \overline{M}(t)\Phi^{-1}(t) \int_0^t \Phi(s)r(s)\, ds\,. \tag{3.3}$$

Solving backward in time the adjoint equation

$$\phi'(s) = -A^T(s)\phi(s), \quad \phi(t) = \overline{M}^T(t), \quad 0 \le s < t, \tag{3.4}$$

and taking into account that $\Phi^{-T}(s)$ is the fundamental matrix of this equation if (and only if) $\Phi(t)$ is the fundamental matrix of (2.3) defined in (2.5), one gets

$$\phi^T(s) = \overline{M}(t)\Phi^{-1}(t)\Phi(s)\,. \tag{3.5}$$

Thus

$$\triangle M(t) = \phi^T(0)e(0) + \int_0^t \phi^T(s)r(s)\, ds, \quad 0 \le t \le T. \tag{3.6}$$

Formula (3.6) serves as the fundamental relation to derive a strategy for global error estimation. The adjoint solution value $\phi^T(0)$ measures sensitivity of $M(w(t))$ with respect to $e(0)$. Likewise, the integral term measures sensitivity with respect to the defects.

. . . . 3.1. In principle one may consider quantities $M(w(t)) = \xi_i^T w(t)$, $i = 1, \ldots, m$, where ξ_i is the ith unit vector in $\mathbb{R}^m$. Then $\triangle M(t) = \xi_i^T e(t) = e^{(i)}(t)$, the ith component of the error vector $e(t)$. Thus $\overline{M}^T(t) = \xi_i$. Denoting by ψ_i the solution of (3.4) with ξ_i as the initial value, one gets from (3.6)

$$\xi_i^T e(t) = e^{(i)}(t) = \psi_i^T(0)e(0) + \int_0^t \psi_i^T(s)r(s)\,ds\,. \tag{3.7}$$

In this way all components of the error vector $e(t)$ could be computed at the price of solving the adjoint equation m times. However, if $m \gg 1$, one would have to solve a tremendous number of adjoint systems, making this method impractical. The best choice would be $M(w(t)) = e^T(t)w(t)/(m\|e(t)\|)$ [2]. Then $\triangle M(t) = \|e(t)\|$ directly from (3.1), but one does not have $e(t)$. The choice of appropriate initial conditions for the adjoint system is the main challenge for the adjoint approach [2, 7].

. . . . 3.2. To set up the adjoint equation (3.4) we have to replace $A^T(s)$ by a suitable approximation in the neighborhood of $v(s)$. Thus the adjoint equation depends on the solution of the original ODE. A first possibility is to store the forward solution for every time step to determine the adjoint equation. Alternatively, the solution is stored at only a few selected times $0 = T_0 < T_1 < \cdots < T_K = T$. As the adjoint equation solver marches backwards in time from T_k to T_{k-1}, one recomputes the solution over that time interval using the previously stored solution T_{k-1} as the initial value. This approach reduces the storage requirements at the price of a second forward solution. The need to make the forward equation available to the adjoint equation is a drawback of the adjoint approach. Solution storage is of course not needed for linear systems (1.1) of type $w' = A(t)w + g(t)$. On the other hand, in all cases all defects must be stored. Another drawback of the adjoint approach is that it is defined for single output times t at which estimation is wanted. In other words, if estimation is wanted at multiple times t, the adjoint solution must be computed separately for each value of t.

3.2. Global error estimation. We have implemented global error estimation applying ROS3P to solve (1.1) as described in section 2.3 and using (3.6) and the small sample statistical method proposed by Cao and Petzold for backward differentiation formua (BDF) methods [2] (see also [14] for more details). The main idea is to replace the vectors ξ_i in (3.7) by a small number of orthonormal vectors $z_1, z_2, \ldots, z_k$, which are selected uniformly and randomly from the unit sphere S_{m-1} in m dimensions. Instead of computing accurate error estimates of global errors, we try to approximate them with a high probability.

Let $\eta_i(t) = |z_i^T e(t)|$. Then an estimate for $\|e(t)\|$ is given by

$$\|e(t)\| \approx g_k(t) = \frac{E_k}{E_m}\sqrt{\frac{1}{m}\Big(\eta_1^2(t) + \eta_2^2(t) + \cdots + \eta_k^2(t)\Big)}\,, \tag{3.8}$$

where $E_1 = 1$, $E_2 = 2/\pi$, and for $n > 2$

$$E_n = \begin{cases} \dfrac{1\cdot 3\cdot 5\cdots(n-2)}{2\cdot 4\cdot 6\cdots(n-1)} & \text{for } n \text{ odd,} \\[2ex] \dfrac{2}{\pi}\cdot\dfrac{2\cdot 4\cdot 6\cdots(n-2)}{1\cdot 3\cdot 5\cdots(n-1)} & \text{for } n \text{ even.} \end{cases} \tag{3.9}$$

E_n can be estimated by $\sqrt{2/(\pi(n-1/2))}$. The "$\approx$" in (3.8) has to be understood in the sense of probability. More precisely, the expected value of the random variable

$g_k(t)$ is given by $E(g_k(t)) = \|e(t)\|$ (see [14], Theorem 3.1). For $k = m$ the vectors $z_1, \ldots, z_k$ form an orthonormal basis in $\mathbb{R}^m$, and hence from the definition in (3.8) we get for $k = m$ directly the identity $g_m(t) = \|e(t)\|$.

The important question now is, what is the probability that the estimator $g_k(t)$ provides upper and lower bounds for $\|e(t)\|$? Let $c > 1$ be a given factor. Then one has for two and three random vectors [14]

$$P\left(\frac{g_2(t)}{c} \le \|e(t)\| \le c\, g_2(t)\right) \approx 1 - \frac{\pi}{4c^2}\,, \tag{3.10}$$

$$P\left(\frac{g_3(t)}{c} \le \|e(t)\| \le c\, g_3(t)\right) \approx 1 - \frac{32}{3\pi^2\, c^3}\,. \tag{3.11}$$

To achieve 99% probability of accuracy, for example, one can use $k = 2$ for $c = 10$ and $k = 3$ for $c = 5$. In [2] it is pointed out that in practice usually at most two or three random vectors are sufficient, although without giving numerical evidence because in their numerical experiments $k = m$. Hence the small sample statistical method was not tested in [2].

3.3. To generate vectors uniformly and randomly on S_{m-1} one can use the following procedure: Let $\lambda^{(1)}, \ldots, \lambda^{(m)}$ be normally distributed independent random variables with mean zero and variance one. Then the vector $\lambda/\|\lambda\|$ is uniformly and randomly distributed over S_{m-1} where $\lambda = (\lambda^{(1)}, \ldots, \lambda^{(m)})^T$ [25].

The small sample statistical method described above thus will be used to estimate $\|e(T)\|$. We first select random vectors $\hat{z}_1, \ldots, \hat{z}_k$ from S_{m-1} and construct an orthonormal basis $z_1, \ldots, z_k$ for their span by using a Gram–Schmidt procedure or a QR decomposition. Then we solve the corresponding adjoint equations

$$\phi_i'(s) = -A^T(s)\,\phi_i(s)\,, \quad \phi_i(T) = z_i\,, \tag{3.12}$$

for $\phi_i, i = 1, \ldots, k$. From (3.1) and (3.6) with $\triangle M(T) = z_i^T e(T)$ we have for each ϕ_i the identity

$$z_i^T\, e(T) = \phi_i^T(0) e(0) + \int_0^T \phi_i^T(s) r(s)\, ds. \tag{3.13}$$

In the following we assume $e(0) = 0$. Using (3.8) and recalling $\eta_i(T) = |z_i^T\, e(T)|$ we get

$$g_k(T) = \frac{E_k}{E_m}\left(\frac{1}{m}\sum_{i=1}^{k}\left(\int_0^T \phi_i^T(s) r(s)\, ds\right)^2\right)^{1/2}. \tag{3.14}$$

The integral terms must be approximated (recall that with $k = m$ and an exact computation of these integrals we have $\|e(T)\| = g_m(T)$). We integrate (3.12) using the second-order implicit midpoint rule on the same grid as selected to solve (1.1) by means of ROS3P, but now backward in time starting from z_i,

$$\begin{aligned} &\tilde{\phi}_{i,n} = 2\,\phi_{i,n+1} + \frac{\tau_n}{2}\, A^T\, \tilde{\phi}_{i,n}\,, \quad A^T = \left(F'\left(t_{n+1/2}, \frac{w_n + w_{n+1}}{2}\right)\right)^T, \\ &\phi_{i,n} = \tilde{\phi}_{i,n} - \phi_{i,n+1}\,, \quad n = N-1, \ldots, 0. \end{aligned} \tag{3.15}$$

Note that for larger problems one cannot store all Jacobians A from the forward integration, so one has to recompute them. The practical need for recomputing Jacobians

is another drawback of the adjoint approach. Further note that, like in the forward approach, A^T is an accurate approximation to the integrated Jacobian $A(t)$ defined in (2.4) as long as the global error $e(t)$ is sufficiently small. The adjoint problems are not coupled and hence can be solved in parallel. Also recomputing Jacobians for use in the backward in time midpoint rule is needed only once, that is, once for all adjoint problems.

To approximate the integrals in (3.14) we use the 1-point, second-order Gaussian formula for each integration interval to obtain

$$g_k(T) \approx \frac{E_k}{E_m} \left(\frac{1}{m} \sum_{i=1}^{k} \left(\sum_{n=0}^{N-1} \tau_n \, \frac{\phi_{i,n}^T + \phi_{i,n+1}^T}{2} \, r(t_{n+1/2}) \right)^2 \right)^{1/2}, \tag{3.16}$$

where the residual $r(t_{n+1/2})$ is computed in exactly the same way as within the classical approach described in sections 2.2 and 2.3; i.e., we take $r(t_{n+1/2}) = \frac{2}{3} d(t_{n+1/2})$.

3.3. The control rule. The possible need for a second forward computation with ROS3P is decided on the same control rule as in the classical approach, which we described in section 2.4. Hence also for the adjoint approach we rely on tolerance proportionality for the global error control. If a second forward computation is decided, then for an additional error check we apply (3.15) once again for the already chosen random vectors z_i taken as initial values, similar to the additional error check in the classical approach.

. . . . 3.4. The conditioning of system (1.1) with respect to small perturbations $r(t)$ can be estimated using (3.14). There holds

$$\|e(T)\| \approx |g_k(T)| \le K_T \cdot \max_{0 \le s \le T} \|r(s)\|, \tag{3.17}$$

with the condition number

$$K_T = K_T(\phi_1, \ldots, \phi_k) = \frac{E_k \sqrt{m}}{E_m} \left(\sum_{i=1}^{k} \left(\int_0^T \|\phi_i(s)\| \, ds \right)^2 \right)^{1/2}. \tag{3.18}$$

If K_T is small, a well-designed defect-based local error control will work out well. But if K_T is large, one could end up with a global error much larger than the imposed local tolerance. As proposed in [5] for parabolic problems, and discussed in [2] for BDF methods, one could tighten the local tolerance for the second ROS3P run within (2.27) through $fac = 1/K_T$. However, when taking norms in (3.17), favorable effects of error cancellation and nearly zero defect values are completely ignored, and therefore the new local tolerance can be extremely pessimistic. Indeed, this is observed for the example system of section 4.4. Its condition number computed from (3.18) with $k = 2$, while using the 1-point, second-order Gaussian formula, is huge, being about 10^{20}. On the other hand, the integral terms in (3.14) are of moderate size due to cancellation and many zero entries in the defect function $r(s)$. So for the example system of section 4.4, the use of the condition number K_T for global error control is impossible. In fact, K_T is of the same size as the a priori condition number $K_{\mu_2} = (e^{\omega T} - 1)/\omega$, based on the logarithmic matrix norm μ_2, given in (2.7). Inserting the accurate bound $\omega = 100$, which in this case is just the largest eigenvalue of the Jacobian matrix due to symmetry [12], yields $K_{\mu_2} \approx 5.2 \times 10^{19}$.

4. Numerical illustrations. Numerical results are given for (i) a 2-dimensional unstable nonstiff example problem from [2], (ii) the 3-dimensional stable stiff Robertson chemical kinetics problem [8], (iii) a 100-dimensional unstable semidiscrete diffusion-reaction problem (an often used test problem from combustion theory [12]), and (iv) a 400-dimensional unstable semidiscrete diffusion-reaction problem (from pattern formation and often called the bistable Allen–Cahn problem). For the semidiscrete problems spatial errors are left out of consideration; i.e., we compare to a highly accurate ODE reference solution.

All four problems are solved with $Tol_A = Tol_R = Tol$ for the four tolerance values $Tol = 10^{-l}, l = 3, 4, 5, 6$, using one and the same initial step size $\tau_0 = 10^{-5}$. If after the first attempt (2.26) is violated with $C_{control} = 1$, a second attempt is carried out with the same τ_0 and the automatically adjusted new value for Tol through (2.27). Needless to say, $C_{control} = 1$ is too stringent. We use it here only for the sake of illustrating the good performance of the global control rule (2.27). It will be clear from the tables of results whether a second attempt was necessary.

The tables of results contain the following quantities $Tol_N = Tol\,(1+\|w_N\|)$ from (2.26): for the classical approach the ratio $\|\varepsilon_N\|/\|e_N\|$ of the true global error and the estimated global error, for the adjoint approach the ratio $\|\varepsilon_N\|/g_k(T)$ of the true global error and the estimated global error defined by (3.16), and for both approaches the ratio $\|\varepsilon_N\|/Tol_N$. The first and second ratios serve to illustrate the quality of the estimation, while the third does this for the control. In addition, the numbers of accepted and rejected ROS3P steps are given.

Finally, for the small sample statistical initialization used in the adjoint method, $k = m$ orthonormal random vectors were used for the two small-sized problems, whereas for the other two much larger problems we used only two random vectors. So only for the two larger problems was the small sample statistical initialization tested, and for the two small-sized problems the classical and adjoint approach should give identical results, except for minor implementation differences. Also observe that the randomness of the initialization will lead to differences in the results when computations are repeated, although minor ones.

4.1. A low-dimensional nonstiff ODE system. The first test problem is the 2-dimensional unstable linear system [2]

$$w' = \begin{pmatrix} \frac{1}{2(1+t)} & -2t \\ 2t & \frac{1}{2(1+t)} \end{pmatrix} w, \quad w(0) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad 0 < t \le T = 10\,, \tag{4.1}$$

with increasing oscillatory solution $w_1(t) = \cos(t^2)\sqrt{1+t}$, $w_2(t) = \sin(t^2)\sqrt{1+t}$. Table 4.1 shows the results for the classical approach. Since $k = m = 2$, the adjoint approach results are identical except for negligible implementation differences. The quality of the global error estimates is very high, and the second runs based on tolerance proportionality yield perfect control for all tolerances.

4.2. A low-dimensional stiff ODE system. Our second test problem is the well-known Robertson problem from the stiff ODE field [8]

(4.2)

$$\begin{aligned} w_1' &= -4.0 \times 10^{-2}\, w_1 + 10^4\, w_2 w_3\,, & w_1(0) &= 1\,, \\ w_2' &= 4.0 \times 10^{-2}\, w_1 - 10^4\, w_2 w_3 - 3.0 \times 10^7\, w_2^2\,, & w_2(0) &= 0\,, \\ w_3' &= 3.0 \times 10^7\, w_2^2\,, & w_3(0) &= 0\,, \quad 0 < t \le T = 1\,. \end{aligned}$$

TABLE 4.1
Low-dimensional nonstiff ODE system: classical approach.

Tol	Tol_N	$\|\epsilon_N\|/Tol_N$	$\|\epsilon_N\|/\|e_N\|$	Accept	Reject
10^{-3}	3.32×10^{-3}	8.16	1.02	1031	4
1.25×10^{-4}	3.34×10^{-3}	1.03	1.01	2044	0
10^{-4}	3.34×10^{-4}	8.23	1.01	2201	0
1.22×10^{-5}	3.34×10^{-4}	1.00	1.00	4415	0
10^{-5}	3.34×10^{-5}	8.20	1.00	4719	0
1.22×10^{-6}	3.35×10^{-5}	1.00	1.00	9419	0
10^{-6}	3.35×10^{-6}	8.19	1.00	10146	0
1.22×10^{-7}	3.35×10^{-6}	1.00	1.00	20426	0

TABLE 4.2
A low-dimensional stiff ODE system: adjoint approach.

Tol	Tol_N	$\|\epsilon_N\|/Tol_N$	$\|\epsilon_N\|/g_3(T)$	Accept	Reject
10^{-3}	1.56×10^{-3}	7.39×10^{-5}	1.05	29	0
10^{-4}	1.56×10^{-4}	1.05×10^{-3}	0.94	31	0
10^{-5}	1.56×10^{-5}	8.68×10^{-3}	1.01	40	1
10^{-6}	1.56×10^{-6}	7.64×10^{-2}	1.02	62	2

This stiff problem is highly stable, resulting in global errors much smaller than imposed local tolerances (the effect of the small initial $\tau_0 = 10^{-5}$ is less strong). So global error control is redundant here, and no control runs were carried out. The global error estimation appears to work very well for this stiff problem. Table 4.2 shows results of the adjoint approach. Since $k = m = 3$, the results of the classical approach again are identical, except for negligible implementation differences. These exist only for the column for the values of $\|\epsilon_N\|/g_3(T)$. The corresponding values $\|\epsilon_N\|/\|e_N\|$ for the classical approach are $(1.07, 1.02, 1.03, 1.04)^T$.

4.3. High-dimensional stiff ODE system I. Our third ODE test system is derived from spatially discretizing the well-known 1D combustion example problem (see, e.g., [12, p. 434] for the 2D version)

$$u_t = u_{xx} + \tfrac{1}{4}(2-u)\, e^{20(1-1/u)}\,, \quad 0 < x < 1\,, \qquad 0 < t \le T = 0.28\,, \tag{4.3}$$

subjected to the initial condition $u(x,0) = 1$, the zero Neumann boundary condition $u_x = 0$ at $x = 0$, and the Dirichlet boundary condition $u = 1$ at $x = 1$. The spatial discretization is done by second-order central differencing on a uniform hybrid grid (because of the Neumann condition) with mesh width $1/100.5$, resulting in a 100-dimensional ODE system $w' = F(w)$. We have chosen this system because it requires variable step sizes and it is unstable. The instability emanates from the reaction term whose derivative ranges between approximately $+1000$ and -5500; see [12] for details. Furthermore, the 100-dimensional ODE system poses a challenging test for the small sample statistical method using $k = 2$ only.

For the classical approach Table 4.3 reveals an excellent estimation of the global error and likewise a high quality of the control for all tolerances. Observe that for $Tol = 10^{-6}$ the second control run is redundant. Regarding the adjoint approach we emphasize the high quality of the small sample statistical method. From inequality (3.10) one would expect the ratios $\|\varepsilon_N\|/g_2(T)$ to lie asymptotically between 0.1 and 10.0 with 99% probability. Table 4.4, however, shows ratio values ranging between

TABLE 4.3
High-dimensional stiff ODE system I: *classical approach.*

Tol	Tol_N	$\|\epsilon_N\|/Tol_N$	$\|\epsilon_N\|/\|e_N\|$	Accept	Reject
10^{-3}	2.83×10^{-3}	2.56	1.25	529	33
4.91×10^{-4}	2.84×10^{-3}	1.03	1.20	680	32
10^{-4}	2.84×10^{-4}	2.64	1.13	1183	18
4.30×10^{-5}	2.84×10^{-4}	1.11	1.09	1586	11
10^{-5}	2.84×10^{-5}	2.09	1.05	2622	5
5.02×10^{-6}	2.84×10^{-5}	0.85	1.03	3318	3
10^{-6}	2.84×10^{-6}	0.91	1.00	5736	3

TABLE 4.4
High-dimensional stiff ODE system I: *adjoint approach.*

Tol	Tol_N	$\|\epsilon_N\|/Tol_N$	$\|\epsilon_N\|/g_2(T)$	Accept	Reject
10^{-3}	2.83×10^{-3}	2.56	1.35	529	33
5.26×10^{-4}	2.84×10^{-3}	1.06	1.38	664	32
10^{-4}	2.84×10^{-4}	2.64	0.92	1183	18
3.47×10^{-5}	2.84×10^{-4}	0.84	0.92	1708	8
10^{-5}	2.84×10^{-5}	2.09	0.76	2622	5
3.66×10^{-6}	2.84×10^{-5}	0.57	0.76	3695	4
10^{-6}	2.84×10^{-6}	0.91	0.72	5736	3
7.93×10^{-7}	2.84×10^{-6}	0.65	0.72	6204	3

0.72 and 1.38. Taking k larger with this large dimension m is no option. For example, doubling k to 4 does not lead to a notable improvement yet increases the costs for the global error estimation. Noting the small deviations from 1.0 of all listed ratios, especially after the control run, both approaches perform excellently with respect to estimation and control.

4.4. High-dimensional stiff ODE system II. Similar to the third, the fourth test problem was chosen because it is unstable, again challenging global error control. It is derived from spatially discretizing the following version of the bistable Allen–Cahn equation:

$$u_t = 10^{-2}\, u_{xx} + 100u\,(1-u^2)\,, \quad 0 < x < 2.5\,, \qquad 0 < t \le T = 0.5\,, \tag{4.4}$$

with the initial function and Dirichlet boundary values taken from the exact wave front solution $u(x,t) = (1 + e^{\lambda\,(x-\alpha\,t)})^{-1}$, $\lambda = 50\sqrt{2}$, $\alpha = 1.5\sqrt{2}$. Uniform second-order central discretization in space yields our ODE test system, now with $m = 400$ components. The instability emanates from the unstable stationary state $u = 0$. Further, since $m = 400$ and $k = 2$, this problem also poses an even more challenging test for the small sample statistical method.

Table 4.5 reveals again a high quality of the global error estimation for the classical approach and the control process also works very well. The ratios for $\|\epsilon_N\|/Tol_N$ lie between 0.71 and 0.93 after the control run. Observe that for $Tol = 10^{-5},\ 10^{-6}$ the control runs are redundant. In actual practice, taking the control parameter $C_{control} > 1$ in (2.26), this would also hold for the other tolerances. For the adjoint approach, the ratios $\|\epsilon_N\|/g_2(T)$ given in Table 4.6 range from 0.53 to 2.56, which is by a factor of 4 better than one would expect from inequality (3.10). The adjoint approach does not require a second run for $Tol = 10^{-4},\ 10^{-6}$, and for the other cases

TABLE 4.5
High-dimensional stiff ODE system II: *classical approach.*

Tol	Tol_N	$\|\epsilon_N\|/Tol_N$	$\|\epsilon_N\|/\|e_N\|$	Accept	Reject
10^{-3}	1.65×10^{-3}	1.29	0.77	373	0
6.01×10^{-4}	1.65×10^{-3}	0.71	0.83	446	0
10^{-4}	1.65×10^{-4}	0.95	0.93	833	0
9.84×10^{-5}	1.65×10^{-4}	0.93	0.93	838	0
10^{-5}	1.65×10^{-5}	0.82	0.97	1835	0
10^{-6}	1.65×10^{-6}	0.76	0.98	3998	0

TABLE 4.6
High-dimensional stiff ODE system II: *adjoint approach.*

Tol	Tol_N	$\|\epsilon_N\|/Tol_N$	$\|\epsilon_N\|/g_2(T)$	Accept	Reject
10^{-3}	1.65×10^{-3}	1.29	0.53	373	0
4.09×10^{-4}	1.65×10^{-3}	0.45	0.57	510	0
10^{-4}	1.65×10^{-4}	0.95	1.40	833	0
10^{-5}	1.65×10^{-5}	0.82	0.78	1835	0
9.45×10^{-6}	1.65×10^{-5}	0.78	0.78	1870	0
10^{-6}	1.65×10^{-6}	0.76	2.56	3998	0

the control is also very efficient (only a factor 0.45 to 0.78 off the imposed tolerance after the control run).

5. Summary and main conclusions. Inspired by [2] and related earlier literature, e.g., [1, 5, 6, 7, 13], we have discussed and compared classical global error estimation based on the first variational equation to a recent more novel approach based on the adjoint equation. The common starting point for both approaches is the perturbed equation with the residual or defect function defined by piecewise, cubic Hermite interpolation. As a base integrator for the comparison we have chosen the third-order, A-stable Runge–Kutta–Rosenbrock method ROS3P. We have also implemented global error control, for which we have used the property of tolerance proportionality for both the classical and the adjoint approaches.

On the basis of the four example problems, ranging from nonstiff to stiff, the computational effort of each of the two approaches, and the insight from the analysis, we have come to three main conclusions. (i) The classical approach is remarkably reliable, with respect to both estimation and control. Although well known in the numerical ODE literature, it seems that the virtue of this approach has been insufficiently brought forward, since as yet this form of global error estimation and control is far less popular than the commonly used local techniques. (ii) Most notable for the adjoint approach is the excellent performance of the small sample statistical method which forms the heart of the method. We have applied it successfully using only $k = 2$ random vectors for dimensions $m = 100, 400$ (in [2] it was not tested since there $k = m \leq 2$). With $k = 2$ the computational costs are only marginally higher than those of the classical approach. Were we to omit the second control run, then our Matlab runs in the adjoint case would be on average only about 15% more costly in CPU time. Omitting the second control run can make a difference, because in the adjoint case the second control run can be significantly more expensive than in the classical case due to the need of reevaluating many Jacobian matrices. (iii) The main

disadvantage of the adjoint approach is the need either to store the whole approximation sequence $(w_n; 0 \leq 1 \leq N)$ or to store part of it and to carry out a second forward computation. When using Jacobians, as is the case for ROS3P, storage becomes truly a handicap with large problems calling for Jacobian reevaluations and hence additional CPU costs.

Something we haven't discussed in this paper is that the adjoint approach can give precious sensitivity information which can be used to advantage for a different step size adaptation (see, e.g., [17]). Such information is not available within the classical approach. Our step size adaptation is standard and based on local error control. Other approaches based on a posteriori information and the adjoint approach do exist and are claimed to give better results for global error control; see [17, 18, 19].

REFERENCES

[1] R. Becker and R. Rannacher, *An optimal control approach to a posteriori error estimation in finite element methods*, Acta Numer., 10 (2001), pp. 1–102.

[2] Y. Cao and L. Petzold, *A posteriori error estimation and global error control for ordinary differential equations by the adjoint method*, SIAM J. Sci. Comput., 26 (2004), pp. 359–374.

[3] K. Dekker and J.G. Verwer, *Stability of Runge-Kutta Methods for Stiff Nonlinear Differential Equations*, North-Holland, Amsterdam, 1984.

[4] W.H. Enright, *Continuous numerical methods for ODEs with defect control*, J. Comput. Appl. Math., 125 (2000), pp. 159–170.

[5] K. Eriksson, C. Johnson, and A. Logg, *Adaptive computational methods for parabolic problems*, in Encyclopedia of Computational Mechanics, E. Stein, R. de Borst, and J.R. Hughes, eds., Wiley, New York, 2004.

[6] D. Estep, *A posteriori error bounds and global error control for approximation of ordinary differential equations*, SIAM J. Numer. Anal., 32 (1995), pp. 1–48.

[7] D. Estep, M. Larsson, and R. Williams, *Estimating the Error of Numerical Solutions of Systems of Reaction–Diffusion Equations*, Mem. Amer. Math. Soc. 146, AMS, Providence, RI, 2000.

[8] E. Hairer, S.P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations* I - *Nonstiff Problems*, second ed., Springer Ser. Comput. Math., Springer, Berlin, 1993.

[9] P. Henrici, *Discrete Variable Methods in Ordinary Differential Equations*, Wiley, New York, 1962.

[10] D.J. Higham, *Robust defect control with Runge-Kutta schemes*, SIAM J. Numer. Anal., 26 (1989), pp. 1175–1183.

[11] D.J. Higham, *Runge-Kutta defect control using Hermite-Birkhoff interpolation*, SIAM J. Sci. Comput., 12 (1991), pp. 991–999.

[12] W. Hundsdorfer and J.G. Verwer, *Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations*, Springer Ser. Comput. Math. 33, Springer, Berlin, 2003.

[13] C. Johnson, *Error estimates and adaptive time–step control for a class of one–step methods for stiff ordinary differential equations*, SIAM J. Numer. Anal., 25 (1988), pp. 908–926.

[14] C.S. Kenney and A.J. Laub, *Small–sample statistical condition estimates for general matrix functions*, SIAM J. Sci. Comput., 15 (1994), pp. 36–61.

[15] J. Lang, *Adaptive Multilevel Solution of Nonlinear Parabolic PDE Systems. Theory, Algorithm and Applications*, Lect. Notes Comput. Sci. Eng. 16, Springer, New York, 1994.

[16] J. Lang and J.G. Verwer, *ROS3P – An accurate third-order Rosenbrock solver designed for parabolic problems*, BIT, 41 (2001), pp. 731–738.

[17] K.-S. Moon, A. Szepessy, R. Tempone, and G. Zouaris, *Convergence rates for adaptive approximation of ordinary differential equations*, Numer. Math., 96 (2003), pp. 99–129.

[18] K.-S. Moon, A. Szepessy, R. Tempone, and G. Zouaris, *A variational principle for adaptive approximation of ordinary differential equations*, Numer. Math., 96 (2003), pp. 131–152.

[19] K.-S. Moon, E. Von Schwerin, A. Szepessy, and R. Tempone, *An adaptive algorithm for ordinary stochastic and partial differential equations*, Contemp. Math. 383 (2006), pp. 325–344.

[20] L.F. Shampine, *Numerical Solution of Ordinary Differential Equations*, Chapman and Hall, New York, 1994.

[21] L.F. Shampine, *Error estimation and control for ODEs*, J. Sci. Comp., 25 (2005), pp. 3–16.
[22] L.F. Shampine, *Solving ODEs and DDEs with residual control*, Appl. Numer. Math., 52 (2005), pp. 113–127.
[23] R.D. Skeel, *Thirteen ways to estimate global error*, Numer. Math., 48 (1986), pp. 1–20.
[24] H.J. Stetter, *Economical global error estimation*, in Stiff Differential Systems, R.A. Willoughby, ed., Plenum, New York, 1974, pp. 245–258.
[25] G. Watson, *Statistics on Spheres*, Univ. Arkansas Lect. Notes Math. Sci. 6, Wiley, New York, 1983.
[26] P.E. Zadunaisky, *On the estimation of errors propagated in the numerical integration of ordinary differential equations*, Numer. Math., 27 (1976), pp. 21–39.

SIAM J. SCI. COMPUT.
Vol. 29, No. 4, pp. 1476–1494
© 2007 Society for Industrial and Applied Mathematics

INCOMPLETE LU PRECONDITIONING WITH THE MULTILEVEL FAST MULTIPOLE ALGORITHM FOR ELECTROMAGNETIC SCATTERING*

TAHİR MALAS† AND LEVENT GÜREL†‡

Abstract. Iterative solution of large-scale scattering problems in computational electromagnetics with the multilevel fast multipole algorithm (MLFMA) requires strong preconditioners, especially for the electric-field integral equation (EFIE) formulation. Incomplete LU (ILU) preconditioners are widely used and available in several solver packages. However, they lack robustness due to potential instability problems. In this study, we consider various ILU-class preconditioners and investigate the parameters that render them safely applicable to common surface integral formulations without increasing the $\mathcal{O}(n \log n)$ complexity of MLFMA. We conclude that the no-fill ILU(0) preconditioner is an optimal choice for the combined-field integral equation (CFIE). For EFIE, we establish the need to resort to methods depending on drop tolerance and apply pivoting for problems with high condition estimate. We propose a strategy for the selection of the parameters so that the preconditioner can be used as a black-box method. Robustness and efficiency of the employed preconditioners are demonstrated over several test problems.

Key words. preconditioning, incomplete LU preconditioners, multilevel fast multipole algorithm, electromagnetic scattering

AMS subject classifications. 31A10, 65F10, 78A45, 78M05

DOI. 10.1137/060659107

1. Introduction.

A popular approach in studying wave scattering phenomena in computational electromagnetics (CEM) is to solve discretized surface integral equations, which give rise to large, dense, complex systems in the form of $\overline{\mathbf{A}} \cdot \mathbf{x} = \mathbf{b}$. Two kinds of surface integral equations are commonly used. The electric-field integral equation (EFIE) can be used for both open and closed geometries, but it results in poorly conditioned systems, especially when the geometry is large in terms of the wavelength. On the other hand, the combined-field integral equation (CFIE) produces well-conditioned systems, but it is applicable to closed geometries only [21].

For the solution of such dense systems, direct methods based on Gaussian elimination are still widely used due to their robustness [34]. However, the large problem sizes confronted in computational electromagnetics prohibit the use of these methods which have $\mathcal{O}(n^2)$ memory and $\mathcal{O}(n^3)$ computational complexity for n unknowns. On the other hand, by making use of the multilevel fast multipole algorithm (MLFMA) [12], the dense matrix-vector products required at least once in each step of the iterative solvers can be performed in $\mathcal{O}(n \log n)$ time and by storing only the sparse near-field matrix elements, rendering these solvers very attractive for large problems.

However, the iterative solver may not converge, or convergence may require too many iterations. We need to have a suitable preconditioner to reach convergence in

*Received by the editors May 3, 2006; accepted for publication (in revised form) January 19, 2007; published electronically June 21, 2007. This work was supported by the Scientific and Technical Research Council of Turkey (TUBITAK) under research grant 105E172, by the Turkish Academy of Sciences in the framework of the Young Scientist Award Program (LG/TUBA-GEBIP/2002-1-12), and by contracts from ASELSAN and SSM.

http://www.siam.org/journals/sisc/29-4/65910.html

†Department of Electrical and Electronics Engineering, Bilkent University, TR-06800, Bilkent, Ankara, Turkey (tmalas@ee.bilkent.edu.tr, lgurel@bilkent.edu.tr).

‡Computational Electromagnetics Research Center (BiLCEM), Bilkent University, TR-06800, Bilkent, Ankara, Turkey.

a reasonable number of iterations and retain the $\mathcal{O}(n \log n)$ complexity of MLFMA. We can use the sparse near-field matrix $\overline{\mathbf{N}}$ to construct a preconditioner. Since this matrix is the best available approximation to the coefficient matrix $\overline{\mathbf{A}}$, it makes sense to use the near-field matrix as a preconditioner and solve (for example) the left-preconditioned system

$$\overline{\mathbf{N}}^{-1} \cdot \overline{\mathbf{A}} \cdot \mathbf{x} = \overline{\mathbf{N}}^{-1} \cdot \mathbf{b}. \tag{1.1}$$

The inversion of the near-field matrix $\overline{\mathbf{N}}$ can be accomplished using direct methods which decompose the matrix into a product of a unit lower-triangular matrix $\overline{\mathbf{L}}$ and an upper-triangular matrix $\overline{\mathbf{U}}$. However, during the factorization of sparse matrices, in general, fill-in occurs and the resulting factors lose their sparsity [28]. This may make it difficult to preserve the $\mathcal{O}(n \log n)$ complexity of MLFMA. Nevertheless, we can discard part of the fill-in and partially incorporate the robustness of the LU factorization into the iterative method by using the incomplete factors of $\overline{\mathbf{N}}$ as a preconditioner. This is the general idea behind the incomplete LU (ILU) preconditioners.

In a general setting, depending on the dropping strategy, we can talk about two kinds of ILU-class preconditioners. The first one depends on the matrix structure and the entries are dropped by their position. A "levels of fill-in" concept is introduced and stronger preconditioners can be constructed by increasing the level of fill-in [31]. Since this technique does not consider numerical values, it becomes ineffective in predicting the locations of the largest entries, particularly for matrices that are far from being diagonally dominant and indefinite [13]. This is the case for the matrices arising from the EFIE formulation. Alternatively, one can drop the matrix elements depending on their magnitudes, and the zero pattern is generated dynamically during the factorization. Among such methods, ILUT(τ, p) proposed by Saad has been successful for many general systems [3]. During the factorization, ILUT drops matrix elements that are smaller than τ times the 2-norm of the current row; and of all the remaining entries no more than the p largest ones are kept. ILUT is known to yield more accurate factorizations than the level-of-fill methods with the same amount of fill-in [13].

Although ILUT is more robust than its counterparts depending on the level of fill-in, it may occasionally encounter problems of instability for real-life problems. Even when factorization terminates normally, the resulting incomplete factors may sometimes be unstable. The common reasons of instability are in general excessive dropping and small pivots [13]. If the problem is related to the small pivots, one can significantly increase the quality of the ILUT preconditioner by using partial pivoting as in the complete factorization case. The resulting preconditioner is called ILUTP [31].

In order to understand the quality of the preconditioner, or to understand the reason for failure when it occurs, we can use $\|(\overline{\mathbf{L}} \cdot \overline{\mathbf{U}})^{-1} \cdot \mathbf{e}\|_\infty$, where $\mathbf{e}$ is the vector of ones. This statistic is called *condest* (for condition estimate) and it provides an upper bound for $\|(\overline{\mathbf{L}} \cdot \overline{\mathbf{U}})^{-1}\|_\infty$ [13]. If the *condest* value is not very high, but the preconditioner still does not work, one can deduce that fill-in should be increased to achieve a successful preconditioner. On the other hand, if the *condest* value is high, one can first try pivoting to remedy the situation instead of including more elements in the incomplete factors.

Considering the remarkable success of ILU-class preconditioners for general non-symmetric and indefinite systems [13] and the wide availability of ILU-class preconditioners in various packages [2, 24, 6], the present study aims to develop a strategy for

both selecting the most appropriate ILU-class preconditioner and determining their parameters to use them as black-box preconditioners for CFIE and EFIE formulations. We perform tests on canonical, quasi-canonical, and real-life problems with increasing number of unknowns and show that when these preconditioners fail for the reasons stated earlier, the failure can be circumvented using pivoting strategies without increasing the memory cost. We also show that the *condest* value is very useful for determining the quality of the resulting factorization before starting the iterative solution.

ILU-class preconditioners have been tested for electromagnetic problems in [8, 32, 26]. In [8], ILU(0) was tried on systems resulting from EFIE formulation with discouraging results in all test problems. Sertel and Volakis [32] tried ILU(0) on two model problems. For the very small problem of 480 unknowns, ILU(0) was successful with the EFIE and CFIE formulations, but clearly such a small problem is not representative of large-scale CEM simulations. They presented only CFIE results for the 50,000-unknown problem; in this case, ILU(0) was quite successful in reducing the number of iterations. Probably the most impressive results are those of Lee, Zhang, and Lu [26], who tried the ILUT preconditioner on hybrid surface-volume integral equations and showed it to be successful on many test problems. However, they neither tried commonly used EFIE or CFIE formulations nor applied pivoting or any other techniques to increase the effectiveness of the preconditioner.

The rest of the paper is organized as follows. In the next section, we outline the surface integral equations of CEM, their discretizations, resulting matrix equations, and MLFMA. Spectral properties of such matrices are analyzed in section 3. Then, in section 4, we comment on the preconditioning of the systems arising from integral-equation formulations. In section 5, we present experimental results for the CFIE and EFIE formulations. In the last section, we make some suggestions for the effective and safe usage of ILU-class preconditioners with EFIE and CFIE.

2. Surface integral equations and fast solvers. EFIE and CFIE are surface integral equations that are formed from the application of physical boundary conditions; they are formulated to solve the radiation and scattering problems of arbitrarily shaped geometries. This leads to the reduction of a three-dimensional problem into a two-dimensional problem, but the resulting matrix equation becomes fully populated.

The boundary condition stating that the total tangential electric field should vanish on a conducting surface can be mathematically expressed to obtain EFIE as

$$\hat{\boldsymbol{t}} \cdot \int_{S'} \mathrm{d}\mathbf{r}' \overline{\mathbf{G}}(\mathbf{r}, \mathbf{r}') \cdot \mathbf{J}(\mathbf{r}') = \frac{i}{k\eta} \hat{\boldsymbol{t}} \cdot \mathbf{E}^{inc}(\mathbf{r}), \tag{2.1}$$

where $\mathbf{E}^{inc}$ represents the incident electric field, S' is the surface of the object, $\hat{\boldsymbol{t}}$ is any tangential unit vector on S', $\mathbf{J}(\mathbf{r}')$ is the unknown induced current residing on the surface,

$$\overline{\mathbf{G}}(\mathbf{r}, \mathbf{r}') = \left[\bar{\mathbf{I}} + \frac{\nabla\nabla}{k^2} \right] g(\mathbf{r}, \mathbf{r}') \tag{2.2}$$

is the dyadic Green's function, and

$$g(\mathbf{r}, \mathbf{r}') = \frac{e^{ik|\mathbf{r}-\mathbf{r}'|}}{4\pi|\mathbf{r}-\mathbf{r}'|} \tag{2.3}$$

is the scalar Green's function for the three-dimensional scalar Helmholtz equation. Green's function represents the response at $\mathbf{r}$ due to a point source located at $\mathbf{r}'$.

Similarly, the boundary condition for the tangential magnetic field on a conducting surface is used to derive the magnetic-field integral equation (MFIE) [22, 14] as

$$\mathbf{J}(\mathbf{r}) - \hat{\boldsymbol{n}} \times \int_{S'} \mathrm{d}\mathbf{r}' \mathbf{J}(\mathbf{r}') \times \nabla' g(\mathbf{r}, \mathbf{r}') = \hat{\boldsymbol{n}} \times \mathbf{H}^{inc}(\mathbf{r}), \tag{2.4}$$

where $\hat{\boldsymbol{n}}$ is any unit normal on S' and $\mathbf{H}^{inc}(\mathbf{r})$ is the incident magnetic field. It should be noted that EFIE is valid for both open and closed geometries, whereas MFIE is valid only for closed surfaces [18, 16].

Combining EFIE and MFIE, we obtain CFIE, i.e.,

$$\mathrm{CFIE} = \alpha \mathrm{EFIE} + (1 - \alpha)\mathrm{MFIE}, \tag{2.5}$$

where α is a parameter between 0 and 1. CFIE is free from the internal resonance problems of both EFIE and MFIE and leads to well-conditioned systems [19]. On the other hand, CFIE is not applicable to open geometries since it contains MFIE. Therefore, CFIE is preferred over MFIE for closed geometries, but EFIE is the only choice for open geometries. CFIE can also be extended for geometries containing both closed and open surfaces [23].

Surface integral equations can be converted to linear systems of equations using the method of moments (MOM). In fact, MOM can be used to reduce any linear-operator equation to a matrix equation. Using a linear operator L, we can represent EFIE and CFIE as

$$L\{\mathbf{f}\} = \mathbf{g}, \tag{2.6}$$

where $\mathbf{f}$ is the unknown vector function, and $\mathbf{g}$ is the excitation. To discretize this equation, $\mathbf{f}$ is first approximated by a set of vector basis functions via the expansion

$$\mathbf{f} \approx \sum_{j=1}^{n} x_j \mathbf{f}_j, \tag{2.7}$$

where x_j is the unknown coefficient of the jth basis function. Then this discretized equation is tested at n points and a linear system is obtained, i.e.,

$$\sum_{j=1}^{n} x_j \langle \mathbf{t}_i, L\{\mathbf{f}_j\} \rangle = \langle \mathbf{t}_i, \mathbf{g} \rangle, \qquad i = 1, \ldots, n. \tag{2.8}$$

In (2.8), $\mathbf{t}_i$ is the vector testing function and the inner product is defined as

$$\langle \mathbf{t}, \mathbf{f} \rangle = \int_S \mathrm{d}\mathbf{r}\, \mathbf{t}(\mathbf{r}) \cdot \mathbf{f}(\mathbf{r}). \tag{2.9}$$

By defining the elements of the coefficient matrix (or the so-called impedance matrix) as

$$A_{ij} = \langle \mathbf{t}_i, L\{\mathbf{f}_j\} \rangle \tag{2.10}$$

and the elements of the right-hand-side vector (or the excitation vector) as

$$b_i = \langle \mathbf{t}_i, \mathbf{g} \rangle, \tag{2.11}$$

the linear system of equations can be expressed as

$$\overline{\mathbf{A}} \cdot \mathbf{x} = \mathbf{b}. \tag{2.12}$$

In order to implement a Galerkin approach for discretization, we use Rao–Wilton–Glisson (RWG) functions [29] for both basis and testing functions. RWG functions are linearly varying vector functions defined on planar triangular domains. They are widely used in MOM applications to discretize surface current distributions.

Surfaces of geometries are meshed with planar triangles in accordance with the RWG functions. Each basis function is associated with an edge; hence the number of unknowns n is equal to the total number of edges in a triangulation. For high accuracy, triangle sizes have to be small, but this leads to problems with large numbers of unknowns. As a rule of thumb, we choose the average size of the mesh about one-tenth of the wavelength.

Since the integrodifferential operator L for both EFIE and CFIE involves long-distance interactions, the resulting coefficient matrix is dense, which is expensive to store and to solve. However, when we use iterative solvers, direct solution is replaced with a series of matrix-vector multiplications, and we have the opportunity to perform the matrix-vector product with $\mathcal{O}(n \log n)$ complexity using MLFMA.

MLFMA can be described as the multilevel application of the fast multipole method (FMM). An important concept, on which FMM relies, is the diagonalized factorization of the Green's function, i.e.,

$$g(\mathbf{r}, \mathbf{r}') = \frac{e^{ik|\mathbf{r}-\mathbf{r}'|}}{4\pi|\mathbf{r}-\mathbf{r}'|} = \frac{e^{ik|\mathbf{D}+\mathbf{d}|}}{4\pi|\mathbf{D}+\mathbf{d}|} \approx \frac{1}{4\pi} \int d^2\hat{\boldsymbol{k}} e^{i\hat{\boldsymbol{k}}\cdot\mathbf{d}} T_L(k, D, \psi), \tag{2.13}$$

where $D = |\mathbf{D}| < d = |\mathbf{d}|$, the integration is on the unit sphere, $\hat{\boldsymbol{k}}$ is the unit vector normal to the unit sphere, and T_L is the truncated sum

$$T_L(k, D, \psi) = \frac{ik}{4\pi} \sum_{l=0}^{L} i^l (2l+1) h_l^{(1)}(kD) P_l \cos(\psi), \tag{2.14}$$

which is called the translation function [17]. In (2.14), $h_l^{(1)}(x)$ is the spherical Hankel function of the first kind, P_l is the Legendre polynomial, and ψ is the angle between unit vectors $\hat{\boldsymbol{k}}$ and $\hat{\boldsymbol{D}}$. The truncation number L is determined by the formula given in [12],

$$L \approx kd + 1.8 d_0^{2/3} (kd)^{1/3}, \tag{2.15}$$

where d_0 is the number of accurate digits required in the matrix-vector multiplication.

In a physical setting, we can think of a matrix-vector multiplication as a set of electromagnetic interactions between the basis and testing functions. When the basis and testing functions are clustered according to the proximity of their locations in space, the same translation function can be used for all interactions between pairs of functions in any two clusters, instead of calculating the interactions separately. For this purpose, radiation patterns of the basis functions $\mathbf{f}_j$ weighted with the corresponding coefficients x_j are evaluated on the unit sphere at K directions. Next, in each basis group, the radiation patterns are added at each direction, a process called aggregation. For each pair of basis and testing groups, a translation function is defined. The overall radiation pattern of each basis group is multiplied by a translation function and thereby translated to the center of a testing cluster. Following

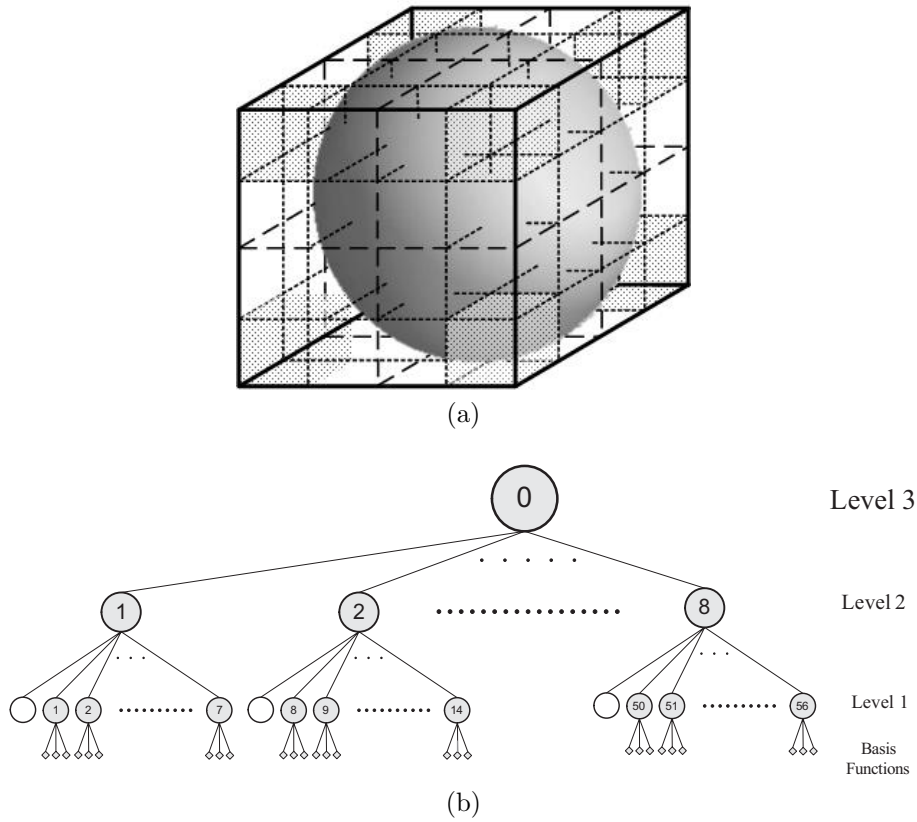


FIG. 2.1. (a) *Multilevel partitioning of the scatterer for the case of a sphere with diameter* 1λ. *The shaded boxes are empty.* (b) *Tree structure of MLFMA for the sphere. Unfilled nodes correspond to empty boxes.*

the translations, radiation patterns of the basis groups are added at the center point of each testing group. Finally, the total radiation from all basis functions at each testing cluster is disaggregated onto individual testing functions in order to complete the testing process. Since the factorization of the Green's function is not valid for basis and testing functions that are close to each other, the near-field interactions are calculated directly. In a multilevel scheme, MLFMA requires interpolations and anterpolations to pass from one level to another [15].

In MLFMA, to perform a multilevel application of FMM, the whole geometry is placed in a cube and then this cube is recursively divided into smaller ones until the smallest cube contains only a few basis or testing functions. However, during the partitioning, if any cube becomes empty, recursion stops there. This partitioning defines the levels of MLFMA and corresponds to a tree structure. An example of a sphere with a diameter of one wavelength (1λ) is shown in Figure 2.1.

With the partitioning scheme described, we can easily determine the near-field and far-field clusters. On any level, pairs of same-size cubes touching at any point are in the near-field zone of each other and the others are in the far-field zone. In the MLFMA tree, we refer to the filled cubes as clusters. A cluster contains either some basis functions (in the lowest level) or low-level clusters. In MLFMA, the replacement of the element-to-element interactions with cluster-to-cluster interactions is made in

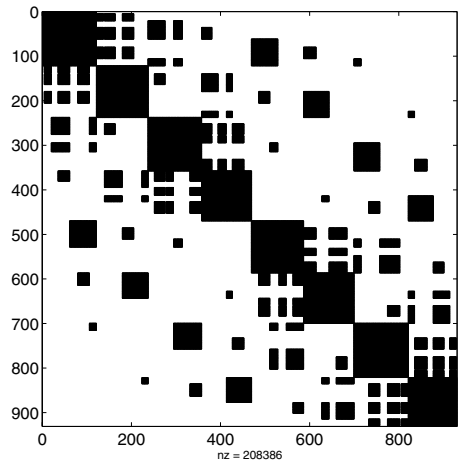


FIG. 2.2. *Near-field pattern of the* 930*-unknown sphere problem.*

a multilevel scheme.

In the lowest level, interactions between the near-field clusters are computed exactly and stored in the sparse near-field matrix. When the ordering of the unknowns is in accordance with the clustering, the near-field matrix is composed of small blocks, but the distribution of the blocks does not exhibit any structured pattern. For the sphere geometry shown in Figure 2.1, the nonzero pattern of the near-field matrix is presented in Figure 2.2. Self-interactions of the second-level clusters can easily be observed in the diagonal blocks of the nonzero pattern.

Interactions among the far-field clusters are computed approximately (but with controllable accuracy) and efficiently. For each level excluding the highest two, radiated fields of each cluster are aggregated at the centers of the clusters. Then, for each pair of far-field clusters whose parents are in the near-field zone of each other, cluster-to-cluster interaction is computed via a single translation. If the parents of the pair of clusters are not in the near-field of each other, the cluster-to-cluster interaction is included in a higher-level translation. Finally, after the translations, the summed values at the centers of the testing clusters are disaggregated towards the testing functions, completing the matrix-vector multiplication.

3. Spectral analysis. For electromagnetic scattering problems, iterative methods with MLFMA provide new opportunities to solve large-scale problems that were previously unsolvable. However, EFIE, MFIE, and CFIE formulations produce indefinite systems for which convergence becomes an issue. Moreover, since we consider the solution of very large problems with millions of unknowns, the condition of the matrix deteriorates (especially for EFIE) as the problem sizes grow. Hence, effective preconditioning is indispensable for attaining convergence in a reasonable time.

Though they are indefinite and nonhermitian, CFIE produces well-conditioned systems that are close to being diagonally dominant. As a consequence, the number of iterations required for convergence has been limited with a simple block-Jacobi preconditioner even for large-scale problems [19]. Nonetheless, for some geometries, the number of iterations is still large. Considering the dominant cost of matrix-vector product for large problems, better preconditioners for CFIE are still desirable.

Systems resulting from EFIE are much more difficult to solve. In addition to being

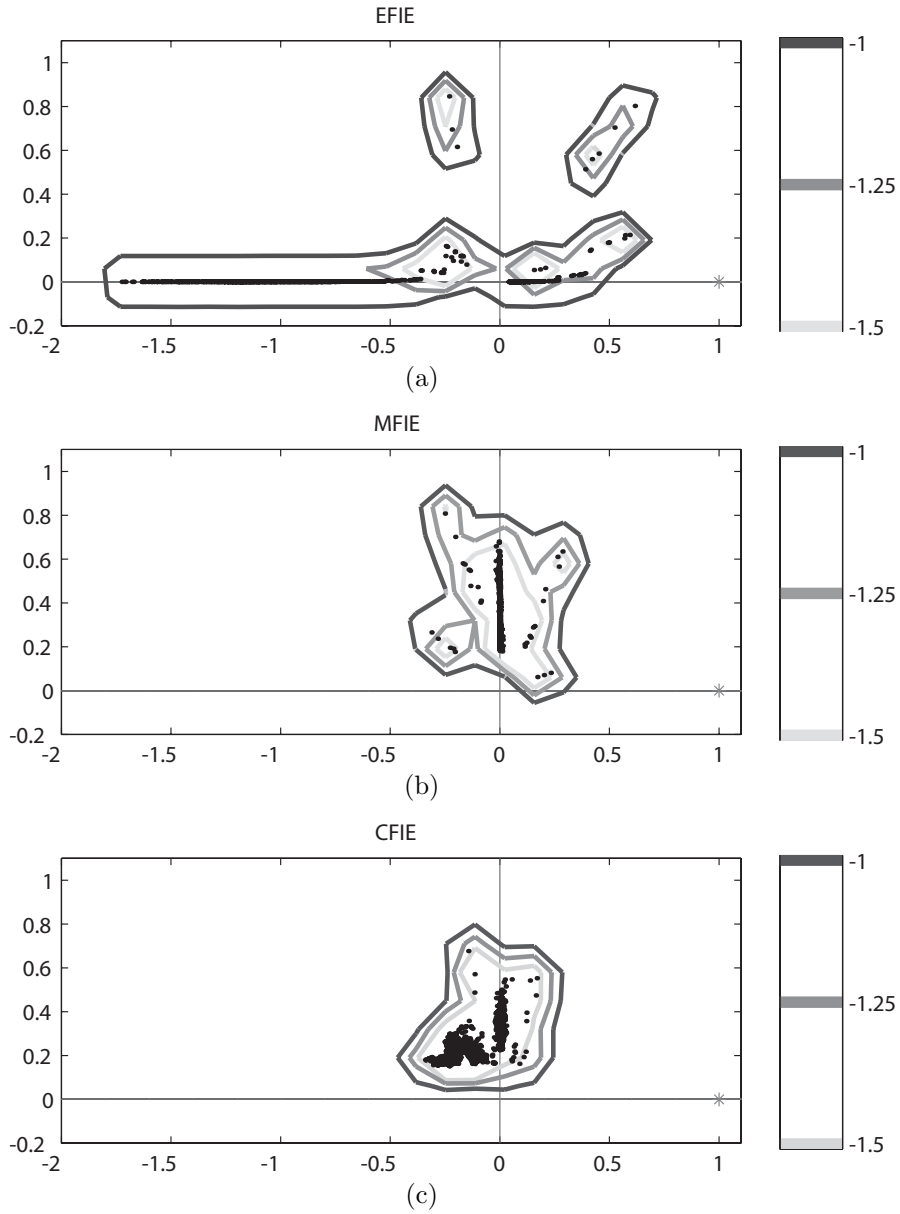


FIG. 3.1. *Pseudospectra of the EFIE, MFIE, and CFIE formulations for three ϵ values, i.e., 10^{-1}, $10^{-1.25}$, and $10^{-1.5}$. The black dots denote the exact eigenvalues of the unperturbed matrices.*

indefinite and nonhermitian, EFIE matrices may have large elements away from the diagonal, and some of the nonstored far-field interactions may be stronger than the near-field interactions.

For a better understanding of the properties of the systems resulting from surface integral equations, we show in Figure 3.1 both the eigenvalues and the pseudospectra of the EFIE, MFIE, and CFIE matrices for the 930-unknown sphere problem. For nonnormal matrices, information obtained from eigenvalues may be misleading, since

they may become highly unstable [33]. More reliable and telling information can be obtained using the ϵ-pseudospectrum, $\wedge_\epsilon(\overline{\mathbf{A}})$, which can be defined as

$$\wedge_\epsilon(\overline{\mathbf{A}}) = \left\{ z \mid z \in \mathbb{C}^n,\ \|(z\overline{\mathbf{I}} - \overline{\mathbf{A}})^{-1}\|_2 \geq \frac{1}{\epsilon} \right\}. \tag{3.1}$$

Denoting the spectrum of $\overline{\mathbf{A}}$ with $\wedge(\overline{\mathbf{A}})$, if an eigenvalue $\lambda \in \wedge(\overline{\mathbf{A}})$, then $\|(z\overline{\mathbf{I}} - \overline{\mathbf{A}})^{-1}\|_2 = \infty$, so $\wedge(\overline{\mathbf{A}}) \subset \wedge_\epsilon(\overline{\mathbf{A}})$ for any $\epsilon > 0$. The pseudospectrum represents the topology of the eigenvalues of the perturbed matrices associated with the exact matrix $\overline{\mathbf{A}}$, and thus gives an idea about the nonnormality.

The ultimate aim in preconditioning is to move all eigenvalues towards the point (1,0). However, if the matrix is close to normal, a spectrum clustered away from the origin also implies rapid convergence for Krylov subspace methods [3, 25]. Comparison of the EFIE and CFIE pseudospectra in Figure 3.1 indicates that combining EFIE with MFIE (to obtain CFIE) has the effect of clustering the distributed eigenvalues and moving them towards the right half-plane. In contrast, most of the eigenvalues of EFIE are scattered in the left half-plane. Moreover, the 0.1-pseudospectrum of the EFIE matrix contains the origin, signaling the near-singularity of the matrix. Hence, effective preconditioning for EFIE becomes more difficult, and also more crucial.

4. ILU-class preconditioners. Various preconditioners have been used for the solution of CEM problems. For CFIE, a block-Jacobi preconditioner is frequently used. In an MLFMA setting, a block-Jacobi preconditioner can be constructed from the self-interactions of the lowest-level clusters. Since there are $\mathcal{O}(n)$ such blocks and each block is composed of a fixed number of unknowns, both the construction and the application of the preconditioner scale with $\mathcal{O}(n)$. Because of its optimal complexity and success with many problems, this simple preconditioner is a common choice for CFIE. However, probably due to the weaker diagonal dominance and indefiniteness of EFIE, the block-Jacobi preconditioner performs even worse than the no-preconditioner case. Sparse approximate inverse (SAI) preconditioners depending on a fixed a priori pattern have been thoroughly studied in some recent works [9, 10, 11, 27, 1]. The electromagnetics community has started to use SAI preconditioners more frequently because of ease of parallelization. However, the construction cost of SAI can become prohibitively large unless one chooses the prefiltering and postfiltering threshold parameters carefully [27]. Hence, it is not suitable for use as a black-box preconditioner; there is still the need for a more easily attainable preconditioner, particularly for sequential implementations.

As an alternative, the ILU-class preconditioners have been widely used and included in several solver packages. They were historically developed for positive-definite and structured matrices arising from the discretization of partial differential equations. For general systems, the failure rate of ILU-class preconditioners is still high. Nonetheless, there have been many improvements to increase their robustness [13, 4, 5].

For iterative solvers utilizing MLFMA, the near-field matrix $\overline{\mathbf{N}}$ is the natural candidate to generate the incomplete factors. Consider an incomplete factorization of the near-field matrix, $\overline{\mathbf{N}} \approx \overline{\mathbf{L}} \cdot \overline{\mathbf{U}}$. If we let the sparsity patterns of $\overline{\mathbf{N}}$ and $\overline{\mathbf{L}} + \overline{\mathbf{U}}$ be the same, that is, if we retain nonzero values of $\overline{\mathbf{L}}$ and $\overline{\mathbf{U}}$ only at the nonzero positions of $\overline{\mathbf{N}}$, we end up with the no-fill LU method, or ILU(0). This simple idea works successfully for well-conditioned matrices [31]. Denser and potentially more effective preconditioners can be obtained by increasing the level of fill-in, but this strategy is unsuccessful in determining the largest entries, particularly for matrices

that are indefinite and far from being diagonally dominant. A more robust strategy is to drop the nonzero elements by comparing the magnitudes during the factorization. Such a strategy discards elements that are small with respect to a suitably chosen drop tolerance τ.

One of the disadvantages of the dropping strategy, which depends on the size of matrix entries, is the difficulty in predicting storage. For this purpose, a dual threshold strategy can be used [30]. The resulting preconditioner, called ILUT(τ, p), retains no more than p elements in the incomplete factors after dropping all the elements that are smaller than τ times the 2-norm of the current row. The threshold parameter τ determines the CPU time and p determines the storage requirement of the preconditioner. This preconditioner is known to be quite powerful and robust.

Despite ILUT's good reputation, there are two important drawbacks preventing its use as a black-box and general library software. The first problem is determining the appropriate parameters. For our specific applications and in the context of MLFMA, we propose to select a small drop tolerance and then set the parameter p so that the preconditioner will have approximately the same number of nonzero elements as the near-field matrix $\overline{\mathbf{N}}$. Once the near-field matrix is generated, this value can easily be found. With this strategy, we aim to obtain a powerful preconditioner with modest storage and low complexity.

Probably the more problematic aspect of threshold-based ILU-class preconditioners is their potential inaccuracy and/or instability. Accuracy refers to how close the incomplete factors are to $\overline{\mathbf{A}}$; this is measured by the norm of the error matrix, i.e., $accuracy = \|\overline{\mathbf{E}}\| = \|\overline{\mathbf{A}} - \overline{\mathbf{L}} \cdot \overline{\mathbf{U}}\|$. Stability refers to how close the preconditioned matrix is to the identity matrix and is measured by the norm of the preconditioned error, i.e., $stability = \|(\overline{\mathbf{L}} \cdot \overline{\mathbf{U}})^{-1} \cdot \overline{\mathbf{E}}\|$. If $\|\overline{\mathbf{L}}^{-1}\|$ or $\|\overline{\mathbf{U}}^{-1}\|$ are extremely large, a factorization may turn out to be accurate but unstable; in that case, the preconditioner may not work even if fill-in is increased [13]. Thus, for general matrices, stability is a more informative measure of the preconditioning quality.

Although we cannot compute these metrics with MLFMA, a rough estimate of $\|(\overline{\mathbf{L}} \cdot \overline{\mathbf{U}})^{-1}\|$, called *condest*, gives a clue about the instability of the triangular factors [13]. This condition estimate is defined as

$$\|(\overline{\mathbf{L}} \cdot \overline{\mathbf{U}})^{-1} \cdot \mathbf{e}\|_\infty, \qquad \mathbf{e} = [1, \ldots, 1]^T. \tag{4.1}$$

One can easily compute *condest* before the iterations, by using a forward substitution followed by a backward substitution, and it provides a strong indicator of the quality of the ILU preconditioner.

When the incomplete factors turn out to be unstable, there are some remedies that can be utilized depending on the cause. Preprocessing steps such as diagonal perturbation, reordering, and scaling can be applied on the coefficient matrix to stabilize the preconditioner. To increase the stability, diagonal perturbations can be used to make the LU factors more diagonally dominant, but quite large perturbations may be required for indefinite systems, and such large perturbations may introduce too much inaccuracy into the preconditioner. Diagonal shifts have already been tried on EFIE systems to increase the robustness of ILU, but the effect of the shift is undetermined, and furthermore it is difficult to select suitable shift parameters [7]. Reorderings aimed at improving the condition of the incomplete factors are widely studied. Indeed, some reordering schemes significantly improve the convergence of the Krylov methods [4]. However, the effect of ordering becomes significant when the incomplete factors are allowed to be denser than the original matrix [3]. We usually

prefer to keep the memory required by the preconditioner bounded by the storage needed for the near-field matrix. Hence, this remedy is not a good candidate because of the storage considerations. Finally, for threshold-based ILU-class preconditioners, it is recommended to scale the matrix so that each column has unit 2-norm, and then scale it again so that each row has unit 2-norm. This suggestion is not applicable in the context of MLFMA, since the preconditioner is constructed from a sparse portion of the coefficient matrix and the coefficient matrix is not explicitly available.

On the other hand, if the instability is caused by the small pivots, partial pivoting is helpful. This is a well-known and much simpler method. Column pivoting can be applied in a rowwise factorization with negligible cost. The resulting preconditioner is known as ILUTP [31]. In some cases, it may be useful to include a permutation tolerance , ., ., and perform the permutation for the ith row when $permtol \times |a_{ij}| > |a_{ii}|$. It is best not to select a very small value for *permtol*; 0.5 is accepted as a good choice [31].

5. Results. In this section, we show the effectiveness of ILU-class preconditioners for electromagnetic scattering problems. We first identify the most appropriate ILU-class preconditioner for the problem type (i.e., open geometries vs. closed geometries, EFIE vs. CFIE), then compare the selected ILU preconditioner with other commonly used preconditioners. For this purpose, we implement an SAI preconditioner, whose sparsity pattern is chosen to be the same as the near-field matrix. In this way, it has the same storage cost as that of ILU(0).

Instead of giving several results with varying parameters for ILUT, we adopt the following strategy for the selection of the parameters. We set the drop tolerance τ to a low value such as 10^{-6} and set p, the maximum number of nonzero elements per row, such that the memory cost of factorization does not exceed that of the no-fill ILU preconditioner. We accomplish this by simply letting p be the average number of nonzero elements in a row of the near-field matrix. In this way, we obtain robust preconditioners with modest computational requirements.

For the specific implementation of the MLFMA considered here, we set the size of the smallest clusters to 0.25λ, the number of accurate digits d_0 to 3, and the α parameter of CFIE to 0.2. The CPU times reported in this section were obtained on a 64-bit server with 1.8 GHz AMD Opteron 244 processors and 4 GB of memory. In addition to performing numerical experiments involving ILU-class preconditioners, we also obtain the exact solution of the near-field matrix to use it as a benchmark preconditioner. This solution, which is denoted by LU, is performed on another 64-bit server with 24 GB of memory. Due to its excessive computational requirements, this LU preconditioner is presented merely for comparison purposes.

For the iterative solver, starting with the zero initial guess, we try to reduce the initial residual norm by 10^{-6} and set the maximum number of iterations at 1,500. We use the generalized minimal residual method (GMRES) with no-restart and apply right-preconditioning in order to minimize the true residual norm. For CFIE solution of closed geometries, the performance of other Krylov subspace methods, such as conjugate gradient squared (CGS), biconjugate gradient (Bi-CG) or biconjugate gradient stabilized (Bi-CGSTAB), approximates GMRES in terms of the number of matrix-vector products. However, for EFIE, other solvers are less robust and do not always converge with preconditioning. Even when they converge, they require more matrix-vector multiplications than GMRES. Though GMRES with no-restart brings extra CPU and memory costs with increasing number of iterations, reduction in the number of matrix-vector products significantly decreases the overall solution time due

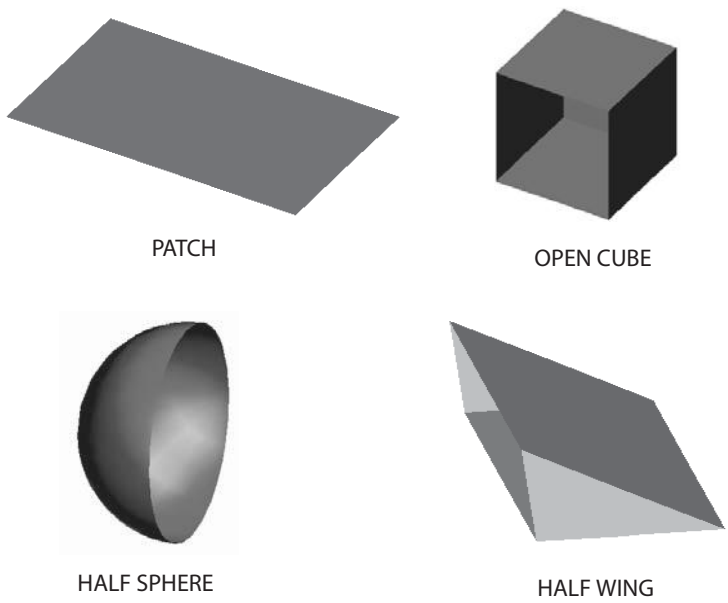


FIG. 5.1. *Open geometries.*

to the high cost of matrix-vector multiplications, particularly for large problems.

5.1. Open geometries. Figure 5.1 displays the open geometries used in the numerical experiments, i.e., a patch (P), a half sphere (HS), an open prism (OP), and an open cube (OC). These geometries are solved at various frequencies, requiring different meshes and numbers of unknowns as shown in Table 5.1. In Table 5.1 the "Size" column stands for the diameter for the spheres, and the maximum side length for others. The subsection sizes of different meshes are consistently selected as one-tenth of the wavelength. As mentioned in section 2, EFIE is the only choice for these geometries.

We compare the ILU-class preconditioners in Table 5.2. The summary of our observations are as follows:

(i) It is easily noticed that as the number of unknowns increases, ILU(0) produces highly unstable and hence useless factorizations. ILU(0) works well for small problems due to the fact that the near-field matrices used to generate the preconditioner and consequently the incomplete factors are nearly dense for such problems.

(ii) ILUT produces stable factors for all geometries except HS3. When we use 0.5 pivoting tolerance (ILUTP5) or 1.0 pivoting tolerance (ILUTP), we overcome the problem. However, for HS3, ILUTP yields a larger *condest* value and requires more iterations compared to ILUTP5. A similar situation is also encountered in some other experiments and the high value of *condest* for full pivoting is related to a poor pivoting sequence [13].

(iii) From the results, we also see a strong relationship between *condest* and the usefulness of the preconditioner. When the *condest* value is very high (i.e., higher than 10^5), the iterative method either requires too many iterations or does not converge at all.

Since ILUTP5 is robust for all our geometries, in Table 5.3 we compare it with other commonly used preconditioners. Block-Jacobi preconditioning performs poorer

TABLE 5.1
Information about the open geometries.

Problem	Frequency (MHz)	Size (λ)	n
P1	2,000	2	1,301
P2	6,000	6	12,249
P3	20,000	20	137,792
OC1	313	1.0	1,690
OC2	781	2.6	16,393
OC3	2,370	7.9	171,655
OP1	683	2.3	1,562
OP2	2,270	7.6	14,705
OP3	6,820	22.7	163,871
HS1	750	1.5	1,101
HS2	2,310	4.6	9,911
HS3	7,890	15.8	116,596

TABLE 5.2
ILU results for open geometries.

		ILU(0)		ILUT		ILUTP5		ILUTP	
Problem	n	*condest*	*iter*	*condest*	*iter*	*condest*	*iter*	*condest*	*iter*
P1	1,301	189	37	83	22	73	21	69	21
P2	12,249	60,855	228	712	42	309	39	5,606	56
P3	137,792	6.3E+09	-	1,398	82	1,350	81	2,545	78
OC1	1,690	59	76	14	37	12	35	11	33
OC2	16,393	2,154	333	52	110	48	109	44	109
OC3	171,655	9.6E+05	-	192	377	240	376	2,892	376
OP1	1,562	198	65	41	27	50	26	151	39
OP2	14,705	1.3E+05	416	161	98	164	92	151	91
OP3	163,871	5.3E+05	-	948	268	835	253	2,424	251
HS1	1,101	47	48	26	26	31	24	27	23
HS2	9,911	990	248	1,095	73	126	46	95	45
HS3	116,596	6.3E+05	-	1.5E+15	-	582	110	22,755	156

than the no-preconditioner case, so the Jacobi preconditioner is used instead. We emphasize the following observations:

(i) Although we use a robust solver, for a simple preconditioner such as Jacobi, either the number of iterations turns out to be very high or convergence is not attained in 1,500 iterations. This is in good agreement with the conclusions derived in section 4.

(ii) ILUTP5 reduces iteration numbers by an order of magnitude compared to the Jacobi preconditioner. Moreover, the iteration numbers of ILUTP5 are not extremely higher than those of LU, indicating that the ILUTP5 preconditioners provide good approximations to the near-field matrices.

(iii) We see that the setup cost of SAI is prohibitively large, proving its inappropriateness for sequential implementations. Moreover, except for OC3, ILUTP5 yields fewer number of iterations compared to SAI.

(iv) Furthermore, the iteration counts reveal that the algebraic scalability of

TABLE 5.3
Comparison of preconditioners for open geometries.

		LU	Jacobi		ILUTP5			SAI		
Problem	n	*iter*	*iter*	*time*	*iter*	*setup*	*time*	*iter*	*setup*	*time*
P1	1,301	15	201	15	21	1	3	25	101	103
P2	12,249	26	431	503	39	33	88	45	1,524	1,573
P3	137,792	53	833	16,209	81	661	2,167	92	19,955	21,384
OC1	1,690	28	224	26	35	4	9	35	569	574
OC2	16,393	97	617	854	109	141	273	114	15,040	15,167
OC3	171,655	332	-	-	376	2,243	9,833	354	207,436	213,619
OP1	1,562	18	315	39	26	2	5	48	663	668
OP2	14,705	78	991	1,894	92	97	224	173	12,301	12,524
OP3	163,871	195	-	-	253	996	6,883	396	57,606	66,093
HS1	1,101	17	187	15	24	3	5	26	165	167
HS2	9,911	38	490	748	46	107	186	61	1,712	1,813
HS3	116,596	93	1052	25,947	110	1,353	3,579	156	22,079	25,066

ILUTP5 is favorable for open geometries. For two orders of increase in the number of unknowns, the iteration numbers increase approximately four times for patch and half sphere, and 10 times for the open cube and open prism.

5.2. Closed geometries. As mentioned in section 2, both EFIE and CFIE can be used for closed geometries. However, CFIE yields better-conditioned systems, so it is usually preferred over EFIE. Nonetheless, we will present some of the results obtained from EFIE for comparison purposes.

Figure 5.2 shows the model problems that we consider for the numerical experiments. These include two canonical geometries, i.e., a sphere (S) and a cube (C); two quasi-canonical geometries, i.e., a thin box (TB) and a wing (W); and two real-life problems, i.e., a helicopter (H) and Flamme (F), which is a stealth target [20]. Table 5.4 presents the operating frequency and the size of the geometries in terms of the wavelength. For Flamme and the helicopter, "Size" denotes the length of the objects in longitudinal direction.

Due to the well-conditioning of CFIE, ILU(0) is expected to be free from instability problems. In Table 5.5, we compare ILU(0) and ILUT by presenting the *condest* values and corresponding number of iterations for the systems obtained with CFIE. Pivoting does not change the iteration counts, and hence is not included in this case.

We note that ILU(0) and ILUT produce very similar preconditioners for CFIE. This is also observed for regular problems arising from the discretization of partial differential equations [35]. Since ILU(0) has a lower computational cost and is easier to implement compared to ILUT, we conclude that ILU(0) is the most appropriate choice among ILU-class preconditioners for CFIE.

When we use EFIE with closed geometries, it becomes even more difficult to solve the linear systems. Table 5.6 shows the [illegible] values and iteration numbers for ILU-class preconditioners. W3 does not converge in 1,500 iterations, and for H2 memory limitation is exceeded during the iterations. All other problems converge with ILUTP5, but with higher iteration counts compared to open geometries.

In Table 5.7, for CFIE, ILU(0) is compared to the block-Jacobi preconditioner (where only the self-interactions of the smallest MLFMA clusters are used in the

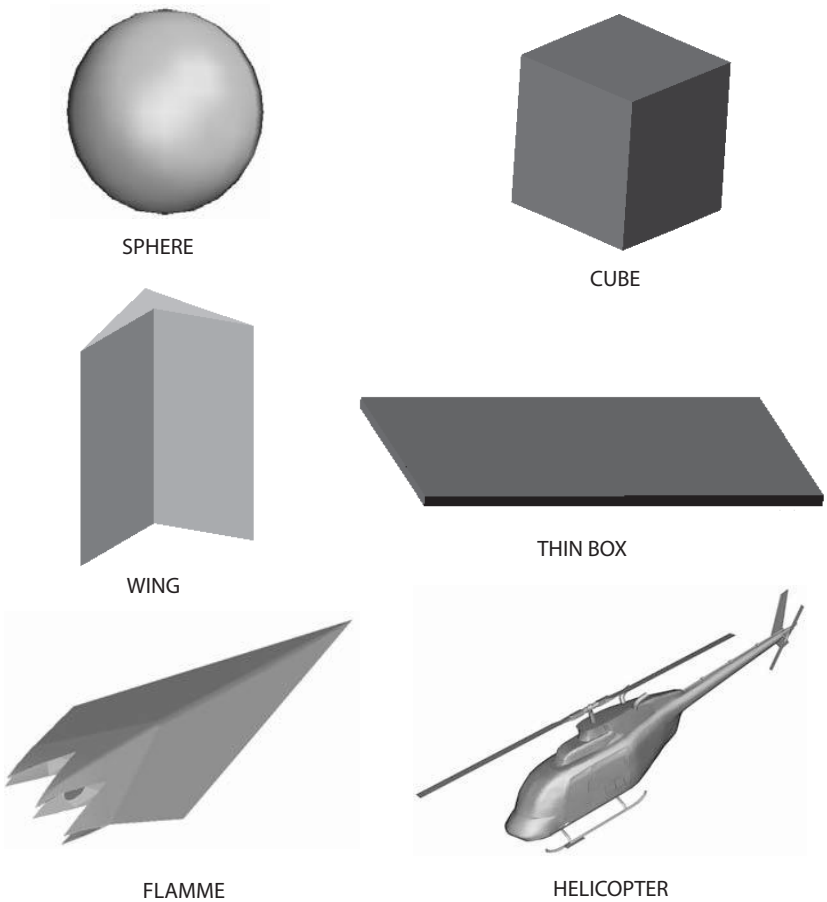


FIG. 5.2. *Closed geometries.*

near-field matrices) and the SAI preconditioner. We summarize the results as follows:

(i) For canonical geometries, compared to the block-Jacobi preconditioner, ILU(0) decreases the iteration numbers slightly. However, due to the larger setup time of ILU(0), total solution times become comparable. For multiple right-hand-side solutions, ILU(0) may be still preferable.

(ii) For quasi-canonical geometries and real-life problems, ILU(0) performs remarkably better compared to the block-Jacobi preconditioner. The number of iterations are halved for the largest problems, and more than halved for smaller sizes. Also, total solution times are significantly smaller.

(iii) Even though SAI has iteration numbers similar to ILU(0), the setup time of SAI is too large.

(iv) ILU(0) iteration numbers are very close to those of LU. Hence, among sequential-algebraic preconditioners for CFIE, ILU(0) emerges as the optimal choice for preconditioning MLFMA in the context of this study.

(v) Finally, even though the near-field matrix becomes sparser as the number of unknowns gets larger, we observe that the algebraic scalability of ILU(0) is surprisingly favorable. For the canonical geometries, as n increases two orders of magnitude, the iteration numbers only double. For quasi-canonical geometries, the iteration numbers

TABLE 5.4
Information about the closed geometries.

Problem	Frequency (MHz)	Size (λ)	n
S1	500	1	930
S2	1,500	3	8,364
S3	6,000	12	132,003
C1	210	0.7	918
C2	600	2.0	8,046
C3	2,410	8.0	131,436
W1	390	1.3	1,050
W2	1,200	4.0	10,512
W3	4,000	13.3	117,945
TB 1	188	1.9	1,650
TB 2	600	6.0	10,122
TB 3	2,400	24.0	147,180
F1	4,000	8	12,750
F2	6,000	12	28,866
F3	10,000	20	78,030
H1	222	9.6	33,423
H2	636	27.6	183,546

TABLE 5.5
ILU results for closed geometries using CFIE.

		ILU(0)		ILUT	
Problem	n	*condest*	*iter*	*condest*	*iter*
S1	930	8	13	3	13
S2	8,364	24	21	9	20
S3	132,003	108	29	108	29
C1	918	3	11	8	12
C2	8,046	9	20	24	20
C3	131,436	34	26	33	26
TB1	1,650	13	11	12	10
TB2	10,122	13	23	14	22
TB3	147,180	97	45	96	42
W1	1,050	8	10	8	9
W2	10,512	26	16	26	15
W3	117,945	83	32	82	32
F1	12,750	174	24	150	23
F2	28,866	198	34	207	33
F3	78,030	327	66	325	65
H1	33,423	6	30	6	30
H2	183,546	18	44	18	44

TABLE 5.6
ILU results for closed geometries using EFIE. "MLE" stands for "memory limitation exceeded."

		ILU(0)		ILUT		ILUTP(0.5)		ILUTP	
Problem	n	*condest*	*iter*	*condest*	*iter*	*condest*	*iter*	*condest*	*iter*
S1	930	65	46	61	37	16	28	20	29
S2	8,364	4.3E+04	416	3.4E+04	246	65	108	427	116
S3	132,003	1.9E+06	-	1.5E+131	-	381	572	1,346	589
C1	918	22	-	9	32	7	30	7	29
C2	8,046	3.1E+05	-	30	77	37	75	99	77
C3	131,436	2.9E+07	-	210	574	181	563	800	557
TB1	1,650	22	37	49	26	72	30	22	27
TB2	10,122	1.8E+05	-	414	169	166	151	7.5E+05	-
TB3	147,180	1.1E+14	-	48,600	1090	13,410	1084	867	709
W1	1,050	128	-	38	23	38	22	43	30
W2	10,512	6.6E+04	-	240	102	88	95	106	91
W3	117,945	4.5E+07	-	538	-	540	-	1,455	-
F1	12,750	4.7E+06	-	1,043	184	623	159	1,006	170
F2	28,866	9.5E+07	-	2,011	421	2,012	393	2,071	440
F3	78,030	1.5E+09	-	2,830	1106	3,066	1042	85,812	1131
H1	33,423	1,359	469	38	206	39	203	59	206
H2	183,546	29,184	MLE	61	MLE	71	MLE	1,799	MLE

TABLE 5.7
Comparisons of preconditioners for closed geometries using CFIE.

		LU	Block-Jacobi		ILU(0)			SAI		
Problem	n	*iter*	*iter*	*time*	*iter*	*setup*	*time*	*iter*	*setup*	*time*
S1	930	13	17	1	13	0	1	14	229	230
S2	8,364	20	23	23	21	6	23	21	1,453	1,474
S3	132,003	29	32	684	29	23	665	29	23,102	23,722
C1	918	11	18	0	11	1	1	12	941	941
C2	8,046	20	25	17	20	2	16	21	2,170	2,184
C3	131,436	26	28	419	26	28	485	27	25,066	25,489
TB1	1,650	9	44	3	11	2	3	20	132	135
TB2	10,122	21	60	40	23	6	22	33	10,468	10,489
TB3	147,180	37	106	1,290	45	271	1,025	64	298,479	299,301
W1	1,050	9	30	1	10	1	1	13	970	971
W2	10,512	15	39	31	16	7	21	21	14,445	14,462
W3	117,945	31	52	779	32	46	542	37	73,100	73,587
F1	12,750	21	77	89	24	11	40	40	22,930	22,976
F2	28,866	32	81	264	34	20	130	45	42,214	42,389
F3	78,030	63	115	1,096	66	43	694	76	96,369	96,369
H1	33,423	30	125	326	30	40	142	51	94,150	94,282
H2	183,546	42	106	3,081	44	145	1,739	61	234,614	236,463

increase by a factor of only 3 or 4. For Flamme, as the number of unknowns increases 15 times, the iteration number only triples. For the helicopter, the increase in the iteration number is 1.4 compared to 5.5 times increase in the number of unknowns.

6. Conclusion. For iterative solvers, ILU-class preconditioners have been intensively studied and widely used. However, potential instability is still a shortcoming that reduces their reliability. We show that this drawback can be eliminated when it occurs, and ILU-class preconditioners can be safely applied to CEM problems employing MLFMA.

For open geometries, EFIE is the only choice of formulation. For the resulting systems, ILUT works remarkably well (about 10 times faster than Jacobi and with disproportionately lower setup time compared to SAI), but sometimes incomplete factors turn out to be unstable. We show that this situation can be handled by pivoting without incurring significant CPU costs; 0.5 pivoting tolerance gives the best results. We also show that the *condest* value is a strong indicator of the resulting preconditioner. Hence, considering the extra cost of pivoting (though not very significant), we propose the following strategy for the solution of problems involving open geometries. Before the iterations begin, compute *condest* for ILUT. If the condition estimate is not high (such as less than 10^4), use ILUT as the preconditioner. Otherwise, switch to ILUTP5. With this strategy, we have obtained robust and effective preconditioners for all our test problems.

CFIE can be used for closed geometries and yields linear systems that are well-conditioned. ILU(0) and ILUT produce very similar factorizations, and therefore cheaper ILU(0) should be preferred. With ILU(0), overall solution times have been decreased by at least one-half compared to the commonly used block-Jacobi preconditioner for real-life problems. Iteration numbers obtained with ILU(0) are very close to those of the exact solution of the near-field matrix, showing that ILU(0) is the optimum preconditioner in the context of this study. Though EFIE can also be used with closed geometries, it becomes harder to obtain fast convergence even with the exact solution of the near-field matrix.

REFERENCES

[1] G. Alléon, M. Benzi, and L. Giraud, *Sparse approximate inverse preconditioning for dense linear systems arising in computational electromagnetics*, Numer. Algorithms, 16 (1997), pp. 1–15.

[2] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, *PETSc Users Manual*, Technical report ANL-95/11, Revision 2.1.5, Argonne National Laboratory, Argonne, IL, 2004.

[3] M. Benzi, *Preconditioning techniques for large linear systems: A survey*, J. Comput. Phys., 182 (2002), pp. 418–477.

[4] M. Benzi, D. B. Szyld, and A. van Duin, *Orderings for incomplete factorization preconditioning of nonsymmetric problems*, SIAM J. Sci. Comput., 20 (1999), pp. 1652–1670.

[5] M. Bollhöfer, *A robust and efficient ILU that incorporates the growth of the inverse triangular factors*, SIAM J. Sci. Comput., 25 (2003), pp. 86–103.

[6] M. Bollhöfer and Y. Saad, *ILUPACK—Preconditioning Software Package*, 2004; available online from http://www.math.tu-berlin.de/ilupack/.

[7] B. Carpentieri, *Sparse Preconditioners for Dense Complex Linear Systems in Electromagnetic Applications*, Ph.D. thesis, Institut National Polytechnique de Toulouse, Toulouse, France, 2002.

[8] B. Carpentieri, I. S. Duff, and L. Giraud, *Experiments with Sparse Preconditioning of Dense Problems from Electromagnetic Applications*, Technical report TR/PA/00/04, CERFACS, Toulouse, France, 1999.

[9] B. Carpentieri, I. S. Duff, and L. Giraud, *Robust preconditioning of dense problems from electromagnetics*, in Revised Papers from the Second International Conference on

Numerical Analysis and Its Applications, L. Vulkov, J. Waśniewski, and P. Yalamov, eds., Springer, London, 2000, pp. 170–178.

[10] B. Carpentieri, I. S. Duff, and L. Giraud, *Sparse pattern selection strategies for robust Frobenius-norm minimization preconditioners in electromagnetism*, Numer. Linear Algebra Appl., 7 (2000), pp. 667–685.

[11] B. Carpentieri, I. S. Duff, L. Giraud, and M. M. Made, *Sparse symmetric preconditioners for dense linear systems in electromagnetism*, Numer. Linear Algebra Appl., 11 (2004), pp. 753–771.

[12] W. C. Chew, J.-M. Jin, E. Michielssen, and J. Song, eds., *Fast and Efficient Algorithms in Computational Electromagnetics*, Artech House, Norwood, MA, 2001.

[13] E. Chow and Y. Saad, *Experimental study of ILU preconditioners for indefinite matrices*, J. Comput. Appl. Math., 86 (1997), pp. 387–414.

[14] Ö. Ergül and L. Gürel, *Improved testing of the magnetic-field integral equation*, IEEE Microwave Wireless Comp. Lett., 15 (2005), pp. 615–617.

[15] Ö. Ergül and L. Gürel, *Enhancing the accuracy of the interpolations and anterpolations in MLFMA*, IEEE Trans. Antennas Propagat. Lett., 5 (2006), pp. 467–470.

[16] Ö. Ergül and L. Gürel, *Improving the accuracy of the magnetic-field integral equation with the linear-linear basis functions*, Radio Sci., 41 (2006), article RS4004 (doi: 10.1029/2005RS003307).

[17] Ö. Ergül and L. Gürel, *Optimal interpolation of translation operator in multilevel fast multipole algorithm*, IEEE Trans. Antennas and Propagation, 54 (2006), pp. 3822–3826.

[18] Ö. Ergül and L. Gürel, *The use of curl-conforming basis functions for the magnetic-field integral equation*, IEEE Trans. Antennas and Propagation, 54 (2006), pp. 1917–1926.

[19] Ö. Ergül, *Fast Multipole Method for the Solution of Electromagnetic Scattering Problems*, Master's thesis, Bilkent University, Ankara, Turkey, 2003.

[20] L. Gürel, H. Bağci, J.-C. Castelli, A. Cheraly, and F. Tardivel, *Validation through comparison: Measurement and calculation of the bistatic radar cross section of a stealth target*, Radio Science, 38 (2003), pp. 1046–1058.

[21] L. Gürel and Ö. Ergül, *Comparisons of FMM implementations employing different formulations and iterative solvers*, in Proceedings of the IEEE Antennas and Propagation Society International Symposium, Vol. 1, 2003, pp. 19–22.

[22] L. Gürel and Ö. Ergül, *Singularity of the magnetic-field integral equation and its extraction*, IEEE Antennas Wireless Propagat. Lett., 4 (2005), pp. 229–232.

[23] L. Gürel and Ö. Ergül, *Extending the applicability of the combined-field integral equation to geometries containing open surfaces*, IEEE Antennas Wireless Propagat. Lett., 5 (2006), pp. 515–516.

[24] M. A. Heroux and J. M. Willenbring, *Trilinos Users Guide*, Technical report SAND2003-2952, Sandia National Laboratories, Albuquerque, NM, 2003.

[25] I. C. F. Ipsen and C. D. Meyer, *The idea behind Krylov methods*, Amer. Math. Monthly, 105 (1998), pp. 889–899.

[26] J. Lee, J. Zhang, and C.-C. Lu, *Incomplete LU preconditioning for large scale dense complex linear systems from electromagnetic wave scattering problems*, J. Comput. Phys., 185 (2003), pp. 158–175.

[27] J. Lee, J. Zhang, and C.-C. Lu, *Sparse inverse preconditioning of multilevel fast multipole algorithm for hybrid integral equations in electromagnetics*, IEEE Trans. Antennas and Propagation, 52 (2004), pp. 2277–2287.

[28] G. Meurant, *Computer Solution of Large Linear Systems*, Stud. Math. Appl. 28, North-Holland, Amsterdam, 1999.

[29] S. M. Rao, D. R. Wilton, and A. W. Glisson, *Electromagnetic scattering by surfaces of arbitrary shape*, IEEE Trans. Antennas and Propagation, 30 (1982), pp. 409–418.

[30] Y. Saad, *ILUT: A dual threshold incomplete LU factorization*, Numer. Linear Algebra Appl., 1 (1994), pp. 387–402.

[31] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.

[32] K. Sertel and J. L. Volakis, *Incomplete LU preconditioner for FMM implementation*, Microw. Opt. Tech. Lett., 28 (2000), pp. 265–267.

[33] L. N. Trefethen, *Computation of pseudospectra*, Acta Numer., 8 (1999), pp. 247–295.

[34] L. N. Trefethen and D. Bau, III, *Numerical Linear Algebra*, SIAM, Philadelphia, 1997.

[35] H. A. van der Vorst, *Iterative Krylov Methods for Large Linear Systems*, Cambridge Monogr. Appl. Comput. Math. 13, Cambridge University Press, Cambridge, UK, 2003.

SIAM J. SCI. COMPUT.
Vol. 29, No. 4, pp. 1495–1510
© 2007 Society for Industrial and Applied Mathematics

GENERAL TOOTH BOUNDARY CONDITIONS FOR EQUATION FREE MODELING*

A. J. ROBERTS† AND I. G. KEVREKIDIS‡

Abstract. We are developing a framework for multiscale computation which enables models at a "microscopic" level of description, for example, lattice Boltzmann, Monte Carlo, or molecular dynamics simulators, to perform modeling tasks at "macroscopic" length scales of interest. The plan is to use the microscopic rules restricted to small "patches" of the domain, the "teeth," using interpolation to bridge the "gaps." Here we explore general boundary conditions coupling the widely separated "teeth" of the microscopic simulation that achieve high order accuracy over the macroscale. We present the simplest case when the microscopic simulator is the quintessential example of a PDE. We argue that classic high order interpolation of the macroscopic field provides the correct forcing in whatever boundary condition is required by the microsimulator. Such interpolation leads to tooth boundary conditions, which achieve arbitrarily high order consistency. The high order consistency is demonstrated on a class of linear PDEs in two ways: first through the eigenvalues of the scheme for selected numerical problems, and second using the dynamical systems approach of holistic discretization on a general class of linear PDEs. Analytic modeling shows that, for a wide class of microscopic systems, the subgrid fields and the effective macroscopic model are largely independent of the tooth size and the particular tooth boundary conditions. When applied to patches of microscopic simulations these tooth boundary conditions promise efficient macroscale simulation. We expect the same approach will also accurately couple patch simulations in higher spatial dimensions.

Key words. multiscale computation, gap-tooth scheme, coupling boundary conditions, high order consistency

AMS subject classifications. 65M20, 65M60, 37M05, 37L10

DOI. 10.1137/060654554

1. Introduction.

The components of physical systems often operate on vastly different space and time scales [5]. We must somehow simulate such systems on the scale of interest and operation. But systems that depend on physical processes at multiple scales pose notorious difficulties. These multiscale difficulties are major obstacles to progress in fields as diverse as environmental and geosciences, climate, materials, combustion, high energy density physics, fusion, bioscience, chemistry, power grids, and information networks [5].

Here we further develop the equation free approach to multiscale modeling [10]. Given a numerical simulator for physical components at much smaller scales than the scale of primary interest, the aim of the methodology is to bridge the space and time scales to simulations resolving the macroscale of interest. Here we focus on bridging [illegible] by improving the accuracy of the gap-tooth methodology for microsimulators [7, 20, 21]. Crucially, our gap-tooth methods must adapt to whatever microsimulator code is provided; one key application of this work is to microsimulators that are tried and tested legacy codes that we do not want to modify.

*Received by the editors March 17, 2006; accepted for publication (in revised form) January 16, 2007; published electronically June 29, 2007.

http://www.siam.org/journals/sisc/29-4/65455.html

†Computational Engineering and Sciences Research Centre, Department of Mathematics and Computing, University of Southern Queensland, Toowoomba, Queensland 4352, Australia (aroberts@usq.edu.au, http://www.sci.usq.edu.au/staff/robertsa).

‡Program in Applied and Computational Mathematics, Princeton University, Princeton, NJ 08544 (yannis@Princeton.edu). The work of this author was supported in part by DARPA and the US DOE.

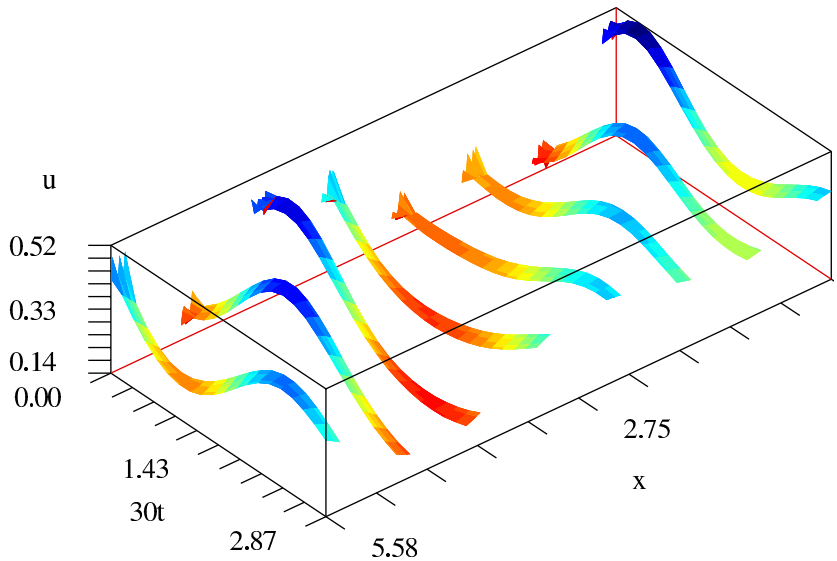


FIG. 1. *Simulation of the Burgers equation using Dirichlet boundary conditions on the teeth (specified u on the edges).*

The equation free approach provides [illegible] closure methods which constitute critical components of, for example, mathematical homogenization (e.g., [21, 8, 1]), renormalization group techniques (e.g., [6, 13, 4]), and multiscale finite elements (e.g., [9, 3]). These closure methods not only need to be computationally efficient but also need to be capable of reproducing the physical dynamics with high fidelity. Our approach avoids inefficiencies in the oversampling methods through using correct coupling conditions, to some order of accuracy, between the macroscale separated teeth. That is, we seek a methodology that can be systematically refined.

Using microscopic simulators of the one dimensional Burgers equation, Roberts and Kevrekidis [19] demonstrated the possibility of achieving high order accuracy in the gap-tooth scheme for macroscale dynamics. The particular microsimulator we use is a fine scale discretization of the PDE which we execute only in the interior of the teeth (see Figure 1). At each time step during execution, the microsimulator within each tooth requires boundary values which must be continuously updated. If the microsimulator was to be executed over the entire macrodomain, these boundary values would naturally come from the immediately neighboring fine grid; this grid is missing in gap-tooth simulation. That pilot study considered only microsimulators which had boundary conditions of specified [illegible] at the edges of their simulation teeth. Here we generalize the analysis to consider microsimulators with

- Dirichlet boundary conditions of specified field u at the tooth edges (section 2),
- mixed boundary conditions of specified $av_j \pm b\partial_x v_j$ at the tooth edges (section 3), or
- nonlocal two-point boundary conditions such as those arising in a microscale discretization of a PDE (section 4).

Consider the gap-tooth scheme (e.g., [7, 20]) illustrated in Figure 1. Let $v_j(x,t)$ be the fine scale, microscopic field in the jth tooth and U_j the jth coarse grid value, that is, the value at the center of each tooth. Let the tooth width be h. Then the edge of

a tooth lies at a distance $h/2$ from its coarse grid point, a fraction $r = h/(2H)$ to the next coarse grid point. The amount of computation performed by microsimulators is proportional to the width of the (microscale) teeth. Hence we aim for the fraction r to be as small as possible so that the teeth are a relatively small part of the physical domain and the computational cost is minimized; we call r the length scale ratio as a small r corresponds to a large disparity in the microscopic and macroscopic length scales. The coupling rule developed in sections 2, 3, and 4 is that you [illegible] macroscopic [illegible]. As a nonlinear example of the coupling we develop, Figure 1 shows a gap-tooth simulation of the nonlinear dynamics of the Burgers equation in one spatial dimension.

This coupling rule promotes a strong connection between classic finite difference discretizations of PDEs, classic finite elements, and the methodology of the gap-tooth scheme. First, the PDE, or a very fine discretization, acts as the quintessential example of a microsimulator in that it informs us of the dynamics in an "infinitesimal patch." The only difference between the gap-tooth scheme and the spatial discretization of PDEs is that the microsimulators in the gap-tooth scheme encode the dynamics on [illegible], whereas the PDE encodes the dynamics on [illegible] patches. Consequently, classic interpolation serves the same role in both: namely, the interpolation appropriately transfers information from the macroscale of interest to the microscale simulators. One result is that our approach reproduces classic finite difference models when applied to the limit of a microsimulator becoming an algebraic PDE. Second, the theoretical support for the gap-tooth scheme is based upon a subgrid scale structure, as is the classic finite element method. Also, the algebraic solvability condition in the construction of the theoretical gap-tooth model is similar to the Galerkin projection of finite elements. But whereas finite elements [illegible] a class of subgrid fields, both the theoretical approach here and the gap-tooth scheme use actual subgrid scale dynamics, obtained from the microsimulator or the PDE, to [illegible] appropriate subgrid scale fields on demand. Thus this approach systematically implements model closures for macroscale discretizations.

In section 5 we prove that classic interpolation accurately connects the teeth across the gaps for the general [illegible] fourth order PDE. The technique of holistic discretization (e.g., [17]) resolves subgrid scale structures to reproduce with high fidelity the dynamics of specified PDEs [18]. The techniques were adapted by Roberts and Kevrekidis [19] to the gap-tooth scheme in the case where the microsimulator requires Neumann boundary conditions of specified slope/flux (see section 3). Using the same techniques, section 5 analyzes the use of classic interpolation of macroscale grid values in the microscale simulators and shows the following desirable properties:

- the approach generates macroscopic discretizations which are consistent with the microscopic dynamics to high order in the macroscopic tooth separation H;
- the macroscopic model and the microscopic solution field are essentially independent of the size of the teeth, measured by r; and
- the macroscopic model and the microscopic solution field are essentially independent of the details of the tooth boundary conditions (TBCs) that couple the teeth together.

Thus our proposed rule generates gap-tooth schemes that may be systematically refined to high order accuracy and gives rise to macroscale simulations that are largely independent of physically irrelevant microscale parameters. Using such high order ac-

curate schemes will reduce errors on the macroscale. Since the microscale boundary forcing is determined from the macroscale coupling, reducing errors on the macroscale will consequently reduce the errors on the microscale.

Our analysis here is only for patches in one spatial dimension. MacKenzie [12] shows that similar coupling of PDEs generates holistic discretizations in two space dimensions. We expect that the coupling here can be straightforwardly generalized to spatial problems in higher dimensions.

2. Dirichlet teeth (specified u). In this section we consider the case of microsimulators that require at each time step the field values on the edge of each spatial patch to be specified. We model this case by PDEs with Dirichlet conditions coupling the dynamics in the teeth. The values for these Dirichlet conditions are obtained by interpolation across the gaps between the teeth using finite difference operators and the exact relationships between the operators. When applied to simple diffusion, the resulting scheme has high order accuracy.

Discrete operators are essential in the analysis. Define the shift operator $Eu(x) = u(x+H)$ and equivalently $EU_j = U_{j+1}$ as appropriate for steps on the coarse grid size H. Then we use the following identities for discrete operators (e.g., [14, p. 65]):

$$\text{mean} \quad \mu = \tfrac{1}{2}(E^{1/2} + E^{-1/2})\,, \tag{1}$$

$$\text{difference} \quad \delta = E^{1/2} - E^{-1/2}\,, \tag{2}$$

$$\text{shift} \quad E = 1 + \mu\delta + \tfrac{1}{2}\delta^2\,, \tag{3}$$

$$\text{derivative} \quad H\partial_x = 2\sinh^{-1}\tfrac{1}{2}\delta = \delta - \tfrac{1}{6}\delta^3 + \mathcal{O}\big(\delta^5\big)\,, \tag{4}$$

$$\mu^2 = 1 + \tfrac{1}{4}\delta^2\,. \tag{5}$$

Formulae involving these operators become more accurate as the differences δ become small. Such small differences arise either as the macroscopic grid size $H \to 0$ or equivalently as the gradients of the physical field u become small.

For example, Roberts and Kevrekidis [19] showed arbitrary order consistent macroscopic dynamics from a gap-tooth scheme as the grid size $H \to 0$. The key to that analysis is the following transformation of the operator for evaluating spatial derivatives $H\partial_x$ at the two patch boundaries $E^{\pm r}$:

$$E^{\pm r}H\partial_x = (1 + \mu\delta + \tfrac{1}{2}\delta^2)^{\pm r} 2\sinh^{-1}\tfrac{1}{2}\delta\,.$$

But this right-hand side, when expanded in a Taylor series in small differences δ, is composed of terms which have an odd number of centered operators δ and μ. Consequently the right-hand side above would require field values halfway between the grid values. These are unknown. Instead, from (5), multiply the right-hand side

by the identity $\mu/\sqrt{1+\delta^2/4}$, and then expand in small differences δ:

$$\begin{aligned}
E^{\pm r}H\partial_x &= (1+\mu\delta+\tfrac{1}{2}\delta^2)^{\pm r}2\sinh^{-1}\tfrac{1}{2}\delta \\
&= \frac{\mu}{\sqrt{1+\frac{1}{4}\delta^2}}(1+\mu\delta+\tfrac{1}{2}\delta^2)^{\pm r}2\sinh^{-1}\tfrac{1}{2}\delta \\
&= \mu\delta \pm r\delta^2 - (\tfrac{1}{6}-\tfrac{1}{2}r^2)\mu\delta^3 \mp r(\tfrac{1}{12}-\tfrac{1}{6}r^2)\delta^4 \\
&\quad + (\tfrac{1}{30}-\tfrac{1}{8}r^2+\tfrac{1}{24}r^4)\mu\delta^5 \pm r(\tfrac{1}{90}-\tfrac{1}{36}r^2+\tfrac{1}{120}r^4)\delta^6 \\
&\quad - (\tfrac{1}{140}-\tfrac{7}{240}r^2+\tfrac{1}{72}r^4-\tfrac{1}{720}r^6)\mu\delta^7 \\
&\quad \mp r(\tfrac{1}{560}-\tfrac{7}{1440}r^2+\tfrac{1}{480}r^4-\tfrac{1}{5040}r^6)\delta^8 + \mathcal{O}\big(\delta^9\big). \qquad (6)
\end{aligned}$$

For microsimulators with Dirichlet boundary conditions, we adapt the earlier analysis of Roberts and Kevrekidis [19]. But instead of determining the slopes at the tooth boundaries as above, the following interpolation of the macroscopic field determines the field values u on the edges of the teeth:

$$\begin{aligned}
E^{\pm r} &= (1+\mu\delta+\tfrac{1}{2}\delta^2)^{\pm r} \\
&= 1 \pm r\mu\delta + \tfrac{1}{2}r^2\delta^2 \pm \tfrac{1}{3!}r(r^2-1)\mu\delta^3 + \tfrac{1}{4!}r^2(r^2-1)\delta^4 \\
&\quad \pm \tfrac{1}{5!}r(r^2-1)(r^2-4)\mu\delta^5 + \tfrac{1}{6!}r^2(r^2-1)(r^2-4)\delta^6 \\
&\quad \pm \tfrac{1}{7!}r(r^2-1)(r^2-4)(r^2-9)\mu\delta^7 \\
&\quad + \tfrac{1}{8!}r^2(r^2-1)(r^2-4)(r^2-9)\delta^8 + \mathcal{O}\big(\delta^9\big). \qquad (7)
\end{aligned}$$

The pattern in the above interpolation formula is clear. Now we explore the numerical performance of a gap-tooth scheme using this formula to determine teeth boundary conditions.

Consider gap-tooth simulations of the simple diffusion equation

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \quad \text{and } 2\pi\text{-periodic in } x. \qquad (8)$$

Imagine we have access to the dynamics only through a microscopic simulator of the diffusion (8), here coded by a fine discretization on n grid points, spaced a distance $\eta = h/(n-1)$ apart, across a tooth of microscopic width $h = 2rH$. The time integration is an explicit scheme with a microscopic time step, typically $\Delta t = 10^{-6}$–10^{-4}. Figure 2 shows an example of the initially rapid microscopic evolution within one tooth; the microsimulator, coupled to its neighbors, rapidly evolves to a smooth state. Figure 3 similarly shows the initial evolution in two neighboring teeth and how the smooth subgrid field arises through the coupling to the neighboring teeth. Similar dynamics take place during the initial instants of the Burgers evolution shown in Figure 1.

First, we implement the following TBC. On the edge of the jth tooth, at $x = X_j \pm rH$, the boundary condition of the fine discretization is that the field

$$v_j = \big[1 \pm r\mu\delta + \tfrac{1}{2}r^2\delta^2 \pm \tfrac{1}{6}r(r^2-1)\mu\delta^3 + \tfrac{1}{24}r^2(r^2-1)\delta^4\big]\,U_j\,. \qquad (9)$$

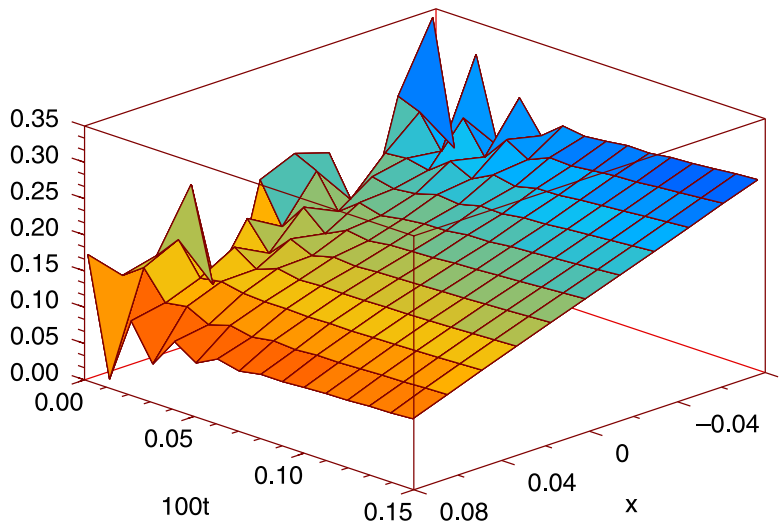


FIG. 2. *View of the initial microscopic evolution within a tooth with dynamics described by the diffusion PDE* (8) *and coupled to its neighbors.*

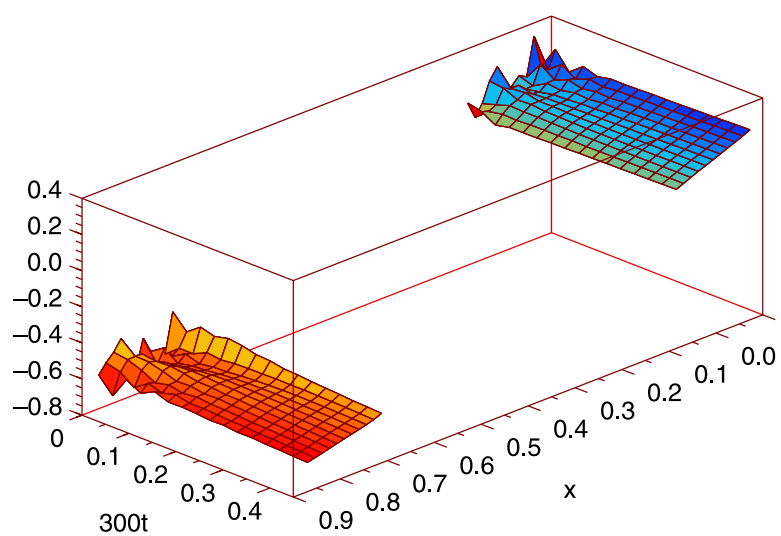


FIG. 3. *View of the initial microscopic evolution within a pair of neighboring teeth with dynamics described by the diffusion PDE* (8) *and also coupled to their neighbors.*

TABLE 1

Growth rates λ of perturbations from steady state $u = 0$: For diffusion (8) *with m teeth, $H = 2\pi/m$; with length scale ratio $r = 0.1$; $n = 11$ points in the microscale grid; and with the fourth order TBC* (9).

m	1	2,3	4,5	6,7	$m+1:2m$
4	$6 \cdot 10^{-12}$	−0.946256	−2.166285	n/a	−397.2
8	$-3 \cdot 10^{-12}$	−0.996073	−3.785024	−7.121435	−1588.
16	$-1 \cdot 10^{-10}$	−0.999750	−3.984293	−8.832102	−6355.
32	0	−0.999986	−3.998999	−8.988613	−25421.

TABLE 2

Growth rates λ of perturbations from steady state $u = 0$: For diffusion (8) with m teeth, $H = 2\pi/m$; with length scale ratio $r = 0.1$; $n = 11$ points in the microscale grid; and with the sixth order TBC.

m	1	2,3	4,5	6,7	$m+1:2m$
4	$-5\cdot10^{-12}$	-0.981981	-2.453767	n/a	-397.2
8	$1\cdot10^{-11}$	-0.999653	-3.927925	-7.835158	$-1588.$
16	$8\cdot10^{-11}$	-1.000001	-3.998611	-8.966332	$-6355.$
32	$8\cdot10^{-10}$	-1.000002	-4.000004	-8.999518	$-25421.$

The first few terms of (7) provide this patch boundary condition by interpolation from the surrounding coarse grid values. For the jth tooth this TBC involves macroscopic grid values $U_{j-2},\ldots,U_{j+2}$ only, and thus we should be able to achieve $\mathcal{O}(H^4)$ consistency with the microsimulator. We numerically linearize the map over one microscopic time step by systematically perturbing each and every microscopic value from zero (there are mn such microscopic values, one for each of n fine grid points in each of m teeth). We then transform the eigenvalues μ of this map to growth rates $\lambda = \log(\mu)/\Delta t$. The mn growth rates fall into n groups of m modes. Each group corresponds to a microscopic [illegible] of the dynamics; the mode is essentially the same in each tooth. Large negative growth rates correspond to rapidly decaying internal modes with significant microscopic structure within each tooth. The group of m modes with [illegible] growth rates corresponds to the relatively slowly evolving [illegible] modes of interest that arise through the coupling of the microscopic dynamics across the teeth. Table 1 shows the leading seven growth rates, and the magnitude of the leading internal growth rate, for various numbers of teeth, $m = 4, 8, 16, 32$. The exact growth rates of the diffusion PDE (8) are $\lambda = -k^2$ for integer k. The table shows that as the number of teeth doubles, the accuracy of the growth rates of the macroscopic modes improves by a factor of about 16. This is consistent with an $\mathcal{O}(H^4)$ method, as predicted for diffusion with TBC (9).

Table 2 shows the even higher order accuracy from implementing sixth order TBCs from (7)—growth rates slightly larger than the ideal seem to be due to the relatively small number of microscopic grid points within the teeth. These sixth order TBCs are used in the simulations of the nonlinear Burgers equation shown in Figure 1. This simulation suggests that gap-tooth schemes employing such TBCs even for nonlinear systems are effective.

3. Mixed boundary conditions for the teeth. Let us explore mixed boundary conditions at the edges of the teeth: suppose the microsimulator requires $av_j \pm b\partial_x v_j$ specified on the edge of the teeth $x = x_j \pm rH$ for some constants a and b. The case $a = 1$ and $b = 0$ constitutes Dirichlet TBCs discussed in the previous section. The case $a = 0$ and $b = 1$ constitutes Neumann TBCs as discussed by Roberts and Kevrekidis [19]: there we used the interpolation formula (6) to specify slopes/fluxes on the edge of each tooth; we obtained spectra of accuracy similar to those in Tables 1 and 2.

For mixed TBCs we propose to simply combine (6) and (7) to give, for example, the fourth order in macroscopic grid size $H = 2\pi/m$ boundary condition

$$\begin{aligned} &av_j \pm b\partial_x v_j \\ &= a\left[1 \pm r\mu\delta + \tfrac{1}{2}r^2\delta^2 \pm \tfrac{1}{6}r(r^2-1)\mu\delta^3 + \tfrac{1}{24}r^2(r^2-1)\delta^4\right]U_j \\ &\pm \frac{b}{H}\left[\mu\delta \pm r\delta^2 - (\tfrac{1}{6}-\tfrac{1}{2}r^2)\mu\delta^3 \mp r(\tfrac{1}{12}-\tfrac{1}{6}r^2)\delta^4\right]U_j \\ &\text{on } x = x_j \pm rH\,. \end{aligned} \tag{10}$$

TABLE 3
Growth rates λ of perturbations from steady state $u = 0$: For diffusion (8) with m teeth, $H = 2\pi/m$; with length scale ratio $r = 0.1$; $n = 11$ points in the microscale grid; and with the mixed TBC (10) with $a = 0.95$ and $b = 0.05$.

m	1	2,3	4,5	6,7	$m+1:2m$
4	$9\cdot10^{-12}$	-0.939448	-2.151772	n/a	-240.7
8	$-2\cdot10^{-11}$	-0.990854	-3.766007	-7.089004	-756.9
16	$7\cdot10^{-11}$	-0.996405	-3.971027	-8.803476	$-2417.$
32	$6\cdot10^{-10}$	-0.998047	-3.991243	-8.971213	$-8153.$

TABLE 4
Growth rates λ of perturbations from steady state $u = 0$: For diffusion (8) with $m = 8$ teeth; with length scale ratio $r = 0.1$; n points in the microscale grid to show variation with microscale resolution; and with the mixed TBC (10) with $a = 0.95$ and $b = 0.05$.

n	1	2,3	4,5	6,7
11	$-2\cdot10^{-11}$	-0.990854	-3.766007	-7.089004
21	$1\cdot10^{-10}$	-0.994896	-3.781289	-7.117588
41	$-3\cdot10^{-10}$	-0.995792	-3.784677	-7.123924

We use $a = 0.95$ and $b = 0.05$ in the mixed TBC: this gives a mixed boundary condition where the effects of the function value v_j and its gradient $\partial_x v_j$ are roughly comparable in the TBC (if the parameter b is significantly larger, then the gradient term dominates the TBC). The numerical eigenvalues given in Table 3 for the diffusion equation (8) with these TBCs again show convergence to the correct eigenvalues as the number of teeth increases, that is, as the macroscopic grid size $H \to 0$. However, the convergence is not as rapid as for Dirichlet TBCs. The poorer convergence as $H \to 0$ seems to be due to the microscopic grid resolution: successive doubling of the number of interior points (see Table 4) demonstrates that there are significant errors of $\mathcal{O}(\eta^2)$ in the microscale grid size η. Thus the total error in this implementation of the mixed TBCs seems to be $\mathcal{O}(H^4, \eta^2)$. This example illustrates that generally there are two types of errors: the error in the microscale simulator, here $\mathcal{O}(\eta^2)$, and the error due to the multiscale coupling, here $\mathcal{O}(H^4)$. But, recall that we adopt the policy that we cannot change the microscale simulator as it is a legacy code handed to us from past development. Thus the errors our methodology seeks to control are those due to the multiscale coupling.

Here the microscale simulation is that of a fine discretization of a PDE. Thus the derivatives in the mixed TBC (10) are subject to the significant errors of numerical differentiation when computed on the microscale. As Table 4 shows, the approximation of derivatives does incur errors; we would be better off without such errors. Higher order formulae for microscale interpolation would reduce the microscale errors in the boundary derivatives, perhaps from $\mathcal{O}(\eta^2)$ to $\mathcal{O}(\eta^4)$, but would ruin the small bandwidth of the microscale simulator. Again, recall that we adopt the policy that we cannot change the microscale simulator. We cannot (do not want to) change the nature nor accuracy of its boundary conditions. Consequently we proceed to address the problem of supplying boundary conditions at the edge of the teeth, [illegible] of the legacy microscale simulator.

4. Teeth with two-point boundary conditions. A microscale simulator may have implemented boundary conditions that do not fit into the classic PDE form of Dirichlet, Neumann, nor mixed Dirichlet–Neumann. Here the microscale simulator

TABLE 5

Growth rates λ of perturbations from steady state $u = 0$: For diffusion (8) with $m = 8$ teeth; with length scale ratio $r = 0.1$; n points in the microscale grid to show variation with microscale resolution $\eta \propto 1/n$; and with the fourth order general TBC (12) with $\beta = 1$.

n	1	2,3	4,5	6,7
11	$1 \cdot 10^{-10}$	-0.999741	-3.984137	-8.831209
21	$-3 \cdot 10^{-10}$	-0.999742	-3.984159	-8.831368
41	$1 \cdot 10^{-9}$	-0.999742	-3.984169	-8.831453

implements a discretization of three-point stencil width. Consequently the simulator has been written so that the supplied boundary conditions depend only upon each of the two extreme pairs of points in each tooth. We thus investigate TBCs that specify a combination of these two point values of the field at the edge of each tooth. This specific case is just one example of the wide range of possible nonlocal TBCs that specific microsimulators may require.

Suppose the microsimulator, here a fine spatial discretization of the diffusion PDE (8), implements a TBC of the (linear) form

$$v_{j,1} + \beta v_{j,2} \quad \text{and} \quad \beta v_{j,n-1} + v_{j,n} \quad \text{are specified,} \tag{11}$$

where $v_{j,i}$ denotes the microscale field value at the ith microscale grid point in the jth tooth. For example, the case $\beta = 1$ approximates Dirichlet boundary conditions at the microscale grid midpoints $x_{j,3/2}$ and $x_{j,n-1/2}$ near the edges of each tooth, which is exactly the case $a = 0$ and $b = 1$ implemented in section 3. Different values of β would [illegible] different mixed boundary conditions of the previous section.

The procedure is straightforward: we interpolate the macroscale grid values to find the specific values required by the boundary conditions (11). Recall that (9) gives a fourth order interpolation from the macroscale grid to points at the tooth boundaries $x = X_j \pm rH$; this gives appropriate values for $v_{j,1}$ and $v_{j,n}$. Get appropriate values for $v_{j,2}$ and $v_{j,n-1}$ through simply replacing in the formula the length scale ratio $r = h/(2H)$ by the ratio required to reach the penultimate microgrid point, namely, $r' = (h/2 - \eta)/H$, where $\eta = h/(n-1)$ is the microgrid size. Thus the fourth order version of the boundary condition (11) is that at $x = X_j \pm rH$

$$\begin{aligned} &(1 + \beta E^{\mp \eta/H}) v_j \\ &= \left[1 \pm r\mu\delta + \tfrac{1}{2} r^2 \delta^2 \pm \tfrac{1}{6} r(r^2 - 1)\mu\delta^3 + \tfrac{1}{24} r^2 (r^2 - 1)\delta^4\right] U_j \\ &\quad + \beta \left[1 \pm r'\mu\delta + \tfrac{1}{2} r'^2 \delta^2 \pm \tfrac{1}{6} r'(r'^2 - 1)\mu\delta^3 + \tfrac{1}{24} r'^2 (r'^2 - 1)\delta^4\right] U_j \,. \end{aligned} \tag{12}$$

Implementing the TBC (12) for the diffusion equation (8) gives a numerical approximation scheme with eigenvalues shown in Table 5 for varying microgrid resolution. See that there is only an extremely weak dependence upon the microgrid size η. [illegible]

Last, Table 6 shows the eigenvalues of the gap-tooth scheme for varying numbers m of teeth in the domain. Note that the eigenvalues converge to their correct values like $\mathcal{O}(H^4)$ as expected by the construction.

Higher order TBCs, in the macroscopic grid size H, would similarly be based upon the expansion (7). We then expect even more rapid convergence as the macroscale grid size $H \to 0$.

TABLE 6

Growth rates λ of perturbations from steady state $u = 0$: For diffusion (8) with m teeth, $H = 2\pi/m$; with length scale ratio $r = 0.1$; $n = 11$ points in the microscale grid; and with the fourth order general TBC (12) with $\beta = 1$.

m	1	2,3	4,5	6,7	$m+1:2m$
4	$-8\cdot10^{-12}$	-0.946069	-2.165068	n/a	-489.5
8	$4\cdot10^{-11}$	-0.996034	-3.784277	-7.118312	$-1958.$
16	$1\cdot10^{-10}$	-0.999741	-3.984137	-8.831209	$-7832.$
32	$8\cdot10^{-10}$	-0.999983	-3.998964	-8.988427	$-31329.$

5. The model is independent of the TBCs. Here we use analytic methods of holistic discretization (e.g., [17]) to explore the gap-tooth scheme on a general class of PDEs with general mixed boundary conditions. The analysis establishes three important properties:

- the approach generates macroscopic models which are consistent with the microscopic dynamics to high orders in grid spacing H;
- the macroscopic model and the microscopic solution field are essentially independent of the size of the teeth, as parametrized by r; and
- the macroscopic model and the microscopic solution field are essentially independent of the details of the TBCs.

5.1. Theory underpins analysis of a PDE with TBCs. We explore solutions of the class of [illegible] hyper–advection-diffusion PDEs

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} - \epsilon\left(c\frac{\partial u}{\partial x} + b\frac{\partial^3 u}{\partial x^3} + a\frac{\partial^4 u}{\partial x^4}\right), \tag{13}$$

where a, b, and c are arbitrary parameters, and where ϵ is introduced as a convenient mechanism to control truncation of the multivariate power series solutions in the parameters a, b, and c. This PDE is solved with mixed TBCs inspired by (10), namely, that on $x = x_j \pm rH$, and in terms of an artificial parameter γ that we explain shortly,

$$\begin{aligned}
&\pm\alpha v_j + \partial_x v_j \\
&= \pm\alpha\left\{1 + \gamma r\left[\pm\mu\delta + \tfrac{1}{2}r\delta^2\right] + \gamma^2\tfrac{1}{6}r(r^2-1)\left[\pm\mu\delta^3 + \tfrac{1}{4}r\delta^4\right]\right. \\
&\quad \left. + \gamma^3\tfrac{1}{120}r(r^2-1)(r^2-4)\left[\pm\mu\delta^5 + \tfrac{1}{6}r\delta^6\right]\right\}U_j \\
&+ \frac{1}{H}\left\{\gamma\left[\mu\delta \pm r\delta^2\right] + \gamma^2\left[-(\tfrac{1}{6}-\tfrac{1}{2}r^2)\mu\delta^3 \mp r(\tfrac{1}{12}-\tfrac{1}{6}r^2)\delta^4\right]\right. \\
&\quad \left. + \gamma^3\left[(\tfrac{1}{30}-\tfrac{1}{8}r^2+\tfrac{1}{24}r^4)\mu\delta^5 \pm r(\tfrac{1}{90}-\tfrac{1}{36}r^2+\tfrac{1}{120}r^4)\delta^6\right]\right\}U_j \\
&+\mathcal{O}\left(\gamma^4\right).
\end{aligned} \tag{14}$$

Explore the structure of this complicated looking TBC: $\pm\alpha v_j + \partial_x v_j$ represents a general linear combination of the microscopic field at the edge of each tooth that needs to be specified for the microscopic simulator; those terms in the right-hand side multiplied by $\pm\alpha$ form the estimate of the field v_j interpolated from the surrounding macroscopic grid values; those terms in the right-hand side multiplied by $1/H$ form the estimate of the field's gradient $\partial_x v_j$ interpolated from the surrounding macroscopic grid values. However, these two interpolations hold only when the artificial parameter

$\gamma = 1$; the value 1 is the physically interesting case. Why then do we introduce the parameter γ? The reason is that, as in "discretization" [16], based around the special values of the parameters $\gamma = \alpha = \epsilon = 0$, the general PDE (13) with TBC (14) possesses a slow center manifold parametrized by the macroscopic grid values. On this slow manifold the evolution of these macroscopic grid values forms a macroscale model of the PDE. This model has rigorous theoretical support based upon $\gamma = 0$, and it becomes physically relevant when evaluated at $\gamma = 1$.

We briefly explain how center manifold theory underpins the macroscale model. Initially set $\gamma = \alpha = \epsilon = 0$; then the PDE + TBC become the diffusion equation with [illegible] boundaries at the edges of the teeth, $x = x_j \pm rH$. Thus, exponentially quickly, all structure within each tooth diffuses away to become constant—but a different constant for each tooth depending upon the initial conditions. See a similar evolution in Figures 2 and 3; but there the teeth are coupled so that the rapid evolution is to a smooth variation in each tooth, whereas here the insulated evolution, $\gamma = \alpha = \epsilon = 0$, is to a constant in each tooth. But we are interested in only fully coupled teeth for which $\gamma = 1$, and in nonzero α and ϵ. Thus from the simple base of piecewise constant fields, we construct a description of the field u and its slow evolution as a power series in the "perturbations" measured by γ, α, and ϵ. The departure of the field u from a constant within each tooth gives the microscopic (subgrid, subtooth) field, as shown, for example, in the smooth fields of Figures 2 and 3 that are quickly established. The slow evolution of the coarse grid values U_j gives the macroscopic model.

The various powers of γ in the TBC (14) are chosen so that truncation of the expressions to errors $\mathcal{O}(\gamma^p)$ will generate a discrete macroscopic model expressing the time derivative $\dot{U}_j$ in terms of only $U_{j-p+1}, \ldots, U_{j+p-1}$ (a spatial stencil of width $2p-1$). Center manifold theory (e.g., [2, 11]) asserts that

- such a model exists,
- through its exponential attractiveness, the model is relevant in some finite neighborhood of $\gamma = \alpha = \epsilon = 0$, and
- we may systematically construct the power series approximation to the model.

Because truncation to errors $\mathcal{O}(\gamma^p)$ results in a model with stencil width $2p-1$, [illegible] $2p-1$ [illegible] $U_{j-p+1}, \ldots, U_{j+p-1}$

Computer algebra[1] performs all the tedious details of constructing the model [15]. We seek a model where the subtooth/subgrid field

$$u(x,t) = v_j(x, \boldsymbol{U}; \alpha, \gamma, \epsilon)\,. \tag{15}$$

That is, the subtooth field has some spatial structure, such as that in Figures 2 and 3, which depends upon the neighboring grid values $U_{j-p+1}, \ldots, U_{j+p-1}$; may depend upon the specific TBC through its parameter α; depends upon the specific PDE though its parameter ϵ; and depends upon the coupling parameter γ. Center manifold theory ensures that the evolution of the system is governed by the evolution of the grid values:

$$\dot{U}_j = g_j(\boldsymbol{U}; \alpha, \gamma, \epsilon); \tag{16}$$

this formula is the macroscopic (closed) discretization. We solve by iteration the PDE (13) with TBC (14) to find the center manifold (15) and its associated coarse

[1] http://www.sci.usq.edu.au/staff/aroberts/CA/burgermixed.red is the source script which was available at the time of writing.

discretization (16). The results are expressions for the microscopic fields v_j and the macroscopic evolution, $\dot{U}_j = g_j$, that are accurate to some specified order in the small parameters α, γ, and ϵ.

5.2. Modeling $\mathcal{O}(\epsilon)$ changes to the PDE. The subgrid fields in each tooth will in general depend upon the coefficients a, b, and c that determine the PDE. For example, there are nontrivial dependencies upon the advection speed that ensure that the macroscale model naturally transforms to an upwind discretization for large advection speeds c [18]—such influences show up in the $\mathcal{O}(\epsilon^2)$ terms that we explore in the next subsection. Here we first explore the models linear in a, b, and c, that is, linear in the general changes to the PDE (13) with the TBC (14).

For example, to errors $\mathcal{O}(\alpha^3, \gamma^4, \epsilon^2)$, computer algebra generates the macroscopic evolution

$$\begin{aligned}\dot{U}_j = {} & \frac{1}{H^2}\left[\gamma\delta^2 - \tfrac{1}{12}\gamma^2\delta^4 + \tfrac{1}{90}\gamma^3\delta^6\right] U_j \\ & - \frac{\epsilon c}{H}\left[\gamma\mu\delta - \tfrac{1}{6}\gamma^2\mu\delta^3 + \tfrac{1}{30}\gamma^3\mu\delta^5\right] U_j \\ & - \frac{\epsilon b}{H^3}\left[\gamma^2\mu\delta^3 - \tfrac{1}{4}\gamma^3\mu\delta^5\right] U_j \\ & - \frac{\epsilon a}{H^4}\left[\gamma^2\delta^4 - \tfrac{1}{6}\gamma^3\delta^6\right] U_j + \mathcal{O}(\alpha^3, \gamma^4, \epsilon^2)\,. \end{aligned} \tag{17}$$

When evaluated at the physically relevant parameter $\gamma = 1$, these are the classical finite difference operators for the PDE (13), truncated to $\mathcal{O}(\delta^7)$. Consequently the terms in the macroscopic model (17) are [illegible] with the PDE (13) to various orders in the macroscopic grid size H. The order of consistency depends upon the order of truncation in the artificial coupling parameter γ and the order of the derivatives in each term. [illegible] independent [illegible] r [illegible] independent [illegible] α [illegible] (14)

Now we explore the microscopic field within the teeth. To low order in the coupling parameter γ and in terms of the microscopic tooth space variable $\xi = (x - X_j)/H$, we find

$$\begin{aligned} v_j = {} & U_j + \gamma\left\{\left[\xi\mu\delta + \tfrac{1}{2}\xi^2\delta^2\right]\right. \\ & \left. + \epsilon c H\left[(\tfrac{1}{6}\xi^3 - \tfrac{1}{2}r^2\xi) + \tfrac{1}{3}H\alpha r^3\xi - \tfrac{1}{3}H^2\alpha^2 r^4\xi\right]\delta^2\right\} U_j \\ & + \mathcal{O}(\alpha^3, \gamma^2, \epsilon^2)\,. \end{aligned} \tag{18}$$

The first line gives the classic quadratic interpolation through the grid values U_j and $U_{j\pm1}$. The second line shows microscopic field structure in the advection speed c. But it exhibits undesirable dependence upon the tooth width r and nature α of the TBC.

However, inspect the next order terms in coupling parameter γ:

$$
\begin{aligned}
v_j = \cdots + \gamma^2 \Bigg\{ & \left[\tfrac{1}{6}(-\xi+\xi^3)\mu\delta^3 + \tfrac{1}{24}(-\xi^2+\xi^4)\delta^4\right] \\
& + \epsilon c H \left[-(\tfrac{1}{6}\xi^3 - \tfrac{1}{2}r^2\xi) - \tfrac{1}{3}H\alpha r^3\xi + \tfrac{1}{3}H^2\alpha^2 r^4\xi\right]\delta^2 \\
& + (-\tfrac{1}{18}\xi^3 + \tfrac{1}{60}\xi^5 + \tfrac{1}{6}r^2\xi - \tfrac{1}{12}r^2\xi^3\tfrac{1}{6}r^4\xi)\delta^4 \\
& + \alpha H(-\tfrac{1}{9}r^3\xi + \tfrac{1}{18}r^3\xi^3 - \tfrac{4}{15}r^5\xi)\delta^4 \\
& + \alpha^2 H^2(\tfrac{1}{9}r^4\xi - \tfrac{1}{18}r^4\xi^3 + \tfrac{17}{45}r^6\xi)\delta^4 \\
& + \frac{\epsilon b}{H}\left[-(\tfrac{1}{6}\xi^3 - \tfrac{1}{2}r^2\xi) - \tfrac{1}{3}H\alpha r^3\xi + \tfrac{1}{3}H^2\alpha^2 r^4\xi\right]\delta^4 \Bigg\} U_j \\
& + \mathcal{O}\big(\alpha^3, \gamma^3, \epsilon^2\big). \qquad (19)
\end{aligned}
$$

The dots denote the terms given in the right-hand side of (18). The first line in the above higher order terms reassuringly contains the classic quartic interpolation formulae. The second line, when we set $\gamma = 1$, cancels all the undesirable α and r dependence in the lower order (18). The third and later lines above describe higher order microscopic structure; some of this undesirably depends upon the TBC through α and the size of the teeth through r, but as far as we have explored, any α and r dependence introduced at any order in γ is canceled by terms at higher orders in γ. Thus any finite truncation of the power series expansion for the model may have undesirable dependence upon α and r, but as the order in the artificial parameter γ is increased, this dependence is removed. In this sense, the microscopic field is "essentially" independent of the details of the TBC and independent of the tooth width r.

5.3. Modeling $\mathcal{O}(\epsilon^2)$ effects in the PDE. In our analysis we find that even order operators in the microscale PDE, such as the diffusion u_{xx} and the hyperdiffusion au_{xxxx}, are represented simply in the macroscale discretization. However, odd order operators in the microscale PDE, such as advection cu_x and the dispersion bu_{xxx}, create nontrivial effects; these first show up in terms quadratic in their amplitude, and hence they first appear in terms of $\mathcal{O}(\epsilon^2)$. We now show that the effects of $\mathcal{O}(\epsilon^2)$ terms typically act to stabilize the $\mathcal{O}(\epsilon)$ discrete model. The implication for the gap-tooth method is that the resolution of the subgrid structures by the microscale simulator will also typically maintain stability of the discrete macroscale model. That is, the microscale simulator will provide successful closure for the macroscale discretization when coupled with the proposed TBCs.

To illustrate this, we constructed the approximate model of the PDE (13) with TBC (14) to errors $\mathcal{O}(\alpha^2, \gamma^4, \epsilon^3)$; that is, we include quadratic effects in the coefficients a, b, and c. The details of the model are too long to record here. However, we find that the equivalent PDE to the macroscale discrete model (17) with its $\mathcal{O}(\epsilon^2)$ modifications

is

$$\begin{aligned}
\frac{\partial u}{\partial t} &= \gamma\frac{\partial^2 u}{\partial x^2} - \epsilon\left[\gamma c\frac{\partial u}{\partial x} + \gamma^2 b\frac{\partial^3 u}{\partial x^3} + \gamma^2 a\frac{\partial^4 u}{\partial x^4}\right] \\
&+ \epsilon^2\left[-(\gamma-\gamma^2)bc\frac{\partial^2 u}{\partial x^2} - (\gamma^2-\gamma^3)b^2\frac{\partial^4 u}{\partial x^4}\right] \\
&+ H^2\left\{\tfrac{1}{12}(\gamma-\gamma^2)\frac{\partial^4 u}{\partial x^4}\right. \\
&\quad + \epsilon\left[-\tfrac{1}{6}(\gamma-\gamma^2)c\frac{\partial^3 u}{\partial x^3} - \tfrac{1}{4}(\gamma^2-\gamma^3)b\frac{\partial^5 u}{\partial x^5} - \tfrac{1}{6}(\gamma^2-\gamma^3)a\frac{\partial^6 u}{\partial x^6}\right] \\
&\quad + \epsilon^2\left[+\tfrac{1}{3}(\gamma-\gamma^2)c^2r^2\frac{\partial^2 u}{\partial x^2} - \tfrac{1}{12}(\gamma-5\gamma^2+4\gamma^3)bc\frac{\partial^4 u}{\partial x^4}\right. \\
&\qquad \left.\left. + \tfrac{2}{3}(\gamma^2-\gamma^3)bcr^2\frac{\partial^4 u}{\partial x^4} - \tfrac{1}{6}(\gamma^2-7\gamma^3+4\gamma^3r^2)b^2\frac{\partial^6 u}{\partial x^6}\right]\right\} \\
&+ \mathcal{O}\big(H^3,\alpha^2,\gamma^4,\epsilon^3\big)\,. \qquad (20)
\end{aligned}$$

Observe that to this level of accuracy there is no dependence upon the TBC parameter α; thus our comments apply for all the TBCs. Now consider the components of (20) in turn. The first line of (20) is the original general linear PDE (13) when evaluated at the physically meaningful $\gamma = 1$. The second line shows some error terms, quadratic in ϵ, that disappear for $\gamma = 1$. The bcu_{xx} error disappears when $\mathcal{O}(\gamma^2)$ terms are retained, which is as soon as the discretization stencil is wide enough to model the third order dispersion term bu_{xxx}. The b^2u_{xxxx} dispersion-induced term shows that the method initially incorporates its effects as enhanced dissipation, as the coefficient of γ^2 is negative; then, when higher order accuracy is requested by retaining $\mathcal{O}(\gamma^3)$ terms, the method proceeds to remove the incurred error via the γ^3 term.

Consider applying these TBCs to a microscale simulator; for slow enough spatial variations, the microsimulator is equivalent to some "infinite order" PDE. For example, the [illegible] discretization $\dot{u}_i = (1/\eta^2)\delta^2 u_i$ is, by (4), equivalent to the PDE $u_t = (4/\eta^2)\sinh^2(\eta\partial_x/2)u$. We expect the behavior of errors in the gap-tooth scheme seen here, induced by the u_{xxx} and u_{xxxx} terms, to be representative of the behavior of errors in the "high order" equivalent terms of any given microscopic simulator.

The remaining terms in (20) are $\mathcal{O}(H^2)$ and hence vanish as the macroscopic grid size $H \to 0$. Nonetheless look at the H^2 terms in the third to sixth lines as they apply to simulations with finite H. The third and fourth lines show that the initial discretization errors of the linear terms are eliminated, for the physical $\gamma = 1$, via the next higher order in coupling parameter γ. The particular focus of this subsection is on the ϵ^2 terms on the next two lines. The c^2u_{xx} terms show that at low order truncations in γ the method treats advection in a manner that increases dissipation, as the coefficient is positive, and thus helps to maintain the stability of the macroscale discretization to high advection speeds c [16, 18]. The interaction between advection cu_x and dispersion bu_{xxx} can maintain stability or be destabilizing depending upon the sign of bc: whether we truncate at γ when $-bcu_{xx}$ dominates (second line) or we truncate at γ^2 when bcu_{xxxx} dominates (fifth and sixth lines), the

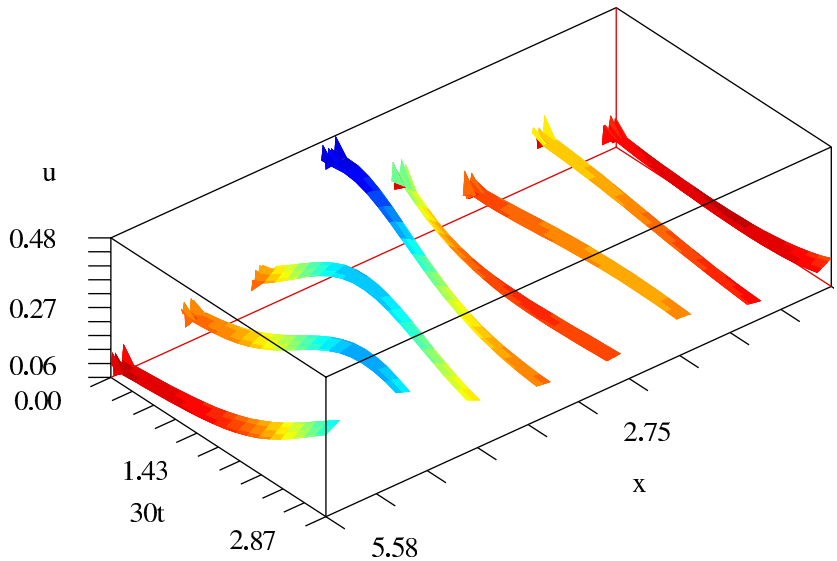


FIG. 4. *Simulation of the Burgers equation using the general* 2*-point boundary condition on the teeth of the fourth order* (12) *with* $\beta = 1$ *demonstrates that the method is stable even for nonlinear PDEs.*

combination is stabilizing whenever $bc < 0$, that is, when the phase velocity of wave-like effects does not change direction as a function of wavenumber. The last term on the sixth line will be dominated by the dissipative $b^2 u_{xxxx}$ term on the second line, and we presume will vanish at higher orders in the coupling parameter γ. Thus, from the equivalent PDE (20) of the macroscale model, we deduce that the subgrid scale interactions between processes in the PDE, and hence for microscale simulators in general, are accounted for in this approach to generate a macroscale model that is typically stable.

Indeed this equivalent PDE (20) confirms support by center manifold theory for the gap-tooth scheme with TBC. Theory asserts that the original system, here the PDE (20), and the slow manifold model, here the macroscale discretization (16), have the same stability. Thus when the microscale system is stable, so will be the macroscale discretization. The caveat is that we can only construct the center manifold approximately; we control the errors to some order in the parameters α, γ, and ϵ, but there will be some error, albeit of high order in the parameters.

6. Conclusion. We use macroscale interpolation based upon the expansions (6) and (7) to determine TBCs for the boundary conditions at the edge of the teeth in the gap-tooth scheme. The interpolation was used to implement whatever boundary conditions are needed by the microscale legacy code during execution. Figure 4 shows a simulation of the nonlinear Burgers equation with two-point boundary conditions at the boundaries of each tooth as an illustrative example. We found that the macroscopic models resulting from the microsimulator and the constructed TBC were consistent, to high order, with the microscopic dynamics and that the macroscopic models and the microscopic (subgrid, subtooth) fields were essentially independent of the tooth size and the detailed nature of the TBC. We expect the same type of TBC to be effective for microsimulations in more than one spatial dimension. Interesting future research would seek TBCs that do not require commu-

nication across the gaps between the teeth at each and every microscale time step and would explore the interplay of TBCs with implicit integration schemes.

Further exciting research would explore issues of existence and performance of TBCs for stochastic microsimulators.

REFERENCES

[1] V. Balakotaiah and H.-C. Chang, *Hyperbolic homogenized models for thermal and solutal dispersion*, SIAM J. Appl. Math., 63 (2003), pp. 1231–1258.

[2] J. Carr, *Applications of Center Manifold Theory*, Applied Math. Sci. 35, Springer-Verlag, New York, 1981.

[3] Z. Chen and T. Y. Hou, *A mixed multiscale finite element method for elliptic problems with oscillating coefficients*, Math. Comp., 72 (2002), pp. 541–576.

[4] A. J. Chorin and P. Stinis, *Problem Reduction, Renormalization, and Memory*, Technical report, http://arXiv.org/abs/math.NA/0503612, 2005.

[5] J. Dolbow, M. A. Khaleel, and J. Mitchell, *Multiscale Mathematics Initiative: A Roadmap. Report from the 3rd DoE Workshop on Multiscale Mathematics*, Technical report, Department of Energy, Washington, DC, http://www.sc.doe.gov/ascr/Research/AM/MultiscaleMathWorkshop3.pdf, 2004.

[6] S.-I. Ei, K. Fujii, and T. Kunihiro, *Renormalization-group method for reduction of evolution equations: Invariant manifolds and envelopes*, Ann. Physics, 280 (2000), pp. 236–298.

[7] C. W. Gear, J. Li, and I. G. Kevrekidis, *The gap-tooth method in particle simulations*, Phys. Lett. A, 316 (2003), pp. 190–195.

[8] B. Gustafsson and J. Mossino, *Non-periodic explicit homogenization and reduction of dimension: The linear case*, IMA J. Appl. Math., 68 (2003), pp. 269–298.

[9] T. Y. Hou and X.-H. Wu, *A multiscale finite element method for elliptic problems in composite materials and porous media*, J. Comput. Phys., 134 (1997), pp. 169–189.

[10] I. G. Kevrekidis, C. W. Gear, J. M. Hyman, P. G. Kevrekidis, O. Runborg, and K. Theodoropoulos, *Equation-free, coarse-grained multiscale computation: Enabling microscopic simulators to perform system level tasks*, Comm. Math. Sci., 1 (2003), pp. 715–762.

[11] Y. A. Kuznetsov, *Elements of Applied Bifurcation Theory*, Appl. Math. Sci. 112, Springer-Verlag, New York, 1995.

[12] T. MacKenzie, *Create Accurate Numerical Models of Complex Spatio-Temporal Dynamical Systems with Holistic Discretisation*, Ph.D. thesis, University of Southern Queensland, Queensland, Australia, 2005.

[13] B. Mudavanhu and R. E. O'Malley, *A new renormalization method for the asymptotic solution of weakly nonlinear vector systems*, University of Washington, Seattle, WA, 2003.

[14] National Physical Laboratory, *Modern Computing Methods*, 2nd ed., Notes on Applied Science 16, Her Majesty's Stationery Office, London, 1961.

[15] A. J. Roberts, *Low-dimensional modeling of dynamics via computer algebra*, Comput. Phys. Comm., 100 (1997), pp. 215–230.

[16] A. J. Roberts, *Holistic discretisation ensures fidelity to Burgers' equation*, Applied Numer. Math., 37 (2001), pp. 371–396.

[17] A. J. Roberts, *Holistic discretization illuminates and enhances the numerical modeling of differential equations*, in Topics in Applied and Theoretical Mathematics and Computer Science, V. V. Kluev and N. E. Mastorakis, eds., WSEAS, Athens, Greece, 2001, pp. 81–89.

[18] A. J. Roberts, *A holistic finite difference approach models linear dynamics consistently*, Math. Comp., 72 (2002), pp. 247–262.

[19] A. J. Roberts and I. G. Kevrekidis, *Higher order accuracy in the gap-tooth scheme for large-scale dynamics using microscopic simulators*, ANZIAM J., 46 (2004/2005), pp. C637–C657.

[20] G. Samaey, I. G. Kevrekidis, and D. Roose, *Damping factors for the gap-tooth scheme*, in Multiscale Modeling and Simulation, S. Attinger and P. Koumoutsakos, eds., Lect. Notes Comput. Sci. Eng. 39, Springer-Verlag, New York, 2004, pp. 93–102.

[21] G. Samaey, D. Roose, and I. G. Kevrekidis, *The gap-tooth scheme for homogenization problems*, Multiscale Model. Simul., 4 (2005), pp. 278–306.

SIAM J. SCI. COMPUT.
Vol. 29, No. 4, pp. 1511–1524
© 2007 Society for Industrial and Applied Mathematics

DUAL FUNCTIONS FOR A PARALLEL ADAPTIVE METHOD*

RANDOLPH E. BANK† AND JEFFREY S. OVALL‡

Abstract. In this paper, we investigate the effects of pollution error on the performance of the parallel adaptive finite element technique proposed by Bank and Holst in 2000 [R. E. Bank and M. Holst, *SIAM J. Sci. Comput.*, 22 (2000), pp. 1411–1443]. In particular, we consider whether the performance of the algorithm as it was originally proposed can be improved through the use of certain dual functions which give an indication of the global influences on subdomain errors.

Key words. dual functions, Bank–Holst algorithm, parallel adaptive grid generation

AMS subject classifications. 65N50, 65N30, 65Y05

DOI. 10.1137/060668304

1. Introduction.

One of the most difficult aspects of adaptive computations in a parallel environment is the issue of load balancing. If an initial mesh is distributed fairly among a number of processors, a good error estimator coupled with adaptive refinement may quickly produce a very bad load imbalance among the processors. A number of static and dynamic load balancing approaches for unstructured meshes have been proposed in the literature [23, 17, 19, 20, 13, 27, 29]. Such approaches tend to be communication intensive, since they must first assess the current state of imbalance among the processors and then perform a redistribution step to equalize the load.

The Bank–Holst adaptive meshing paradigm [9, 10, 8] presented a new way of addressing the issue of load balancing. Their scheme has the additional benefits of keeping communication costs low and using existing sequential adaptive software such as PLTMG without the need for extensive investment in recoding for use in a parallel environment. The paradigm consists of three major steps.

Step 1: Load balancing. We solve a small problem on a coarse mesh and use a posteriori error estimates to partition the mesh into p subdomains—one for each processor. Each subregion has approximately the same error, although subregions may vary considerably in terms of numbers of elements or grid points.

Step 2: Adaptive meshing. Each processor is provided the complete coarse mesh and instructed to sequentially solve the [illegible] problem, with the stipulation that its adaptive refinement should be limited largely to its assigned subdomain. The target number of elements and grid points for each problem is the same. At the end of this step, the global mesh should be regularized so that the global mesh described in Step 3 will be conforming.

Step 3: Global solution. A final mesh is computed using the union of the refined subdomains provided by each processor. A final solution is computed

*Received by the editors August 25, 2006; accepted for publication (in revised form) January 16, 2007; published electronically June 29, 2007.
http://www.siam.org/journals/sisc/29-4/66830.html

†Department of Mathematics, University of California, San Diego, La Jolla, CA 92093-0112 (rbank@ucsd.edu). The work of this author was supported by the National Science Foundation under contract DMS-0511766.

‡Scientific Computation Group, Max Planck Institute for Mathematics in the Sciences, Inselstraße 22-26, D-04103 Leipzig, Germany (ovall@mis.mpg.de). The work of this author was supported by a research fellowship of the Max Planck Society.

using a standard domain decomposition solver or parallel multigrid technique. For the experiments done in this paper, the final global iterative solve is performed by use of the domain decomposition solver described in [6].

Here the load balancing problem is reduced to the numerical solution of a small elliptic problem on a single processor, using a sequential adaptive solver, without requiring any modifications to the sequential solver. The small elliptic problem is used to produce a posteriori error estimates to predict future element densities in the mesh, which are then used to partition the mesh according to equal error. The underlying assumption here is that roughly equal element densities imply roughly equal workload. We note that, although this assumption is seemingly fragile, the procedure does generally produce good workload balance in practice. It is important to emphasize that the adaptive mesh generation in Step 2 is done asynchronously and without communication. The only communication needed prior to the final global solve in Step 3 is the initial fan-out of the coarse mesh at the end of Step 1 and the mesh regularization at the the end of Step 2. (The fan-out could be avoided by having every processor solve the same course problem and compute the same load balance in Step 1.) By providing each processor with the entire problem, information about the global behavior of the solution, which normally would require some communication between processors, is generated independently and without communication on each processor, albeit on a coarse mesh. In effect, extra coarse grid computations are substituted for the interprocessor communication that would otherwise be required. This is likely to be an advantageous trade-off in situations where communication costs are high relative to computation costs, which is a situation typical of small Beowulf clusters.

In early investigations of the algorithm, the issue of restricting refinement on a given processor largely to its own subdomain was addressed by multiplying a posteriori error estimates for elements outside the subdomain by 10^{-6}. This small change allowed sequential adaptive codes (like the PLTMG package used here) to work in parallel without additional changes. This scheme tricks the adaptive refinement routine into thinking errors outside the given subdomain are much smaller than they actually are and hence not good candidates for refinement. Some refinement outside the given processor's subdomain is still required for other reasons, e.g., the shape-regular grading of the mesh from the small elements inside the refined region to the larger elements in the coarse regions.

While initially derived as a simple way to implement and test the concept, the 10^{-6} trick proved to be surprisingly effective for a wide variety of elliptic PDEs. However, because of the important role the course grid plays in the paradigm, it is important to consider heuristics based on more solid mathematical foundations than the 10^{-6} trick to help determine an appropriate refinement of the coarse grid outside the subregion assigned to the processor. It is this point that is the main focus of the present work.

For example, the coarse grid provides information about the effect of singularities outside the given subdomain on the solution inside (so-called pollution effects). See [1, 16, 28] for some discussion of such pollution errors. If convection is present, good information about the solution along the upstream boundaries of the subregion is needed to form an accurate impression of the convective behavior within the subregion. Because of the continuous dependence of the solution of elliptic problems on data throughout the domain, the approximation quality in any given subdomain depends on the approximation quality in the rest of the domain.

Dual functions and goal related adaptivity have a long history in the context of adaptive meshing algorithms [4, 3, 2, 5, 21, 18, 14, 15, 22]. Usually the functional is some quantity of physical interest for a particular application (lift, drag, stress intensity factor, etc.). Our idea here is to adapt this goal oriented adaptive meshing technology to the present situation. In Step 2 of the paradigm, we view each processor as having the "goal" of producing a good mesh in its subregion. Thus the goal related functional in this case is not provided externally by the user but rather is developed internally and independently by the software running on each processor. The dual functions developed on each processor may be quite different from one another, since the subregions assigned to each processor for refinement may require quite different refinement patterns for their respective coarse meshes in order to produce good fine meshes for their assigned subregions. In particular, for this heuristic, we develop functionals intended to express the influence of the solution outside a given processor's subdomain on the solution within. The dual solution, generated by solving a problem with the adjoint operator and given functional as data, should then reflect the influence of the coarse parts of the domain on the processor's own region. The adjoint problem is solved using the same mesh as the primal problem, i.e., fine in its own region and coarse elsewhere. Then values (or local norms) of the dual function are used in place of 10^{-6} to weight the a posteriori error estimates in the coarse subregions, which in turn are used in the adaptive refinement procedure on that processor.

We emphasize that the primary objective of Step 2 in the algorithm is for each processor to generate a well-adapted mesh in its subdomain as part of the global fine mesh. However, a secondary objective is for each processor to contribute to a reasonable initial guess in its subdomain of the solution of the global fine problem in Step 3—this speeds up the convergence of the parallel solver. Clearly these two goals are related, but it is not always necessary to have a good approximate solution in order to generate a well-adapted mesh from the associated error estimates. Indeed, the ability to generate good meshes from relatively inaccurate solutions explains the success of many adaptive methods. However, if the solution quality in a given subdomain is relatively poor and is not improved by further refinement in that subdomain, then the quality of the initial guess for Step 3 will be diminished. This is another motivation for investigating the use of dual weighted error estimates, the topic of section 3, which might better indicate global influences on subdomain error.

The rest of this manuscript is organized as follows. In section 2, we establish notation and assumptions for this study. In section 3, we develop the functionals and corresponding dual solutions suitable for our parallel adaptive meshing algorithm. Finally, in section 4, we present some numerical examples.

2. Notation and assumptions. Let $\Omega \subset \mathbb{R}^2$ be a bounded domain with Lipschitz boundary $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$, and define

$$\mathcal{H} \equiv \left\{ v \in H^1(\Omega) : v|_{\partial\Omega_D} = 0 \text{ in the trace sense} \right\}. \tag{2.1}$$

For simplicity, we will assume that $\partial\Omega$ is a polygon. The usual spaces $W_p^k(\Omega)$ and $H^k(\Omega) \equiv W_2^k(\Omega)$ are equipped with norms $\|\cdot\|_{k,p,\Omega}$ and $\|\cdot\|_{k,\Omega} \equiv \|\cdot\|_{k,2,\Omega}$, respectively. Let data functions $a : \bar{\Omega} \to \mathbb{R}^{2\times 2}$, $\mathbf{b} : \bar{\Omega} \to \mathbb{R}^2$, $c, f : \bar{\Omega} \to \mathbb{R}$, and $g : \partial\Omega_N \to \mathbb{R}$ be

given. The primal problem is to find $u \in \mathcal{H}$ such that

$$B(u,v) = F(v) \text{ for all } v \in \mathcal{H}, \tag{2.2}$$

$$B(u,v) \equiv \int_\Omega a\nabla u \cdot \nabla v + (\mathbf{b}\cdot\nabla u + cu)v\,dx, \tag{2.3}$$

$$F(v) \equiv \int_\Omega fv\,dx + \int_{\partial\Omega_N} gv\,dS. \tag{2.4}$$

We assume that the data functions are smooth and that the matrix a is positive-definite, with the smallest eigenvalue bounded below on Ω by some constant $\gamma > 0$. We make the following standard boundedness and coercivity assumptions for the bilinear form B and linear functional F: There exist constants $\alpha, \nu, \mu > 0$ such that, for all $v, w \in \mathcal{H}$,

$$|F(v)| \le \alpha\|v\|_{1,\Omega},$$

$$|B(v,w)| \le \nu\|v\|_{1,\Omega}\|w\|_{1,\Omega},$$

$$B(v,v) \ge \mu\|v\|^2_{1,\Omega}.$$

Let $\mathcal{T}_h$ denote a shape-regular triangulation of Ω with mesh size $h \in (0,1)$. Let $V_h \subset \mathcal{H}$ denote the space of continuous, piecewise-linear polynomials defined on $\mathcal{T}_h$, and let $\bar{V}_h \subset \mathcal{H}$ denote the continuous, piecewise-quadratic polynomials. Let $u_\ell \in V_h$ and $u_q \in \bar{V}_h$ denote piecewise-linear and -quadratic interpolants of u on $\mathcal{T}_h$. We make the following standard assumptions about their asymptotic approximation quality:

$$\|u - u_\ell\|_{k,\Omega} \lesssim h^{2-k}\|u\|_{2,\Omega}, \tag{2.5}$$

$$\|u - u_q\|_{k,\Omega} \lesssim h^{3-k}\|u\|_{3,\Omega} \tag{2.6}$$

for $0 \le k \le 1$.

In this work we assume the existence of a gradient recovery operator $\mathcal{R}\nabla u_h \in V_h \times V_h$ possessing certain superconvergence properties. Let $u \in H^3(\Omega) \cap W^{2,\infty}(\Omega)$, and let $u_h \in V_h$ be any approximation of u satisfying

$$\|u - u_h\|_{k,\Omega} \lesssim h^{2-k}|u|_{2,\Omega}, \quad k = 0, 1, \tag{2.7}$$

$$\|u - u_h\|_{0,\infty,\Omega} \lesssim h^2|\log h||u|_{2,\infty,\Omega}. \tag{2.8}$$

Then we assume

$$\|\nabla u - \mathcal{R}\nabla u_h\|_{0,\Omega} \lesssim h\delta\left(\|u\|_{3,\Omega} + |u|_{2,\infty,\Omega}\right), \tag{2.9}$$

where $\delta \to 0$ as $h \to 0$.

Some recovery procedures that satisfy bounds like (2.9) are described in [11, 12]. It is important to note that u_h needs only to satisfy the required approximation properties in order for (2.9) or Lemma 2.1 to apply; in particular, there is no requirement that u_h be an approximate solution to (2.2)–(2.4). That stated, for the rest of this work we will use $u_h \in V_h$ to denote the finite element solution of the variational problem

$$B(u_h, v) = F(v) \text{ for all } v \in V_h. \tag{2.10}$$

For tables and figures, we will use the abbreviation $e_h = u - u_h$ to denote the error in the finite element approximation.

Although the details of $\mathcal{R}$ are not needed in the analysis here, for completeness we briefly summarize the scheme analyzed in [12]. In this case

$$\mathcal{R}\nabla u_h = S^m Q_h \nabla u_h,$$

where Q_h is the operator corresponding to the L^2 projection onto V_h, and $S : V_h \to V_h$ is a smoothing operator. In other words, the discontinuous, piecewise constant gradient ∇u_h is projected into the space of continuous piecewise-linear polynomials and then smoothed with a multigrid-like smoothing operator. Typically $m = 1$ or $m = 2$ smoothing steps are used.

Given a superconvergent gradient approximation, one can form $\mathcal{R}\nabla u_h - \nabla u_h$ to approximate the error $\nabla u - \nabla u_h$. In many cases we can expect such an estimate to be asymptotically exact.

LEMMA 2.1. [illegible] $u \in H^3(\Omega) \cap W^{2,\infty}(\Omega)$ [illegible] $u_h \in V_h$ [illegible] (2.7)–(2.8), [illegible]

$$\|\nabla u - \nabla u_h\|_{0,\Omega} \geq c_0 h$$

[illegible] $c_0 > 0$ [illegible]

$$\lim_{h \to 0} \frac{\|(\mathcal{R} - I)\nabla u_h\|_{0,\Omega}}{\|\nabla u - \nabla u_h\|_{0,\Omega}} = 1.$$

[illegible]. It holds that

$$\left| \frac{\|(\mathcal{R} - I)\nabla u_h\|_{0,\Omega}}{\|\nabla u - \nabla u_h\|_{0,\Omega}} - 1 \right| \leq \frac{\|\mathcal{R}\nabla u_h - \nabla u\|_{0,\Omega}}{\|\nabla u - \nabla u_h\|_{0,\Omega}} \lesssim \delta \,,$$

and $\delta \to 0$ as $h \to 0$. □

The local error indicators $\|(\mathcal{R} - I)\nabla u_h\|_{0,\tau}$ form the basis of the standard adaptive refinement algorithm for PLTMG [7], the PDE software used for the experiments in this work.

3. Duality and global influence on local error. In many applications, it is not the PDE solution u which is of primary interest, but rather some functional $G(u)$, where $G \in \mathcal{H}^*$. The examples often given in the literature are related to physical quantities of interest which can be computed from u, such as the lift or drag on a wing of an airplane during cruising conditions (see, for example, [25]). The use of appropriate dual (adjoint) problems for the accurate approximation of functional quantities and for driving adaptive algorithms (goal oriented adaptive refinement) has been an active area of research for some time [21, 22, 25, 26, 24]. The sort of functionals that are of interest to us in the context of the Bank–Holst paradigm are those where we can use the dual solution to extract information concerning outside influences on the solution u within a given subdomain. We include some of the basic theory below for completeness before presenting the dual error estimator used in this work.

Given a functional G, we define the related dual problem

$$B^*(\omega, v) = G(v) \text{ for all } v \in \mathcal{H}, \tag{3.1}$$

where $B^*(w,v) \equiv B(v,w)$. The key relation between the solutions of the primal and dual problems is that

$$G(u-u_h) = B^*(\omega, u-u_h) \tag{3.2}$$

$$= B(u-u_h, \omega) \tag{3.3}$$

$$= F(\omega) - B(u_h, \omega). \tag{3.4}$$

In this context, ω provides a weighting of the global error (3.2), (3.3) or of the global residual (3.4) that indicates the influence of global approximation errors on the subdomain error $G(u-u_h)$.

Because we cannot generally compute ω, we must approximate $G(u-u_h)$ using one of the expressions (3.2), (3.3), (3.4). For the computations in this paper, we use the dual error estimate weighting technique (DEW) introduced in [24]. In order to make this paper more self-contained, we briefly present the pertinent details below.

Let $\omega_h \in V_h$ be the piecewise-linear finite element solution of the dual problem

$$B^*(\omega_h, v) = G(v) \text{ for all } v \in V_h. \tag{3.5}$$

Noting that

$$B(u-u_h,\omega) = B(u-u_h, \omega-\omega_h) = \int_\Omega a\nabla(u-u_h)\cdot\nabla(\omega-\omega_h)\,dx + \mathcal{O}(h^3),$$

we define the approximator $\mathcal{G}_h \approx G(u-u_h)$ by

$$\mathcal{G}_h = \int_\Omega a(\mathcal{R}-I)\nabla u_h \cdot (\mathcal{R}-I)\nabla\omega_h\,dx. \tag{3.6}$$

We bound the approximation error, $|G(u-u_h)-\mathcal{G}_h|$, in the following theorem.

THEOREM 3.1. [illegible] $u, \omega \in H^3(\Omega)\cap W^{2,\infty}(\Omega)$ [illegible] $u_h, \omega_h \in V_h$ [illegible] (2.7))–((2.8), [illegible]

$$|G(u-u_h)-\mathcal{G}_h| \lesssim h^2\delta\left(\|u\|_{3,\Omega} + |u|_{2,\infty,\Omega}\right)\left(\|\omega\|_{3,\Omega} + |\omega|_{2,\infty,\Omega}\right), \tag{3.7}$$

[illegible] $\delta \to 0$ [illegible] $h \to 0$ [illegible]. We have, by the triangle and Cauchy–Schwarz inequalities,

$$\begin{aligned}|G(u-u_h)-\mathcal{G}_h| \lesssim{}& \|\nabla u - \mathcal{R}\nabla u_h\|_{0,\Omega}\|\nabla(\omega-\omega_h)\|_{0,\Omega} \\ &+ \|(\mathcal{R}-I)\nabla u_h\|_{0,\Omega}\|\nabla\omega - \mathcal{R}\nabla\omega_h\|_{0,\Omega} + \mathcal{O}(h^3).\end{aligned}$$

We have the bounds

$$\|\nabla u - \mathcal{R}\nabla u_h\|_{0,\Omega} \lesssim h\delta\left(\|u\|_{3,\Omega} + |u|_{2,\infty,\Omega}\right),$$

$$\|(\mathcal{R}-I)\nabla u_h\|_{0,\Omega} \lesssim h\left(\|u\|_{3,\Omega} + |u|_{2,\infty,\Omega}\right),$$

$$\|\nabla\omega - \mathcal{R}\nabla\omega_h\|_{0,\Omega} \lesssim h\delta\left(\|\omega\|_{3,\Omega} + |\omega|_{2,\infty,\Omega}\right),$$

$$\|\nabla(\omega-\omega_h)\|_{0,\Omega} \lesssim h\|\omega\|_{2,\Omega}.$$

Combining these four estimates completes the proof. ∎

We have the following immediate corollary.

COROLLARY 3.2. [illegible] 3.1, [illegible] $c_0 > 0$ [illegible] $|G(u-u_h)| \geq c_0 h^2$ [illegible] $\mathcal{G}_h$ [illegible] $G(u-u_h)$

$$\frac{\mathcal{G}_h}{G(u-u_h)} \to 1.$$

[illegible]. It holds that

$$\left|1 - \frac{\mathcal{G}_h}{G(u-u_h)}\right| = \left|\frac{G(u-u_h) - \mathcal{G}_h}{G(u-u_h)}\right| \lesssim \delta \left(\|u\|_{3,\Omega} + |u|_{2,\infty,\Omega}\right)\left(\|\omega\|_{3,\Omega} + |\omega|_{2,\infty,\Omega}\right),$$

which implies the conclusion. □

We motivate our local error indicators for duality-based adaptive refinement, where the objective is to reduce $|G(u-u_h)|$, by observing that

$$\begin{aligned}|\mathcal{G}_h| &= \left|\sum_{\tau\in\mathcal{T}} \int_\tau a(\mathcal{R}-I)\nabla u_h \cdot (\mathcal{R}-I)\nabla\omega_h \, dx\right| \\ &\leq \sum_{\tau\in\mathcal{T}} \left|\int_\tau a(\mathcal{R}-I)\nabla u_h \cdot (\mathcal{R}-I)\nabla\omega_h \, dx\right| \\ &\leq \sum_{\tau\in\mathcal{T}} \|a(\mathcal{R}-I)\nabla\omega_h\|_{0,\tau}\|(\mathcal{R}-I)\nabla u_h\|_{0,\tau}.\end{aligned}$$

So we use the local indicators

$$\|a(\mathcal{R}-I)\nabla\omega_h\|_{0,\tau}\|(\mathcal{R}-I)\nabla u_h\|_{0,\tau} \tag{3.8}$$

for adaptive refinement in this context. The local error estimates $\|(\mathcal{R}-I)\nabla u_h\|_{0,\tau}$ for the primal problem are the basis for the standard adaptive refinement procedure in PLTMG, so the quantities $\|a(\mathcal{R}-I)\nabla\omega_h\|_{0,\tau}$ can be thought of as dual weights which indicate global influences on the functional error.

We describe below the choice of functional which we will later use in our parallel experiments. Suppose that we are given a subdomain $\Omega_s \subset \Omega$. For purposes of practical implementation, we assume that Ω_s has a polygonal boundary and that all triangulations of Ω line up with $\partial\Omega_s$ perfectly. We have in mind subdomains Ω_s of the sort generated during the load balancing phase of the Bank–Holst algorithm. We define our dual functional by

$$G(v) = \int_{\Omega_s} fv - a\nabla u_h \cdot \nabla v - (\mathbf{b}\cdot\nabla u_h + cu_h)v \, dx + \int_{\partial\Omega_s} a\nabla u_h \cdot \mathbf{n} v \, ds. \tag{3.9}$$

If the function u is a classical solution of the PDE in question, then we have

$$\begin{aligned}G(u-u_h) &= \int_{\Omega_s} a\nabla(u-u_h)\cdot\nabla(u-u_h) + (\mathbf{b}\cdot\nabla(u-u_h) \\ &\qquad + c(u-u_h))(u-u_h)\, dx - \int_{\partial\Omega_s} a\nabla(u-u_h)\cdot\mathbf{n}(u-u_h)\, ds \\ &= \int_{\Omega_s} a\nabla(u-u_h)\cdot\nabla(u-u_h)\, dx + \mathcal{O}(h^3|\log h|).\end{aligned}$$

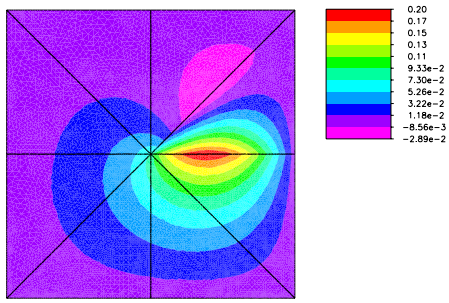


FIG. 1. *A contour plot of the approximate dual solution on a mesh with* 16036 *vertices.*

TABLE 1
The subdomain error $|e_h|^2_{1,\Omega_s}$*, the dual approximation* $\mathcal{G}_h$*, and its effectivity.*

N	$\lvert e_h\rvert^2_{1,\Omega_s}$	$\mathcal{G}_h$	EFF
244	1.97e-3	1.93e-3	0.98
996	3.36e-4	2.18e-4	0.65
4004	7.35e-5	5.78e-5	0.79
16036	1.41e-5	1.18e-5	0.84

We point out that the definition of G depends on the approximate solution u_h.

It is difficult to give a priori predictions of the smoothness of the dual solution ω. Therefore, we do not know that the assumptions of Theorem 3.1 generally apply. In other words, it is difficult to assess the effectivity of the approximation

$$\mathcal{G}_h \approx \int_{\Omega_s} a\nabla(u-u_h)\cdot\nabla(u-u_h)\,dx \tag{3.10}$$

apart from experimentation. We provide empirical evidence of the effectivity of the dual approximator in the following example. The example, henceforth called the ˛imple Problem, has primal problem

$$\begin{aligned} -\Delta u &= \sin(2\pi y)(2+4\pi^2 x(1-x)) \quad && \text{in } \Omega, \\ u &= 0 && \text{on } \partial\Omega, \end{aligned}$$

where Ω is the unit square $(0,1)\times(0,1)$. The primal solution is $u = x(1-x)\sin(2\pi y)$. For the dual problem, we choose the subdomain Ω_s with $x > y$ and $x > 0$. The approximate dual solution is pictured in Figure 1 on a mesh with approximately 16,000 vertices.

In Table 1 we see the results of several stages of the dual weighted functional error estimation for the Simple Problem. We use N for the number of unknowns, e_h for the finite element error $u-u_h$, $\mathcal{G}_h$ as our dual approximator, and EFF as the effectivity in approximation. The final row of data corresponds to the dual function pictured in Figure 1. We see that the initial effectivity is exceptionally good, decreases after the first refinement, and then begins to improve again upon successive refinements—all the while remaining in a useful range.

4. Parallel experiments. During Step 2 of the Bank–Holst algorithm as presented in [9, 10], the adaptive refinement done on processor k is driven by the weighted

error estimates

$$\eta_\tau = w_\tau \|(\mathcal{R} - I)\nabla u_h\|_{0,\tau} \text{ with } w_\tau = \begin{cases} 1 & \text{if } \tau \in \mathcal{T}_k, \\ 10^{-6} & \text{otherwise,} \end{cases}$$

where $\mathcal{T}_k$ is the portion of the triangulation coinciding with the subdomain Ω_k assigned to processor k. In this section we compare that weighting scheme with dual weighting,

$$\eta_\tau = w_\tau \|(\mathcal{R} - I)\nabla u_h\|_{0,\tau} \text{ with } w_\tau = \begin{cases} 1 & \text{if } \tau \in \mathcal{T}_k, \\ \|a(\mathcal{R} - I)\nabla \omega_h^k\|_{0,\tau} & \text{otherwise,} \end{cases}$$

where ω_h^k is the solution of the dual problem presented in the previous section associated with the subdomain Ω_k assigned to processor k.

This choice of weighting scheme deserves further explanation. In particular, why not use the dual weights in $\mathcal{T}_k$ as well? We recall that the goal of Step 2 of the algorithm is to produce a well-adapted, globally fine mesh and initial guess for use in the final global iterative solve. In other words, Step 2 should provide a good set-up for Step 3. Evidence that the dual weighting scheme has provided a better set-up than the original approach would include reduced solve times and better load balance during Step 3. Implicit in the original, 10^{-6}, version of Step 2 is the assumption that only coarse information outside Ω_k is needed for processor k to make a good contribution to the final global problem; so any defects due to essentially ignoring finer-scale pollution errors will be adequately addressed in Step 3, with negligible negative effects on its performance. In contrast, refinement on processor k which uses the dual weights both inside and outside $\mathcal{T}_k$ is acting as though there will be no Step 3 at all—as if the ultimate quality of the solution within its assigned subdomain is completely dependent upon what can be accomplished in Step 2. What one could reasonably expect at the end of Step 2 in this case is a mesh which is not significantly finer in its "fine" region than in its "coarse" region, with the effect that many degrees of freedom are discarded when the final global problem is constructed from the fine portions on each processor. The weighting scheme used for our experiments is one of many potential compromises between these two extremes, which aims to use duality to better address finer-scale pollution effects while still recognizing that the independent adaptive refinement phase is not the final phase of the algorithm.

We ran a series of experiments on two problems using PLTMG on a cluster of Dual AMD Opteron 250s with 2.4 GHz CPU clockrate, 4GB RAM, and a 2x Gigabit Ethernet Interconnect. The first problem is the Simple Problem introduced in the previous section. The second problem, which we call the -C-R Problem (diffusion-convection-reaction), is given by

$$-2(u_{xx} + u_{ux} + u_{yy}) + u_x + u_y + 3u = 2e^x(5\sin 2y - \cos 2y)$$

in $(0,1) \times (0,1)$, with Dirichlet conditions on the top and bottom of the square and Neumann conditions on the left and right chosen to match the solution $u = e^x \sin 2y$.

For Step 1 of the algorithm a coarse problem of size N_C is solved, and error estimates are used to partition the domain into p subdomains—one for each processor. During Step 2 of the algorithm, error estimates weighted by either the 10^{-6} or the dual schemes mentioned above are used for adaptive refinement, until a problem having the target size N_T is reached on each processor. We expect that most of the refinement done in Step 2 will be in the interior of the assigned subdomains on each processor; so the final global problem size N should be close to, though smaller than,

$p(N_T - N_C) + N_C$. We use the domain decomposition algorithm of Bank and Lu [6] for the final global solve in Step 3.

We take $N_C = 1000$ for Step 1 and refine until we reach a target problem size of $N_T = 50\,000,\ 100\,000,\ 150\,000,\ 200\,000,\ 250\,000$. This is done on $p = 4,\ 8,\ 16$ processors. The data for these experiments is collected in Tables 2 and 3, and is organized into blocks of six pieces of information in the following form:

$$\boxed{\begin{array}{ccc} N & |e_h|_{1,\Omega} & \|e_h\|_{0,\Omega} \\ T_{fin} & T_{tot} & T_{dif} \end{array}}, \tag{4.1}$$

where N is the final global problem size, $e_h = u - u_h$ is the error in the finite element solution, and T_{fin}, T_{tot}, and T_{dif} are the average processor time to perform the domain decomposition solve, the average processor time for the entire algorithm, and the difference between the slowest and fastest processor times for the entire algorithm respectively. For greater ease in reading the numerical results, we also include the legend (4.1) at the bottom of Tables 2 and 3.

The initial guess for the global domain decomposition solve is constructed from the fine portion of the solution on each of the processors. It is, therefore, multivalued (hence discontinuous) along the interfaces between subdomains. One of the objectives of the domain decomposition algorithm, which is based on a mortar finite element formulation, is to move toward continuity at these interfaces upon convergence. The errors are computed as

$$|e_h|^2_{1,\Omega} = \sum_{k=1}^{p} |e_h|^2_{1,\Omega_k} \quad \text{and} \quad \|e_h\|^2_{0,\Omega} = \sum_{k=1}^{p} \|e_h\|^2_{0,\Omega_k}.$$

This is a slight abuse of notation for $|e_h|_{1,\Omega}$ because e_h may remain discontinuous at the interfaces upon convergence, so $|e_h|_{1,\Omega}$ is actually infinite. We include $\|e_h\|_{0,\Omega}$ in the data primarily because, if the measure $|e_h|_{1,\Omega}$ is nearly the same for both weighting schemes then a smaller measure $\|e_h\|_{0,\Omega}$ for one of the schemes probably indicates smaller jump-discontinuities at the interfaces.

4.1. The Simple Problem in parallel. Looking at Table 2, we first point out that the global fine problem size N is approximately the same for both weighting schemes in each run of the program. The problem size is slightly smaller for the dual method than it is for the 10^{-6} method, which is expected because the dual weighting on a given processor is likely to cause more refinement outside of the region assigned to that processor. We also note that the values $|e_h|_{1,\Omega}$ are essentially identical in each case, but the values for $\|e_h\|_{0,\Omega}$ tend to be better for the dual method—sometimes dramatically better. As mentioned above, this is an indication that the final computed solution is more nearly continuous for the dual method than for the 10^{-6} method.

We see the greatest difference between the two methods in terms of the execution times for the algorithm. The dual method generally tends to lead to better times T_{fin}, T_{tot} for the final solve and the algorithm in total, as well as in the measure of load balance/imbalance T_{dif}. The most dramatic case is $p = 4$, $N_T = 250\,000$, where T_{fin} for the dual method is less than half of its counterpart! Because the problem sizes are roughly equivalent for both methods, the better values for T_{fin} indicate that using the dual method during Step 2 of the algorithm provides a better set-up for Step 3. In fact, comparing the difference $T_{tot} - T_{fin}$ for both methods shows that the time to reach Step 3 is essentially the same, so the only real difference is how well each method sets up this final global solve in Step 2.

TABLE 2
Data for the Simple Problem.

Simple 4	10^{-6}			Dual		
50000	194621	3.97e-3	2.89e-6	193436	3.98e-3	2.90e-6
	9.86	17.10	0.72	9.88	17.56	0.67
100000	393298	2.87e-3	1.52e-6	390749	2.87e-3	1.53e-6
	32.09	55.54	2.94	24.70	50.08	0.78
150000	592464	2.32e-3	1.01e-6	588500	2.33e-3	1.01e-6
	55.06	88.34	2.01	42.42	78.31	5.49
200000	791793	1.98e-3	1.06e-6	785629	1.99e-3	7.31e-7
	131.57	175.51	10.20	71.10	117.55	3.80
250000	990954	1.78e-3	9.71e-7	982117	1.79e-3	5.85e-7
	178.84	234.18	22.63	81.95	140.01	4.61

Simple 8	10^{-6}			Dual		
50000	385830	2.90e-3	1.58e-6	382321	2.92e-3	1.59e-6
	12.17	19.75	2.28	10.34	18.24	1.68
100000	782601	2.04e-3	2.57e-6	775796	2.05e-3	7.77e-7
	55.62	79.91	14.13	30.07	55.94	5.96
150000	1180258	1.70e-3	1.80e-6	1170255	1.71e-3	5.41e-7
	91.33	125.70	16.32	56.18	91.93	9.84
200000	1577903	1.44e-3	2.72e-6	1563607	1.44e-3	3.90e-7
	134.90	179.76	31.34	93.18	139.95	12.02
250000	1975825	1.23e-3	1.67e-6	1955312	1.23e-3	2.99e-7
	182.19	238.68	38.76	118.18	176.58	22.07

Simple 16	10^{-6}			Dual		
50000	768628	2.05e-3	7.77e-7	758557	2.06e-3	7.93e-7
	16.20	23.75	5.43	14.14	21.97	3.70
100000	1560383	1.47e-3	1.37e-6	1543297	1.47e-3	9.78e-7
	57.67	81.41	27.47	57.95	83.50	20.92
150000	2355278	1.19e-3	2.11e-6	2330699	1.19e-3	7.99e-7
	105.32	139.32	50.59	98.42	133.74	35.39
200000	3150792	1.02e-3	2.10e-6	3115681	1.02e-3	1.98e-7
	149.24	194.36	72.32	93.81	139.90	25.62
250000	3946044	9.14e-4	2.81e-6	3898144	9.16e-4	7.2e-7
	184.29	239.57	55.61	169.58	226.63	51.94

N	$\lvert e_h\rvert_{1,\Omega}$	$\lVert e_h\rVert_{0,\Omega}$
T_{fin}	T_{tot}	T_{dif}

We make one final point before moving to the next problem. As the number of processors increased, the difference between the solve times for the two methods decreased. However, we also point out that the tendency is for the difference in execution times to increase in favor of the dual method. The only real violation of this tendency is the case $p = 16$, $N_T = 200\,000$, where the dual method is unusually fast.

4.2. The D-C-R Problem in parallel. We first note in Table 3 that, in contrast to the behavior for the Simple Problem, here we see a noticeable difference in the final problem size generated by the two methods, with the dual method producing smaller problems for the global parallel solve. This is due to the fact that the dual weighting on each processor causes more refinement to be done outside of its assigned subdomain than the 10^{-6} weighting during Step 2, so the "fine" subdomains on each processor tend to be less fine for the dual method. In Figure 2 we see a contour plot of the dual solution and a plot of the element sizes during a stage of Step 2, correspond-

TABLE 3
Data for the D-C-R Problem.

D-C-R 4	10^{-6}			Dual		
50000	194496	0.18	2.89e-4	161018	0.20	3.52e-4
	12.90	21.16	0.51	10.33	18.70	1.33
100000	392900	0.13	1.99e-4	343059	0.14	1.72e-4
	54.70	80.31	9.83	28.92	56.10	2.32
150000	592156	0.10	1.03e-4	524186	0.11	1.14e-4
	81.38	117.28	10.93	49.47	87.03	6.78
200000	791366	8.77e-2	2.13e-4	698025	9.36e-2	8.20e-5
	152.19	199.64	35.22	70.67	118.84	5.86
250000	990514	7.75e-2	3.85e-4	866089	8.44e-2	6.65e-5
	204.86	262.93	26.58	98.93	158.10	27.38

D-C-R 8	10^{-6}			Dual		
50000	386063	0.13	2.72e-4	312943	0.14	1.82e-4
	25.20	33.49	4.37	12.77	21.04	1.72
100000	782568	9.08e-2	3.56e-4	674527	0.10	9.13e-5
	58.73	84.22	13.83	31.27	58.32	4.56
150000	1180401	7.32e-2	4.95e-4	1029826	7.84e-2	5.71e-5
	102.49	137.94	28.36	58.60	96.20	10.19
200000	1578249	6.28e-2	5.35e-4	1367799	6.75e-2	5.13e-5
	149.35	195.34	40.41	135.70	183.13	7.87
250000	1976366	5.63e-2	5.74e-4	1703877	6.13e-2	5.99e-5
	210.02	267.00	64.16	178.57	236.46	36.54

D-C-R 16	10^{-6}			Dual		
50000	768158	9.10e-2	8.83e-5	565141	0.11	1.05e-4
	18.13	26.26	5.46	11.23	19.54	4.60
100000	1559940	6.43e-2	3.85e-4	1257098	7.17e-2	4.68e-5
	64.26	89.62	19.13	40.28	67.15	8.81
150000	2354749	5.27e-2	4.29e-4	1924354	5.83e-2	4.58e-5
	110.96	146.35	36.09	97.34	134.64	28.79
200000	3150111	4.49e-2	6.05e-4	2563542	5.04e-2	3.99e-5
	165.74	211.21	65.21	143.11	191.21	29.61
250000	3945421	3.97e-2	7.65e-4	3195221	4.51e-2	3.27e-5
	224.25	279.77	97.58	193.13	252.28	47.88

N	$\lvert e_h\rvert_{1,\Omega}$	$\lVert e_h\rVert_{0,\Omega}$
T_{fin}	T_{tot}	T_{dif}

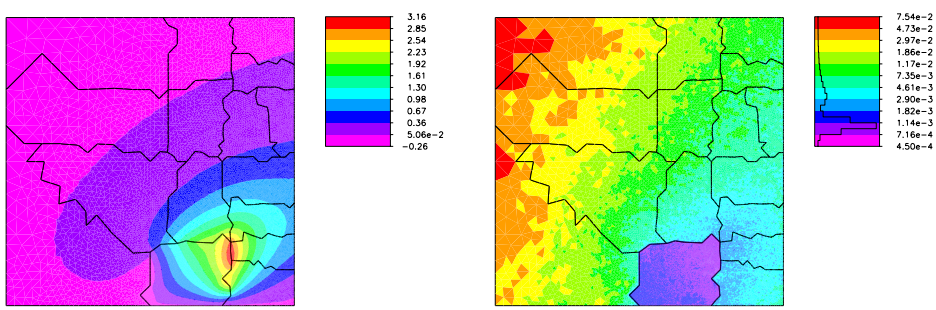


FIG. 2. *A contour plot of the approximate dual solution (left) and a plot of the element sizes.*

ing to $p = 16$, $N_T = 100\,000$. We see that the dual solution is more pronounced in the direction of the convection and that the refinement is also heavier in this direction.

Because the final global problem sizes for the two methods are noticeably different, it is not surprising that the final errors are also different. In this case $|e_h|_{1,\Omega}$ tends to be slightly smaller for the 10^{-6} method, but $\|e_h\|_{0,\Omega}$ tends to be smaller for the dual method—often by an order of magnitude! To compare the gradient errors on more equal footing we consider the ratios $|e_h|_{1,\Omega}/\sqrt{N}$, because it is expected that $|e_h|_{1,\Omega} \sim h$ and $N \sim h^2$. The ratios $|e_h|_{1,\Omega}/\sqrt{N}$ are similar in each case for both methods, with the 10^{-6} method having slightly smaller values but the dual method catching up as N_T and/or p is increased. We emphasize that the smaller values of the function error $\|e_h\|_{0,\Omega}$ for the dual method indicate a final solution which is more nearly continuous—as it should be.

Concerning the timing comparisons, the situation is similar to that for the Simple Problem. We see that the dual method is uniformly better in terms of time for the domain decomposition solve T_{fin} and total time for the algorithm T_{tot}, and tends to be better in terms of load balance T_{dif} as well. The differences between T_{fin} can be dramatic in some cases, with the value for $p = 4$, $N_T = 250\,000$ for the dual method being less than half of its counterpart!

REFERENCES

[1] I. Babuška, F. Ihlenburg, T. Strouboulis, and S. K. Gangaraj, *A posteriori error estimation for finite element solutions of Helmholtz' equation*. II. *Estimation of the pollution error*, Internat. J. Numer. Methods Engrg., 40 (1997), pp. 3883–3900.

[2] I. Babuška and A. Miller, *The post-processing approach in the finite element method. Part* 1. *Calculation of displacements, stresses and other higher derivatives of the displacements*, Internat. J. Numer. Methods Engrg., 20 (1984), pp. 1085–1109.

[3] I. Babuška and A. Miller, *The post-processing approach in the finite element method. Part* 2. *The calculation of stress intensity factors*, Internat. J. Numer. Methods Engrg., 20 (1984), pp. 1111–1129.

[4] I. Babuška and A. Miller, *The post-processing approach in the finite element method. Part* 3. *A posteriori error estimates and adaptive mesh selection*, Internat. J. Numer. Methods Engrg., 20 (1984), pp. 2311–2324.

[5] I. Babuška and T. Strouboulis, *The Finite Element Method and Its Reliability*, Numerical Mathematics and Scientific Computation, Oxford University Press, New York, 2001.

[6] R. Bank and S. Lu, *A domain decomposition solver for a parallel adaptive meshing paradigm*, SIAM J. Sci. Comput., 26 (2004), pp. 105–127.

[7] R. E. Bank, *PLTMG: A Software Package for Solving Elliptic Partial Differential Equations, Users' Guide* 9.0, Tech. report, University of California, San Diego, CA, 2004.

[8] R. E. Bank, *Some variants of the Bank-Holst parallel adaptive meshing paradigm*, Comput. Vis. Sci., 9 (2006), pp. 133–144.

[9] R. E. Bank and M. Holst, *A new paradigm for parallel adaptive meshing algorithms*, SIAM J. Sci. Comput., 22 (2000), pp. 1411–1443.

[10] R. E. Bank and M. J. Holst, *A new paradigm for parallel adaptive meshing algorithms*, SIAM Rev., 45 (2003), pp. 291–323.

[11] R. E. Bank and J. Xu, *Asymptotically exact a posteriori error estimators*. I. *Grids with superconvergence*, SIAM J. Numer. Anal., 41 (2003), pp. 2294–2312.

[12] R. E. Bank and J. Xu, *Asymptotically exact a posteriori error estimators*. II. *General unstructured grids*, SIAM J. Numer. Anal., 41 (2003), pp. 2313–2332.

[13] M. W. Beall and M. S. Shephard, *A general topology-based mesh data structure*, Internat. J. Numer. Methods Engrg., 40 (1997), pp. 1573–1596.

[14] R. Becker and R. Rannacher, *A feed-back approach to error control in finite element methods: Basic analysis and examples*, East-West J. Numer. Math., 4 (1996), pp. 237–264.

[15] R. Becker and R. Rannacher, *An optimal control approach to a posteriori error estimation in finite element methods*, Acta Numer., 10 (2001), pp. 1–102.

[16] S. C Brenner and L. Ridgway Scott, *The Mathematical Theory of Finite Element Methods*, Texts in Applied Mathematics 15, Springer-Verlag, New York, 1994.

[17] H. L. deCougny, K. D. Devine, J. E. Flaherty, R. M. Loy, C. Ozturan, and M. S. Shephard, *Load balancing for the parallel adaptive solution of partial differential equations*, Appl. Numer. Math., 16 (1994), pp. 157–182.

[18] D. J. Estep, M. J. Holst, and M. Larson, *Generalized Green's functions and the effective domain of influence*, SIAM J. Sci. Comput., 26 (2005), pp. 1314–1339.

[19] J. E. Flaherty, R. M. Loy, C. Ozturan, M. S. Shephard, B. K. Szymanski, J. D. Teresco, and L. H. Ziantz, *Parallel structures and dynamic load balancing for adaptive finite element computation*, Appl. Numer. Math., 26 (1998), pp. 241–263.

[20] G. C. Fox, R. D. Williams, and P. C. Messina, *Parallel Computing Works!*, Morgan-Kaufmann, San Francisco, CA, 1994.

[21] M. Giles and E. Süli, *Adjoint methods for PDEs: A posteriori error analysis and postprocessing by duality*, Acta Numer., 11 (2002), pp. 145–236.

[22] V. Heuveline and R. Rannacher, *Duality-based adaptivity in the hp-finite element method*, J. Numer. Math., 11 (2003), pp. 95–113.

[23] S. Kohn, J. Weare, M. E. Ong, and S. B. Baden, *Software abstractions and computational issues in parallel structured adaptive mesh methods for electronic structure calculations*, in Workshop on Structured Adaptive Mesh Refinement Grid Methods, Institute for Mathematics and Its Applications 117, Springer-Verlag, New York, 2000, pp. 75–95.

[24] J. S. Ovall, *Asymptotically exact functional error estimators based on superconvergent gradient recovery*, Numer. Math., 102 (2006), pp. 543–558.

[25] N. A. Pierce and M. B. Giles, *Adjoint recovery of superconvergent functionals from PDE approximations*, SIAM Rev., 42 (2000), pp. 247–264.

[26] S. Prudhomme and J. T. Oden, *On goal-oriented error estimation for elliptic problems: Application to control of pointwise errors*, Comput. Methods Appl. Mech. Engrg., 176 (1999), pp. 313–331.

[27] P. M. Selwood, M. Berzins, and P. M. Dew, 3*D parallel mesh adaptivity: Data structures and algorithms*, in Proceedings of the 8th SIAM Conference on Parallel Processing for Scientific Computing, Minneapolis, MN, CD-ROM, SIAM, Philadelphia, 1997.

[28] L. B. Wahlbin, *Local behavior in finite element methods*, in Handbook of Numerical Analysis, Vol. II, North–Holland, Amsterdam, 1991, pp. 353–522.

[29] C. Walshaw and M. Berzins, *Dynamic load balancing for pde solvers on adaptive unstructured meshes*, Concurrency: Practice and Experience, 7 (1995), pp. 17–28.

SIAM J. SCI. COMPUT.
Vol. 29, No. 4, pp. 1525–1549
© 2007 Society for Industrial and Applied Mathematics

BOUNDARY PRESERVING SEMIANALYTIC NUMERICAL ALGORITHMS FOR STOCHASTIC DIFFERENTIAL EQUATIONS*

ESTEBAN MORO† AND HENRI SCHURZ‡

Abstract. Construction of splitting-step methods and properties of related nonnegativity and boundary preserving semianalytic numerical algorithms for solving stochastic differential equations (SDEs) of Itô type are discussed. As the crucial assumption, we oppose conditions such that one can decompose the original system of SDEs into subsystems for which one knows either the exact solution or its conditional transition probability. We present convergence proofs for a newly designed splitting-step algorithm and simulation studies for numerous well-known numerical examples ranging from stochastic dynamics occurring in asset pricing in mathematical finance (Cox–Ingersoll–Ross (CIR) and constant elasticity of variance (CEV) models) to measure-valued diffusion and super-Brownian motion (stochastic PDEs (SPDEs)) as met in biology and physics.

Key words. stochastic differential equations, absorbing boundary, numerical methods, super-Brownian motion, splitting-step algorithm, convergence, nonnegativity, simulation

AMS subject classifications. 65C30, 60H10, 60H15

DOI. 10.1137/05063725X

1. Introduction and examples.

Stochastic differential equations (SDEs) are fundamental to describe and understand random phenomena in different areas of physics, engineering, economics, etc. Particularly, they serve as models for price fluctuations as in the famous Black–Scholes option pricing model, description of erratic movements of particles as in the Langevin equation, or spatial processes like the super-Brownian motion (sBm). In most cases, explicit solutions of SDEs are very difficult to obtain and numerical approximations need to be exploited. In fact a wealth of methods for numerically integrating SDEs are known and tested [27, 28, 31, 40, 41, 42, 43, 46].

In some practical applications autonomous Itô-type SDEs of the form

$$dX(t) = f(X(t))\,dt + \sigma(X(t))\,dW(t), \quad X(0) = X_0 \in \mathcal{D} \tag{1.1}$$

are well defined only with certain boundary conditions at the boundaries of the domain $\mathcal{D}$ in which $X(t)$ lives for all t (almost surely). These boundary conditions could be additionally imposed or naturally inherent in (1.1). For example, if $X(t)$ is the price of a stock and (1.1) gives its time-evolution, then $\mathcal{D} = [0, \infty)$, where $X(t) = 0$ implies an absorbing or reflecting state. Or, if $X(t)$ models the number of certain species in a noisy environment, then $\mathcal{D} = [0, K]$, where K is an attracting carrying capacity of the environment. When the boundaries are unattainable in finite time, conditions at the boundary are not needed to define the problem (1.1). This is the case of [illegible], according to Feller's classification [14, 15, 16, 17].

The aim of this paper is to present an alternative strategy, based on [illegible] to numerically integrate SDEs (1.1) subject to natural boundary conditions,

*Received by the editors August 1, 2005; accepted for publication (in revised form) December 20, 2006; published electronically July 4, 2007. This work was supported by the Ministerio de Ciencia and Tecnología (Spain) through projects FIS2004-01001 and BFM2002-04474-C02.
http://www.siam.org/journals/sisc/29-4/63725.html

†Grupo Interdisciplinar de Sistemas Complejos (GISC) and Departamento de Matemáticas, Universidad Carlos III de Madrid, Avenida de la Universidad 30, 28911 Leganés, Spain (emoro@math.uc3m.es).

‡Department of Mathematics, Southern Illinois University, 1245 Lincoln Drive, Carbondale, IL 62901 (hschurz@math.siu.edu).

that guarantees both that the numerical solutions live in the domain $\mathcal{D}$ and that the character of the boundary is preserved. Our analysis focuses mainly on the case of naturally inherent boundaries in the exact solution and its numerical approximation. It is important to stress that, although our method relies heavily on the knowledge of the exact solution of related stochastic processes and thus its validity is not strictly general, a wide variety of paramount problems known in the literature ranging from finance to physics or biology can be treated efficiently. Moreover, the consistency of this new method shall be mathematically justified by two convergence theorems, namely, one for sufficiently smooth Lipschitzian coefficients of involved SDEs (Theorem 3.1 in the main text) and another one (Theorem C.1 in the appendix) based on the variation-of-parameters formula (VOP). By those theorems we are going to establish the convergence rates which are equal to or higher than standard methods (e.g., 1.0 for global L^2-convergence in contrast to the rate 0.5 of Euler-type methods). Finally, the computational method is complemented with a very fast algorithm to sample an important class of conditional transition probabilities found in important problems.

The standard (unrestricted) Brownian motion on $\mathbb{R}$ is the most obvious example of natural boundaries at infinity. Reasonable population models in ecology, interest rates, or asset prices in finance possess the natural boundary at 0 (i.e., nonnegativity). The situation is different when the solution of (1.1) attains the boundaries in finite time. In this case, boundary conditions can be naturally inherent or imposed such as [illegible] —for example, the Brownian motion on $\mathbb{R}^+$ for which a boundary condition at $x = 0$ needs to be specified to completely define its solution.

The problem of how to handle boundary conditions also appears in the numerical approximations of (1.1). Naturally inherent boundary conditions do need to be incorporated in the construction of its numerical approximations too. In fact, standard numerical methods such as Euler methods may fail to meet these boundary conditions; see [39]. This is also true for higher order converging Taylor-type methods [27, 28, 41]. Numerical time-discretizations face two kinds of different (though related) problems when a boundary condition is specified. The first one is to restrict the values of the numerical approximations to live within the domain $\mathcal{D}$; second, they also have to preserve the character of the boundary (natural, reflecting, absorbing, etc.).

There are several methods of numerically approximating the solutions of (1.1) when boundary conditions are imposed at the boundaries of $\mathcal{D}$. In the case of natural boundaries (related to Neumann boundary conditions in approximations of parabolic PDEs), methods involving projection techniques [29], penalty methods [26, 38], or symmetrized schemes [1] have been proved successful. For absorbing boundary conditions (related to Dirichlet boundary conditions in PDEs), the strategies involving direct Euler schemes [13, 19, 20] or exploiting the well-known Feynman–Kac representation formula [3] are applied. Finally, in the case of natural boundaries, i.e., when the diffusion does not reach the boundary, balanced implicit methods (see [32, 40, 41]), balanced Milstein methods [25], or composition methods based on Lie algebra techniques [33] are efficient methods of incorporating boundary restrictions. In other specific cases, numerical approximations of (1.1) in $\mathcal{D}$ are obtained by using standard methods and modifying the equation whenever the approximation lies outside $\mathcal{D}$ (e.g., [21, 8]).

In contrast to all forementioned citations, we shall follow a rather different, semi-analytic numerical approach to systems of SDEs (1.1). Our approach heavily relies on the exploitation of the specific structure of the original system (1.1) which al-

lows a natural splitting into explicitly known parts of the underlying dynamics with well-known boundary behavior and the remaining parts to be treated numerically with naturally incorporated boundary behavior (or even without restrictions). The method is especially efficient at approximating solutions of SDEs (1.1) where boundary conditions are naturally inherent. To exemplify the class of problems and its numerical approximations, consider the well-known square-root diffusion of Cox, Ingersoll, and Ross [5],

$$dX(t) = [a + bX(t)]\,dt + \sigma\sqrt{X(t)}\,dW(t), \quad X(0) = x_0 \geq 0 \tag{1.2}$$

with real parameters $a \geq 0$, $b \in \mathbb{R}$, and $\sigma > 0$, which is widely used for interest rate modeling or as an alternative to geometric Brownian motion occurring in the Black–Scholes model of dynamic asset pricing in mathematical finance. It is well known that the (strong) solution of (1.2) is unique, preserves the nonnegativity of the initial data, and has different boundary behavior at $X(t) = 0$ depending on the values of a and σ (see section 5.1). For the real-valued numerical implementation of the $\sqrt{X(t)}$-term, the nonnegativity of numerical approximations of (1.2) is of special importance. In fact, the nonnegativity (and hence the exclusive real-valued numerical evaluation) can be easily achieved in numerical approximations by means of balanced implicit methods (see [32, 40, 41]), balanced Milstein methods [25], and composition methods based on Lie algebra techniques [33]. However, the latter-mentioned or other methods may fail to incorporate the right boundary properties in the numerical paths. Other methods, while converging in the limit $\Delta t \to 0$, do not even conserve the nonnegativity of the solution like that of the straightforward fixing by taking $\sqrt{|X(t)|}$ instead of the diffusion term $\sqrt{X(t)}$ in (1.2) (as done in [21] or [8]). Recall that convergence as $\Delta t \to 0$ is an asymptotic property, but nonnegativity or boundary preserving property is more of qualitative nature and important from the point of view of adequate modeling interpretation (while using nonzero step sizes as in computational practice).

The following sections are structured as follows. We introduce and discuss the general splitting-step algorithm in section 2. Section 3 reports on a general convergence theorem and its mathematical proof. Several numerical experiments in sections 4–6 strongly support the suggested algorithm and results from previous sections by ordinary SDEs. In section 4 we present some simple models and related experiments. Section 5 is devoted to the simulation of processes as often met in dynamic asset pricing in mathematical finance. In section 6 we mention some applications to measure-valued diffusions and reaction-diffusion equations demonstrating the potential range of the splitting-step algorithm for numerical treatment of random PDEs. Section 7 concludes with some supplemental remarks. Eventually, there are three small appendices on aspects of random number generation and strong path approximations and on a convergence theorem based on the VOP with an illustrative example.

2. Splitting-step algorithm. The general structure of a splitting-step algorithm, which is based on the idea in [37] (see also [34, 10]), is as follows. Suppose that the more general equation which is to be integrated is of the form

$$dX(t) = [\alpha(X(t), t) + \beta(X(t), t)]\,dt + \sigma(X(t), t)\,dW(t). \tag{2.1}$$

We then decompose the above equation into the two equations

$$dX_1(t) = \beta(X_1(t), t)\,dt + \sigma(X_1(t), t)\,dW(t), \tag{2.2}$$

$$dX_2(t) = \alpha(X_2(t), t)\,dt, \tag{2.3}$$

where the splitting is done assuming that we know the exact strong solution for $X_1(t)$ or the conditional probability $\mathcal{P}[X_1(t)|X_1(0)]$. Thus, we can approximate the solution of (2.1) by a stochastic process Y_t along time intervals $[t, t+\Delta t]$ using the following two-step algorithm for each Δt, which we call the [illegible]:

Step 1. Knowing the value of Y_t we obtain an intermediate value $\tilde{Y}_t$ which is obtained through the exact integration of (2.2) and $\tilde{Y}_t = X_1(t+\Delta t)$ and with initial condition $X_1(t) = Y_t$.

Step 2. Then $\tilde{Y}_t$ is used as the initial condition for (2.3) which is now integrated using any converging deterministic numerical algorithm to get $\tilde{X}_2$ (at least of deterministic order 1). Then $Y_{t+\Delta t} = \tilde{X}_2(t+\Delta t)$.

As said before, we [illegible] that either the analytic expression of the exact strong solution of X_1 governed by (2.2) or its conditional transition probabilities $\mathcal{P}[X_1(t)|X_1(0)]$ are explicitly known. Thus, the advantage of this splitting-step technique for SDEs subject to boundary conditions is that if (2.2) is simple enough and we know the solution X_1 of (2.1), then the stochastic part of the problem can be handled correctly. For example, for the case $\beta(X_1, t) = 0$ and $\sigma(X_1, t) = \sqrt{X_1}$, it is known that the conditional probability distribution is given by

$$\mathcal{P}[X_1(t)|X_1(0)] = \frac{2}{t}\left(\frac{X_1(0)}{X_1(t)}\right)^{1/2} I_1\left(\frac{4}{t}\sqrt{X_1(t)X_1(0)}\right) e^{-\frac{2}{t}[X_1(t)+X_1(0)]} + e^{-\frac{4}{t}X_1(0)}\delta(X_1(t)), \tag{2.4}$$

where I_1 is the modified Bessel function and $\delta(x)$ is the Dirac delta function.

This cdf can be sampled using the rejection or inverse methods but this is computationally expensive. However, we introduce here a very simple method [34] (see also [2]) for obtaining $X_1(t)$ by noting that the variable $Z(t) = \frac{4}{t}X_1(t)$ has a probability distribution given by the noncentral χ^2-distribution, that is,

$$\mathcal{P}[Z|Z_0] = \sum_{j=1}^{\infty} \frac{(\lambda/2)^j e^{-\lambda/2}}{j!} \mathcal{P}_{\chi^2_{2j}}(Z) + e^{-\lambda/2}\delta(Z), \tag{2.5}$$

where $\lambda = \frac{4}{t}Z_0$ and $\mathcal{P}_{\chi^2_{2j}}(X)$ is the χ^2-pdf with $2j$ degrees of freedom. Equation (2.5) reveals that the probability distribution for Z is a linear combination of χ^2-pdfs with Poisson weights. This fact can be exploited to generate $X_1(t)$ efficiently. If we choose K from a Poisson distribution with mean $\lambda/2$, then

$$X_1(t) = \frac{1}{2k}\begin{cases} 0 & \text{if } K = 0, \\ \sum_{i=1}^{2K} z_i^2 & \text{if } K \neq 0, \end{cases} \tag{2.6}$$

where z_i are independent Gaussian random numbers with zero mean and unit variance. Computationally, it is faster to sample the random number $\sum_{i=1}^{2K} z_i^2$ using standard algorithms for random number generation of the χ^2 distribution. Other examples of this sampling can be found in section 4 and the appendix.

The obvious advantage of the splitting algorithm is that we exploit the structure of the original equation more efficiently. For example, in the above example, we get numbers from (2.6) which are nonnegative and thus, if $\alpha(X, t)$ has nice properties, the approximated values for $X(t)$ are nonnegative, too. Such a splitting algorithm is not known from the literature to the best of our current knowledge, although the idea of

splitting is not new for dynamical systems and their numerical integration. A different kind of splitting technique has been suggested in [22]. However, their algorithm, called the split-step Euler method, is related to another subclass of splitting of SDEs and their resulting split-step algorithm is of lower order 0.5 of mean square convergence, which is restricted by their use of the (drift-implicit) backward Euler method. Their method indirectly refers to the splitting

$$dX_1(t) = [\alpha(X_1(t),t) + \beta(X_1(t),t)]\,dt,$$
$$dX_2(t) = \sigma(X_2(t),t)\,dW(t),$$

of the original system (1.1) in our set-up, where both equations for X_1 and X_2 are numerically integrated in a separated fashion. In contrast to them, by a more efficient choice of splitting, we even suggest removing stochastic integrals from numerical integration by appropriate splitting of (1.1) in order to achieve a higher order of convergence, and our technique is not restricted only to techniques of direct pathwise simulation. It may be noted that the explicit removal of stochastic integrals from numerical integration by splitting techniques is always possible and leads to converging numerical approximations with higher order under some appropriate assumptions on the diffusion coefficient σ (for example, $\sigma\sigma\prime$ has sufficiently many bounded derivatives) in $\mathbb{R}^1$. Another form of splitting has been suggested in [6]. They present an algorithm which also takes advantage of techniques of numerical integration of ODEs. However, neither [6] nor [22] discusses the issue of pathwise preservation of nonnegativity, monotonicity, and boundedness by their numerical approximation techniques.

3. A convergence theorem of splitting algorithm for nonautonomous equations. Let $C^{i,j}(\mathbb{R}^d \times [0,T])$ denote the vector space of continuous functions $f = f(x,t)$ which are i times continuously differentiable with respect to the space-coordinate $x_k \in \mathbb{R}$ $(k = 1,2,\ldots,d)$ and j times continuously differentiable with respect to time-coordinate $t \in [0,T]$. Recall that the original equation is

$$dX(t) = [\alpha(X(t),t) + \beta(X(t),t)]\,dt + \sigma(X(t),t)\,dW(t). \tag{3.1}$$

For the proof of splitting techniques below, we refer the reader to the splitting

$$dX_1(t) = \beta(X_1(t),t)\,dt + \sigma(X_1(t),t)\,dW(t), \tag{3.2}$$
$$dX_2(t) = \alpha(X_2(t),t)\,dt. \tag{3.3}$$

THEOREM 3.1. [illegible] $\alpha, \beta \in C^{2,1}(\mathbb{R}^d \times [0,T])$ $\sigma \in C^{3,2}(\mathbb{R}^d \times [0,T])$ [illegible]

$$\sup_{0\le t\le T} \mathbb{E}\left[|X(t)|^2 + |\alpha(X(t),t)|^2 + |\beta(X(t),t)|^2 + |\sigma(X(t),t)|^2\right] < +\infty$$

[illegible] $T > 0$ [illegible] 1 2 [illegible] 1.0 [illegible] $[0,T]$ [illegible]. For simplicity, suppose that $d = 1$. Let

$$0 = t_0 < t_1 < \cdots < t_n < t_{n+1} < \cdots < t_N = T$$

be any nonrandom partition of the given time-interval $[0,T]$ with sufficiently small maximum step size $\Delta = \max_{i=1,2,\ldots,N} |t_i - t_{i-1}| \le 1$. Define the local pathwise error

by $\varepsilon_{n+1}^{loc} = X(t_n + \Delta_n t) - \tilde{X}(t_n + \Delta_n t)$ assuming that both exact solution X and its approximation $\tilde{X}$ have started at the same value $X(t_n)$ at time t_n. To investigate this error, apply the stochastic Taylor formula to the processes X, X_1, and X_2—an idea which originates from the Wagner–Platen expansion [47] and was popularized by [27] (more precisely speaking, this exploits the idea of an iterative application of the Itô formula). For the sake of abbreviation, we shall write z_n or z_u for all occurring coefficients or processes (not referring to partial derivatives here); hence $X_n = X(t_n)$, $\alpha_n = \alpha(X(t_n), t_n)$, $\beta_n = \beta(X(t_n), t_n)$, and $\sigma_n = \sigma(X(t_n), t_n)$, and similarly, $\alpha_u = \alpha(X(u), u)$, $\beta_u = \beta(X(u), u)$, $\sigma_u = \sigma(X(u), u)$, and so on, unless it is stated differently wherever convenient. Define the partial differential operators

$$
\begin{aligned}
L_0^0 f(x,t) &= \frac{\partial f(x,t)}{\partial t} + [\alpha(x,t) + \beta(x,t)] \frac{\partial f(x,t)}{\partial x} + \frac{1}{2}[\sigma(x,t)]^2 \frac{\partial^2 f(x,t)}{\partial x^2}, \\
L_0^1 f(x,t) &= \sigma(x,t) \frac{\partial f(x,t)}{\partial x}, \quad L_1^1 f(x,t) = L_0^1 f(x,t), \\
L_1^0 f(x,t) &= \frac{\partial f(x,t)}{\partial t} + \beta(x,t) \frac{\partial f(x,t)}{\partial x} + \frac{1}{2}[\sigma(x,t)]^2 \frac{\partial^2 f(x,t)}{\partial x^2}, \\
L_2^0 f(x,t) &= \frac{\partial f(x,t)}{\partial t} + \alpha(x,t) \frac{\partial f(x,t)}{\partial x},
\end{aligned}
$$

where L_i^0 is mapping from $C^{2,1}(\mathbb{R}^d \times [0,T])$ to $C^{0,0}(\mathbb{R}^d \times [0,T])$ and L_i^1 from $C^{1,0}(\mathbb{R}^d \times [0,T])$ to $C^{0,0}(\mathbb{R}^d \times [0,T])$ for $i = 1, 2$, and L_2^0 from $C^{1,1}(\mathbb{R}^d \times [0,T])$ to $C^{0,0}(\mathbb{R}^d \times [0,T])$. First, apply Taylor-type approximations to the solutions of (3.1) to obtain

$$
\begin{aligned}
X_{n+1} &= X_n + \int_{t_n}^{t_{n+1}} [\alpha(X(s), s) + \beta(X(s), s)]\, ds + \int_{t_n}^{t_{n+1}} \sigma(X(s), s)\, dW(s) \\
&= X_n + [\alpha_n + \beta_n]\Delta_n t + \sigma_n \Delta_n W \\
&\quad + \int_{t_n}^{t_{n+1}} \int_{t_n}^{s} L_0^0(\alpha_u + \beta_u)\, du\, ds + \int_{t_n}^{t_{n+1}} \int_{t_n}^{s} L_0^1(\alpha_u + \beta_u)\, dW(u)\, ds \\
&\quad + \int_{t_n}^{t_{n+1}} \int_{t_n}^{s} L_0^0 \sigma_u\, du\, dW(s) + \int_{t_n}^{t_{n+1}} \int_{t_n}^{s} L_0^1 \sigma_u\, dW(u)\, dW(s) \\
&= X_n + [\alpha_n + \beta_n]\Delta_n t + \sigma_n \Delta_n W + \frac{1}{2} L_0^1 \sigma_n [(\Delta_n W)^2 - \Delta_n t] + R_{0,n} \qquad (3.4)
\end{aligned}
$$

with the remainder term

$$
\begin{aligned}
R_{0,n} &= \int_{t_n}^{t_{n+1}} \int_{t_n}^{s} L_0^0(\alpha_u + \beta_u)\, du\, ds + \int_{t_n}^{t_{n+1}} \int_{t_n}^{s} L_0^1(\alpha_u + \beta_u)\, dW(u)\, ds \\
&\quad + \int_{t_n}^{t_{n+1}} \int_{t_n}^{s} L_0^0 \sigma_u\, du\, dW(s) + \int_{t_n}^{t_{n+1}} \int_{t_n}^{s} \int_{t_n}^{u} L_0^0 L_0^1 \sigma_v\, dv\, dW(u)\, dW(s) \\
&\quad + \int_{t_n}^{t_{n+1}} \int_{t_n}^{s} \int_{t_n}^{u} L_0^1 L_0^1 \sigma_v\, dW(v)\, dW(u)\, dW(s).
\end{aligned}
$$

Now, we compare (3.4) with what we get from the expansion of the splitting method. In the latter case, by application of the Wagner–Platen expansion [47] again, we arrive at

$$
X_{1,n+1} = X_n + \int_{t_n}^{t_{n+1}} \beta(X_1(s), s)\, ds + \int_{t_n}^{t_{n+1}} \sigma(X_1(s), s)\, dW(s)
$$

$$
\begin{aligned}
&= X_n + \beta_n \Delta_n t + \sigma_n \Delta_n W \\
&\quad + \int_{t_n}^{t_{n+1}} \int_{t_n}^{s} L_1^0 \beta_u \, du \, ds + \int_{t_n}^{t_{n+1}} \int_{t_n}^{s} L_1^1 \beta_u \, dW(u) \, ds \\
&\quad + \int_{t_n}^{t_{n+1}} \int_{t_n}^{s} L_1^0 \sigma_u \, du \, dW(s) + \int_{t_n}^{t_{n+1}} \int_{t_n}^{s} L_1^1 \sigma_u \, dW(u) \, dW(s) \\
&= X_n + \beta_n \Delta_n t + \sigma_n \Delta_n W + \frac{1}{2} L_1^1 \sigma_n [(\Delta_n W)^2 - \Delta_n t] + R_{1,n} \qquad (3.5)
\end{aligned}
$$

with the remainder term

$$
\begin{aligned}
R_{1,n} = &\int_{t_n}^{t_{n+1}} \int_{t_n}^{s} L_1^0 \beta_u \, du \, ds + \int_{t_n}^{t_{n+1}} \int_{t_n}^{s} L_1^1 \beta_u \, dW(u) \, ds \\
&+ \int_{t_n}^{t_{n+1}} \int_{t_n}^{s} L_1^0 \sigma_u \, du \, dW(s) + \int_{t_n}^{t_{n+1}} \int_{t_n}^{s} \int_{t_n}^{u} L_1^0 L_1^1 \sigma_v \, dv \, dW(u) \, dW(s) \\
&+ \int_{t_n}^{t_{n+1}} \int_{t_n}^{s} \int_{t_n}^{u} L_1^1 L_1^1 \sigma_v \, dv \, dW(u) \, dW(s).
\end{aligned}
$$

Here, the coefficients β and σ involved in the above integrals are evaluated at the arguments $(X_1(u), u)$ and $(X_1(v), v)$, respectively. Similarly, by deterministic Taylor expansion for the local approximation of (3.3) in the framework of the splitting method, one gets

$$
X_{2,n+1} = X_{1,n+1} + \int_{t_n}^{t_{n+1}} \alpha(X_2(s), s) \, ds = X_{1,n+1} + \alpha(X_{1,n+1}, t_n) \Delta_n t + R_{2,n} \qquad (3.6)
$$

with the remainder term

$$
\begin{aligned}
R_{2,n} &= \int_{t_n}^{t_{n+1}} \int_{t_n}^{s} L_2^0 \alpha_u \, du \, ds \\
&= \int_{t_n}^{t_{n+1}} \int_{t_n}^{s} \left[\frac{\partial \alpha(X_2(u), u)}{\partial u} + \alpha(X_2(u), u) \frac{\partial \alpha(X_2(u), u)}{\partial x} \right] du \, ds.
\end{aligned}
$$

Here, the coefficients α involved in the above integrals are evaluated at the arguments $(X_2(u), u)$. An expansion of $\alpha(X_{1,n+1}, t_n)$ with respect to space-variable x gives

$$
\alpha(X_{1,n+1}, t_n) = \alpha_n + \int_{t_n}^{t_{n+1}} L_1^0 \alpha(X_1(s), s) \, ds + \int_{t_n}^{t_{n+1}} L_1^1 \alpha(X_1(s), s) \, dW(s). \qquad (3.7)
$$

Now, plug expansions (3.7) and (3.5) into (3.6) in order to obtain

$$
X_{2,n+1} = X_n + [\alpha_n + \beta_n] \Delta_n t + \sigma_n \Delta_n W + \frac{1}{2} L_1^1 \sigma_n [(\Delta_n W)^2 - \Delta_n t] + R_{3,n} \qquad (3.8)
$$

with the remainder term

$$
R_{3,n} = R_{1,n} + R_{2,n} + \left[\int_{t_n}^{t_{n+1}} L_1^0 \alpha_s \, ds + \int_{t_n}^{t_{n+1}} L_1^1 \alpha_s \, dW(s) \right] \Delta_n t.
$$

Consequently, the local pathwise error $\varepsilon_{n+1}^{loc} = X_{n+1} - X_{2,n+1}$ can be represented by

$$\begin{aligned}
\varepsilon_{n+1}^{loc} &= X_{n+1} - X_{2,n+1} = R_{0,n} - R_{3,n} \\
&= \int_{t_n}^{t_{n+1}} \int_{t_n}^{s} L_0^0(\alpha_u + \beta_u)\, du\, ds + \int_{t_n}^{t_{n+1}} \int_{t_n}^{s} L_0^1(\alpha_u + \beta_u)\, dW(u)\, ds \\
&\quad + \int_{t_n}^{t_{n+1}} \int_{t_n}^{s} L_0^0 \sigma_u\, du\, dW(s) + \int_{t_n}^{t_{n+1}} \int_{t_n}^{s} \int_{t_n}^{u} L_0^0 L_0^1 \sigma_v\, dv\, dW(u)\, dW(s) \\
&\quad + \int_{t_n}^{t_{n+1}} \int_{t_n}^{s} \int_{t_n}^{u} L_0^1 L_0^1 \sigma_v\, dW(v)\, dW(u)\, dW(s) \\
&\quad - \int_{t_n}^{t_{n+1}} \int_{t_n}^{s} L_1^0 \beta_u\, du\, ds - \int_{t_n}^{t_{n+1}} \int_{t_n}^{s} L_1^1 \beta_u\, dW(u)\, ds \\
&\quad - \int_{t_n}^{t_{n+1}} \int_{t_n}^{s} L_1^0 \sigma_u\, du\, dW(s) - \int_{t_n}^{t_{n+1}} \int_{t_n}^{s} \int_{t_n}^{u} L_1^0 L_1^1 \sigma_v\, dv\, dW(u)\, dW(s) \\
&\quad - \int_{t_n}^{t_{n+1}} \int_{t_n}^{s} \int_{t_n}^{u} L_1^1 L_1^1 \sigma_v\, dv\, dW(u)\, dW(s) - \int_{t_n}^{t_{n+1}} \int_{t_n}^{s} L_2^0 \alpha_u\, du\, ds \\
&\quad - \left[\int_{t_n}^{t_{n+1}} L_1^0 \alpha_s\, ds + \int_{t_n}^{t_{n+1}} L_1^1 \alpha_s\, dW(s) \right] \Delta_n t. \qquad (3.9)
\end{aligned}$$

Now recall that α, β, σ have exclusively uniformly bounded derivatives and their second moments along the solution of (3.1) are bounded on $[0, T]$. Therefore, all operators L_i^j applied to coefficients α, β, and σ have images with uniformly bounded second moments. This implies that $\mathbb{E}\,\varepsilon_n^{loc} = \mathcal{O}([\Delta_n t]^2)$ and $\mathbb{E}\,(\varepsilon_n^{loc} - \mathbb{E}\,\varepsilon_n^{loc})^2 = \mathcal{O}([\Delta_n t]^3)$. For mean square convergence, it remains to apply fairly general convergence theorems as known from [31, 42, 43] with local rates $r_1 = 2$ and $r_2 = 1.5$ in order to conclude the global strong (i.e., in the L^2-sense) convergence order $r = 1.0$. More precisely, we have

$$\max_{n=0,1,\ldots,N} \Big(\mathbb{E}\,|X(t_n) - \tilde{X}(t_n)|^2 \Big)^{1/2} = \mathcal{O}(\Delta)$$

provided that the initial errors $\mathbb{E}\,|X(0) - \tilde{X}(0)|^2 = \mathcal{O}(\Delta^2)$ started at L^2-integrable initial values which are independent of the σ-algebra generated by the underlying Wiener process W. Eventually, the global order $r_w = 1.0$ of weak convergence with respect to smooth, polynomially bounded test functions is rather obvious from the fact that the local weak rate $r_w = 2.0$ (exploiting standard techniques known from [17] and [31] on weak convergence analysis). Consequently, the splitting-step algorithm has the claimed convergence orders, and the proof is complete. □

4. First illustrative numerical experiments. In this section we give several illustrative and important examples of applying our new stochastic numerical schemes to SDEs and test the rate of convergence obtained in section 3.

4.1. Using the transition probability. Consider the splitting-step method applied to the SDE

$$dX(t) = (1 + X(t))\, dt + 2\sqrt{X(t)}\, dW(t) \qquad (4.1)$$

with $\alpha(X,t) = 1 + X$, $\beta(X,t) = 0$, and $\sigma(X,t) = 2\sqrt{X}$. The conditional transition probability for the stochastic part of the splitting algorithm is sampled using (2.6).

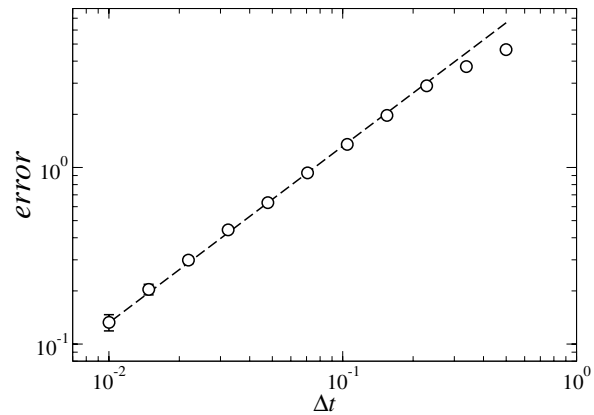


FIG. 4.1. *Absolute value of the error* (4.3) *as a function of* Δt *for* $t = 1.0$ *and* $X(0) = 1$ *obtained using the splitting-step algorithm. The dashed line is proportional to* Δt.

The conditional mean value is given by

$$\mathbb{E}\,(X(t)|X(0) = x_0) = (x_0 + 1)e^t - 1 \tag{4.2}$$

for any nonrandom value $x_0 \geq 0$. In Figure 4.1 the error

$$\varepsilon_1 = |\mathbb{E}\,\tilde{X}(t) - [(x_{0-1})e^t - 1]| \tag{4.3}$$

versus the decreasing uniform step size Δt is depicted, where $\tilde{X}(t)$ is the solution obtained through the numerical approximation. Figure 4.1 shows statistical numerical evidence that the method has weak order 1.0 while using constant step sizes Δt.

4.2. Using the exact solution. Consider the splitting-step algorithm applied to

$$dX(t) = (X(t) - [X(t)]^3)\,dt + X(t)\,dW(t), \tag{4.4}$$

which is called the stochastic planar Ginzburg–Landau equation. In this case we take advantage of the known exact solution for the linear part of this equation which is

$$dX_1(t) = X_1(t)\,dt + X_1(t)\,dW(t) \Rightarrow X_1(t) = X_1(s)\exp\Big((t-s)/2 + W(t) - W(s)\Big). \tag{4.5}$$

To integrate the second part of the split system

$$dX_2(t) = -[X_2(t)]^3\,dt \tag{4.6}$$

we use a partial-implicit nonstandard technique which is a nonnegativity preserving one[1] given by

$$X_2(t + \Delta t) = X_2(t) - \frac{\Delta t}{2}X_2^2(t)[X_2(t) + X_2(t + \Delta t)]. \tag{4.7}$$

[1]This is true if current step size $\Delta_n t$ is chosen such that $[X_2(t_n)]^2\Delta_n t < 2$.

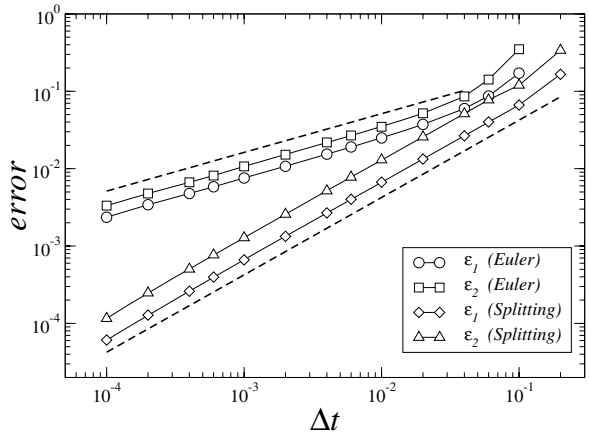


FIG. 4.2. *Value of the error* (4.8) *as a function of* Δt *for* $t = 5.0$ *and* $X(0) = 1.0$ *using the splitting-step and Euler algorithms. Dashed lines are the power laws* Δt *and* $\sqrt{\Delta t}$.

Our results for the strong error

$$\varepsilon_k(t) = (\mathbb{E}\,|X(t) - \tilde{X}(t)|^k)^{1/k} \tag{4.8}$$

are shown in Figure 4.2 for $k = 1, 2$ and compared to the Euler algorithm for the same equation. Obviously, our results indicate that our proposed splitting method indeed is of strong order 1.0 while using constant step sizes Δt.

5. Stochastic models in finance. In this section we apply the splitting-step method to some fundamental models in mathematical finance. This application will also serve to introduce other algorithmically simple samplings of conditional probability transitions as for the $\sqrt{X_1(t)}$ case shown in the introduction.

5.1. Interest rate model of Cox, Ingersoll, and Ross. An interesting example in which the splitting-step scheme is particularly efficient is the Cox–Ingersoll–Ross (CIR) model [5] for stochastic interest rates

$$dX(t) = [a + bX(t)]\,dt + \sigma\sqrt{X(t)}\,dW(t), \quad X(0) = x_0 \geq 0 \tag{5.1}$$

with real parameters $a \geq 0$, $b \in \mathbb{R}$, and $\sigma > 0$. As stated in the introduction, strong solutions of (5.1) are nonnegative. However, depending on the specific values of parameters a, b, and σ, distinct behavior at the boundary is possible.

- If $a \geq \sigma^2/2$, then the solution is always positive. $X(t) > 0$ if $x_0 > 0$, because the boundary $X_(t) = 0$ becomes unattainable.
- If $a < \sigma^2/2$, there are infinite many values of $t > 0$ for which $X(t) = 0$. The boundary becomes attainable, but it is (instantaneously) reflecting. That is, when a sample path reaches 0, then it returns immediately to the interior of the state space in a reflecting manner.

The exact transition density for the CIR process is known, but its sampling can be difficult depending on the parameters a, b, and σ. Here we may exploit the simplest splitting which appropriately reflects the boundary behavior of the CIR process by SDEs

$$dX_1(t) = a\,dt + \sigma\sqrt{X_1(t)}\,dW(t), \tag{5.2}$$

$$dX_2(t) = bX_2(t)\,dt, \tag{5.3}$$

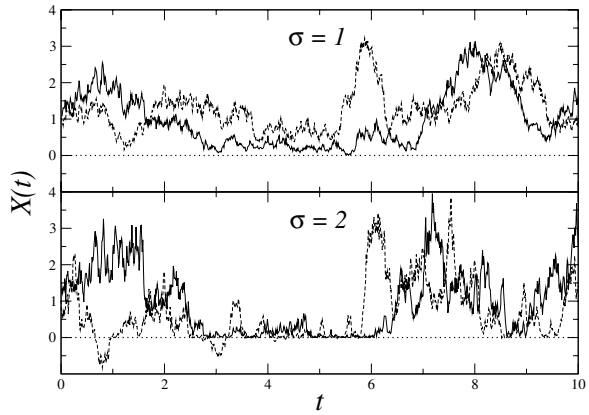


FIG. 5.1. *Numerical approximation of the CIR process* (5.1) *using the splitting-step method* (5.2) *(solid line) and the method proposed in* [21] *(dashed line) with* $a = b = 1$, $\Delta t = 10^{-2}$, $X(0) = 1$, *and different values of* σ*: The upper panel shows the case* $a \geq \sigma^2/2$ *where the boundary at zero becomes unattainable, while the lower panel is for* $a < \sigma^2/2$. *The splitting-step numerical approximation reproduces the attainability of the boundary condition while preserving the nonnegativity of the solution of* (5.1), *whereas the method in* [21] *has trajectories which take negative values for* $X(t)$.

which can be easily inferred by noting that the boundary behavior does not depend on the parameter b. To integrate the first step in the splitting-step system (5.2)–(5.3) we note that the process defined by (5.2) represents an a-dimensional Bessel process [17]. Its transition density $\mathcal{P}[X_1(t+\Delta t)|X_1(t)]$ can be written in terms of a noncentral χ^2 distribution and, in particular, we have

$$X_1(t+\Delta t) = \frac{\sigma^2 \Delta t}{4} \chi'^2_d(\lambda), \tag{5.4}$$

where

$$\lambda = \frac{4X_1(t)}{\sigma^2 \Delta t}, \qquad d = \frac{4a}{\sigma^2}, \tag{5.5}$$

and $\chi'^2_d(\lambda)$ random numbers can be sampled using the algorithms in the appendix. The last part of the splitting-step scheme (5.3) can be integrated exactly or using numerical approximations. In our case we use deterministic Euler approximations where nonnegativity is preserved if Δt is small enough. Our simulations for the CIR process are shown in Figure 5.1. The boundary behavior for different values of a and σ is reproduced and, at the same time, nonnegativity is conserved. Note that other usual integration strategies such as that of [21] eventually produce negative values for $X(t)$ (for finite Δt) which lack any possible interpretation in the context of finance and could induce severe errors in option valuation.

5.2. Constant elasticity volatility models. Another important stochastic process in finance is the constant elasticity of variance (CEV) diffusion to model asset prices. This process, first introduced to finance by Cox [4], is capable of reproducing the volatility smile observed in the empirical data unlike other standard price models such as the Black–Scholes–Merton geometric Brownian motion. The process is defined as

$$dX(t) = \mu X(t)\, dt + \sigma X(t)^\gamma\, dW(t), \quad X(0) = x_0. \tag{5.6}$$

The CEV model includes the geometric Brownian model of Black, Scholes, and Merton ($\gamma = 1$) and the square-root model of Cox and Ross ($\gamma = 1/2$). Contrary to the Black–Scholes–Merton model, the CEV model incorporates a variance adjustment that causes the absolute level of the variance to decline as the stock price rises and to rise as the stock price declines, as seen empirically in most equity and interest rate volatilities.

For our purposes, the CEV model encompasses most of the boundary conditions we can implement at $X(0) = 0$. Depending on γ, the boundary $X(0) = 0$ is the following:

- a [illegible] for $\gamma \geq 1$, which means that boundary is unattainable in finite time;
- an [illegible] for $1/2 \leq \gamma < 1$; i.e., $X(t)$ reaches zero with finite probability in finite time and gets absorbed at it;
- a [illegible] for $\gamma < 1/2$. Now the boundary can be reached in finite time, and we need to specify a boundary condition at $X(0) = 0$. Typical choices are reflecting or absorbing ones.

The transition density for the CEV process is known for general values of γ and μ (see [4]). However, due to the fact that the boundary behavior of $X(t)$ does not depend on μ, we propose here the following splitting-step system:

$$dX_1 = \sigma X_1^{\gamma}\, dW(t), \tag{5.7}$$

$$dX_2 = \mu X_2\, dt. \tag{5.8}$$

The classification of the boundary $X_1(t) = 0$ for (5.7) is the same as for (5.6). We discuss now the sampling of the transition density for each value of γ.[2]

5.2.1. $\gamma > 1$, natural boundary. In this case $X_1(t) = 0$ is unattainable and its probability transition density is given by

$$\mathcal{P}[x_t|x_0] = \frac{x_t^{1/2-2\gamma} x_0^{1/2}}{\sigma^2(\gamma-1)} \exp\left[-\frac{x_0^{2(1-\gamma)} + x_t^{2(1-\gamma)}}{2\sigma^2(1-\gamma)^2 t}\right] I_{\frac{1}{2(\gamma-1)}}\left[\frac{x_0^{1-\gamma} x_t^{1-\gamma}}{\sigma^2(1-\gamma)^2 t}\right]. \tag{5.9}$$

To sample this distribution, we can consider the relationship between the CEV and the CIR processes. Since $X_1(t) = 0$ is not accessible, we can make the change of variables $Y_1 = X_1^{2(1-\gamma)}/[4(\gamma-1)^2\sigma^2]$ to find that $dY_1 = \lambda\, dt + \sqrt{Y_1}\, dW(t)$ where $\lambda = (1-2\gamma)/[4(1-\gamma)]$. Thus Y_1 is a Bessel process which can be sampled using the noncentral χ^2 distribution. At the end we have that

$$X_1(t+\Delta t) = \left[(\gamma-1)^2\sigma^2 \Delta t\, \chi'^2_d\left(\frac{X_1(t)^{2(1-\gamma)}}{\sigma^2(\gamma-1)^2\Delta t}\right)\right]^{\frac{1}{2(1-\gamma)}}, \tag{5.10}$$

where $d = (1-2\gamma)/(1-\gamma) \geq 1/2$. A typical path of the CEV process for $\gamma > 1$ is shown in Figure 5.2.

5.2.2. $1/2 \leq \gamma < 1$, absorbing boundary. In this case $X_1(t) = 0$ is an absorbing boundary. The transition probability density was found by Cox [4] and is given by

$$\mathcal{P}[x_t|x_0] = \frac{x_t^{1/2-2\gamma} x_0^{1/2}}{\sigma^2(1-\gamma)} \exp\left[-\frac{x_0^{2(1-\gamma)} + x_t^{2(1-\gamma)}}{2\sigma^2(1-\gamma)^2 t}\right] I_{\frac{1}{2(1-\gamma)}}\left[\frac{x_0^{1-\gamma} x_t^{1-\gamma}}{\sigma^2(1-\gamma)^2 t}\right]. \tag{5.11}$$

[2]We do not consider the case $\gamma = 1$, since this can be integrated using the strong solution (4.5).

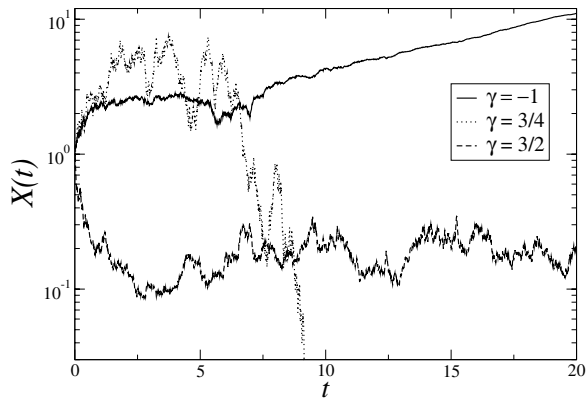


FIG. 5.2. *Numerical approximation of the CEV process* (5.6) *using the splitting-step method* (5.7)–(5.8) *for different values of* γ. *The case* $\gamma = -1$ *is the solution of* (5.6) *with reflecting boundary conditions, while the case* $\gamma = 3/4$ *gets absorbed at zero at finite time (not shown in the logarithmic scale). Parameters are* $\mu = 0.1$, $\Delta t = 10^{-2}$.

This transition probability does not integrate to one because there is a finite probability that the trajectory gets absorbed at zero given by [4]:

$$\mathcal{P}[0|X_1(t)] = G\left(\frac{1}{2(1-\gamma)}, \frac{x_1(t)^{-2(1-\gamma)}}{2\sigma^2(1-\gamma)^2\Delta t}\right), \tag{5.12}$$

where $G(\nu, x)$ is the complementary Gamma function. For this equation, we know that trajectories of (5.7) get absorbed at zero with probability one while taking the limit $\Delta t \to \infty$. Actually, this is also the case for the solutions of (5.6) for $1/2 \le \gamma < 1$. This is counterintuitive since the mean $\mathbb{E}[X_1(t)]$ is constant for (5.7) or grows exponentially like $x_0 e^{\mu t}$ for (5.6). This intriguing feature of (5.6) hampers the numerical simulations of this process, and in fact only exact sampling of the probability transition density (5.11) and (5.12) correctly accounts for it at finite Δt.

While similar to (5.9), the transition probability density (5.11) cannot be sampled using the relationship to the CIR process, since the solution of $dX_1 = X_1^\gamma dW(t)$ has a finite probability to be absorbed at $X_1(t) = 0$. However, using the relationship $I_n(x) = I_{-n}(x)$, $n \in \mathbb{N}$, for the modified Bessel function, we have that if $1/[2(1-\gamma)] = n$, X_1 can be sampled from a noncentral χ^2 distribution with negative (integer) number of degrees of freedom (see the appendix):

$$X_1(t+\Delta t) = \left[(\gamma-1)^2\sigma^2\Delta t\, \chi'^2_d\left(\frac{X_1(t)^{2(1-\gamma)}}{\sigma^2(\gamma-1)^2\Delta t}\right)\right]^{\frac{1}{2(1-\gamma)}}, \tag{5.13}$$

where $d = 2 - 2n = 0, -2, -4, \ldots$, and then $\gamma = 1 - 1/2n = 1/2, 3/4, 5/6, \ldots$. In Figure 5.2 we show a typical path for the CEV process for $\gamma = 3/4$ ($d = -2$) in which we see that it gets absorbed at zero for finite time.

5.2.3. $\gamma < 1/2$, regular boundary. Now the boundary is attainable and, contrary to the previous case, a boundary condition should be provided. If we consider the simplest cases, namely, an absorbing or reflecting boundary condition, then the transition probability density is given by

$$\mathcal{P}_\pm[x_t|x_0] = \frac{x_t^{1/2-2\gamma}x_0^{1/2}}{\sigma^2|1-\gamma|}\exp\left[-\frac{x_0^{2(1-\gamma)}+x_t^{2(1-\gamma)}}{2\sigma^2(1-\gamma)^2 t}\right] I_{\pm\frac{1}{2|1-\gamma|}}\left[\frac{x_0^{1-\gamma}x_t^{1-\gamma}}{\sigma^2(1-\gamma)^2 t}\right], \tag{5.14}$$

where $\mathcal{P}_+$ corresponds to the absorbing case and $\mathcal{P}_-$ to the reflecting one. In the latter case, the relationship of the CEV and CIR processes can be exploited to sample the probability distribution (5.14), with the same result as in (5.10). Simulation of the CEV process in this case is shown in Figure 5.2. However, in the absorbing case, $\mathcal{P}_-$ cannot be sampled either by using the corresponding CIR process or by using [illegible] as in the previous case, since for $\gamma < 1/2$ we have that $1 < 1/[2(\gamma-1)] \leq 0$. Thus $\mathcal{P}_-$ should be sampled using rejection or transformation methods.

6. Measure valued diffusions / reaction-diffusion problems. During the last decade much attention has been devoted to stochastic spatial models of interacting and branching particles like the contact model, the voter model, and the normal and oriented percolation [45, 23, 7]. Despite its simplicity these models display interesting critical properties at high spatial dimensions and serve as universality classes for more complicated situations. While much analytical progress has been made in the study of these models, some of their properties must be understood by using numerical methods. In this respect, several efficient particle stochastic simulations have been proposed and studied. An alternative approach is to study the convergence of the particle process to a continuum measure value diffusion in which the system is described by the (stochastic) concentration of particles $u(x,t)$ at a given spatial location x. This accelerates the numerical simulations and also could help to identify the relevant dynamics at the coarse-grained dynamics. Useful representations of these measure-valued diffusions are the ones interpreted as solutions of a martingale problem in terms of stochastic PDEs (SPDEs). Despite its clear interpretation, this representation is not usually considered in numerical simulations. The reason for that is the inability of usual numerical methods to handle correctly the nonnegativity character and the Poissonian fluctuations of the concentration of particles close to $u = 0$ [34, 18]. In this respect, we will see that the splitting scheme provides a very efficient and accurate method to study these models.

6.1. Super-Brownian motion. The most simple and studied measured-valued diffusion is the super-Brownian motion (sBm). The sBm arises as the scaling limit in various critical branching systems when the interaction between them is weak, i.e., when the system is above some [illegible] [45]. Above this critical dimension we expect a Gaussian limit and indeed the sBm is the Gaussian limit of a number of models: the voter model above two dimensions, the contact process above four dimensions, oriented percolation above four dimensions, and percolation over six dimensions.

sBm can be studied analytically by using the log-Laplace transform that maps its dynamics into a nonlinear PDE, a result due to Dynkin [12]. More general situations or particular properties of the sBm can be reached only through numerical simulations. To our knowledge there is no numerical simulation of the martingale problem of the sBm so far. In one dimension, the martingale problem of the sBm is described by the SPDE

$$du(x,t) = \Delta u(x,t)\,dt + \sqrt{\sigma u(x,t)}\,dW(x,t), \tag{6.1}$$

where $W(x,t)$ is a Wiener sheet. It is well known that the solutions of the sBm die in finite time almost surely. Another interesting property is that the support of the solution, i.e., the set for which $u(x,t) > 0$, is compact, provided that the initial condition has a compact support. This compact support property is related to the absorbing nature of $u(x,t) = 0$. Thus it can only be recovered in the numerical

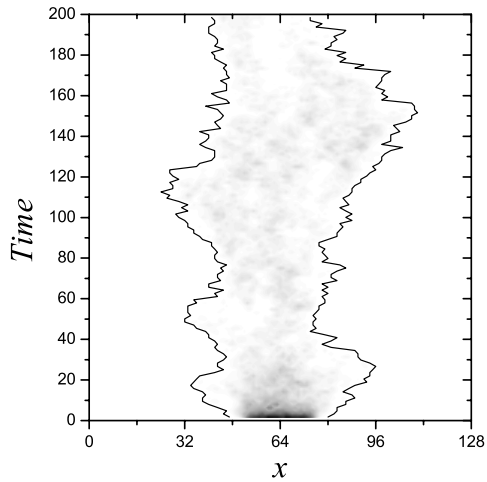


FIG. 6.1. *Realization of the sBm in one dimension. The figure shows density plots for $u(x,t)$ as a function of time with initial condition $u(x,0) = 0.1$. The solid line depicts the extremes of the support at each time. Parameters are $\Delta x = 1$, $\Delta t = 0.1$, and $\sigma = 1.0$ in an $L = 128$ lattice.*

approximations if the behavior of the boundary is incorporated. For example, the method of Gaines [18], while implementing the nonnegativity of the solutions, fails to reproduce the compact support property for the solutions of (6.1).

We use the splitting-step method for approximating the strong path solutions of (6.1). To this end we approximate the sBm by the superrandom walk on $\mathbb{Z}^d$ (see [7])

$$du_i(t) = \Delta_i u_i(t)\, dt + \sqrt{\frac{\sigma u_i(t)}{(\Delta x)^d}}\, dW_i(t), \tag{6.2}$$

where Δ_i is the discrete Laplacian operator

$$\Delta_i u = \frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta x)^2}$$

in $\mathbb{Z}^d$, Δx is the lattice spacing, and $W_i(t)$ are independent Wiener processes in time t. Equation (6.2) represents a model of interacting Feller diffusions. The splitting-step algorithm in this case is based on

$$du_i^{(1)}(t) = \sqrt{\frac{\sigma u_i^{(1)}(t)}{(\Delta x)^d}}\, dW_i(t), \quad du_i^{(2)}(t) = \Delta_i u_i^{(2)}, \tag{6.3}$$

where the first step is integrated using the transition probability (2.5) and its sampling (2.6). The second equation can be integrated using standard schemes. In Figure 6.1 we observe a simulation of the sBm in one dimension. Our method not only provides an accurate description in both the strong and weak sense of the sBm; it also incorporates one of its main properties, namely, the compact support property.

In two dimensions (6.1) is not well defined, but still we can study (6.2) in the lattice $\mathbb{Z}^2$. As our simulations show (see Figure 6.2), the compact support properties of sBm in two dimensions are preserved, and we also see the clusters' formation and their disappearance at large times.

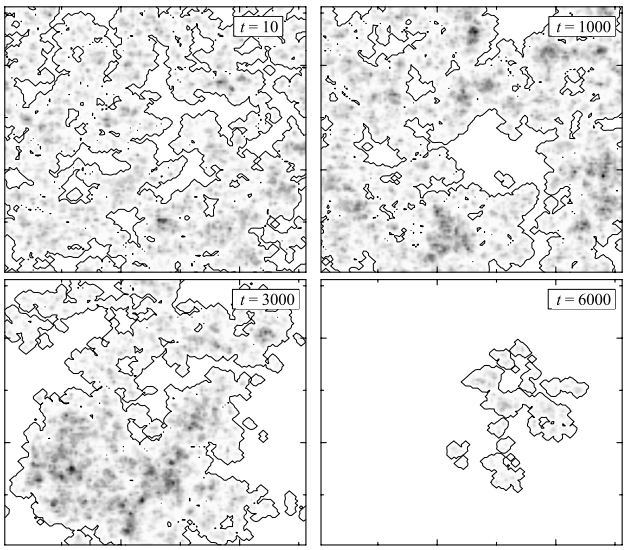


FIG. 6.2. *Strong path approximation of the sBm in two dimensions. The figures show density plots of $u(x,t)$ surrounded by the border of the support (solid line) at different times. The initial condition is $u(\vec{r},0) = 0.1$. Parameters are $\Delta x = 1$, $\Delta t = 0.1$ in a 256×256 lattice.*

6.2. Contact process. The contact process is a model of the spreading of infection in a lattice in which a site can be infected via contact with an infected site in its neighborhood [45]. Infection and recovery happen at different rates and there is a critical value of them for which an infection started from a single infected individual will die out in finite time. The contact process with a finite range of infection has a critical dimension $d_c = 4$. At higher dimensions the critical contact process converges to the sBm, but not below d_c. At lower dimensions, it is not known what the scaling limits should be.

However, long-range interaction with suitable scalings shows that the contact process converges to the sBm for $d \geq 2$; see [11]. In one dimension, Mueller and Tribe [35, 36] showed that the rescaled density of particles for the long-range contact process weakly converges to the solution of the SPDE

$$du(x,t) = [\Delta u(x,t) + \theta u(x,t) - u(x,t)^2]\, dt + \sqrt{u}\, dW(x,t), \tag{6.4}$$

where $W(x,t)$ denotes a space-time Wiener process. Moreover, they showed that the above equation undergoes a phase transition at a critical value of θ_c for which

$$\mathcal{P}(u(x,t) \text{ survives}) \begin{cases} = 0 & \text{if } \theta < \theta_c, \\ > 0 & \text{if } \theta > \theta_c. \end{cases} \tag{6.5}$$

The nontrivial behavior of the solution of (6.4) is believed to represent the well-known universality class for contact processes (also named the directed percolation universality class [23, 34]).

As before, numerical simulations of (6.4) can be now addressed using the splitting scheme and the efficient random number generators for the conditional probability [34]. In particular, we discretize the spatial operators in a lattice $\mathbb{Z}$ (as in (6.2)) and split the dynamics as follows:

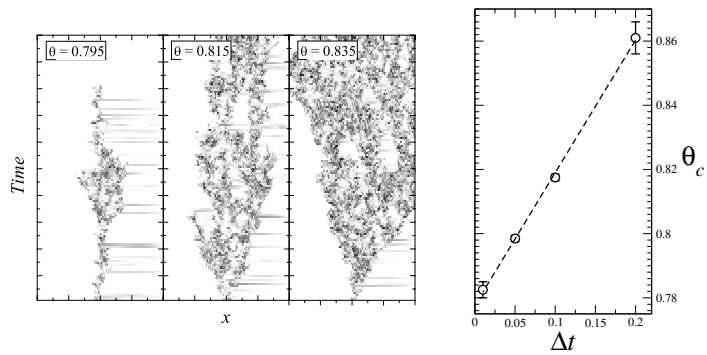


FIG. 6.3. *(Left) Strong path approximation of* (6.4) *for* $\Delta x = 1$ *and* $L = 2^{14}$ *and different values of* θ *below and above the phase transition. (Right) Critical value of* θ_c *as defined in* (6.5) *as a function of* Δt*. The straight line is a linear fit to the data. Parameters are* $\Delta x = 1$*,* $L = 2^{14}$*.*

$$du_i^{(1)} = \sqrt{u_i^{(1)}}\, dW(t), \quad du_i^{(2)} = [\Delta_i u_i^{(2)} + \theta u_i^{(2)} - (u_i^{(2)})^2]\, dt, \tag{6.6}$$

where the last equation is numerically integrated using Euler approximations with sufficiently small step sizes. Results for strong approximations of $u_i(t)$ are shown in Figure 6.3, where we can see a typical realization for the subcritical (infection dies out), critical, and supercritical (infection spreads) for different values of θ.

We can calculate the critical value of θ_c in one dimension using our algorithm. To this end, we identify the critical point θ_c using (6.5) and finite scaling techniques of statistical mechanics. In particular, we found that $\theta_c = 0.777 \pm 0.001$ for $\Delta x = 1$, and the convergence to this value is of order one.

7. Conclusion. The general idea of this paper is to propose a splitting of SDE (1.1) into two new SDEs for which one can keep nonnegativity during integration of both subsystems. This is achieved by either solving one subsystem exactly in the pathwise sense or using its transition probabilities, and solving the other (here nonrandom) subsystem by nonnegativity preserving numerical methods. In this way one is able to preserve nonnegativity and a maximum of convergence order 1.0 in both weak and strong sense. For the efficiency of our splitting algorithm, it is crucial to find a splitting into appropriate subsystems. For this purpose, one also tries to incorporate more complicated boundary conditions of the original SDE (1.1) into an explicitly solvable subsystem such that a fairly easier numerical integration of the remaining subsystem remains to be implemented. For example, for SDEs

$$dX(t) = [\alpha(X(t), t) + \lambda X(t)]\, dt + \sigma X(t)\, dW(t),$$

one makes use of the splitting into

$$dX_1(t) = \lambda X_1(t)\, dt + \sigma X_1(t)\, dW(t), \quad dX_2(t) = \alpha(X_2(t), t)\, dt,$$

where the explicit solution of the first component X_1 is given by

$$X_1(t) = X_1(0) \exp\left(\left(\lambda - \frac{1}{2}\sigma^2\right) t + \sigma W(t)\right),$$

which possesses the monotone property of leaving the positive axes $[0, +\infty)$ invariant by this type of random mapping (almost surely). This idea can be easily extended to

nonlinear systems of SDEs with its splitting into linear and nonlinear subsystems in several dimensions. Another type of splitting is found for nonlinear systems

$$dX(t) = f(X(t),t)\,dt + \sigma(X(t),t)\,dW(t) \tag{7.1}$$

with continuously differentiable coefficients σ as follows. Rewrite this equation as

$$dX(t) = \left[f(X(t),t) - \frac{1}{2}\sigma(X(t),t)\frac{\partial\sigma(X(t),t)}{\partial x} + \frac{1}{2}\sigma(X(t),t)\frac{\partial\sigma(X(t),t)}{\partial x}\right]dt + \sigma(X(t),t)\,dW(t)$$

and set

$$\alpha(x,t) = f(x,t) - \frac{1}{2}\sigma(x,t)\frac{\partial\sigma(x,t)}{\partial x}, \quad \beta(x,t) = \frac{1}{2}\sigma(x,t)\frac{\partial\sigma(x,t)}{\partial x}.$$

Then the splitting-step algorithm is applied to system (X_1, X_2) satisfying (2.2) and (2.3) with coefficients α and β as defined above. This works at least efficiently if $\sigma(x,t) = \sigma(x)$ does not depend on t and the invertible integral $H(z) = \int^z [b(z)]^{-1}\,dz$ exists on the domain of definition of the original equation (7.1) for X. In this case one finds $X_1(t) = H^{-1}(W(t) + H(X_1(0)))$.

Once an appropriate splitting is found then it is relatively easy to implement the related numerical algorithm. The proposed splitting-step method efficiently works since its implementation essentially relies on the well-known variation-of-parameters formula for perturbed dynamical systems which extends to SDEs. Recall that, by this formula, if the equation

$$dX(t) = \beta(X(t),t)\,dt + \sigma(X(t),t)\,dW(t)$$

has known fundamental solution $\Phi = \Phi(t, X_0)$ then the exact solution of the original equation (2.1) possesses the pathwise representation

$$\begin{aligned} X(t+\Delta t) &= \Phi(t+\Delta t, X(t)) + \Phi(t+\Delta t, X(t)) \int_t^{t+\Delta t} [\Phi(s, X(t))]^{-1}\alpha(X(s),s)\,ds \\ &\approx \Phi(t+\Delta t, X(t)) + \alpha(X(t),t)\Delta t \end{aligned}$$

on each subinterval $[t, t+\Delta t] \subset [0,T]$. Thus, the motivation of our splitting-step technique is apparent by finding Φ and numerical integration of $\int \alpha(X(s),s)\,ds$.

Final remark on applicability of our splitting algorithm. The applicability of our splitting algorithm heavily relies on the knowledge on the exact solution of SDEs. An extensive list of analytically known solutions is found in textbooks such as [27]. Of course, in one dimension one knows more about explicitly solvable SDEs. However, our technique is also applicable to multidimensional problems. For example, when noise terms are linear, decoupled, commutative, or diagonal, one can apply it successfully, too. In particular, the knowledge of explicitly solvable linear multidimensional stochastic subsystems from appropriately split systems can be exploited (see also Appendix C). So we are not exclusively restricted to one-dimensional problems (although we must admit that the range of highly efficient applicability may shrink with the increase of dimensions). A more detailed discussion on multidimensional extensions of splitting techniques is left to research in the future. We have deliberately chosen some well-known examples from the literature such that the interested

reader may easily compare the performance of his preferred numerical technique to our semianalytic numerical splitting technique.

Appendix A. Noncentral chi-square distribution random number generation. The splitting-step method proposed in this paper relies on the exact numerical sampling of transition probability density for some processes. In particular, efficient generation of random numbers $\chi'^2_d(\lambda)$ with a noncentral chi-square distribution with d degrees of freedom whose probability density function is found by

$$\mathcal{P}[\chi'^2_d(\lambda) = x] = p(x; d, \lambda) = \frac{e^{-(\lambda+x)/2}}{2} \left(\frac{x}{\lambda}\right)^{(d-2)/4} I_{(\nu-2)/2}(\sqrt{\lambda x}), \quad x > 0, \tag{A.1}$$

in [24]. This distribution is properly defined for any positive d and was extended to the case $d = 0$ by Siegel [44]. Here we shall extend it to the case $d = -2, -4, \ldots$ and show how to sample this distribution.

To this end, we use the fact that the distribution (A.1) can be expressed also as a mixture of central χ^2 variables with Poisson weights

$$p(x; d, \lambda) = \sum_{j=0}^{\infty} \frac{(\lambda/2)^j e^{-\lambda/2}}{j!}\, p_0(x; d + 2j), \quad d > 0, \tag{A.2}$$

where $p_0(x; d)$ is the distribution of a chi-square random variable χ^2_d with d degrees of freedom. This expression suggests a simple and efficient procedure to obtain $\chi'^2_d(\lambda)$ random variables:

1. Choose K from a Poisson distribution with mean $\lambda/2$ so that $\mathcal{P}[K = k] = e^{-k/2}(\lambda/2)^k/k!$ $(k = 0, 1, \ldots)$.
2. Then take $\chi'^2_d(\lambda) = \chi^2_{d+2K}$, which can be done using the any standard random number generator of the χ^2_d distribution.

In the $d = 0$ case, the $\chi'^2_d(\lambda)$ distribution has a discrete component at zero with mass $e^{\lambda/2}$ (which represents the probability to get absorbed at zero in our stochastic processes); see [44]. We have

$$p(x; 0, \lambda) = \sum_{j=1}^{\infty} \frac{(\lambda/2)^j e^{-\lambda/2}}{j!}\, p_0(x; 2j) + e^{-\lambda/2}\delta(x), \quad d = 0. \tag{A.3}$$

The procedure above can be modified to account for this discrete component by taking the convention that the central χ^2_d distribution is identically zero when $d = 0$. This convention can be extended to even negative dimensions to get

$$p(x; d, \lambda) = \sum_{j=|d|/2+1}^{\infty} \frac{(\lambda/2)^j e^{-\lambda/2}}{j!}\, p_0(x; d + 2j) + \delta(x) \sum_{j=0}^{|d|/2} \frac{(\lambda/2)^j e^{-\lambda/2}}{j!} \tag{A.4}$$

for $d = 0, -2, -4, \ldots$.

Summarizing, if K is a Poisson random number with mean $\lambda/2$ we have

$$\chi'^2_d(\lambda) = \chi^2_{d+2K}, \qquad d > 0, \tag{A.5}$$

and

$$\chi'^2_d(\lambda) = \begin{cases} 0 & \text{if } d + 2K \le 0, \\ \chi^2_{d+2K} & \text{if } d + 2K > 0, \end{cases} \qquad d = 0, -2, -4, \ldots. \tag{A.6}$$

This sampling of the probability distribution function is exact and should be used especially when λ is small. However, when λ is large, the $\chi'^2_d(\lambda)$ distribution asymptotically converges to the Gaussian distribution and other approximations (like the ones in [24]) might be considered to improve the speed of our algorithm.

Appendix B. Numerical generation of strong path solutions for some processes. The success of the splitting-step method requires the exact numerical integration of step 1 which can be problematic in some situations. That is why we have applied in previous numerical examples such equations which allow us to represent the solution from step 1 in terms of pure functions of the Wiener process and initial values such as $X(t) = F(X(0), W(t))$ or simple deterministic integrals as seen with the example of Bessel-type diffusions $X(t) = (\sqrt{X(0)} + W(t))^2$ or the geometric Brownian motion $X(t) = X(0)\exp((\alpha - \sigma^2/2)t + \sigma W(t))$. Also, for the Ornstein–Uhlenbeck processes or logistic equations, one may use the well-known integration-by-parts formula and / or the information on the exact distribution of involved stochastic integrals with deterministic differentials which can be pathwisely treated by deterministic quadrature methods. As another alternative, one could exploit the Doss representation (see [9]) to develop a semianalytic approach or an ODE-PDE approach to decompose the original problem into a random ODE and a deterministic PDE in order to figure out which numerical implementation is more efficient in conjunction with our splitting technique. However, in general one produces additional discretization errors which might influence the accuracy of the computations using the splitting algorithm as well. Then it is a must to consider its stability properties too. Such a fairly complex and very problem-dependent work is left to the future.

Appendix C. A general theorem on L^2-convergence based on the VOP. To exploit the specific knowledge on solvability of bilinear stochastic systems, once may also make use of the well-known variation-of-constants formula (VOP) (this formula is also sometimes called the variation-of-parameters formula). This possibility is justified through this section. For this purpose, suppose that the original equation is

$$dX(t) = [\alpha(X(t), t) + \beta(t)X(t)]\,dt + \sigma_2(X(t), t)\,dW_2(t) + \sigma_1(t)X(t)\,dW_1(t), \tag{C.1}$$

where W_1 and W_2 are independent Wiener processes. Consider the splitting

$$dX_1(t) = \beta(t)X_1(t)\,dt + \sigma_1(t)X_1(t)\,dW_1(t), \tag{C.2}$$

$$dX_2(t) = \alpha(X_2(t), t)\,dt + \sigma_2(X_2(t), t)\,dW_2(t) \tag{C.3}$$

with Borel-measurable real-valued matrices $\beta(t), \sigma_1(t) \in \mathbb{R}^{d\times d}$. Let $C^0_{locLip}(S)$ denote the vector space of local Lipschitz continuous functions on the open set S.

THEOREM C.1. [illegible] $\alpha, \sigma_2 \in C^0_{locLip}(\mathbb{D} \times [0,T])$ $\beta, \sigma_1 \in \mathbb{R}^{d\times d}$ [illegible] X [illegible] (C.1) $\bar{\mathbb{D}}$

$$\sup_{0\le t\le T} \mathbb{E}\left[|X(t)|^2\right] + \sup_{0\le t<T} \sup_{t\le s\le T} \mathbb{E}\left[|\Phi(s, X(t))|^2\right] < +\infty,$$

$$\sup_{0\le t\le T} \mathbb{E}\left[\int_t^T |[\Phi(s, X(t))]^{-1}\alpha(X(s), s)|^2\,ds + \int_t^T |[\Phi(s, X(t))]^{-1}\sigma_2(X(s), s)|^2\,ds\right] < +\infty$$

[illegible] $T > 0$ [illegible] Φ [illegible] (C.2) [illegible] $r_H \ge 0.5$ [illegible]

1 2 $r_1 \geq 1.0$ $r_2 \geq 0.5$ $\min(r_1, r_H + 1.0) \geq \min(r_2, r_H + 1.0) + 0.5$ 1 2

$$r_g \geq \min(1.0, \min(r_2, r_H + 1.0) - 0.5)$$

L^2 $[0,T]$. The VOP is needed for the proof. This formula for the perturbation of Itô-interpreted fundamental systems

$$dX(t) = A(t)X(t)\,dt + \sum_{j=1}^{m} B_j(t)X(t)\,dW_j(t) \tag{C.4}$$

with real-valued matrices $A(t)$ and $B(t) \in \mathbb{R}^{d\times d}$ by

$$dX(t) = [A(t)X(t) + f(t)]\,dt + \sum_{j=1}^{m}[B_j(t)X(t) + g_j(t)]\,dW_j(t) \tag{C.5}$$

with vector-valued random functions f and g_j can be found as Theorem 3.1 from Mao [30, pp. 96–97]. It says that systems (C.5) have the representation

$$\begin{aligned} X(s) = \Phi(s)\Bigg(X(t) + \int_t^s [\Phi(u)]^{-1}\left[f(u) - \sum_{j=1}^{m} B_j(u)g_j(u)\right]du \\ + \sum_{j=1}^{m}\int_t^s [\Phi(u)]^{-1} g_j(u)\,dW_j(u)\Bigg) \end{aligned} \tag{C.6}$$

in terms of the fundamental solution Φ related to linear system (C.4), where $s \geq t$. Thus, it heavily relies on the specific knowledge of properties of the flow Φ of underlying linear fundamental systems (C.4). Note that autonomous systems (C.4) with $m = 1$, constant matrices A, B, and commutativity condition $[A, B] = AB - BA = 0$ possess the explicit representation in terms of matrix exponentials (for $s \geq t$)

$$\Phi(s, X(t)) = \exp\left(\left(A - \frac{1}{2}B^2\right)(s-t) + B(W(s) - W(t))\right)X(t).$$

. Suppose that (C.2) has known fundamental solution $\Phi = \Phi(t, X_0)$ with a.s. Hölder exponent r_H. Then, following the VOP (C.6) (see the above remark), the exact solution of the original equation (C.1) possesses the pathwise representation

$$X(t+\Delta t) = \begin{cases} \Phi(t+\Delta t, X(t)) \\ + \Phi(t+\Delta t, X(t)) \displaystyle\int_t^{t+\Delta t} [\Phi(s, X(t))]^{-1}\alpha(X(s), s)\,ds \\ + \Phi(t+\Delta t, X(t)) \displaystyle\int_t^{t+\Delta t} [\Phi(s, X(t))]^{-1}\sigma_2(X(s), s)\,dW_2(s) \end{cases}$$

on each subinterval $[t, t+\Delta t] \subset [0, T]$. Note that formula (C.6) is applied with

$$\begin{gathered} A(s) = \beta(s), \quad B_1(s) = \sigma_1(s), \quad B_2(s) = 0, \\ f(s) = \alpha(X(s), s), \quad g_1(s) = 0, \quad g_2(s) = \sigma(X(s), s). \end{gathered}$$

Now, let $Y(t)$ denote the value of right-continuous numerical approximation of splitting step algorithm (which we always can construct using step functions in a standard way). The main idea is to apply the fairly general L^2-convergence theory known from [31, 41, 42, 43]. For this purpose, we need to study the local conditional accuracy of our algorithm. Locally, we may suppose that $X(t) = Y(t) = x$ is deterministic ($\mathcal{F}_t$-adapted). First, consider the local conditional mean accuracy. We find that

$$\begin{aligned}
&\left|\mathbb{E}\left[X(t+\Delta t) - Y(t+\Delta t)\middle|\mathcal{F}_t\right]\right| \\
&= \left|\mathbb{E}\left[\Phi(t+\Delta t,x)\int_t^{t+\Delta t}[\Phi(s,x)]^{-1}\alpha(X(s),s)\,ds\right.\right. \\
&\qquad \left.\left. + \Phi(t+\Delta t,x)\int_t^{t+\Delta t}[\Phi(s,x)]^{-1}\sigma_2(X(s),s)\,dW_2(s) - (Y(t+\Delta t)-x)\middle|\mathcal{F}_t\right]\right| \\
&\le \left|\mathbb{E}\left[\int_t^{t+\Delta t}\Big(\Phi(t+\Delta t,x)-\Phi(s,x)\Big)\,[\Phi(s,x)]^{-1}\alpha(X(s),s)\,ds\middle|\mathcal{F}_t\right]\right| \\
&\quad + \left|\mathbb{E}\left[\int_t^{t+\Delta t}\Big(\Phi(t+\Delta t,x)-\Phi(s,x)\Big)\,[\Phi(s,x)]^{-1}\sigma_2(X(s),s)\,dW_2(s)\middle|\mathcal{F}_t\right]\right| \\
&\quad + \left|\mathbb{E}\left[\int_t^{t+\Delta t}\alpha(X(s),s)\,ds + \int_t^{t+\Delta t}\sigma_2(X(s),s)\,dW_2(s) - (Y(t+\Delta t)-x)\middle|\mathcal{F}_t\right]\right| \\
&\le K_1[\Delta t]^{\min(r_1,(2r_H+1)/2+0.5)},
\end{aligned}$$

where K_1 is a real constant; hence the local conditional mean accuracy rate

$$\min(r_1,(2r_H+1)/2+0.5) = \min(r_1, r_H+1.0)$$

of the splitting algorithm can be verified. Second, consider the local conditional mean square accuracy of the splitting algorithm:

$$\begin{aligned}
&\Big(\mathbb{E}\left[|X(t+\Delta t) - Y(t+\Delta t)|^2\middle|\mathcal{F}_t\right]\Big)^{1/2} \\
&= \left(\mathbb{E}\left[\left|\Phi(t+\Delta t,x)\int_t^{t+\Delta t}[\Phi(s,x)]^{-1}\alpha(X(s),s)\,ds\right.\right.\right. \\
&\qquad \left.\left.\left. + \Phi(t+\Delta t,x)\int_t^{t+\Delta t}[\Phi(s,x)]^{-1}\sigma_2(X(s),s)\,dW_2(s) - (Y(t+\Delta t)-x)\right|^2\middle|\mathcal{F}_t\right]\right)^{1/2} \\
&\le \left(\mathbb{E}\left[\left|\int_t^{t+\Delta t}\Big(\Phi(t+\Delta t,x)-\Phi(s,x)\Big)\,[\Phi(s,x)]^{-1}\alpha(X(s),s)\,ds\right|^2\middle|\mathcal{F}_t\right]\right)^{1/2} \\
&\quad + \left(\mathbb{E}\left[\left|\int_t^{t+\Delta t}\Big(\Phi(t+\Delta t,x)-\Phi(s,x)\Big)\,[\Phi(s,x)]^{-1}\sigma_2(X(s),s)\,dW_2(s)\right|^2\middle|\mathcal{F}_t\right]\right)^{1/2} \\
&\quad + \left(\mathbb{E}\left[\left|\int_t^{t+\Delta t}\alpha(X(s),s)\,ds + \int_t^{t+\Delta t}\sigma_2(X(s),s)\,dW_2(s) - (Y(t+\Delta t)-x)\right|^2\middle|\mathcal{F}_t\right]\right)^{1/2} \\
&\le K_2[\Delta t]^{\min(r_2,(2r_H+1)/2+0.5)},
\end{aligned}$$

where K_2 is a real constant; hence the local conditional mean square accuracy rate

$$\min(r_2, (2r_H + 1)/2 + 0.5) = \min(r_2, r_H + 1.0)$$

of the splitting algorithm can be derived. Now, apply the general convergence theory from [31, 41, 42, 43] to conclude the worst case estimate of global mean square convergence rate r_g as stated in Theorem C.1 along nonrandom partitions of $[0, T]$. □

. Consider the Itô-interpreted example

$$dX(t) = (1 + X(t))\,dt + \sqrt{2}X(t)\,dW_1(t) + 2\sqrt{X(t)}\,dW_2(t). \tag{C.7}$$

Clearly, SDE (C.7) is of the form (C.1) with $\alpha(X,t) = 1$, $\beta(t) = 1$, $\sigma_2(X,t) = 2\sqrt{X(t)}$, and $\sigma_1(t) = \sqrt{2}$. Moreover, this example satisfies the assumptions of Theorem C.1 with $r_H = 0.5$. To see this fact, use the splitting into the subsystems

$$dX_1(t) = X_1(t)\,dt + \sqrt{2}X_1(t)\,dW_1(t), \tag{C.8}$$

$$dX_2(t) = dt + 2\sqrt{X_2(t)}\,dW_2(t). \tag{C.9}$$

The stochastic flows Φ_i related to linear (C.8) and nonlinear subsystems (C.9) are

$$\Phi_1(s,x) = \exp(\sqrt{2}W_1(s))x, \quad \Phi_2(s,x) = (\sqrt{x} + W_2(s))^2$$

as direct applications of Itô's formula. Besides, both flows have uniformly bounded moments of all orders for nonrandom initial data and they are mean square Hölder-continuous with exponent $r_H = 0.5$. For example, for the second subsystem (C.9), we have $\Phi_2(t, X(0)) - \Phi_2(s, X(0)) = 2\sqrt{X(0)}(W_2(t) - W_2(s))$ and

$$\mathbb{E}\,|\Phi_2(t, X(0)) - \Phi_2(s, X(0))|^2 = 2\,\mathbb{E}\,[X(0)]\,\mathbb{E}\,|W_2(t) - W_2(s)|^2 = 2\,\mathbb{E}\,[X(0)]\,|t - s|$$

provided that $X(0) \geq 0$ (a.s.) is independent of the processes W_2. Therefore, the related splitting-step algorithm can be applied and achieve a global L^2-convergence order $r_g = 1.0$ since the splitting-step integration has trivially at least local accuracy rates $r_1 = 2.0$ and $r_2 = 1.0$ thanks to the exact knowledge on the analytic expression of both subflows and the formula for r_g given by Theorem C.1. Moreover, the positivity-preserving character of such a splitting technique is obvious from the explicit formulas of subflows Φ_i. Similarly, we can proceed for other equations with $\sqrt{(.)}$- or other Hölder-continuous terms with Hölder exponent ≥ 0.5.

Acknowledgments. We are grateful to C. Doering and P. Smereka for comments and discussions. The authors would also like to thank the anonymous referees for numerous helpful comments.

REFERENCES

[1] M. Bossy, E. Gobet, and D. Talay, *A symmetrized Euler scheme for an efficient approximation of reflected diffusions*, J. Appl. Probab., 41 (2004), pp. 877–889.

[2] M. Broadie and Ö. Kaya, *Exact simulation of stochastic volatility and other affine jump diffusion processes*, Oper. Res., 54 (2006), pp. 217–231.

[3] F. M. Buchmann and W. P. Petersen, *Solving Dirichlet problems numerically using the Feynman–Kac representation*, BIT, 43 (2003), pp. 519–540.

[4] J. C. Cox, *Notes on option pricing* I: *Constant elasticity of variance diffusions*, J. Portfolio Management, 22 (1996), pp. 15–17.

[5] J. C. Cox, J. E. Ingersoll, and S. A. Ross, *A theory of term structure of interest rates*, Econometrica, 53 (1985), pp. 385–407.

[6] F. Castell and J. Gaines, *The ordinary differential equation approach to asymptotically efficient schemes for solution of stochastic differential equations*, Ann. Inst. H. Poincaré Probab. Statist., 32 (1996), pp. 231–250.

[7] D. A. Dawson and E. A. Perkins, *Measured-valued processes and renormalization of branching particle systems*, in Stochastic Partial Differential Equations: Six Perspectives, R. A. Carmona and B. Rozovskii, eds., Math. Surveys Monogr. 64, AMS, Providence, RI, 1998, pp. 45–102.

[8] C. R. Doering, K. V. Sargsyan, and P. Smereka, *A numerical method for some stochastic differential equations with multiplicative noise*, Phys. Lett. A, 344 (2005), pp. 149–155.

[9] H. Doss, *Liens entre équations différentielles stochastiques et ordinaires*, Ann. Inst. H. Poincaré Sect. B (N.S.), 13 (1977), pp. 99–125.

[10] I. Dornic, H. Chaté, and M. A. Muñoz, *Integration of Langevin equations with multiplicative noise and the viability of field theories for absorbing phase transitions*, Phys. Rev. Lett., 94 (2005), 100601.

[11] R. Durrett and E. A. Perkins, *Rescaled contact processes converge to super-Brownian motion in two or more dimensions*, Probab. Theory Related Fields, 114 (1999), pp. 309–399.

[12] E. B. Dynkin, *Superprocesses and partial differential equations*, Ann. Probab., 21 (1993), pp. 1185–1262.

[13] A. Dzougoutov, K.-S. Moon, E. von Schwerin, A. Szepessy, and R. Tempone, *Adaptive Monte Carlo algorithms for stopped diffusion*, in Multiscale Methods in Science and Engineering, Lect. Notes Comput. Sci. Eng. 44, Springer-Verlag, Berlin, 2005, pp. 59–88.

[14] W. Feller, *The general diffusion operator and positivity preserving semi-groups in one dimension*, Ann. of Math. (2), 60 (1954), pp. 417–436.

[15] W. Feller, *Diffusion processes in one dimension*, Trans. Amer. Math. Soc., 77 (1954), pp. 1–31.

[16] W. Feller, *The parabolic differential equations and the associated semi-groups of transformations*, Ann. of Math. (2), 55 (1952), pp. 468–519.

[17] W. Feller, *Probability Theory and Its Applications*, 2nd ed., Wiley, New York, 1971.

[18] J. G. Gaines, *Numerical experiments with S(P)DE's*, in Stochastic Partial Differential Equations, A. M. Etheridge, ed., London Math. Soc. Lecture Note Ser. 216, Cambridge University Press, Cambridge, UK, 1995, pp. 55–71.

[19] E. Gobet, *Weak approximation of killed diffusion using Euler schemes*, Stochastic Process. Appl., 87 (2000), pp. 167–197.

[20] E. Gobet and S. Menozzi, *Exact approximation rate of killed hypoelliptic diffusions using the discrete Euler scheme*, Stochastic Process. Appl., 112 (2004), pp. 201–223.

[21] D. J. Higham and X. Mao, *Convergence of Monte Carlo simulations involving the mean-reverting square root process*, J. Comput. Finance, 8 (2005), pp. 35–62.

[22] D. J. Higham, X. Mao, and A. M. Stuart, *Strong convergence of Euler-type methods for nonlinear stochastic differential equations*, SIAM J. Numer. Anal., 40 (2002), pp. 1041–1063.

[23] H. Hinrichsen, *Non-equilibrium critical phenomena and phase transitions into absorbing states*, Adv. in Phys., 49 (2000), pp. 815–958.

[24] N. L. Johnson, S. Kotz, and N. Balakrishnan, *Continuous Univariate Distributions*, Vol. II, Wiley, New York, 1994.

[25] C. Kahl and H. Schurz, *Balanced Milstein methods for ordinary SDEs*, Monte Carlo Methods Appl., 12 (2006), pp. 143–170.

[26] S. Kanagawa and Y. Saisho, *Strong approximation of reflecting Brownian motion using penalty method and its application to computer simulation*, Monte Carlo Methods Appl., 6 (2000), pp. 105–114.

[27] P. E. Kloeden and E. Platen, *Numerical Solution of Stochastic Differential Equations*, Springer-Verlag, New York, 1998.

[28] P. E. Kloeden, E. Platen, and H. Schurz, *Numerical Solution of SDEs through Computer Experiments*, Springer-Verlag, New York, 1994.

[29] D. Lépingle, *Un schéma d'Euler pour équations différentielles stochastiques réfléchies*, C. R. Acad. Sci. Paris Sér. I Math., 316 (1993), pp. 601–605.

[30] X. Mao, *Stochastic Differential Equations & Applications*, Horwood Publishing, Chichester, UK, 1997.

[31] G. N. Milstein, *Numerical Integration of Stochastic Differential Equations*, Kluwer, Dordrecht, The Netherlands, 1995.

[32] G. N. Milstein, E. Platen, and H. Schurz, *Balanced implicit methods for stiff stochastic systems*, SIAM J. Numer. Anal., 35 (1998), pp. 1010–1019.

[33] T. Misawa, *A Lie algebraic approach to numerical integration of stochastic differential equations*, SIAM J. Sci. Comput., 23 (2001), pp. 866–890.

[34] E. Moro, *Numerical schemes for continuum models of reaction–diffusion systems subject to internal noise*, Phys. Rev. E (3), 70 (2004), 045102.

[35] C. Mueller and R. Tribe, *A phase transition for a stochastic PDE related to the contact process*, Probab. Theory Related Fields, 100 (1994), pp. 131–156.

[36] C. Mueller and R. Tribe, *Stochastic PDE's arising from the long range contact and long range voter process*, Probab. Theory Related Fields, 102 (1995), pp. 519–545.

[37] L. Pechenik and H. Levine, *Interfacial velocity correction due to multiplicative noise*, Phys. Rev. E (3), 59 (1999), pp. 3893–3900.

[38] R. Pettersson, *Penalization schemes for reflecting stochastic differential equations*, Bernoulli, 3 (1997), pp. 403–414.

[39] H. Schurz, *Numerical regularization for SDEs: Construction of nonnegative solutions*, Dynam. Systems Appl., 5 (1996), pp. 323–352.

[40] H. Schurz, *Stability, Stationarity, and Boundedness of Some Implicit Numerical Methods for Stochastic Differential Equations and Applications*, Logos-Verlag, Berlin, 1997.

[41] H. Schurz, *Numerical analysis of SDEs without tears* (invited chapter), in Handbook of Stochastic Analysis and Applications, V. Lakshmikantham and D. Kannan, eds., Marcel Dekker, Basel, 2002, pp. 237–359.

[42] H. Schurz, *General theorems for numerical approximation of stochastic processes on the Hilbert space $H_2([0,T])$*, Electron. Trans. Numer. Anal., 16 (2003), pp. 50–69.

[43] H. Schurz, *An axiomatic approach to numerical approximations of stochastic processes*, Int. J. Numer. Anal. Model., 3 (2006), pp. 459–480.

[44] A. F. Siegel, *The noncentral chi-squared distribution with zero degrees of freedom and testing for uniformity*, Biometrika, 66 (1979), pp. 381–386.

[45] G. Slade, *Scaling limits and super-Brownian motion*, Notices Amer. Math. Soc., 49 (2002), pp. 1056–1067.

[46] D. Talay, *Simulation of stochastic differential systems*, in Probabilistic Methods in Applied Physics, P. Krée and W. Wedig, eds., Lecture Notes in Phys. 451, Springer-Verlag, Berlin, 1995, pp. 54–96.

[47] W. Wagner and E. Platen, *Approximation of Itô Integral Equations*, Technical report at ZIMM, Academy of Sciences GDR, Berlin, 1978 (obtainable from library at WIAS, Berlin, Germany, http://www.wias-berlin.de/).

SIAM J. SCI. COMPUT.
Vol. 29, No. 4, pp. 1550–1566
© 2007 Society for Industrial and Applied Mathematics

A VARIATIONAL APPROACH TO VIDEO SEGMENTATION FOR BOTANICAL DATA*

AARON LUTTMAN† AND JOHN BARDSLEY‡

Abstract. In order to engage in photosynthesis, plant leaves absorb CO_2 via the opening of pores in their surfaces called *stomata*. Water evaporates through open stomata, however, which is a detriment to plant function. Thus a plant will seek a stomatal aperture that balances its need for CO_2 with its aversion to H_2O loss. In order to visualize a particular leaf's stomatal aperture, an experimentalist injects the leaf with dye so that it fluoresces. Regions with a higher relative intensity that do not correspond to veins in the leaf then correspond to areas in which the stomata are closed and the darker nonvein regions to areas in which the stomata are open. A camera is used to collect the emitted light, and a fluorescence pattern is measured. Images are continually recorded as these patterns change with time, resulting in a video sequence. The goal of this paper is to segment one such video sequence into fluorescing and nonfluorescing regions. After preprocessing the video, we take a variational approach to the segmentation problem and solve the associated three-dimensional evolution equation using a semi-implicit numerical scheme. A fixed-point formulation of the method is presented as are results of the segmentation for actual leaf data.

Key words. image segmentation, shape reconstruction, variational methods

AMS subject classifications. 92B99, 65K10, 65M32

DOI. 10.1137/060653792

1. Introduction. A plant's leaves have pores on their surfaces called [illegible] that occur at densities of tens to hundreds per mm^2 [28]. Carbon dioxide (CO_2), which is used in photosynthetic energy storage, is absorbed when the stomata are open, and, in general, the more open the pores are, the more CO_2 is absorbed. When the stomata are opened, however, water is lost through evaporation, which is detrimental to plant function. Thus, in response to external conditions, the stomatal apertures in a leaf must be regulated in such a way that the plant's need to absorb CO_2 is balanced with its aversion to losing H_2O. Peak, et al. [28] make the following statement regarding this process: "A central paradigm of plant biology is that, in the face of spatially heterogeneous and temporally varying environmental conditions, a plant continually adjusts its stomatal aperture so that, over time, it maximizes CO_2 uptake for a fixed amount of water loss." Moreover, it is thought that this optimization occurs over the entire leaf rather than for each stoma individually. It is not yet understood how plants solve this global optimization problem, but it has been hypothesized (cf. [28]) that the stomatal aperture depends on external environmental conditions as well as on interactions between neighboring stomata. The latter claim is somewhat controversial among plant biologists, and its primary motivation is the observation—made in over 200 plant species—that, even under spatially varying environmental conditions, stomatal apertures are often synchronized into spatially extended patches. This behavior is observed even though patchiness often results in less than optimal

*Received by the editors March 6, 2006; accepted for publication (in revised form) January 12, 2007; published electronically July 4, 2007. This work was partially supported by the NSF under grant DMS-0504325 and by Montana NSF EPSCoR.

http://www.siam.org/journals/sisc/29-4/65379.html

†Department of Mathematical Sciences, University of Montana, Missoula, MT, and Division of Science and Mathematics, Bethany Lutheran College, Mankato, MN (luttman@blc.edu).

‡Department of Mathematical Sciences, University of Montana, Missoula, MT (bardsleyj@mso.umt.edu).

local CO_2 uptake for fixed water loss. At this point it is not known how, or even if, neighboring stomata communicate, but it is believed that a study of stomatal patch dynamics in particular leaves may provide insight into this phenomena.

In order to study stomatal patch dynamics, this phenomenon must be observed in a particular leaf. The data analyzed in this paper was collected by the [illegible] research lab at Utah State University [28]. A cocklebur ([illegible]) leaf is placed in a light-penetrating chamber so that the temperature is controlled and gasses can be independently regulated and measured both on the top and bottom surfaces of the leaf. A dye is injected into the leaf that fluoresces when the stomata close, and stomatal dynamics are initiated by decreasing the concentration of H_2O on the upper surface of the leaf. This causes the stomata to close, since a decrease in environmental humidity causes an increase in H_2O loss through evaporation. This results in increased fluorescence, which is measured by a charge-coupled device (CCD) camera. Each image captured by the camera consists of a 512×512 array of pixels. Regions of high relative intensity correspond to areas in which the stomata are closed, whereas regions of low relative intensity correspond to areas in which the stomata are open. The images in the sequence were collected every 40 seconds over a period of several hours. (For more details on the data collection procedure and chlorophyll fluorescence, see [14, 25, 28].) Four sample images from the video sequence analyzed in this paper are shown in Figure 1.1.

In frames 150, 275, and 400, the light (closed stomata) and dark (open stomata) patches are evident. Our task in this paper is to provide a binary segmentation of the video into these light and dark regions. From such a segmentation, a study of patch dynamics can then be performed.

The first step in processing the data is denoising each of the images in the video sequence. For this we use the PDE-based denoising method of Rudin–Osher–Fatemi [30], with L^1-fidelity as proposed by Chan and Esedoglu in [8]. We choose this method because it is effective at denoising an image without degrading its inherent structure. It is necessary to use such a method so that resolution is not lost in the images, particularly at the boundaries of the stomatal patches and veins, which correspond to the darkest regions in the images of Figure 1.1. Additional noise enters the problem in the form of global lighting changes that occur during the data collection experiment, so the data must also be scaled and normalized against such effects.

Having preprocessed the video data, our final task is to extract the stomatal patches from the video sequence by segmenting the video into the fluorescing and nonfluorescing regions. To do this, we take a variational approach inspired by the methods found in [10, 11, 18, 19]. In contrast to these methods, our segmentation is computed in three dimensions (over the entire video sequence) and segments based on contrast to an [illegible] background model generated from the experiment. We perform the segmentation in three dimensions, because the algorithm is then able to globally compare the changes between the video sequence and the background model. When each frame is considered individually, no such comparison is possible, and, as a result, differences between the images and the background model tend to be overemphasized in frames with little or no dynamics. Such frames occur in the early stages of the video analyzed here. Other standard segmentation approaches include threshold-based methods [31], edge-based methods [3, 4, 17], region-based methods [16, 27], and hybrid techniques [35, 39] that combine several approaches. The reason that such approaches fail for our problem is that vein and nonvein regions fluoresce differently. In particular, the standard methods tend to either ignore the veins, by incorporating them in a background model which is subtracted off, or overemphasize the veins

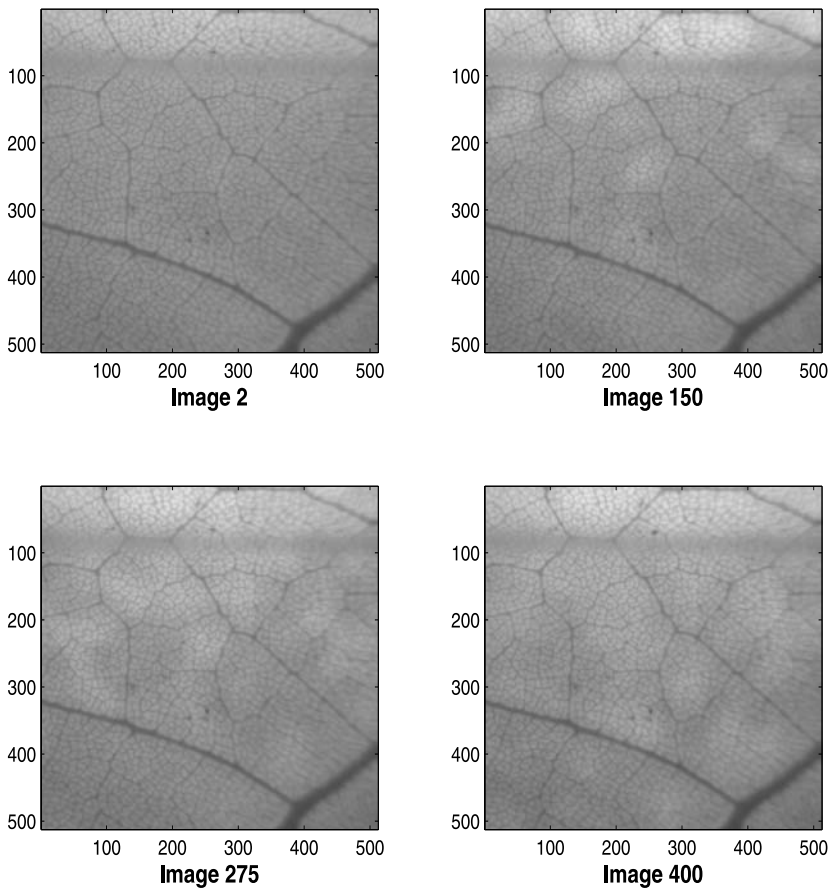


FIG. 1.1. *Four* 512×512 *images from the video sequence of* 436 *frames.*

by applying the same segmentation standard to the vein pixels as the nonvein pixels. The algorithm used here incorporates both the measured data and a static background model into the segmentation, so that the veins are neither ignored nor overemphasized.

The video preprocessing is discussed in section 2. In section 3, we present the variational segmentation method. The numerical implementation of the method is detailed, we present our results in section 4, and conclusions are given in section 5.

2. Data preprocessing. Before segmentation, it is necessary to remove noise from each frame of the video, and two such types of noise will be accounted for. The first is the noise incurred during image formation due to the CCD camera instrumentation. This noise is high frequency and will be removed using the method discussed in section 2.1. In section 2.2 we discuss removal of lower frequency noise due to global lighting changes that occurred during the data collection experiment.

2.1. Denoising. The images in the video sequence of interest are formed via the counting of photons by a mechanism in each pixel of the camera's CCD array.

The imprecision of this counting process together with stochastic instrumental errors results in collected images that are noisy. (For more details on CCD camera noise statistics, see [33].) Instrumentation noise is high frequency and can be removed using a wide array of denoising methods. Due to the fact that the images in the video have edges at the vein boundaries, the method chosen must preserve edges. Several methods of this type have recently been developed [6,12,29], and one of the most popular choices is that of Rudin, Osher, and Fatemi (ROF) [30], which seeks to minimize the sum of the total variation of the digital signal and an L^2 data-fidelity term. The popularity of ROF is due to the fact that the approach is effective, and computationally efficient algorithms exist for its solution [7, 9, 36]. It has been noticed, however, that if an L^1-fidelity is used instead, as is done in [8], less degradation occurs in the structure of the denoised images (see [2] and the references therein). Moreover, the L^1 denoising technique is completely contrast invariant. Since our images are highly structured at multiple contrast scales, we take this approach, minimizing the total variation regularized L^1-fidelity functional using the gradient descent method of [8]. Although we are able to obtain satisfactory results in a computationally feasible manner using this algorithm, it was pointed out to us by the referees that the numerical approach of [8] is typically very slow to converge and that a faster method is presented [2].

Let $I_0(\vec{x})$, where $\vec{x} = (x, y)$, be the input image corresponding to a single frame of the video sequence. Then the denoising corresponds to computing a minimum of the energy functional

$$E(I) = \int_\Omega |\nabla I| \, d\vec{x} + \lambda \int_\Omega |I_0 - I| \, d\vec{x}, \tag{2.1}$$

where $\lambda > 0$ is the regularization parameter and $\Omega \subset \mathbb{R}^2$ is the image domain. The first term on the right-hand side of (2.1) is the [illegible] of I which acts as a local smoothing operator, and the second term is the fidelity term which ensures that the resulting I approximates the input image I_0 in L^1. The minimization cannot be performed directly due to numerical instability, so a differentiable approximation of the functional, given by

$$E(I) = \int_\Omega \sqrt{(\nabla I)^2 + \delta} \, d\vec{x} + \lambda \int_\Omega \sqrt{(I_0 - I)^2 + \epsilon} \, d\vec{x}, \tag{2.2}$$

with parameters δ and ϵ small and positive, is minimized instead (see [11]). The minimization procedure is performed using the gradient descent method for (2.2), which involves evolving the image $I(\vec{x}; t)$ according to

$$I_t = \operatorname{div}\left(\frac{\nabla I}{|\nabla I| + \delta}\right) + \lambda \frac{I_0 - I}{\left((I_0 - I)^2 + \epsilon\right)^{1/2}}, \qquad I_0(\vec{x}) = I(\vec{x}, 0), \tag{2.3}$$

where t is an artificial time parameter. This is done for each frame in the video sequence.

Total-variation regularization is used for image denoising when it is assumed that the image has jump discontinuities. The total-variation penalization strengthens jump discontinuities, which, for our data, means that the veins are enhanced. Enhancing the veins is good, because that will improve our background model for vein pixels and also make the edges along the vein boundaries sharper. Jump discontinuities in our data along the time dimension are experimental artifacts caused by global lighting

changes between frames (recall that the frames are generated every 40 seconds). These artifacts must be removed, so applying the denoising algorithm to the data in 3D would not be appropriate, as this would enhance these experimental artifacts. This is the motivation behind denoising each image in the video sequence individually. Our method for removing these global lighting artifacts is discussed in the next section.

2.2. Median-change normalization. After denoising, the video data must be normalized against global lighting effects. The leaf experiments run over several hours, and there are large-scale, global lighting changes that must be accounted for. We make the following assumption, which is reasonable for our data: the number of pixels that have increasing intensity due to fluorescence dynamics at any given time is less than half of the total number of pixels, and likewise less than half of all pixels demonstrate a decrease in intensity due to dynamics. Given that this assumption is correct, the median intensity changes from one frame to the next are primarily due to global effects. Thus we normalize each frame in the video sequence by subtracting off the median intensity change. If $I(\vec{x}, n)$ represents the nth image in the video sequence, then we replace $I(\vec{x}, n)$ by $I^*(\vec{x}, n)$, where

$$I^*(\vec{x}, n) = I(\vec{x}, n) - \underset{\vec{x}}{\text{median}} \left(I(\vec{x}, n) - I^*(\vec{x}, n-1) \right),$$

with $I^*(\vec{x}, 1) = I(\vec{x}, 1)$. Note that this is not equivalent to simply normalizing each image to have the same median value. Lastly, the video sequence is linearly scaled so that the intensity values in the sequence take up the full range $[0, 255]$.

The preprocessed images corresponding to the original images in Figure 1.1 are given in Figure 2.1. In these images, the fluorescing stomatal patches are much clearer. Furthermore, it is evident that the patches on the leaf change in time, as a large portion of the image in frame 400 is quite bright whereas essentially all of frame 2 is much darker. The goal is to extract the bright regions from each frame in order to determine the evolution of the fluorescence patterns over time. Our method for this segmentation step is presented in the next section.

3. Variational segmentation of the leaf data. Let $\vec{x} = (x, y, z)$ be the coordinates of the video sequence, with x and y denoting pixel coordinates within each frame and z corresponding to frame number. We define $I(\vec{x})$ to be the preprocessed video sequence, and let $D(\vec{x}) = |I(\vec{x}) - B(x, y)|$, where B is a static background model generated using a sequence of frames from the beginning of the video. The value of $B(x, y)$ at pixel (x, y) is taken to be the mean intensity of $I(x, y, z)$ for values of z corresponding to these early frames (the first 40 frames are used for the data analyzed here). The early frames in this video sequence contain no noticeable dynamics, so the background model contains no bright regions. Thus any bright regions that occur in later frames are in contrast to the background model, except at the veins, which do not fluoresce and remain dark even when they are within a bright region.

We define $\Omega \subset \mathbb{R}^3$ to be the (open) domain of D and characterize all possible segmentations of the video sequence by piecewise-constant functions u on Ω that take on two distinct values, c_1 and c_2, such that the sets $\Omega_1 = \{\vec{x} \in \Omega \mid u(\vec{x}) = c_1\}$ and $\Omega_2 = \{\vec{x} \in \Omega \mid u(\vec{x}) = c_2\}$ have piecewise-smooth boundaries of finite surface area. From any such u, a natural segmentation of the video results; namely, Ω_1 corresponds to the bright regions and Ω_2 to the dark regions of the image, or vice versa.

3.1. The segmentation functional. We assume that the optimal segmentation of the video is given by the function u, together with values c_1 and c_2, that minimize

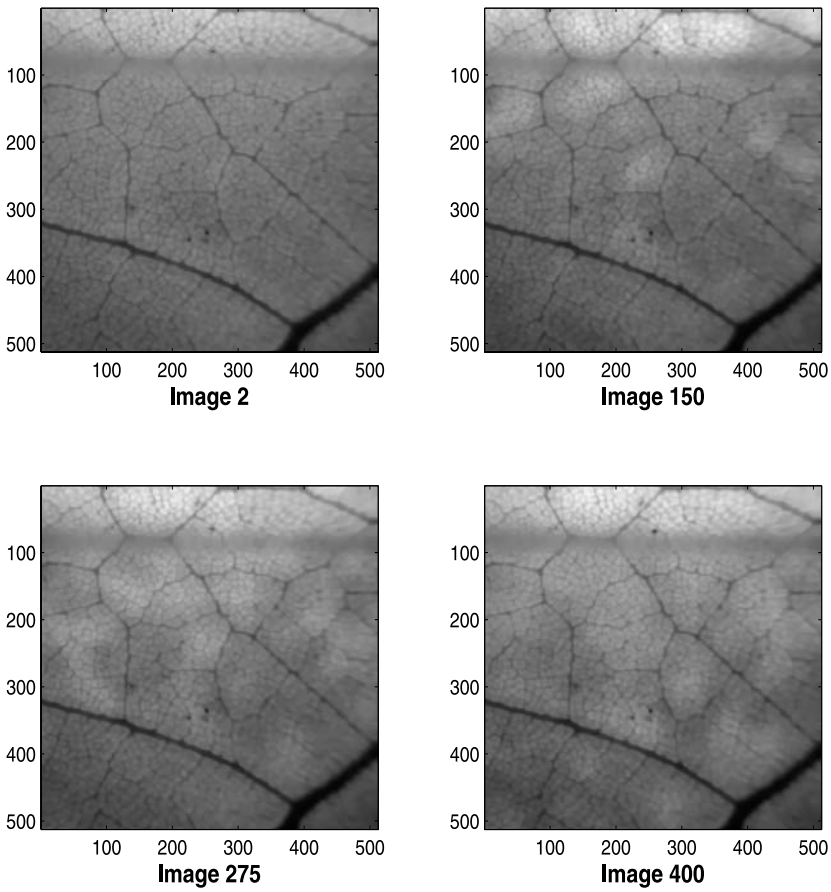


FIG. 2.1. *Images* 2, 150, 275, *and* 400 *after preprocessing.*

the squared L^2 norm of the difference between D and u, i.e.,

$$E(u, c_1, c_2) = \|D(\vec{x}) - u(\vec{x})\|^2_{L^2(\Omega)} = \int_{\Omega_1} (D(\vec{x}) - c_1)^2 \, d\vec{x} + \int_{\Omega_2} (D(\vec{x}) - c_2)^2 \, d\vec{x}. \tag{3.1}$$

Following Kimmel in [18, 19], we reduce E by computing the optimal values of c_1 and c_2 for fixed Ω_1 and Ω_2. Assuming that u and c_2 are fixed, the minimizer of $E(u, c_1, c_2)$ occurs where its derivative with respect to c_1 is zero. Since

$$\begin{aligned} E(u, c_1, c_2) &= \int_{\Omega_1} (D(\vec{x}) - c_1)^2 \, d\vec{x} + \int_{\Omega_2} (D(\vec{x}) - c_2)^2 \, d\vec{x}, \\ &= \int_{\Omega_1} D(\vec{x}) \, d\vec{x} - 2c_1 \int_{\Omega_1} D(\vec{x}) \, d\vec{x} + c_1^2 \int_{\Omega_1} d\vec{x} + \int_{\Omega_2} (D(\vec{x}) - c_2)^2 \, d\vec{x}, \end{aligned}$$

and u and c_2 are assumed fixed, it is readily seen that the derivative of E with respect to c_1 is zero when

$$c_1 = \int_{\Omega_1} D(\vec{x})\, d\vec{x} \Big/ \int_{\Omega_1} d\vec{x}\,, \tag{3.2}$$

which is the mean value of D on Ω_1. Note that E is bounded below with respect to c_1 but not from above. Thus, since (3.2) corresponds to the only local extremum of E, it must be a global minimum with respect to c_1 for fixed c_2 and u. A similar argument shows that, for fixed c_1 and u, the minimizing value of c_2 is the mean value of D on Ω_2,

$$c_2 = \int_{\Omega_2} D(\vec{x})\, d\vec{x} \Big/ \int_{\Omega_2} d\vec{x}. \tag{3.3}$$

In what follows, we will use c_1 to denote (3.2) and c_2 to denote (3.3). The resulting functional E then depends explicitly only on u.

Computing a minimizer of E analytically on the set of piecewise-constant functions u taking on exactly two values is not possible; hence, a numerical approach must be taken. This can be done efficiently if we recast the minimization problem as an evolution equation.

3.2. Formulating the segmentation problem as an evolution equation. For each function u of the type discussed above, there exists a Lipschitz continuous level set function $\Phi\colon \Omega \to \mathbb{R}$ satisfying

$$\Omega_1 = \{\vec{x} \in \Omega \mid u(\vec{x}) = c_1\} = \{\vec{x} \in \Omega \mid \Phi(\vec{x}) < 0\}, \tag{3.4}$$

$$\Omega_2 = \{\vec{x} \in \Omega \mid u(\vec{x}) = c_2\} = \{\vec{x} \in \Omega \mid \Phi(\vec{x}) > 0\}, \tag{3.5}$$

where set equality is always assumed to be up to a set of Lebesgue measure zero. Such a function Φ is called a level set representation for u, and, though level set representations of u are not unique, each particular Φ yields a unique segmentation of the video sequence via (3.4) and (3.5). Then (3.1) can be rewritten as

$$E(\Phi) = \int_{\Omega} |D - c_1|^2 \,(1 - H(\Phi)) + |D - c_2|^2 H(\Phi)\, d\vec{x}, \tag{3.6}$$

where H is the Heaviside function, i.e., $H(\Phi)$ is 1 for $\Phi > 0$ and is 0 for $\Phi \leq 0$. This is the model proposed in [11], and it is based on the variational segmentation algorithm first developed by Mumford and Shah [26].

In order to obtain efficient numerical methods for minimizing (3.6), we seek to recast this problem as an evolution equation of the form

$$\frac{\partial \Phi}{\partial t} = F(\Phi), \tag{3.7}$$

with initial condition $\Phi(\vec{x}; 0) = \Phi_0(\vec{x})$. The function F dictates the deformation of Φ in time and is chosen so that the steady-state solution of (3.7) corresponds to a solution of the Euler–Lagrange equation

$$\frac{\delta E}{\delta \Phi} = 0, \tag{3.8}$$

where $\delta/\delta\Phi$ denotes the variational derivative. Recalling that the distributional derivative of $H(\Phi)$ is the Dirac delta distribution $\delta(\Phi)$, a straightforward calculation yields

$$\frac{\delta E(\Phi)}{\delta \Phi} = -2\left((c_2 - c_1)\left(D - \frac{1}{2}(c_1 + c_2)\right)\right)\delta(\Phi). \tag{3.9}$$

The standard approach taken in the literature is to approximately solve (3.8) by numerically solving the gradient descent evolution equation

$$\frac{\partial \Phi}{\partial t} = (c_2 - c_1)\left(D - \frac{1}{2}(c_1 + c_2)\right)\delta(\Phi). \tag{3.10}$$

However, we note that (3.8)–(3.9) is satisfied if

$$(c_2 - c_1)\left(D - \frac{1}{2}(c_1 + c_2)\right) = 0, \tag{3.11}$$

which leads, instead, to the simplified evolution equation

$$\frac{\partial \Phi}{\partial t} = (c_2 - c_1)\left(D - \frac{1}{2}(c_1 + c_2)\right). \tag{3.12}$$

To see that (3.12) is a minimization scheme for (3.6), we note that $E(\Phi(\vec{x};t))$ is a nonincreasing function in t provided the variational derivative of E with respect to t, given by

$$\frac{\delta E(\Phi(\vec{x};t))}{\delta t} = \frac{\delta E(\Phi(\vec{x};t))}{\delta \Phi}\frac{\partial \Phi(\vec{x};t)}{\partial t},$$

is nonpositive. A straightforward calculation shows that this is satisfied by both (3.10) and (3.12).

We note that in [11] and related works, approximate solutions of (3.10) are sought, requiring an approximation of the $\delta(\Phi)$ term. In [40], for example, a computationally feasible problem is obtained by replacing $\delta(\Phi)$ with $|\nabla\Phi|$; whereas in [1, 11, 20], $\delta(\Phi)$ is approximated by a piecewise smooth function. Our own numerical experiments suggest, however, that although these approaches can be effective, adequate segmentations and dramatically less computationally intensive algorithms result if (3.12) is used instead. We also note the fact that with the "approximation" $\delta(\Phi) = 1$, (3.10) becomes (3.12).

3.3. Regularizing the segmentation algorithm. There is no guarantee that a solution Φ of (3.12) will yield a segmentation (3.4)–(3.5) in which the boundary of Ω_1 is piecewise-smooth and of finite surface area. To ensure that our solutions have this property, a regularization functional is added to the segmentation functional (3.6). Following Kimmel [18], we begin with the geodesic active contour [5] functional given by

$$E_{\text{reg}}(\Phi) = \oint_{\partial\Omega_1} g \, dS,$$

where $\partial\Omega_1$ denotes the boundary of Ω_1 and g is an edge-indicator function for each image in the video sequence. This corresponds to the regularization used in [11] if $g \equiv 1$. We use, instead, the standard edge indicator function,

$$g(x,y) = \frac{1}{1 + |\nabla I(x,y)|^2}, \tag{3.13}$$

evaluated in each image individually. Note that edges are two-dimensional features, a fact that requires g be evaluated in two dimensions rather than in three dimensions. Moreover, this particular regularization allows us to incorporate edge information directly into the segmentation functional. Using $g = 1$, as is done in [11], does not allow edges of different strengths to be weighted differently. All edges are either enhanced or flattened according to the value of the regularization parameter. Using the edge-indicator function in (3.13), however, allows the segmentation functional to treat edges of different strengths differently, according to the weighting factor determined by g. Thus some edges will be enhanced greatly, some a little bit, some not at all, and some will be flattened. Since determining the impact of the leaf veins on stomatal dynamics is of particular interest, and edges in the data correspond to vein boundaries, this yields a measurement, based on vein size, of the effect of the veins on the segmentation. Moreover, note that the fidelity functional acts on the background-subtracted data, but that g acts on the preprocessed data directly. Thus the regularization allows us to include this vein information directly into the segmentation algorithm, where other segmentation methods that act on background-subtracted data do not.

In order to incorporate the regularization functional into the level set PDE (3.12), it is necessary to compute its variational derivative with respect to Φ, which is (see [5])

$$\frac{\delta E_{\text{reg}}(\Phi)}{\delta \Phi} = -\text{div}\left(g\frac{\nabla\Phi}{|\nabla\Phi|}\right)|\nabla\Phi|. \tag{3.14}$$

In order to optimize computational efficiency (recall that we must numerically solve the evolution equation in three dimensions), we seek an approximation of (3.14) that does not require an evaluation of $|\nabla\Phi|$. Such an approximation can be obtained by appealing to the equality

$$\text{div}\left(g\frac{\nabla\Phi}{|\nabla\Phi|}\right)|\nabla\Phi| = \text{div}(g\nabla\Phi) - \frac{g}{|\nabla\Phi|}\nabla(|\nabla\Phi|).$$

In particular, rather than using the full geodesic active contour model for regularization, we use only the the diffusion term $\text{div}(g\nabla\Phi)$. This then yields the regularized analogue to (3.11), given by the nonlinear equation

$$\alpha\text{div}(g\nabla\Phi) + \beta(c_2 - c_1)\left(D - \frac{1}{2}(c_1 + c_2)\right) = 0 \tag{3.15}$$

and corresponding gradient descent evolution equation

$$\frac{\partial\Phi}{\partial t} = \alpha\text{div}(g\nabla\Phi) + \beta(c_2 - c_1)\left(D - \frac{1}{2}(c_1 + c_2)\right), \tag{3.16}$$

where α and β are weighting factors. Note that (3.16) has the form of (3.7), as desired.

4. Numerical methods and results. The most direct method of solving (3.16) is to use an explicit time discretization. This is not a practical approach for our problem, however, since the corresponding stability criterion is computationally prohibitive. Thus, rather than discretizing (3.16) explicitly, we follow Weickert, et al. [37, 38] and use the [illegible] time discretization

$$\Phi^{n+1} - \alpha\Delta t\text{div}\left(g\nabla\Phi^{n+1}\right) = \Phi^n + \Delta t\beta(c_2^n - c_1^n)\left(D - \frac{1}{2}(c_1^n + c_2^n)\right), \tag{4.1}$$

where c_1^n and c_2^n represent the values of c_1 and c_2 at the nth time step. Recall that c_1 and c_2 depend on Φ. This numerical scheme is called semi-implicit, or linear-implicit, because the regularization term is evaluated at the $n+1$ time step, whereas the fidelity term is evaluated at the nth time step. Scheme (4.1) can be rewritten as

$$(\mathcal{I} - \alpha\Delta t\,(A_x + A_y + A_z))\,\Phi^{n+1} = \Phi^n + \Delta t\beta(c_2^n - c_1^n)\left(D - \frac{1}{2}(c_1^n + c_2^n)\right), \tag{4.2}$$

where $A_s = \frac{\partial}{\partial s} g \frac{\partial}{\partial s}$ for $s \in \{x, y, z\}$ and $\mathcal{I}$ is the identity operator. It is shown in [38] that this discretization leads to a convergent, unconditionally stable numerical scheme.

The unconditional stability of (4.2) results in a relaxed restriction on the size of the time step, which is good for computational efficiency. The drawback is that a large linear system must be solved at every iteration. There are various methods for solving large linear systems of this type efficiently such as multigrid, multiplicative operator splitting (e.g., ADI or a locally one dimensional scheme such as in [18,19]), and Additive Operator Splitting (AOS) [37,38]. We use AOS, where the linear system (4.2) is approximately solved at each iteration using

$$\begin{aligned}&(\mathcal{I} - \alpha\Delta t(A_x + A_y + A_z))^{-1}\\ &\qquad \approx \frac{1}{3}\left(\mathcal{I} - 3\alpha\Delta t A_x\right)^{-1} + \frac{1}{3}\left(\mathcal{I} - 3\alpha\Delta t A_y\right)^{-1} + \frac{1}{3}\left(\mathcal{I} - 3\alpha\Delta t A_z\right)^{-1}.\end{aligned} \tag{4.3}$$

This operator splitting yields an $\mathcal{O}(\Delta t^2)$ approximation to the semi-implicit scheme as can be seen from the Taylor series expansions

$$\left(\mathcal{I} - \alpha\Delta t(A_x + A_y + A_z)\right)^{-1} v = v + \alpha\Delta t(A_x + A_y + A_z)v + \mathcal{O}(\Delta t^2),$$

and

$$\begin{aligned}&\frac{1}{3}(\mathcal{I} - 3\alpha\Delta t A_x)^{-1}v + \frac{1}{3}\left(\mathcal{I} - 3\alpha\Delta t A_y\right)^{-1} v + \frac{1}{3}\left(\mathcal{I} - 3\alpha\Delta t A_z\right)^{-1} v\\ &\qquad = \frac{1}{3}\left(v + 3\alpha\Delta t A_x v\right) + \frac{1}{3}\left(v + 3\alpha\Delta t A_y v\right) + \frac{1}{3}\left(v + 3\alpha\Delta t A_z v\right) + \mathcal{O}(\Delta t^2)\\ &\qquad = v + \alpha\Delta t(A_x + A_y + A_z)v + \mathcal{O}(\Delta t^2).\end{aligned}$$

Thus, since (4.2) is already an $\mathcal{O}(\Delta t^2)$ scheme, the global error does not change when AOS is used in its place.

Applying the AOS approximation to (4.2) yields

$$\Phi^{n+1} = \frac{1}{3}\sum_{s\in\{x,y,z\}} \left(\mathcal{I} - 3\alpha\Delta t A_s\right)^{-1}\left(\Phi^n + \Delta t\beta(c_2 - c_1)\left(D - \frac{1}{2}(c_1 + c_2)\right)\right). \tag{4.4}$$

Convergence and unconditional stability results for this scheme follow directly from the results found in [21, 22], which are obtained using operator semigroup theory. A straightforward proof of these facts is presented in [23] using only the Lax–Equivalence Theorem together with the fact that the eigenvalues of $(\mathcal{I} - 3\alpha\Delta t A_s)^{-1}$ are between 0 and 1 for $s \in \{x, y, z\}$.

In order to use (4.4), we discretize the A_s operators using a standard finite difference approximation (cf. [24, Chapter 6]) with spatial grid size $h = 1$ (the natural

pixel-frame discretization). The A_x approximation is given by

$$(A_x)_{i,j} = \begin{cases} (g_{k,j,l} + g_{k,i,l})/\,2, & j \in \{i+1, i-1\}, \\ -\sum\limits_{n \in \{i-1, i+1\}} (g_{k,n,l} + g_{k,i,l})\,/\,2, & j = i, \\ 0, & \text{otherwise}, \end{cases}$$

where k and l denote the row and frame number, respectively.

Similarly, the A_y approximation has the form

$$(A_y)_{i,j} = \begin{cases} (g_{j,k,l} + g_{i,k,l})/\,2, & j \in \{i+1, i-1\}, \\ -\sum\limits_{n \in \{i-1, i+1\}} (g_{n,k,l} + g_{i,k,l})\,/\,2, & j = i, \\ 0, & \text{otherwise}, \end{cases}$$

where k and l denote the column and frame number, respectively.

Finally, the A_z approximation is given by

$$(A_z)_{i,j} = \begin{cases} (g_{k,l,j} + g_{k,l,i})/\,2, & j \in \{i+1, i-1\}, \\ -\sum\limits_{n \in \{i-1, i+1\}} (g_{k,l,n} + g_{k,l,i})\,/\,2, & j = i, \\ 0, & \text{otherwise}, \end{cases}$$

where k and l denote the row and column number, respectively.

The A_x matrices are generated for every row in each frame; the A_y matrices are generated for every column in each frame; and the A_z matrices are generated for each pixel time series. The associated linear systems are small since each row and column has only 512 pixels, and the time series of each pixel has length 436. Each of these linear systems can be solved independently, and therefore in parallel, using appropriate boundary conditions. We use antireflective boundary conditions [32], which guarantee a C^1 extension of the signal at the boundary. Furthermore, since A_x, A_y, and A_z are tridiagonal and strictly diagonally dominant, the linear systems involving each can be solved efficiently using the well known Thomas algorithm—a special form of Gaussian elimination for tridiagonal systems. Thus, each step of (4.4) can be computed in a computationally efficient manner.

Finally, to further increase computational efficiency, the numerical scheme is implemented at multiple scales. We use a power-of-two pyramid scheme for computing the solution. In this approach, a solution is computed using a subset of the data that is downsampled by a power of 2 in each dimension. This solution is then upsampled to a larger subset of the data and used as an initial condition for the initial-value problem at that scale. The initial condition for the initial-value problem using the full data set is therefore a linearly interpolated version of the computed solution one scale down. In this way, an initial condition $\Phi(\vec{x}, 0)$ that well approximates the steady-state solution of (3.16) is obtained in a computationally efficient manner.

4.1. A fixed-point formulation of the segmentation algorithm. We now make note of the fact that iteration (4.2) can also be formulated as a fixed-point iteration for solving the nonlinear equation (3.15). Specifically, by multiplying (3.15) by Δt and then adding Φ to both sides of the equation, we see that (3.15) is equivalent to

$$(\mathcal{I} - \Delta t\alpha\,(\mathrm{div}(g\nabla)))\,\Phi = \Phi + \Delta t\beta(c_2 - c_1)\left(D - \frac{1}{2}(c_1 + c_2)\right).$$

If we evaluate the right-hand side of this equation at Φ^n and solve for Φ to obtain Φ^{n+1}, the resulting fixed-point iteration is given by (4.2).

The fixed-point iteration (4.2) can also be expressed in quasi-Newton form. To see this, we note that if we define

$$F(\Phi) = \Delta t\alpha \mathrm{div}(g\nabla\Phi) + \Delta t\beta(c_2 - c_1)\left(D - \frac{1}{2}(c_1 + c_2)\right), \tag{4.5}$$

(4.2) can be written

$$\Phi^{n+1} = \Phi^n - MF(\Phi^n), \tag{4.6}$$

where $M = (\Delta t\alpha\,(\mathrm{div}(g\nabla)) - \mathcal{I})^{-1}$. In our numerical implementation, however, (4.4) is used, which is equivalent to (4.6) with

$$M = \frac{1}{3}\sum_{s\in\{x,y,z\}} (\mathcal{I} - 3\alpha\Delta t A_s^n)^{-1} - \mathcal{I}. \tag{4.7}$$

Equation (4.6) can be viewed as a quasi-Newton method in which the operator M in (4.7) serves as an approximation of the inverse of the Jacobian of F at Φ in (4.5).

Formulation (4.6)–(4.7) of (4.2) allows us to make some observations regarding the convergence properties of this algorithm. First, we recall from above that (4.4) is convergent and unconditionally stable. Time-step size restrictions are therefore greatly relaxed when compared to the corresponding fully explicit formulation, which is given by (4.6) with $M = \mathcal{I}$. This fact, together with the quasi-Newton formulation (4.6)–(4.7), suggests that we can expect (4.4) to converge more rapidly than the fully explicit method. Our numerical experiments support this observation, though a more thorough convergence analysis is saved for a future work.

Before continuing, we make mention of a few other fast methods for minimizing the Chan–Vese segmentation functional: these include the hybrid k-means level set algorithm of [15], which reduces the segmentation problem to the solution of an ODE in order to speed up convergence; the fast algorithm of [34], which does not require the evaluation of the gradient of the segmentation functional (3.6); and the two-stage algorithm of [13], which requires the solution of a related PDE that allows for fast solutions.

4.2. Results. As was noted above, the algorithm is implemented in three dimensions. Because of this both computational and memory issues arise. In particular, although the data I and background model B can be stored as images using relatively small amounts of memory, and D can be computed directly from I and B, the level set function must be stored using floating point values at each node in the data array. For a video sequence of 436 images of 512×512 pixels, this requires almost a gigabyte of memory when floating point values are 64-bit. In addition, the computation is

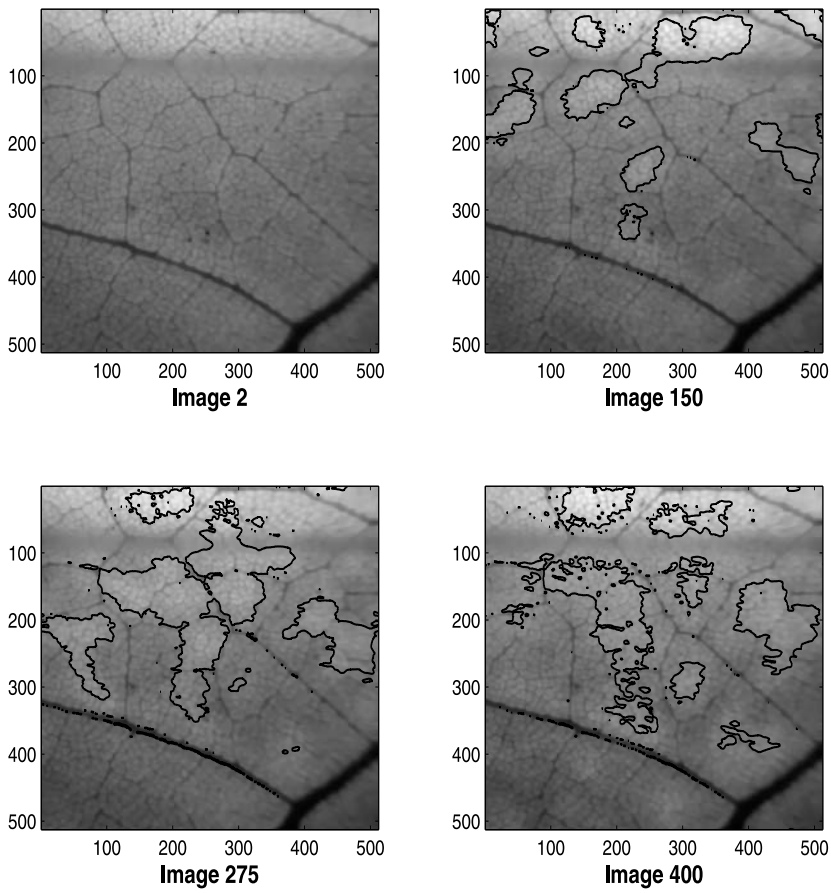


FIG. 4.1. *Segmented images* $2, 150, 275,$ *and* 400.

intensive due, primarily, to the large number of linear system backsolves required at each time iteration (708608 for this particular video sequence at the highest scale).

Our algorithm is implemented on an SGI Origin 2000 with 32 processors and 32 GB of shared memory and is fully parallelized using OpenMP. Though none of the individual processors on this machine is fast by current standards (300 Mhz), the large memory capacity and parallel capabilities make the machine suitable for this application. Though each iteration at the highest scale is quite slow, few iterations are required at the highest scale, since the approximate solution computed at the lower scales is already quite good.

Since the numerical scheme is unconditionally stable with respect to Δt, it is possible to choose Δt to be quite large. For the experiments shown in this work, $\Delta t = 0.5$, though it is also possible to start the algorithm with a larger value of Δt, decreasing it incrementally as the solution stabilizes. The ratio of the weighting factors α and β in (3.16) determines the amount of regularization used in the numerical

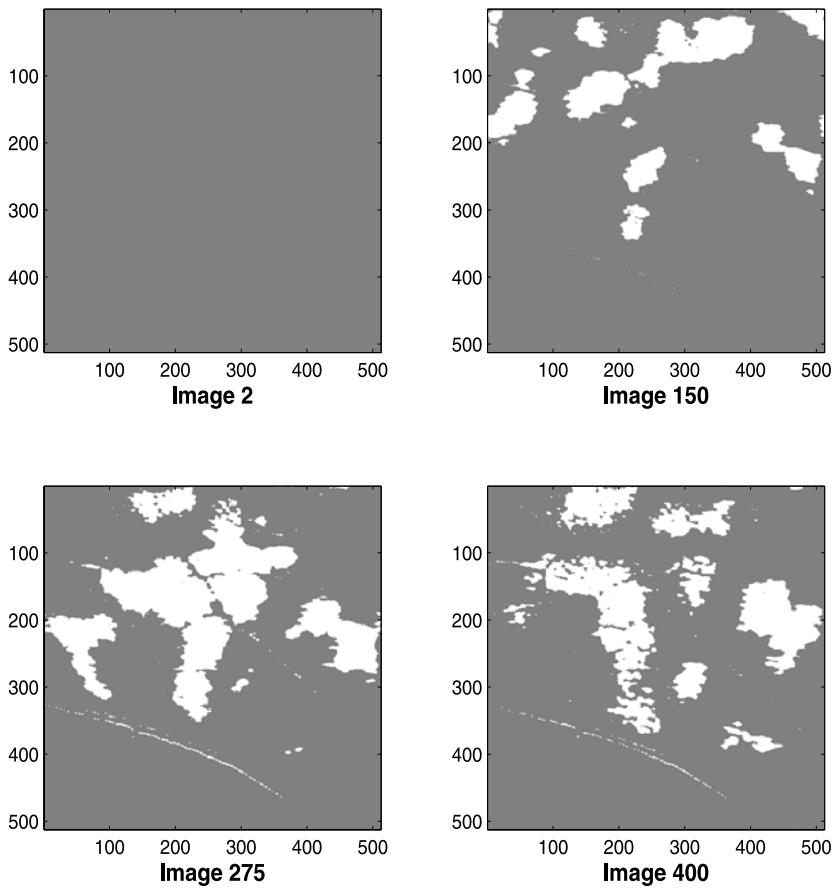


FIG. 4.2. *Binary segmentation for images* $2, 150, 275,$ *and* 400.

implementation, and for the examples shown here, $\alpha = 1$ and $\beta = 2$.

The last important detail of the numerical implementation is the initial surface from which we evolve our solution. For the examples shown here, the initial surface is a sphere contained completely inside the computational domain, and the initial level set representation of the surface, $\Phi(\vec{x}; 0)$, is defined to be the signed distance between $\vec{x}$ and the sphere boundary. The distance is taken to be negative inside the sphere and positive outside the sphere.

The segmented images can be seen in Figure 4.1[1]. Note that there is no segmentation in image 2. This is due to the fact that there are no dynamics in the early images of the video, and it is correctly captured because the algorithm is working on the full three-dimensional video sequence. When a two-dimensional version of the segmentation algorithm is applied to each individual frame, meaningless segmenta-

[1]The full videos of the original, preprocessed, and segmented data can be seen at http://hans.blc.edu/ luttman/.

tion artifacts are obtained in the early frames. This difference is due to the fact that in three-dimensions the algorithm is able to globally compare the changes between the video sequence and the background model, whereas, when each frame is considered individually, no such comparison is possible, resulting in differences between the images and the background model being overemphasized in frames with little-to-no dynamics.

The main goal in this project is to provide a binary segmentation of the video into light and dark regions, since, from such a segmentation, a study of two-dimensional patch dynamics can more easily be done. This is easily obtained from the results given in Figure 4.1 and can be seen in Figure 4.2. The regions of fluorescence are in white and the nonfluorescent regions are in gray.

5. Conclusions. The formation and dynamics of stomatal patches in plant leaves is a phenomenon that is not fully understood. In order to study this process, plant biologists collect video in which dynamically changing light and dark stomatal patches are evident. An in-depth study of the dynamics of these patches is desirable and requires a segmentation of such video into fluorescing (light) and nonfluorescing (dark) regions.

In this paper, we have presented an algorithm that provides such a segmentation for a video sequence with example frames given in Figure 1.1. The algorithm consists of several stages. First, due to experimental inconsistencies, the video in our example requires preprocessing. This is accomplished by first using a denoising algorithm for the removal of high frequency (instrumental) noise. The video is then normalized against global lighting changes, which removes low frequency (experiment specific) noise. Images from the preprocessed video are given in Figure 2.1.

The three-dimensional video segmentation is formulated as a variational inverse problem. Since the problem is ill-posed, a regularization term is added so that solutions of the corresponding minimization problem provide a sufficiently smooth segmentation of the video. The desired minimizer of the segmentation functional is computed via a time-continuous gradient descent method in the form of a three-dimensional evolution equation that is solved efficiently using a semi-implicit time discretization with additive operator splitting. A fixed-point formulation of the numerical scheme is also presented. Frames 2, 150, 275, and 400 of the segmented video are given in Figures 4.1 and 4.2.

Acknowledgments. The authors would like to thank Professor Emily Stone of the University of Montana for introducing us to this problem and for many helpful discussions regarding the leaf data and also Professors David Peak and Keith Mott of Utah State University for providing the leaf data and for helpful discussions regarding the current theories of stomatal patchiness. The authors would also like to thank the referees for their helpful comments and suggestions.

REFERENCES

[1] E. Angelini, R. Otsuka, S. Homma, and A. Laine, *Comparison of ventricular geometry for two real-time 3d ultrasound machines with three-dimensional level set*, in Proceedings of the IEEE International Symposium on Biomedical Imaging (ISBI), Arlington, VA, Vol. 1, 2004, pp. 1323–1326.

[2] J. F. Aujol, G. Gilboa, T. Chan, S. Osher, *Structure-texture image decomposition - modeling, algorithms, and parameter selection*, International Journal of Computer Vision (IJCV), 67, (2006), pp. 111–136.

[3] A. Blake, *Introduction to Active Contours and Visual Dynamics*, in Web Reference, 1999.

[4] J. Canny, *A computational approach to edge detection*, IEEE Trans. Pattern Anal. Mach. Intell., 8 (1986) pp. 679–698.
[5] V. Caselles, R. Kimmel, and G. Sapiro, *On geodesic active contours*, International Journal of Computer Vision, 22 (1997), pp. 61–79.
[6] M. Ceccarelli, V. D. Simone, and A. Murli, *Well-posed anisotropic diffusion for image denoising*, IEE Proceedings Vision, Image and Signal Processing, 149 (2002), pp. 244–252.
[7] A. Chambolle and J. Darbone, *Algorithms for Total Variation Minimization*, Mathematical Models for Multi-Channel Image Processing, Beijing, 2006.
[8] T. F. Chan and S. Esedoglu, *Aspects of total variation regularized L^1 function approximation*, SIAM J. Appl. Math., 65 (2005), pp. 1817–1837.
[9] T. F. Chan, G. H. Golub, and P. Mulet, *A nonlinear primal-dual method for total variation based image restoration*, SIAM J. Sci. Comput., 20 (1999), pp. 1964–1977.
[10] T. F. Chan and J. Shen, *Image Processing and Analysis in* Variational, PDE, Wavelet, and Stochastic Methods, SIAM, Philadelphia, 2005.
[11] T. F. Chan and L. A. Vese, *Active contours without edges*, IEEE Trans. Image Process., 10 (2001), pp. 266–277.
[12] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr, *Anisotropic feature-preserving denoising of height fields and bivariate data*, Graphics Interface, 2000, pp. 145–152.
[13] S. Esedoglu and Y.-H. Tsai., *Threshold dynamics for the piecewise constant Mumford–Shah functional*, J. Comput. Phys., 211 (2006), pp. 367–384.
[14] B. Genty and S. Meyer, *Quantitative mapping of leaf photosynthesis using chlorophyll fluorescence imaging*, Australian Journal of Plant Physiology, 22 (1994), pp. 277–284.
[15] F. Gibou and R. Fedkiw, *A fast hybrid k-means level set algorithm for segmentation*, in Proceedings of the 4th Annual Hawaii International Conference on Statistics and Mathematics, Waikiki Beach, HI, 2005, pp. 281–291.
[16] H. Jin, A. J. Yezzi, and S. Soatto, *Region-based segmentation on evolving surfaces with application to 3D reconstruction of shape and piecewise constant radiance*, in Proceedings of the European Conference on Computer Vision, Prague, Czech Republic, 2004 pp. 114–125.
[17] M. Kass, A. Witkin, and D. Terzopoulos, *Snakes: Active contour models*, International Journal of Computer Vision, 1 (1987), pp. 321–331.
[18] R. Kimmel, *Fast edge integration*, in Geometric Level Set Methods, S. Osher and N. Paragios, eds., Springer-Verlag, New York, 2003, pp. 59–77.
[19] R. Kimmel, *Numerical Geometry of Images,* in Theory, Algorithms, and Applications, Springer-Verlag, New York, 2004.
[20] C. Li, C. Xu, C. Gui, and M. D. Fox, *Level set evolution without re-initialization: A new variational formulation*, in IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), San Diego, CA, 2005, pp. 430–436.
[21] T. Lü, P. Neittaanmäki, and X-C. Tai, *A parallel splitting up method and its application to Navier–Stokes equations*, Appl. Math. Lett., 4 (1991), pp. 25–29.
[22] T. Lü, P. Neittaanmäki, and X-C. Tai, *A parallel splitting-up method for partial differential equations and its application to Navier–Stokes equations*, RAIRO Model. Math. Anal. Numér., 26 (1992), pp. 673–708.
[23] A. Luttman, *A Three-Dimensional Variational Approach to Video Segmentation*, Ph.D. thesis, University of Montana, Missoula, MT, 2006.
[24] K. W. Morton and D. F. Mayers, *Numerical Solutions of Partial Differential Equations An Introduction*, Cambridge University Press, Cambridge, UK, 1994.
[25] K. A. Mott, Z. G. Cardon, and J. A. Berry, *Asymmetric patchy stomatal closure for the two surfaces of Xanthium strumarium L. leaves at low humidity*, Plant, Cell, and Environment, 16 (1993), pp. 25–34.
[26] D. Mumford and J. Shah, *Optimal approximation by piecewise smooth functions and associated variational problems*, Comm. Pure Appl. Math., 42 (1989), pp. 577–685.
[27] N. Paragios and R. Deriche, *Coupled geodesic active regions for image segmentation: A level set approach*, in Proceedings of the European Conference on Computer Vision, 2, 2000 Dublin, Ireland, pp. 224–240.
[28] D. Peak, J. West, S. Messinger, and K. Mott, *Evidence for complex, collective dynamics and emergent, distributed computation in plants*, Proceedings of the National Academy of Sciences, 101 (2004), pp. 918–927.
[29] P. Perona and J. Malik, *Scale-space and edge detection using anisotropic diffusion*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 12 (1990), pp. 629–639.
[30] L. I. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Physica D, 60 (1992), pp. 259–268.

[31] P. K. Sahoo, S. Soltani, and A. K. C. Wong, *A survey of thresholding techniques*, Computer Vision, Graphics, and Image Processing, 41 (1988), pp. 233–260.

[32] S. Serra-Capizzano, *A note on antireflective boundary conditions and fast deblurring models*, SIAM J. Sci. Comput., 25 (2003), pp. 1307–1325.

[33] D. L. Snyder, A. M. Hammond, and R. L. White, *Image recovery from data acquired with a charged-coupled-device camera*, Journal of the Optical Society of America A, 10 (1993), pp. 1014–1023.

[34] B. Song and T. F. Chan, *A Fast Algorithm for Level Set Based Optimization*, UCLA CAM Report 02-68, UCLA, Los Angeles, CA, 2002.

[35] B. Sumengen, B. S. Manjunath, and C. Kenney, *Image segmentation using multi-region stability and edge strength*, in IEEE International Conference on Image Processing (ICIP), vol. 3, 2003, pp. 429–432.

[36] C. R. Vogel and M. E. Oman, *Iterative methods for total variation denoising*, SIAM J. Sci. Comput., 17 (1996), pp. 227–238.

[37] J. Weickert and G. Kühne, *Fast methods for implicit active contour models*, in Geometric Level Set Methods, S. Osher and N. Paragios, eds., Springer-Verlag, New York, 2003, pp. 43–58.

[38] J. Weickert, B. M. ter Haar Romeny, and M. A. Viergever, *Efficient and reliable schemes for nonlinear diffusion filtering*, IEEE Trans. Image Process., 7 (1998).

[39] Q. Wu and Y. Yu, *Two-level image segmentation based on region and edge integration*, in Proceedings of the Seventh Digital Image Computing Conference: Techniques and Applications, Sydney, Australia, 2003, pp. 957–964.

[40] H. Zhao, T. F. Chan, B. Merriman, and S. Osher, *A variational level set approach to multiphase motion*, J. Comput. Phys., 127 (1996), pp. 179–195.

SIAM J. SCI. COMPUT.
Vol. 30, No. 1, pp. 548
© 2008 Society for Industrial and Applied Mathematics

ERRATUM: A VARIATIONAL APPROACH TO VIDEO SEGMENTATION FOR BOTANICAL DATA*

AARON LUTTMAN† AND JOHN BARDSLEY‡

Abstract. In this erratum, we correct the previous explanation of the data-collection procedure as outlined in [*SIAM J. Sci. Comput.*, 29 (2007), pp. 1550–1566].

DOI. 10.1137/070700036

In [1], the authors incorrectly described the data-collection procedure for capturing the botanical fluorescence data.

- In the abstract, the sentence
  "In order to visualize a particular leaf's stomatal aperture, an experimentalist injects the leaf with a dye so that it fluoresces."

  should be reworded as

  "In order to visualize a particular leaf's stomatal aperture, fluorescence is induced corresponding to changes in average stomatal aperture size, and the data are collected at regular intervals."

- On page 2, in the first full paragraph, the sentence
  "A dye is injected into the leaf that fluoresces when the stomata close, and stomatal dynamics are initiated by decreasing the concentration of H_2O on the upper surface of the leaf."

  should be replaced with

  "Chlorophyll molecules in an intact leaf that are excited by visible light, upon deexcitation, dissipate part of the excitation energy as infrared fluorescence. Stomatal dynamics are then initiated by decreasing the concentration of H_2O on the upper surface of the leaf."

These changes reflect that the data-collection procedure used for the data in this work was updated, and the data actually shown in this work are derived from the updated procedure.

REFERENCES

[1] A. LUTTMAN AND J. BARDSLEY, *A variational approach to video segmentation for botanical data*, SIAM J. Sci. Comput., 29 (2007), pp. 1550–1566.

*Received by the editors August 13, 2007; accepted for publication September 14, 2007; published electronically February 8, 2008. This work was partially supported by the NSF under grant DMS-0504325 and by Montana NSF EPSCoR.
http://www.siam.org/journals/sisc/30-1/70003.html
†Department of Mathematical Sciences, University of Montana, Missoula, MT 59812, and Division of Science and Mathematics, Bethany Lutheran College, Mankato, MN 56001 (luttman@blc.edu).
‡Department of Mathematical Sciences, University of Montana, Missoula, MN 59812 (bardsleyj@mso.umt.edu).

SIAM J. SCI. COMPUT.
Vol. 29, No. 4, pp. 1567–1592
© 2007 Society for Industrial and Applied Mathematics

BOUNDARY CONDITIONS IN THE SCHUR COMPLEMENT PRECONDITIONING OF DISSIPATIVE ACOUSTIC EQUATIONS*

MIKA MALINEN†

Abstract. This paper is concerned with the development of robust and efficient iterative techniques for solving the coupled systems of linear equations which arise from a mixed finite element discretization of dissipative acoustic equations. The system of linearized, time-harmonic acoustic equations we consider is based on the Navier–Stokes, continuity, and energy equations accompanied by suitable equations of state. In this paper, we focus on the design of preconditioners that can be used in combination with the Krylov subspace methods. Our starting point is the Schur complement formulation of the mixed problem. One of the difficulties encountered with the approach we consider is generally associated with the issue of defining boundary conditions for the associated Schur complement problem. To derive suitable boundary conditions for this problem, we first consider an iterative solution method which is based on the direct use of the Schur complement formulation. This development naturally leads us to address the key question of what boundary conditions should be adjoined to the Schur complement preconditioner. We give numerical examples which demonstrate the effectiveness and robustness of the proposed preconditioner. The examples include both two- and three-dimensional problems with boundary conditions of various types.

Key words. acoustics, iterative algorithms, preconditioning, finite element method, dissipation

AMS subject classifications. 65N22, 65N30, 76N99

DOI. 10.1137/050629720

1. Introduction.

In this paper we consider the development of robust and efficient iterative techniques for solving the coupled systems of linear equations which arise from a mixed finite element discretization of dissipative acoustic equations. The system of linearized acoustic equations we consider characterizes the flow of a viscous and thermally conducting fluid. This coupled model can be derived by starting from the Navier–Stokes, continuity, and energy equations accompanied by suitable equations of state. Here we limit our consideration to the time-harmonic analysis.

The full system of acoustic equations we consider has not customarily been used as a starting point for the development of numerical methods. The fact that for fluids of low viscosity and conductivity (for example, air) the dissipative effects of viscosity and heat conduction are pronounced only in a thin boundary layer adjacent to a solid boundary has motivated the use of simplified equations. These simplified models approximate the full system of equations for the region either next to the boundaries or outside the boundary layers (for an overview of simplified models see, for example, [1]). Our use of the full model has been motivated by a need for modeling the propagation of sound in small cavities, which nevertheless are so large that the boundary layer equations cannot be used to obtain a proper approximation throughout the entire cavity. Problems posed on regions of such size arise, for example, in the advanced design of mobile phones.

We note that in principle the use of the full model could be circumvented by incorporating the dissipative effects of viscosity and heat conduction into the boundary conditions on the solution mode which dominates in the region outside the boundary

*Received by the editors April 21, 2005; accepted for publication (in revised form) April 11, 2007; published electronically July 11, 2007. This work was supported by Nokia Research Center.
http://www.siam.org/journals/sisc/29-4/62972.html
†CSC–Scientific Computing Ltd., P.O. Box 405, FI-02101 Espoo, Finland (mika.malinen@csc.fi).

layers [2]. Except for simple problems, such approximate boundary conditions are not directly available, but they should be found iteratively. One possible segregated solution strategy is outlined in [2]. We have found nevertheless that to find appropriate boundary conditions by using such a method is not straightforward, in particular for the range of frequencies near resonance. The avoidance of these difficulties has also motivated our direct start from the full system of acoustic field equations.

The wider range of applicability of the full model with the easiness of specifying suitable boundary conditions comes with the cost of the solution of a larger linear system that arises from the discretization. For most cases of practical interest the system of linear equations is so large that direct solution methods cannot be applied. Instead, iterative methods such as the Krylov subspace methods [3] have to be used. In this paper, we focus on the development of robust and efficient preconditioners that are required to obtain or improve the convergence of iterative methods.

One possible preconditioning strategy leading to preconditioners that are straightforward to implement would be to use the incomplete LU factorizations [4]. The drawback of the basic preconditioners based on this approach is that, when they are applied to the problem we consider, the use of a low level of fill may lead to slow convergence, while the memory usage grows rapidly as more fill is allowed. Often they also perform less advantageously when the discretization is refined. Experience gained from the solution of the systems of linear equations which arise from the approximation of related problems such as the Stokes and incompressible Navier–Stokes equations (see, for example, [5, 6]) indicates that more efficient preconditioning is possible by exploiting the structure of the systems. A preconditioning strategy we explore in this paper is based on the Schur complement approach leading us to considering preconditioners that have block-triangular structure [7].

In general, one of the key components for the design of Schur complement preconditioners is the development of an accurate and inexpensive way to approximate the action of the inverse of an associated Schur complement operator. Another central issue associated with the preconditioner design is the question of defining suitable boundary conditions for the approximation of the Schur complement operator [7, 6]. It seems that this question has not been fully explored. A more special feature of the problem considered is that the associated linear systems are complex-valued. The preconditioning of complex linear systems has been less investigated than the case of real linear systems.

Developing suitable boundary conditions for the Schur complement preconditioner will be the main subject of this paper. General mixed boundary conditions with the surface force prescribed on a part of the boundary are considered, as such boundary conditions are often met in acoustic analyses in the form of impedance boundary conditions. We note that the key principles we use here appear to be generally applicable, although the type of flow problem (such as incompressibility) can have an effect on the details of derivation in a particular case. The demonstration of these observations will be the subject of a further study.

The paper is structured as follows. In section 2 we introduce the system of acoustic equations we consider. It is noted that here we do not make the common assumption of a perfect gas but allow the consideration of more general fluids. Suitable equations of state are obtained by linearized versions of relations that characterize the fluctuations of the pressure and entropy about an equilibrium state. We then transform the system of field equations into a dimensionless form. This development is useful, not only

because it gives insight into the relative importance of various terms in the equations, but because it provides guidelines for constructing reliable finite element methods for the approximation of the system.

The construction of reliable finite element methods for the system of dissipative acoustic equations has already been considered in our previous work [8]. The mixed formulation which is used as a starting point for the finite element approximation is presented in section 3. In this paper, we do not discuss the discretization of the mixed problem in detail. It should be noted that the preconditioner we propose is flexible in the sense that it can be adapted to handle various discretizations which utilize different finite element spaces for the approximation of the velocity, temperature, and an auxiliary unknown introduced in connection with the mixed formulation.

The main development of this paper is contained in section 4, where the derivation of preconditioners is considered. By splitting the velocity solution to the flow problem into a rotational and an irrotational part, we first derive an alternative formulation for the mixed system of field equations. This formulation may be used to separate the solution of the temperature and auxiliary unknown from that of the velocity. By arranging the variables, the system may thus be written in the form of a block-triangular system with the matrix differential operator acting on the temperature and auxiliary unknown now being the exact Schur complement. In principle, this Schur complement formulation opens up a possibility for solving the acoustic problem. However, this approach involves a practical difficulty as the correct boundary conditions for the Schur complement are not generally available. Although the triangular system cannot be employed to directly obtain the solution, we show that this system supplemented by suitable boundary conditions can be used to develop preconditioners. The resulting preconditioner can be employed in combination with iteration methods that are applied directly to the linear system which result from the discretization of the boundary-value problem.

The results of numerical experiments are presented in section 5. The examples include both two- and three-dimensional problems with boundary conditions of various types. The results show that efficient preconditioning is possible, in particular when the proposed preconditioner is applied in combination with nested versions of iterative methods based on the minimal residual approach. The results indicate also that convergence rates are independent of the discretization mesh size. Finally, in section 6 we give a brief summary of the main conclusions and mention some open issues.

2. The mathematical model. In the following we consider a three-dimensional body Ω occupied by a fluid. We denote the outward unit normal vector to its boundary $\partial\Omega$ by $\mathbf{e}_n$ and let $\mathbf{e}_s$ and $\mathbf{e}_t$ be mutually orthogonal unit vectors tangential to the boundary. In addition, the velocity, density, pressure, and temperature fields associated with the fluid flow are denoted by $\mathbf{V}(\mathbf{x},t)$, $\rho(\mathbf{x},t)$, $p(\mathbf{x},t)$, and $T(\mathbf{x},t)$, respectively, with $\mathbf{x}$ and t being place and time. Notations ρ_0, p_0, and T_0 will be used for the values of the density, pressure, and temperature, respectively, at an equilibrium state.

2.1. The boundary-value problem. We consider the acoustic equations based on the linearized equation of motion, the constitutive equation relating the stress to the motion, the kinematic relation, the linearized continuity equation, and the

linearized energy equation:

$$
\begin{aligned}
\rho_0 \frac{\partial \mathbf{V}}{\partial t} &= \operatorname{div} \mathbf{T} + \rho_0 \mathbf{b}, \\
\mathbf{T} &= -p\mathbf{I} + \lambda(\operatorname{div} \mathbf{V})\mathbf{I} + 2\mu \mathbf{D}(\mathbf{V}), \\
\mathbf{D}(\mathbf{V}) &= \frac{1}{2}(\operatorname{grad} \mathbf{V} + \operatorname{grad} \mathbf{V}^T), \\
\frac{\partial \rho}{\partial t} &= -\rho_0 \operatorname{div} \mathbf{V}, \\
\rho_0 \frac{du}{dt} &= \kappa \operatorname{div} \operatorname{grad} T - p_0 \operatorname{div} \mathbf{V} + \rho_0 h.
\end{aligned} \tag{1}
$$

Here $\mathbf{T}$ and $\mathbf{I}$ are the stress and identity tensor, respectively, $\mathbf{b}$ is the body force (per unit mass), λ and μ are parameters characterizing the viscosity of the fluid, u is the specific internal energy, κ is the heat conductivity, and h is the internal supply of heat.

To supplement the system (1) by suitable equations of state, we assume that the properties of the medium are expressible as functions of two state variables, say, the temperature and density. We denote the specific entropy (entropy per unit mass) and its equilibrium value by s and s_0, respectively, and assume that the relation

$$
du = T_0 ds + (p_0/\rho_0^2) d\rho \tag{2}
$$

is valid. In addition, we approximate the equations which give the changes of pressure and specific entropy in terms of the changes of the state variables by (cf. [2, equation (6.4.21)])

$$
p - p_0 = \frac{(\gamma - 1)\rho_0 C_V}{T_0 \beta}(T - T_0) + \frac{(\gamma - 1)C_V}{T_0 \beta^2}(\rho - \rho_0) \tag{3}
$$

and

$$
s - s_0 = \frac{C_V}{T_0}(T - T_0) - \frac{C_V(\gamma - 1)}{T_0 \rho_0 \beta}(\rho - \rho_0), \tag{4}
$$

where C_V is the specific heat at constant volume (per unit mass), γ is the ratio of the specific heats at constant pressure and constant volume, and β is the coefficient of thermal expansion:

$$
\beta = -\frac{1}{\rho}\Big(\frac{\partial \rho}{\partial T}\Big)_p. \tag{5}
$$

We approximate the value of β by evaluating (5) for the equilibrium values of ρ and p.

We confine our consideration to the time-harmonic case, so that the solutions of the primary unknowns are assumed to be of the form

$$
\begin{aligned}
\mathbf{V}(\mathbf{x}, t) &= \mathbf{V}(\mathbf{x}) \exp(i\omega t), \\
\rho(\mathbf{x}, t) &= \rho_0 + \rho(\mathbf{x}) \exp(i\omega t), \\
T(\mathbf{x}, t) &= T_0 + T(\mathbf{x}) \exp(i\omega t),
\end{aligned} \tag{6}
$$

where i is the imaginary unit and ω is the angular frequency. By the substitution of (6), the system of field equations based on (1)–(4) may be reduced to a system where

the only unknown fields are the amplitudes $\mathbf{V}(\mathbf{x})$ and $T(\mathbf{x})$ of the disturbances of the velocity and temperature fields. The reduced system may be written as

$$
\begin{aligned}
i\omega\rho_0\mathbf{V} + \frac{(\gamma-1)C_V\rho_0}{\beta T_0}\operatorname{grad} T - \left(\lambda+\mu - \frac{i(\gamma-1)C_V\rho_0}{\omega T_0\beta^2}\right)\operatorname{grad}\operatorname{div}\mathbf{V} \\
- \mu\operatorname{div}\operatorname{grad}\mathbf{V} = \rho_0\mathbf{b}, \\
-\kappa\operatorname{div}\operatorname{grad} T + i\omega\rho_0 C_V T + \frac{(\gamma-1)C_V\rho_0}{\beta}\operatorname{div}\mathbf{V} = \rho_0 h.
\end{aligned} \tag{7}
$$

It is noted that after the solution of the velocity and temperature the amplitudes $p(\mathbf{x})$ and $\rho(\mathbf{x})$ of the disturbances of the pressure and density fields can readily be obtained from the relations

$$
\begin{aligned}
p &= \frac{(\gamma-1)C_V\rho_0}{\beta T_0}\left(T + \frac{i}{\omega\beta}\operatorname{div}\mathbf{V}\right), \\
\rho &= \frac{i\rho_0}{\omega}\operatorname{div}\mathbf{V}.
\end{aligned} \tag{8}
$$

To solve the system (7), suitable boundary conditions have to be adjoined to it. The most straightforward boundary conditions suitable for fluid flow in a region completely enclosed by rigid walls consist of the requirement that

$$\mathbf{V} = \widehat{\mathbf{V}} \quad \text{and} \quad T = 0 \tag{9}$$

on the boundary, with $\widehat{\mathbf{V}}$ a prescribed function on the boundary. The solution methods considered here admit the consideration of general mixed boundary conditions, so that at each point of the boundary one may prescribe

$$\mathbf{V}\cdot\mathbf{e}_n \quad \text{or} \quad \mathbf{T}\mathbf{e}_n\cdot\mathbf{e}_n \text{ but not both,} \tag{10}$$

$$\mathbf{V}\cdot\mathbf{e}_s \quad \text{or} \quad \mathbf{T}\mathbf{e}_n\cdot\mathbf{e}_s \text{ but not both, and} \tag{11}$$

$$\mathbf{V}\cdot\mathbf{e}_t \quad \text{or} \quad \mathbf{T}\mathbf{e}_n\cdot\mathbf{e}_t \text{ but not both,} \tag{12}$$

where $\mathbf{T}\mathbf{e}_n$ is the surface force vector. Similarly, mixed thermal boundary conditions may also be specified so that at each point of the boundary one prescribes

$$T \quad \text{or} \quad \frac{\partial T}{\partial n} \quad \text{but not both,} \tag{13}$$

where $\partial/\partial n$ denotes the directional derivative in the direction of $\mathbf{e}_n$.

We note that in acoustic simulations the normal surface force is often prescribed by using the impedance boundary condition

$$\mathbf{T}\mathbf{e}_n\cdot\mathbf{e}_n = Z_V(\mathbf{V}\cdot\mathbf{e}_n), \tag{14}$$

where Z_V is a complex quantity which we will refer to as the specific acoustic impedance. We express a similar thermal boundary condition as

$$\frac{\partial T}{\partial n} = Z_T T, \tag{15}$$

with the complex quantity Z_T referred to as the specific thermal impedance.

For simplicity of presentation we consider in the following the model problem where the Dirichlet boundary conditions (9) are imposed on the part Γ_D of the

boundary. On the remainder $\Gamma_I = \partial\Omega \setminus \Gamma_D$ the boundary conditions are given by (14) and (15) together with the tangential conditions

$$\mathbf{T}\mathbf{e}_n \cdot \mathbf{e}_s = \mathbf{T}\mathbf{e}_n \cdot \mathbf{e}_t = 0. \tag{16}$$

Other combinations of boundary conditions could be treated without difficulty.

2.2. Dimensionless field equations. It will be convenient to transform the system (7) into a dimensionless form. To this end, we introduce the dimensionless quantities

$$\theta = kL, \qquad \epsilon = \left(\frac{\mu}{\rho_0 \omega L^2}\right)^{1/2}, \qquad \delta = \left(\frac{C_V \mu}{\kappa}\right)^{1/2}, \qquad \eta = \lambda/\mu, \tag{17}$$

where $L = \operatorname{diam}(\Omega)$ and $k = \omega/c$, with c the adiabatic sound speed defined by the relation

$$T_0 \beta^2 c^2 = \gamma(\gamma - 1) C_V. \tag{18}$$

We also introduce dimensionless unknowns by setting

$$\mathbf{v} = (1/c)\mathbf{V}, \qquad \tau = \frac{\beta T_0}{\gamma\theta}\left(\frac{T}{T_0}\right) \tag{19}$$

and define scaled coordinates x_i' by $x_i' = L^{-1} x_i$.

When the dimensionless variables are introduced, the field equations (7) take the form

$$\begin{aligned} &\mathbf{v} - i\nabla\tau + \frac{1}{\gamma\theta^2}\nabla(\nabla\cdot\mathbf{v}) + i\epsilon^2(\eta+1)\nabla(\nabla\cdot\mathbf{v}) + i\epsilon^2\Delta\mathbf{v} = -\left(\frac{i}{\omega c}\right)\mathbf{b}, \\ &-\frac{i\gamma\epsilon^2\theta^2}{(\gamma-1)\delta^2}\Delta\tau - \frac{\gamma\theta^2}{\gamma-1}\tau + i\nabla\cdot\mathbf{v} = \left(\frac{i\gamma\theta^2}{cL\beta T_0\omega^2}\right) h. \end{aligned} \tag{20}$$

Here ∇ denotes the gradient with respect to the dimensionless coordinates and $\Delta = \nabla \cdot \nabla$.

We close this section by noting that the dimensionless parameter θ characterizes the ratio of the characteristic length L to the acoustic wavelength, while ϵ characterizes the ratio of the thickness of the viscous boundary layer to L. Similarly, the parameter δ characterizes the ratio of the thickness of the viscous boundary layer to that of the thermal boundary layer.

3. Mixed formulation. Our exploration of numerical solution methods for (7) or, equivalently, (20) is motivated by the modeling of the propagation of sound in small cavities. The largest dimension of the cavity typically is at most of the same order of magnitude as the minimum wavelength of the sound over the frequency range considered. When low frequencies are considered, the dimensionless parameter θ thus tends to take small values in comparison to unity. Usually we also have $\epsilon \ll 1$ and $\delta \sim 1$, so that the boundary layers are thin as compared with the characteristic length L.

In [8] we demonstrated that, when $\theta \ll 1$, the straightforward finite element discretization of (20) based on the use of low-degree polynomials does not lead to a reliable approximation method for the solution mode which dominates in the region outside the boundary layers. This was shown to be due to the fact that nearly

incompressible velocity fields need to be approximated. These observations led us to formulate the problem as a mixed problem for which reliable finite element methods were shown to be available.

We write here the mixed formulation of (20) as

$$\begin{aligned} \mathbf{v} - i\nabla\tau - i\nabla\phi + i\epsilon^2\nabla(\nabla\cdot\mathbf{v}) + i\epsilon^2\Delta\mathbf{v} &= i\mathbf{f}, \\ -\frac{i\epsilon^2}{(\gamma-1)\delta^2}\Delta\tau - \frac{1}{\gamma-1}\tau + \frac{1}{1+i\gamma\theta^2\epsilon^2\eta}\phi &= g, \\ i\nabla\cdot\mathbf{v} - \frac{\gamma\theta^2}{1+i\gamma\theta^2\epsilon^2\eta}\phi &= 0, \end{aligned} \tag{21}$$

where we have set

$$\mathbf{f} = -\left(\frac{1}{\omega c}\right)\mathbf{b} \quad \text{and} \quad g = \left(\frac{i}{cL\beta T_0\omega^2}\right)h. \tag{22}$$

Setting $\hat{\mathbf{v}} = (1/c)\widehat{\mathbf{V}}$, the Dirichlet boundary conditions corresponding to our model problem are then

$$\mathbf{v} = \hat{\mathbf{v}} \quad \text{and} \quad \tau = 0 \tag{23}$$

on Γ_D. On the complementary part Γ_I the boundary conditions are given by

$$\begin{aligned} -i\tau - i\phi + 2i\epsilon^2\mathbf{d}(\mathbf{v})\mathbf{e}_n\cdot\mathbf{e}_n &= z_v(\mathbf{v}\cdot\mathbf{e}_n), \\ 2i\epsilon^2\mathbf{d}(\mathbf{v})\mathbf{e}_n\cdot\mathbf{e}_s &= 0, \\ 2i\epsilon^2\mathbf{d}(\mathbf{v})\mathbf{e}_n\cdot\mathbf{e}_t &= 0, \end{aligned} \tag{24}$$

and

$$\frac{i\epsilon^2}{(\gamma-1)\delta^2}\frac{\partial\tau}{\partial n} = z_\tau\tau, \tag{25}$$

where $\mathbf{d}(\mathbf{v}) = 1/2(\nabla\mathbf{v} + \nabla\mathbf{v}^T)$ is the dimensionless stretching tensor and

$$z_v = \frac{iZ_V}{\rho_0\omega L}, \qquad z_\tau = \frac{i\epsilon^2 LZ_T}{(\gamma-1)\delta^2}. \tag{26}$$

It is also noted that the pressure, which often is of practical interest, may now be obtained readily in terms of τ and the auxiliary unknown ϕ from

$$p = \theta\rho_0 c^2\left(\tau + \frac{1}{1+i\gamma\theta^2\epsilon^2\eta}\phi\right). \tag{27}$$

4. Iterative solution methods. We start this section by deriving an alternative formulation for the mixed system of field equations. We then show how this alternative formulation can be used to develop preconditioners for the iterative solution of discrete systems which result from the finite element approximation of the mixed problem.

4.1. The Schur complement formulation of the mixed problem. It will be convenient to split the velocity solution $\mathbf{v}$ to (21) into components $\mathbf{w} \in W$ and $\mathbf{u} \in U$, with W and U defined by

$$W = \{\mathbf{w} \mid \nabla \times \mathbf{w} = \mathbf{0}\}$$

and

$$U = \{\mathbf{u} \mid \nabla \cdot \mathbf{u} = 0\}.$$

In addition, to introduce the similar representation for the body force, we denote the body-force potential, the gradient of which gives the conservative part of the scaled body force $\mathbf{f}$, by φ. Then $\mathbf{f}$ may be written as

$$\mathbf{f} = \nabla\varphi + (\mathbf{f} - \nabla\varphi). \tag{28}$$

It is noted that these representations for the velocity and body force are unique.

By substituting $\mathbf{v} = \mathbf{w} + \mathbf{u}$ and (28) into (21) and using the identity

$$\Delta\mathbf{v} = \nabla(\nabla \cdot \mathbf{v}) - \nabla \times \nabla \times \mathbf{v},$$

it is found that $\mathbf{w}$, τ, and ϕ satisfy the field equations

$$\begin{aligned} \mathbf{w} - i\nabla\tau - i\nabla\phi + 2i\epsilon^2\,\nabla(\nabla \cdot \mathbf{w}) &= i\nabla\varphi, \\ -\frac{i\epsilon^2}{(\gamma-1)\delta^2}\Delta\tau - \frac{1}{\gamma-1}\tau + \frac{1}{1+i\gamma\theta^2\epsilon^2\eta}\phi &= g, \\ i\nabla \cdot \mathbf{w} - \frac{\gamma\theta^2}{1+i\gamma\theta^2\epsilon^2\eta}\phi &= 0. \end{aligned} \tag{29}$$

As we now have

$$\mathbf{w} = \nabla(i\tau) + \frac{1+i\gamma\theta^2\epsilon^2(2+\eta)}{1+i\gamma\theta^2\epsilon^2\eta}\nabla(i\phi) + \nabla(i\varphi), \tag{30}$$

we may reduce (29) into a system involving only τ and ϕ:

$$\begin{aligned} -\frac{i\epsilon^2}{(\gamma-1)\delta^2}\Delta\tau - \frac{1}{\gamma-1}\tau + \frac{1}{1+i\gamma\theta^2\epsilon^2\eta}\phi &= g, \\ -\Delta\tau - \frac{1+i\gamma\theta^2\epsilon^2(2+\eta)}{1+i\gamma\theta^2\epsilon^2\eta}\Delta\phi - \frac{\gamma\theta^2}{1+i\gamma\theta^2\epsilon^2\eta}\phi &= \Delta\varphi. \end{aligned} \tag{31}$$

We may thus separate the solution of τ and ϕ from that of $\mathbf{v}$, so that the system (21) may be rewritten as

$$\begin{aligned} A(\mathbf{v}) + B^*(\tau) + B^*(\phi) &= i\mathbf{f}, \\ C(\tau) + D(\phi) &= g, \\ E(\tau) + G(\phi) &= \Delta\varphi, \end{aligned} \tag{32}$$

where

$$\begin{aligned} &A = I + i\epsilon^2\,\nabla(\nabla\cdot\,) + i\epsilon^2\Delta, \quad B^* = -i\nabla, \\ &C = -\frac{i\epsilon^2}{(\gamma-1)\delta^2}\Delta - \frac{1}{\gamma-1}, \quad D = \frac{1}{1+i\gamma\theta^2\epsilon^2\eta}I, \\ &E = -\Delta, \quad G = -\frac{1+i\gamma\theta^2\epsilon^2(2+\eta)}{1+i\gamma\theta^2\epsilon^2\eta}\Delta - (\gamma\theta^2)D. \end{aligned} \tag{33}$$

We note that the matrix differential operator

$$S = \begin{pmatrix} C & D \\ E & G \end{pmatrix}, \tag{34}$$

which appears in the block-triangular system (32), is the Schur complement for the system of acoustic equations.

In principle, the Schur complement formulation (32) opens up a possibility for solving (21). However, this approach involves a practical difficulty as the correct boundary condition for the last equation of (32) depends on the normal component of the unknown part $\mathbf{w}$. This can also be seen from the variational formulation corresponding to this equation. If we take the inner product of the equation with $q \in H^1(\Omega)$ (the space $H^1(\Omega)$ is defined in the usual way [9]) and integrate by parts, we see that the solution of the acoustic problem satisfies

$$\begin{aligned} &-\frac{\gamma\theta^2}{1+i\gamma\theta^2\epsilon^2\eta}\int_\Omega \phi q\,d\Omega + \int_\Omega \nabla\tau\cdot\nabla q\,d\Omega \\ &+\frac{1+i\gamma\theta^2\epsilon^2(2+\eta)}{1+i\gamma\theta^2\epsilon^2\eta}\int_\Omega \nabla\phi\cdot\nabla q\,d\Omega + \int_\Omega \nabla\varphi\cdot\nabla q\,d\Omega + i\int_{\partial\Omega}\mathbf{w}\cdot\mathbf{n}q\,dS = 0. \end{aligned} \tag{35}$$

By substituting $\mathbf{w} = \mathbf{v} - \mathbf{u}$ we see that we have also

$$\begin{aligned} &-\frac{\gamma\theta^2}{1+i\gamma\theta^2\epsilon^2\eta}\int_\Omega \phi q\,d\Omega + \int_\Omega \nabla\tau\cdot\nabla q\,d\Omega + \frac{1+i\gamma\theta^2\epsilon^2(2+\eta)}{1+i\gamma\theta^2\epsilon^2\eta}\int_\Omega \nabla\phi\cdot\nabla q\,d\Omega \\ &+\int_\Omega \nabla\varphi\cdot\nabla q\,d\Omega - i\int_\Omega \mathbf{u}\cdot\nabla q\,d\Omega + i\int_{\partial\Omega}\mathbf{v}\cdot\mathbf{n}q\,dS = 0 \end{aligned} \tag{36}$$

for any function $q \in H^1(\Omega)$.

Although the system (32) cannot be employed to directly obtain the solution, it can be used to develop iteration methods for the solution of the original system (21). The next section shows that the variational principle (36) plays a key role in the derivation of iterative algorithms.

4.2. The iterative solution using consistent splitting approach. We now turn to the iterative solution of the system (21). The Schur complement formulation (32) is a natural starting point for the derivation of decoupled solution strategies where the solution of the temperature and auxiliary unknown is separated from that of the velocity. Our experiences with iteration methods that directly rely on the Schur complement formulation have nevertheless been discouraging. We have found that the convergence of such methods can be problematic, in particular for the range of frequencies near resonance. These difficulties have turned our interest to developing solution methods where decoupled solution techniques are used only as preconditioners in connection with other iteration methods. Such a possibility arises naturally when the Krylov subspace methods are applied directly to the linear systems which arise from the discretization of the mixed problem.

Although our primary interest here is the preconditioner design, we start our discussion by considering an iteration method which is based on the direct use of the Schur complement formulation (32). This will not be the method we are focusing on, but this introductory consideration gives insight into the key question of developing

suitable boundary conditions for the associated Schur complement operator. It should be noted that the method we consider in the following is closely related to the so-called consistent splitting algorithm [10], originally designed for solving the time-discretized incompressible Navier–Stokes equations.

Now let us assume that the solution of the system (21) is attempted by an iterative process where the new velocity $\mathbf{v}^{(k+1)}$ is computed using the previous approximation of (τ, ϕ) from

$$\mathbf{v}^{(k+1)} - i\nabla\tau^{(k)} - i\nabla\phi^{(k)} + i\epsilon^2\nabla(\nabla\cdot\mathbf{v}^{(k+1)}) + i\epsilon^2\Delta\mathbf{v}^{(k+1)} = i\mathbf{f}, \tag{37}$$

together with the boundary conditions

$$\mathbf{v}^{(k+1)} = \hat{\mathbf{v}} \quad \text{on } \Gamma_D \tag{38}$$

and

$$\begin{aligned} -i\tau^{(k)} - i\phi^{(k)} + 2i\epsilon^2\mathbf{d}(\mathbf{v}^{(k+1)})\mathbf{e}_n\cdot\mathbf{e}_n &= z_v(\mathbf{v}^{(k+1)}\cdot\mathbf{e}_n),\\ 2i\epsilon^2\mathbf{d}(\mathbf{v}^{(k+1)})\mathbf{e}_n\cdot\mathbf{e}_s &= 0,\\ 2i\epsilon^2\mathbf{d}(\mathbf{v}^{(k+1)})\mathbf{e}_n\cdot\mathbf{e}_t &= 0 \end{aligned} \tag{39}$$

on Γ_I. The new pair $(\tau^{(k+1)}, \phi^{(k+1)})$ is then constructed such that, in analogy with (36), we have

$$\begin{aligned} &-\frac{\gamma\theta^2}{1+i\gamma\theta^2\epsilon^2\eta}\int_\Omega \phi^{(k+1)} q\,d\Omega + \int_\Omega \nabla\tau^{(k+1)}\cdot\nabla q\,d\Omega\\ &+\frac{1+i\gamma\theta^2\epsilon^2(2+\eta)}{1+i\gamma\theta^2\epsilon^2\eta}\int_\Omega \nabla\phi^{(k+1)}\cdot\nabla q\,d\Omega + \int_\Omega \nabla\varphi\cdot\nabla q\,d\Omega\\ &-i\int_\Omega \mathbf{u}^{(k+1)}\cdot\nabla q\,d\Omega + i\int_{\partial\Omega}\mathbf{v}^{(k+1)}\cdot\mathbf{n}q\,dS = 0 \end{aligned} \tag{40}$$

for any function $q \in H^1(\Omega)$. Here $\mathbf{u}^{(k+1)}$ is the rotational velocity corresponding to $\mathbf{v}^{(k+1)}$.

It should be noted that the boundary condition resulting from the principle (40) is consistent. If the current velocity iterate is the solution of the problem, then the new pair $(\tau^{(k+1)}, \phi^{(k+1)})$ resulting from the principle (40) together with the second equation of (32) solves the acoustic problem.

Here the key issue is how to compute in (40) the term depending on $\mathbf{u}^{(k+1)}$. Surprisingly, this can be done without explicitly forming $\mathbf{u}^{(k+1)}$. To show this, note that from (37) we get the equation

$$i\mathbf{v}^{(k+1)} + \nabla\tau^{(k)} + \nabla\phi^{(k)} - 2\epsilon^2\nabla(\nabla\cdot\mathbf{v}^{(k+1)}) + \epsilon^2\nabla\times\nabla\times\mathbf{v}^{(k+1)} = -\mathbf{f}, \tag{41}$$

which may also be written as

$$i\mathbf{v}^{(k+1)} + \nabla\tau^{(k)} + \nabla\phi^{(k)} - 2\epsilon^2\nabla(\nabla\cdot\mathbf{v}^{(k+1)}) - i\mathbf{u}^{(k+1)} = -\nabla\varphi. \tag{42}$$

Following the idea presented in [10], we now take the inner product of (42) with ∇q, where $q \in H^1(\Omega)$. In view of the Helmholtz–Hodge decomposition, the resulting

equation simply expresses the requirement that the projection of $i\mathbf{u}^{(k+1)} - i\mathbf{v}^{(k+1)}$ on the set irrotational fields coincides with the field

$$\nabla\tau^{(k)} + \nabla\phi^{(k)} - 2\epsilon^2\nabla(\nabla\cdot\mathbf{v}^{(k+1)}) + \nabla\varphi.$$

Taking the inner product and subtracting the result from (40), we then obtain

$$\begin{aligned}
&-\frac{\gamma\theta^2}{1+i\gamma\theta^2\epsilon^2\eta}\int_\Omega \phi^{(k+1)} q\,d\Omega + \int_\Omega \nabla\tau^{(k+1)}\cdot\nabla q\,d\Omega \\
&+\frac{1+i\gamma\theta^2\epsilon^2(2+\eta)}{1+i\gamma\theta^2\epsilon^2\eta}\int_\Omega \nabla\phi^{(k+1)}\cdot\nabla q\,d\Omega - i\int_\Omega \mathbf{v}^{(k+1)}\cdot\nabla q\,d\Omega \\
&-\int_\Omega \nabla\tau^{(k)}\cdot\nabla q\,d\Omega - \int_\Omega \nabla\phi^{(k)}\cdot\nabla q\,d\Omega + 2\epsilon^2\int_\Omega \nabla(\nabla\cdot\mathbf{v}^{(k+1)})\cdot\nabla q\,d\Omega \\
&+ i\int_{\partial\Omega}\mathbf{v}^{(k+1)}\cdot\mathbf{n} q\,dS = 0.
\end{aligned} \tag{43}$$

Integrating by parts and rearranging the terms, we finally get

$$\begin{aligned}
&-\frac{\gamma\theta^2}{1+i\gamma\theta^2\epsilon^2\eta}\int_\Omega (\phi^{(k+1)}-\phi^{(k)}) q\,d\Omega + \int_\Omega \nabla(\tau^{(k+1)}-\tau^{(k)})\cdot\nabla q\,d\Omega \\
&+\frac{1+i\gamma\theta^2\epsilon^2(2+\eta)}{1+i\gamma\theta^2\epsilon^2\eta}\int_\Omega \nabla(\phi^{(k+1)}-\phi^{(k)})\cdot\nabla q\,d\Omega \\
&=\int_\Omega\left(\frac{\gamma\theta^2}{1+i\gamma\theta^2\epsilon^2\eta}\phi^{(k)} - i\nabla\cdot\mathbf{v}^{(k+1)}\right) q\,d\Omega \\
&-2i\epsilon^2\int_\Omega \nabla\left(\frac{\gamma\theta^2}{1+i\gamma\theta^2\epsilon^2\eta}\phi^{(k)} - i\nabla\cdot\mathbf{v}^{(k+1)}\right)\cdot\nabla q\,d\Omega.
\end{aligned} \tag{44}$$

It is noted that the last term of (44) depends on the gradient field of the residual corresponding to the last equation of the system (21). It can be difficult to approximate this term directly in connection with the finite element method. This difficulty can be circumvented by approximating the residual by a continuous function ψ. We thus come to the following iterative method for solving (21) associated with the boundary conditions (23)–(25): Set $\tau^{(0)} = \phi^{(0)} = 0$ and $k = 0$. Then perform the following iteration steps:

1. Find $\mathbf{v}^{(k+1)} \in V$ such that

$$\begin{aligned}
&\int_\Omega \mathbf{v}^{(k+1)}\cdot\mathbf{z}\,d\Omega + i\int_\Omega \tau^{(k)}\nabla\cdot\mathbf{z}\,d\Omega + i\int_\Omega \phi^{(k)}\nabla\cdot\mathbf{z}\,d\Omega \\
&-2i\epsilon^2\int_\Omega \mathbf{d}(\mathbf{v}^{(k+1)})\cdot\mathbf{d}(\mathbf{z})\,d\Omega + i\int_{\Gamma_I} z_v(\mathbf{v}^{(k+1)}\cdot\mathbf{e}_n)(\mathbf{z}\cdot\mathbf{e}_n)\,dS = i\int_\Omega \mathbf{f}\cdot\mathbf{z}\,d\Omega
\end{aligned} \tag{45}$$

for all $\mathbf{z} \in Z$. Here we have

$$V = \{\ \mathbf{v} \mid \mathbf{v} \in [H^1(\Omega)]^3 \text{ and } \mathbf{v} = \hat{\mathbf{v}} \text{ on } \Gamma_D\}$$

and $Z = [H_D^1(\Omega)]^3$, with

$$H_D^1(\Omega) = \{\ z \mid z \in H^1(\Omega) \text{ and } z = 0 \text{ on } \Gamma_D\}.$$

2. Find $(\tau^{(k+1)}, \phi^{(k+1)}, \psi) \in H_D^1(\Omega) \times H^1(\Omega) \times H^1(\Omega)$ such that

$$\begin{aligned}
&\frac{i\epsilon^2}{(\gamma-1)\delta^2}\int_\Omega \nabla(\tau^{(k+1)} - \tau^{(k)}) \cdot \nabla r \, d\Omega - \frac{1}{\gamma-1}\int_\Omega (\tau^{(k+1)} - \tau^{(k)}) r \, d\Omega \\
&+ \frac{1}{1 + i\gamma\theta^2\epsilon^2\eta}\int_\Omega (\phi^{(k+1)} - \phi^{(k)}) r \, d\Omega - \int_{\Gamma_I} z_\tau (\tau^{(k+1)} - \tau^{(k)}) r \, dS \\
&= \int_\Omega g r \, d\Omega - \frac{i\epsilon^2}{(\gamma-1)\delta^2}\int_\Omega \nabla\tau^{(k)} \cdot \nabla r \, d\Omega + \frac{1}{\gamma-1}\int_\Omega \tau^{(k)} r \, d\Omega \\
&- \frac{1}{1 + i\gamma\theta^2\epsilon^2\eta}\int_\Omega \phi^{(k)} r \, d\Omega + \int_{\Gamma_I} z_\tau \tau^{(k)} r \, dS, \qquad (46)
\end{aligned}$$

$$\begin{aligned}
&- \frac{\gamma\theta^2}{1 + i\gamma\theta^2\epsilon^2\eta}\int_\Omega (\phi^{(k+1)} - \phi^{(k)}) q \, d\Omega + \int_\Omega \nabla(\tau^{(k+1)} - \tau^{(k)}) \cdot \nabla q \, d\Omega \\
&+ \frac{1 + i\gamma\theta^2\epsilon^2(2+\eta)}{1 + i\gamma\theta^2\epsilon^2\eta}\int_\Omega \nabla(\phi^{(k+1)} - \phi^{(k)}) \cdot \nabla q \, d\Omega \\
&+ 2i\epsilon^2 \int_\Omega \nabla\psi \cdot \nabla q \, d\Omega = \int_\Omega \Big(\frac{\gamma\theta^2}{1 + i\gamma\theta^2\epsilon^2\eta}\phi^{(k)} - i\nabla \cdot \mathbf{v}^{(k+1)}\Big) q \, d\Omega, \qquad (47)
\end{aligned}$$

and

$$\int_\Omega \psi s \, d\Omega = \int_\Omega \left(\frac{\gamma\theta^2}{1 + i\gamma\theta^2\epsilon^2\eta}\phi^{(k)} - i\nabla \cdot \mathbf{v}^{(k+1)}\right) s \, d\Omega \qquad (48)$$

for all $(r, q, s) \in H_D^1(\Omega) \times H^1(\Omega) \times H^1(\Omega)$. If residuals are small enough, stop the iteration. Otherwise, set $k = k + 1$ and go to Step 1.

Obviously, the first step of the algorithm could also be written in the form of residual correction. It is also noted that the iteration method could be discretized easily by choosing suitable finite element spaces for approximation. In practice, the auxiliary unknown ψ may be solved independently from (48), and the new solution pair $(\tau^{(k+1)}, \phi^{(k+1)})$ can thus be obtained from the coupled system of (46) and (47).

We have seen that the use of the principle (40) leads us to add an additional forcing term to the residual-correction system. We conclude this discussion by demonstrating what velocity boundary conditions underlie this modification. To this end, we split the new velocity solution into three components so that $\mathbf{v}^{(k+1)} = \mathbf{u}^{(k+1)} + \mathbf{w}^{(k)} + \boldsymbol{\delta}\mathbf{w}^{(k+1)}$, where $\mathbf{w}^{(k)}$ is the irrotational velocity corresponding to the pair $(\tau^{(k)}, \phi^{(k)})$. The component $\boldsymbol{\delta}\mathbf{w}^{(k+1)}$ is thus the consistency error giving the difference between the irrotational component of $\mathbf{v}^{(k+1)}$ and $\mathbf{w}^{(k)}$. It follows then that, when the new pair $(\tau^{(k+1)}, \phi^{(k+1)})$ satisfying (40) is solved, the normal component of the velocity is adjusted so that

$$(\mathbf{w}^{(k+1)} - \mathbf{w}^{(k)}) \cdot \mathbf{e}_n = \boldsymbol{\delta}\mathbf{w}^{(k+1)} \cdot \mathbf{e}_n. \qquad (49)$$

4.3. Preconditioned iteration methods for discrete systems. The ideas presented in the preceding section can easily be adapted to develop Schur complement preconditioners for the iterative solution of the discrete system

$$\mathbf{KU} = \mathbf{F} \tag{50}$$

which result from the finite element discretization of the system (21). In the following we consider the use of Krylov subspace methods in combination with right-oriented preconditioning.

The development of the preceding section implies that an approximation to the current error may be solved from the preconditioning system of the form

$$\begin{bmatrix} \mathbf{A} & \mathbf{B}^* & \mathbf{B}^* & \mathbf{0} \\ \mathbf{0} & \mathbf{C} & \mathbf{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{E} & \mathbf{G} & \mathbf{H} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{M} \end{bmatrix} \begin{bmatrix} \mathbf{s}_v \\ \mathbf{s}_\tau \\ \mathbf{s}_\phi \\ \boldsymbol{\psi} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_v \\ \mathbf{r}_\tau \\ \mathbf{r}_\phi \\ \mathbf{r}_\phi \end{bmatrix}. \tag{51}$$

Here the vector $\boldsymbol{\psi}$ contains the unknowns corresponding to the field ψ, and $\mathbf{r}_v$, $\mathbf{r}_\tau$, and $\mathbf{r}_\phi$ are the discrete residuals corresponding to the block structure of (50). In addition, the matrices $\mathbf{M}$ and $\mathbf{H}$ result from the finite element approximation of the identity and Laplace operators, and $\mathbf{A}$, $\mathbf{B}^*, \ldots$ result from the finite element approximation of the operators A, B^*, etc. The boundary conditions adjoined to these operators are as defined in (45)–(48).

By eliminating $\boldsymbol{\psi}$ from (51), we see that the corrections to the primary unknowns are obtained from the system

$$\mathbf{Ps} = \mathbf{r},$$

which has the block structure

$$\begin{bmatrix} \mathbf{A} & \mathbf{B}^* & \mathbf{B}^* \\ \mathbf{0} & \mathbf{C} & \mathbf{D} \\ \mathbf{0} & \mathbf{E} & \mathbf{G} \end{bmatrix} \begin{bmatrix} \mathbf{s}_v \\ \mathbf{s}_\tau \\ \mathbf{s}_\phi \end{bmatrix} = \begin{bmatrix} \mathbf{r}_v \\ \mathbf{r}_\tau \\ \hat{\mathbf{r}}_\phi \end{bmatrix},$$

with $\hat{\mathbf{r}}_\phi = \mathbf{r}_\phi - \mathbf{HM}^{-1}\mathbf{r}_\phi$. In practice the inverse is not constructed as it suffices to solve iteratively the system

$$\mathbf{M}\boldsymbol{\psi} = \mathbf{r}_\phi. \tag{52}$$

This is a cheap operation even when a stringent stopping criterion is used, the principal cost being the allocation of the matrix $\mathbf{M}$. It should also be noted that in practice the explicit construction of the preconditioner in the form (51) is not needed, since a part of the preconditioner matrix is already available.

To make the application of the preconditioner efficient, an inexpensive method for approximating the action of the inverse of $\mathbf{P}$ is required. In practice, in addition to solving (52), we need effective solvers for the linear systems of the type

$$\begin{bmatrix} \mathbf{C} & \mathbf{D} \\ \mathbf{E} & \mathbf{G} \end{bmatrix} \begin{bmatrix} \mathbf{s}_\tau \\ \mathbf{s}_\phi \end{bmatrix} = \begin{bmatrix} \mathbf{r}_\tau \\ \hat{\mathbf{r}}_\phi \end{bmatrix} \tag{53}$$

and

$$\mathbf{As}_v = \hat{\mathbf{r}}_v. \tag{54}$$

In our numerical experiments we will consider the solution of these subproblems by using another Krylov method.

The solution of the subproblems involving $\mathbf{A}$ being relatively inexpensive, the overall cost of the preconditioner depends to a large extent on the cost of the assembly and solution of the Schur complement systems. The development of optimal solution methods for this problem is an open issue. Nevertheless, due to the fact that the size of the Schur complement matrix is significantly smaller than that of $\mathbf{K}$, reasonably efficient solvers can be obtained by using iterative methods where the preconditioning is based on incomplete factorizations of the coefficient matrix. A similar strategy may also be applied to the solution of the subproblems involving $\mathbf{A}$, but it may be possible to solve these systems even without any extra storage by using the Jacobi preconditioning.

We note that ideally it should be possible to make use of rather inaccurate iterative solutions of the preconditioning systems. Otherwise, the computational cost of the preconditioning can be large. We have observed that, although methods based on the exact preconditioning may perform equally well, there can be drastic differences in the performances of their inexact versions. We have found that, when BICGSTAB or BICGSTAB(l) is used as the outer iterative method, accurate solutions of preconditioning systems are needed in order to maintain convergence. This is a serious drawback limiting the usefulness of these methods in this connection.

We have found that iterative methods which perform robustly with respect to inaccuracies in the preconditioning step with reduced storage requirements can be developed by employing nested GCR iterations. The numerical examples of the next section were solved using the following nested GCR algorithm which utilizes the Schur complement preconditioning in inner GCR iterations: Given an initial guess $\mathbf{U}^{(0)}$ and a convergence tolerance ϵ_{outer}, perform the iteration

$\mathbf{r}^{(0)} = \mathbf{F} - \mathbf{K}\mathbf{U}^{(0)}$
$k = 0$
`while` $\|\mathbf{r}^{(k)}\|_\infty < \epsilon_{\text{outer}}(\|\mathbf{K}\|_\infty \|\mathbf{U}^{(k)}\|_\infty + \|\mathbf{F}\|_\infty)$
  $\mathbf{s}^{(k+1)} = GCR(\mathbf{K}, \mathbf{P}, \mathbf{r}^{(k)}, m, \delta_{\text{inner}}, \epsilon_{\text{inner}})$
  $\mathbf{v}^{(k+1)} = \mathbf{K}\mathbf{s}^{(k+1)}$
  `do` $j = 1, k$
    $\mathbf{v}^{(k+1)} = \mathbf{v}^{(k+1)} - \langle \mathbf{v}^{(j)}, \mathbf{v}^{(k+1)} \rangle \mathbf{v}^{(j)}$
    $\mathbf{s}^{(k+1)} = \mathbf{s}^{(k+1)} - \langle \mathbf{v}^{(j)}, \mathbf{v}^{(k+1)} \rangle \mathbf{s}^{(j)}$
  `end do`
  $\mathbf{v}^{(k+1)} = \mathbf{v}^{(k+1)} / \|\mathbf{v}^{(k+1)}\|$
  $\mathbf{s}^{(k+1)} = \mathbf{s}^{(k+1)} / \|\mathbf{v}^{(k+1)}\|$
  $\mathbf{U}^{(k+1)} = \mathbf{U}^{(k)} + \langle \mathbf{v}^{(k+1)}, \mathbf{r}^{(k)} \rangle \mathbf{s}^{(k+1)}$
  $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \langle \mathbf{v}^{(k+1)}, \mathbf{r}^{(k)} \rangle \mathbf{v}^{(k+1)}$
  $k = k + 1$
`end while.`

Here we have $\langle \mathbf{v}, \mathbf{r} \rangle = \overline{\mathbf{v}} \cdot \mathbf{r}$, and the vector and matrix norms are defined in the usual manner. The function $GCR(\mathbf{K}, \mathbf{P}, \mathbf{r}^{(k)}, m, \delta_{\text{inner}}, \epsilon_{\text{inner}})$ returns the approximate solution of the residual equation

$$\mathbf{K}\mathbf{s}^{(k+1)} = \mathbf{r}^{(k)} \tag{55}$$

obtained by allowing at most m steps of the preconditioned GCR algorithm with the preconditioner $\mathbf{P}$ and the stopping criterion

$$\|\mathbf{r}^{(k)} - \mathbf{K}\mathbf{s}^{(k+1)}\| < \delta_{\text{inner}} \|\mathbf{r}^{(k)}\|. \tag{56}$$

The parameter ϵ_{inner} defines the error tolerance used in connection with the Schur complement preconditioning. In the case of the subproblem (54) the stopping criterion is given by

$$\|\hat{\mathbf{r}}_v - \mathbf{A}\mathbf{s}_v\| < \epsilon_{\text{inner}}\|\hat{\mathbf{r}}_v\|.$$

The stopping criterion for the subproblem (53) is analogous.

While the stopping criterion for the solution of (53) and (54) can be chosen to be mild, it can be advantageous to use a stringent stopping criterion for the solution of (52). As there is no principal cost associated with the solution of the subproblem (52), in the following we always solve these subproblems accurately so that we have

$$\|\mathbf{r}_\phi - \mathbf{M}\boldsymbol{\psi}\| < 10^{-7}\|\mathbf{r}_\phi\|.$$

We close this section by noting that the nested GCR algorithm we consider is similar to the GMRESR algorithm [11], provided the inner iteration of GMRESR is based on a version of GMRES that allows variable preconditioning.

5. Numerical examples. The proposed preconditioner has been implemented in Elmer [12], which is finite element software for the modeling of multiphysical problems. In the following we give the results of numerical experiments which demonstrate the performance of the proposed preconditioner in the case of both two- and three-dimensional problems. Boundary conditions of various types are considered. It is noted that some of the problems were solved using a version capable of handling axially symmetric problems.

The mixed problem is discretized using an enhanced MINI finite element of the type considered already in [8]. This method uses continuous, equal-order approximations for all of the unknown fields with the velocity approximation enriched by elementwise-defined bubble functions. The local bubble basis we use here is analogous to that of the MINI element [9], so that the velocity approximation is augmented by a single bubble degree of freedom.

To facilitate the computation of incomplete LU factorizations, the matrix $\mathbf{A}$ in the preconditioner (51) was constructed explicitly with the contribution resulting from the bubble functions neglected. It is emphasized that the double allocation of the $\mathbf{A}$-block could be avoided easily, so that the only additional assembly required by the preconditioner would be related to the discretization of the Schur complement operator and the construction of the matrices $\mathbf{H}$ and $\mathbf{M}$. In the computations, a row equilibration strategy was also used to reduce the condition numbers of the system matrix $\mathbf{K}$ and the coefficient matrices that arise in the preconditioning. The preconditioning systems were solved using the BICGSTAB(l) method.

5.1. Two-dimensional examples. We consider here a small system consisting of two cavities and a narrow duct which all have circular cross sections. We use axial symmetry to reduce the problem to two dimensions. The computational domain is given by $\Omega = \{\mathbf{x} = (x_1, x_2) \mid \mathbf{x} \in (0, R_1)\times(0, L_1)\cup(0, R_2)\times[L_1, L_2]\cup(0, R_3)\times(L_2, L_3)\}$, where x_1 and x_2 refer to the radial and axial coordinates, respectively, and $R_1 = 9.25 \cdot 10^{-3}$, $L_1 = 2.5 \cdot 10^{-2}$, $R_2 = 7.5 \cdot 10^{-4}$, $L_2 = 2.9 \cdot 10^{-2}$, $R_3 = 5.0 \cdot 10^{-3}$, and $L_3 = 4.4 \cdot 10^{-2}$. We assume that the medium is air at standard conditions and take $T_0 = 293$, $\beta = 1/T_0$, $\rho_0 = 1.21$, $\mu = 1.82 \cdot 10^{-5}$, $\lambda = -2/3\mu$, $\kappa = 2.6 \cdot 10^{-2}$, $C_V = 717.17$, and $\gamma = 1.4$. Here all values are expressed in SI units.

The wave source is defined by prescribing a nonvanishing velocity amplitude on the edge $x_2 = L_3$. We take $V_1(x_1) = 0$ and $V_2(x_1) = J_0(\alpha x_1)$, where J_0 is the Bessel

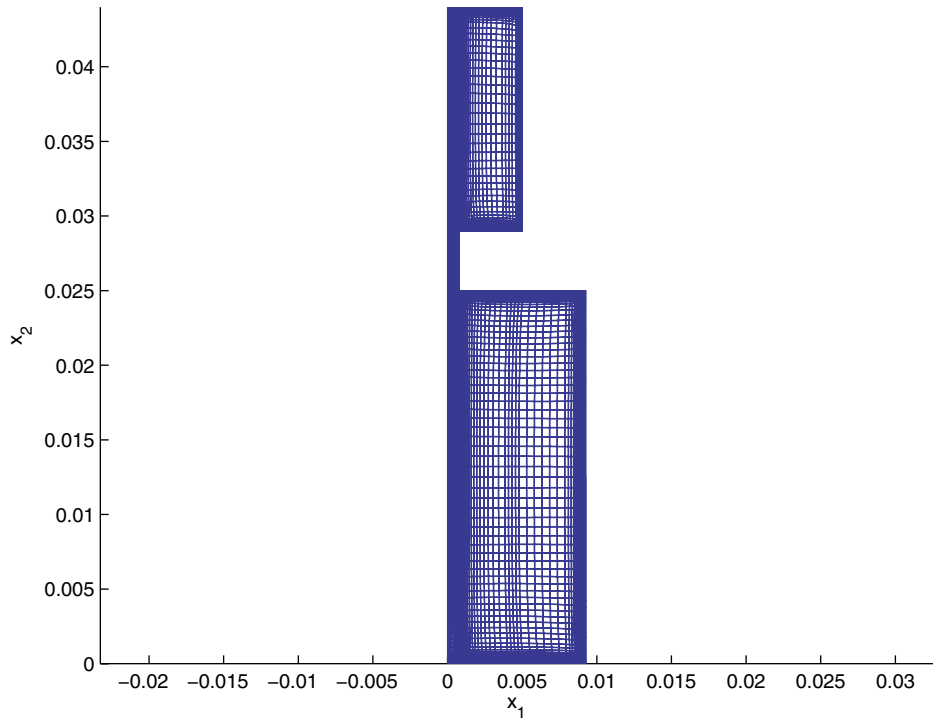


FIG. 5.1. *The finite element mesh.*

function with the maximum value scaled to the value $1.0 \cdot 10^{-3}$. In addition, the scaling factor α is defined so that $x_1 = \alpha R_3$ is the first root of $J_0(x_1)$. We assume that the fluctuation of the temperature vanishes on this edge. On the edge $x_1 = 0$, the symmetry boundary conditions are used (they correspond to the case of $Z_V \longrightarrow \infty$ and $Z_T = 0$). On the remaining part of the boundary, the conventional wall boundary conditions are imposed; i.e., all of the components of the velocity vector as well as the disturbance of the temperature are prescribed to vanish on this part.

We consider frequencies ranging from $f = 2.0 \cdot 10^2$ to $f = 10^4$, where $f = \omega/(2\pi)$. It is noted that the frequency $f = 1000$ is an approximate resonance frequency of the system. The same finite element mesh shown in Figure 5.1 was used to solve the problem for various frequencies. The mesh consists of bilinear quadrilaterals, and the total number of nodes is $N = 8848$ (leading to 70784 real unknowns).

We first demonstrate that the performance of the preconditioner depends drastically on the boundary conditions of the preconditioner. In addition to the proposed preconditioner, we also consider a variant where the terms depending on the auxiliary field ψ are omitted (in this case we have $\mathbf{H} = \mathbf{0}$). It is noted that this version corresponds to using the boundary condition

$$(\mathbf{w}^{(k+1)} - \mathbf{w}^{(k)}) \cdot \mathbf{e}_n = 0$$

instead of (49). We consider the example linear system which results from taking $f = 200$. In Figure 5.2 we illustrate the performances of the two versions by plotting

$$\frac{\|\mathbf{D}_K(\mathbf{F} - \mathbf{K}\mathbf{U}^{(k)})\|}{\|\mathbf{D}_K(\mathbf{F} - \mathbf{K}\mathbf{U}^{(0)})\|}, \tag{57}$$

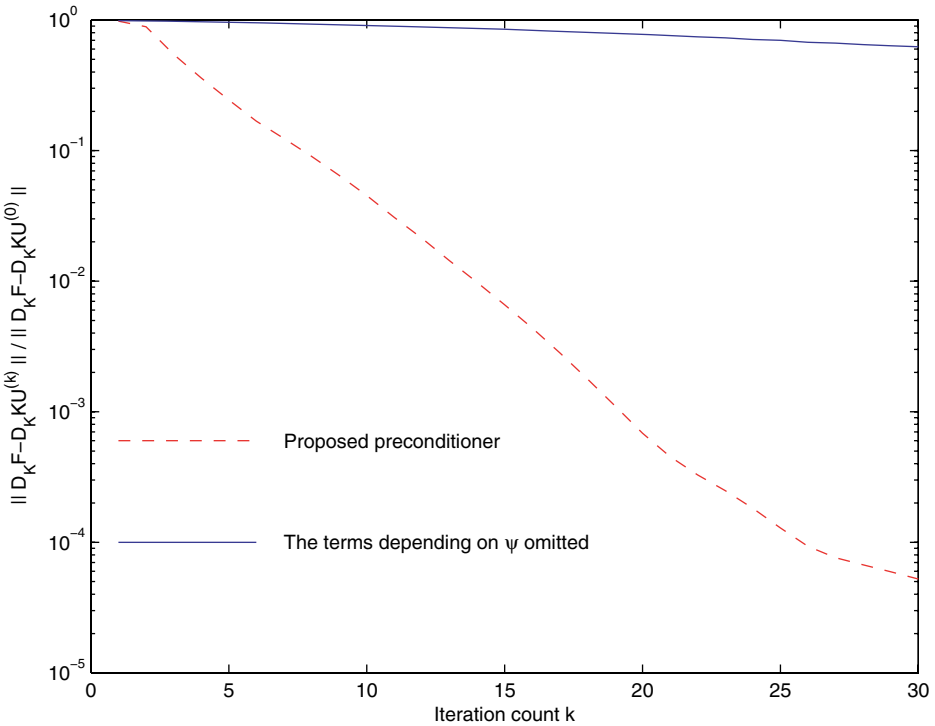


FIG. 5.2. *Convergence histories of the preconditioned GCR method. Two preconditioning strategies were used. The first one was based on omitting all of the terms which depend on ψ, while the second one was based on the proposed preconditioner.*

where $\mathbf{D}_K$ is the diagonal scaling matrix and $\mathbf{U}^{(k)}$, $k > 0$, are generated by the preconditioned GCR algorithm with $\epsilon_{\text{inner}} = 10^{-7}$. It is observed that the terms depending on ψ play a key role, as the omission of these terms essentially prohibits the convergence of residual-minimizing algorithms.

The key feature of the GCR method in connection with the Schur complement preconditioning is that the preconditioning systems can be solved inaccurately without essentially affecting the rate of convergence. Therefore the performance of our nested GCR algorithm depends only mildly on the value of ϵ_{inner}. This makes efficient preconditioning possible. It also follows that the outer iteration count required for convergence can effectively be controlled using the solver parameters m and δ_{inner}. We have found that often an efficient algorithm results when one takes, for example, $m = 5$ or $m = 10$ and $\delta_{\text{inner}} = 10^{-1}$ or $\delta_{\text{inner}} = 10^{-2}$.

To demonstrate the robustness of the nested GCR method with respect to inaccuracies in the preconditioning step, we consider the solution of the example linear system ($f = 200$) when $m = 10$, $\delta_{\text{inner}} = 10^{-2}$, and ϵ_{inner} is varied. Figure 5.3 shows the residual after each inner GCR iteration when $\epsilon_{\text{inner}} = 10^{-5}, \ldots, 10^{-2}$. Figure 5.4 shows the same residual vs. CPU time. It is seen from Figure 5.3 that increasing the value of ϵ_{inner} from 10^{-5} to 10^{-3} does not have a significant effect on the reduction of the residual obtained by the inner GCR iteration. Taking $\epsilon_{\text{inner}} = 10^{-2}$ has a clear effect on the residual reduction. In view of computing time the most efficient algorithm results when $\epsilon_{\text{inner}} = 10^{-3}$.

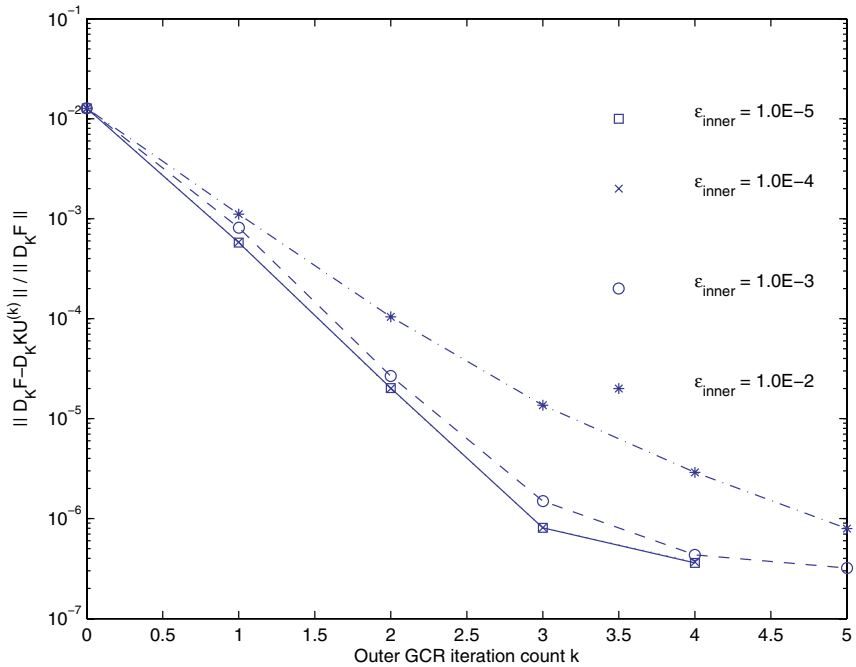


FIG. 5.3. *The residual after each inner GCR iteration when* ϵ_{inner} *is varied in the case of* $f = 200$.

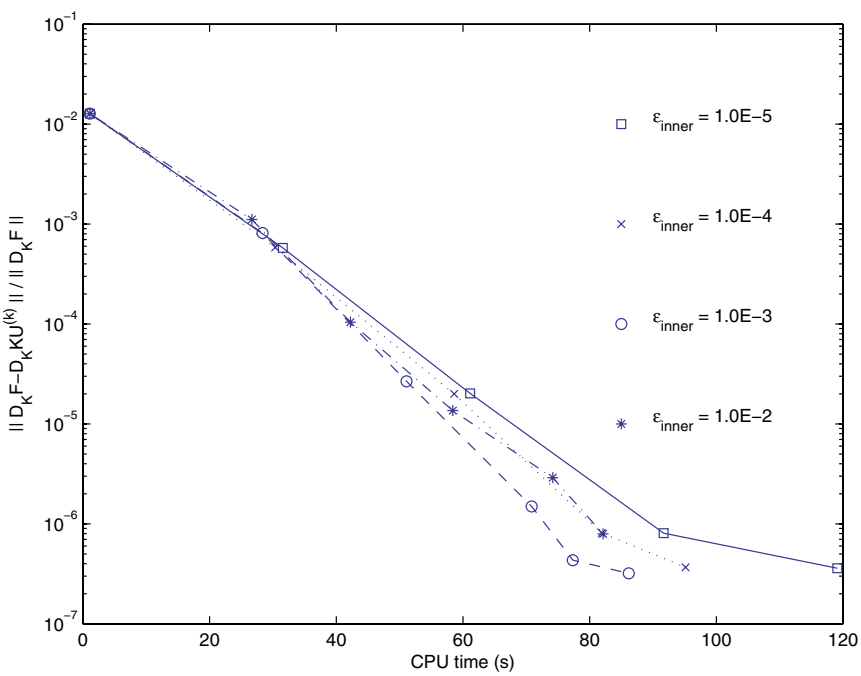


FIG. 5.4. *The residual after each inner GCR iteration vs. CPU time when* ϵ_{inner} *is varied in the case of* $f = 200$.

To demonstrate that the preconditioner is suitable for a wide range of frequencies, we also solved the linear systems which result from the sample frequencies $f = 10^3$ and $f = 10^4$. Figures 5.5–5.8 illustrate the convergence in the case of different values of ϵ_{inner}. It is noted that the results for the frequency $f = 10^4$ are comparable with those for the frequency $f = 200$. A different behavior is seen in the case of the resonance frequency $f = 10^3$, as the preconditioning systems have to be solved more accurately in this case. This can be explained by the greater sensitivity of the Schur complement matrix at the condition of resonance. It is not a surprise that, when the preconditioner is more sensitive, a more stringent stopping criterion is needed to produce a certain level of relative accuracy in the preconditioning.

We finally show how the nested GCR algorithm performs as compared with the BICGSTAB(l) method, which is applied directly to $\mathbf{KU} = \mathbf{F}$ with the preconditioner based on an incomplete LU factorization (ILU) of $\mathbf{K}$. We consider the cases of $l = 2$ and $l = 4$. In Table 5.1 we show the solution time when the fill level n of the incomplete factorization ILU(n) is varied. The iteration was stopped when

$$\|\mathbf{D}_K(\mathbf{F} - \mathbf{K}\mathbf{U}^{(k)})\| < 10^{-6}\|\mathbf{D}_K\mathbf{F}\|.$$

The maximum number of iterations allowed was taken to be 500. The results of Table 5.1 indicate that the incomplete factorization preconditioners perform less advantageously when the frequency and hence θ take smaller values (we have $\theta = 0.16$ when $f = 200$). When $f = 200$, the convergence was attained only in the case of ILU(0). It should be noted that a similar degradation of performance was not observed in the case of the Schur complement preconditioner. By comparing the CPU times, we see that, when incomplete factorization preconditioners with higher levels of fill work,

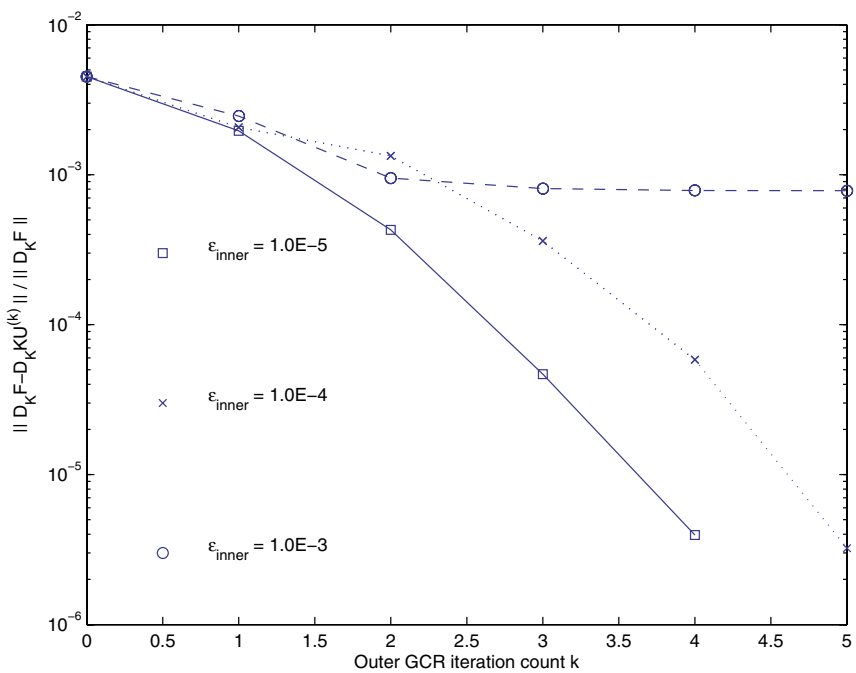


FIG. 5.5. *The residual after each inner GCR iteration when* ϵ_{inner} *is varied in the case of* $f = 10^3$.

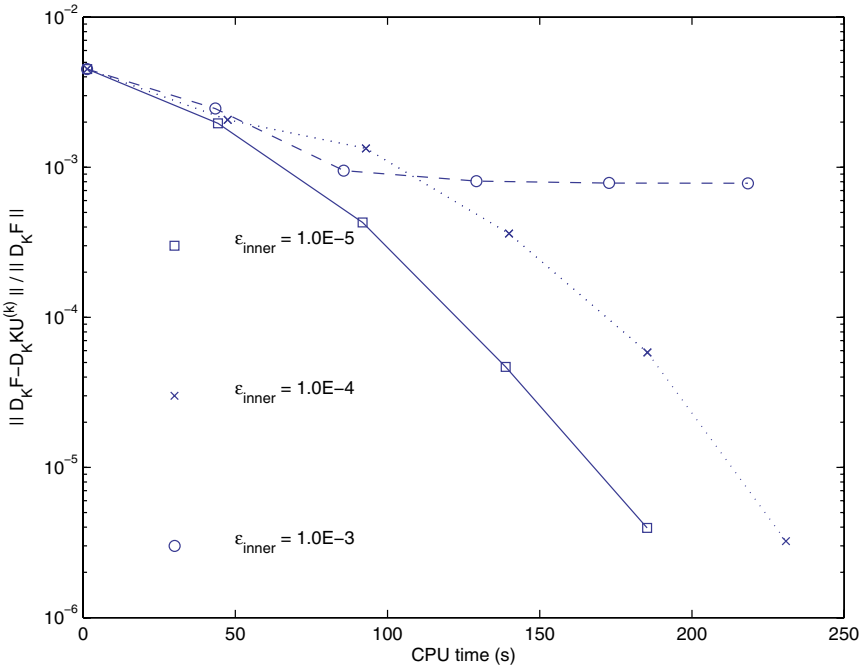


FIG. 5.6. *The residual after each inner GCR iteration vs. CPU time when* ϵ_{inner} *is varied in the case of* $f = 10^3$.

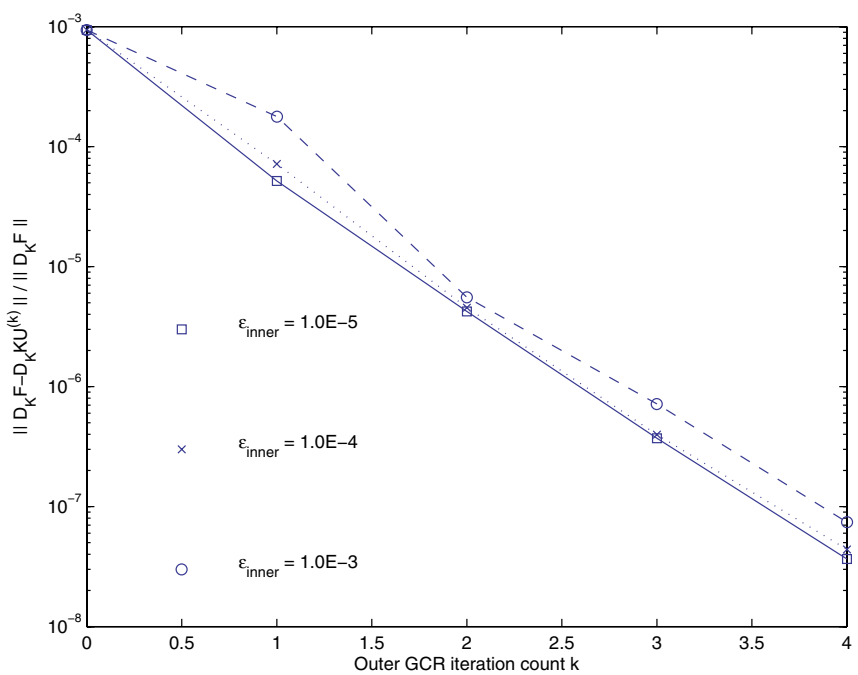


FIG. 5.7. *The residual after each inner GCR iteration when* ϵ_{inner} *is varied in the case of* $f = 10^4$.

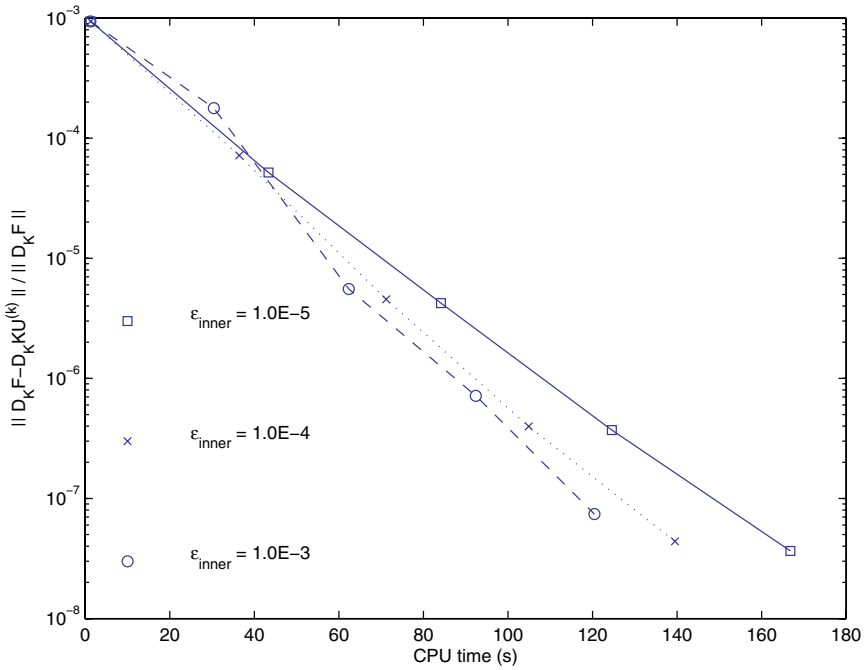


FIG. 5.8. *The residual after each inner GCR iteration vs. CPU time when* ϵ_{inner} *is varied in the case of* $f = 10^4$.

TABLE 5.1
The CPU time (s) to solve the linear system in the case of the BICGSTAB(l*) method.*

	$f = 200$		$f = 10^3$		$f = 10^4$	
	$l = 2$	$l = 4$	$l = 2$	$l = 4$	$l = 2$	$l = 4$
ILU(0)	138.2	109.0	130.6	122.9	82.3	47.9
ILU(1)	—	—	—	—	68.4	61.3
ILU(2)	—	—	60.4	52.8	22.1	20.2
ILU(3)	—	—	44.4	40.6	18.3	15.8

they can be more efficient than the current implementation of the Schur complement preconditioner. To improve the efficiency of the Schur complement preconditioner, a more efficient solver for the linear systems of the type (53) is required.

5.2. Three-dimensional examples. To demonstrate the ability of the preconditioner to handle three-dimensional problems, we now consider a curved tube which makes an angle of $\pi/2$ and has a circular cross section of radius $R = 1.0 \cdot 10^{-3}$. The tube is symmetric with respect to the coordinate plane $x_2 = 0$, so it suffices to consider half of the tube. The other dimensions of the tube are visualized in Figure 5.9, which shows the intersection of the three-dimensional tube with the plane $x_2 = 0$, along with the finite element mesh on that surface.

We assume that the medium is air at standard conditions and consider the case of the frequency $f = 10^4$. At the end $x_3 = 0$, we impose the velocity boundary conditions $V_1 = V_2 = 0$ and $V_3(r) = J_0(\alpha r)$, where $r^2 = x_1^2 + x_2^2$. The maximum value of the Bessel function is scaled to the value $1.0 \cdot 10^{-3}$, and the scaling factor α is defined so that $r = \alpha R$ is the first root of $J_0(r)$. In addition, we assume that the fluctuation of the temperature vanishes on this edge. At the other end of the tube,

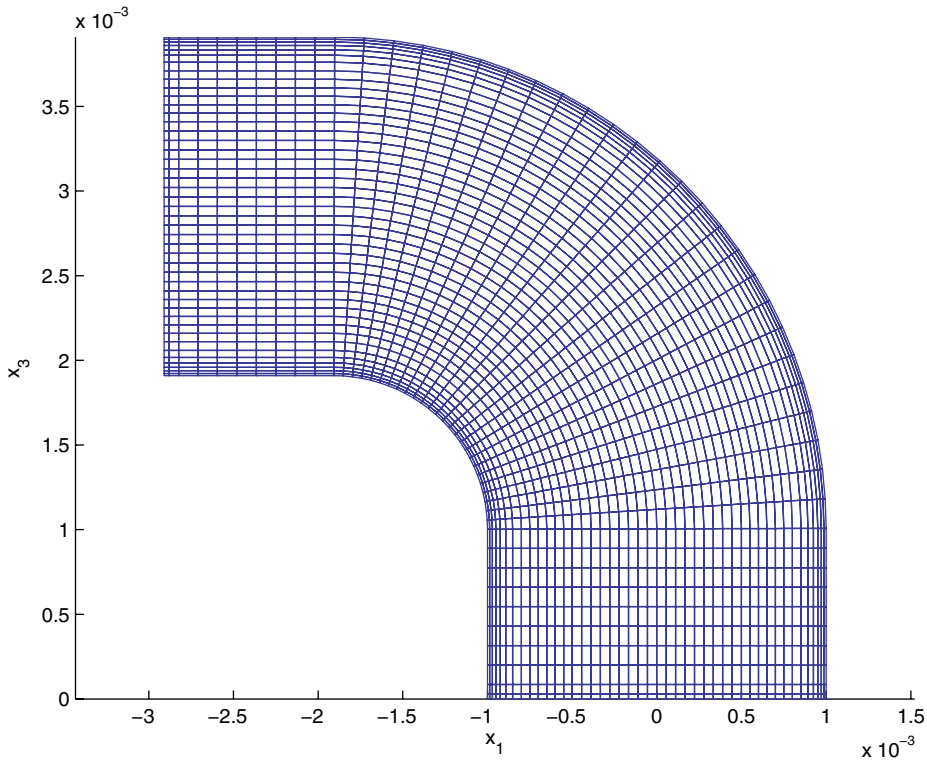


FIG. 5.9. *The intersection of the three-dimensional tube with the plane $x_2 = 0$. The finite element mesh on the surface is also displayed.*

we impose the impedance boundary conditions with the specific acoustic and thermal impedances given by

$$Z_V = -\rho_0 c$$

and

$$Z_T = -i\omega/c.$$

These boundary conditions are supplemented by the tangential velocity boundary conditions $V_2 = 0$ and $V_3 = 0$. On the surface $x_2 = 0$, the boundary conditions are obtained on the basis of symmetry, while on the remaining part of the boundary all of the components of the velocity vector as well as the disturbance of the temperature are prescribed to vanish.

The finite element mesh consists of trilinear elements and is visualized in Figures 5.9 and 5.10, where the intersections of the three-dimensional mesh with the planes $x_2 = 0$ and $x_3 = 0$ are shown. The total number of nodes is $N = 31537$ (leading to 315370 real unknowns). We solved the linear system using the nested GCR algorithm with $m = 10$, $\delta_{\text{inner}} = 10^{-2}$, and $\epsilon_{\text{inner}} = 10^{-2}$. The residual after each inner GCR iteration is again recorded in Table 5.2. We note that the convergence is comparable with that seen in two-dimensional examples.

The axially symmetric problem of section 5.1 was also solved in three dimensions. A quarter of the three-dimensional tube was discretized into tetrahedral elements

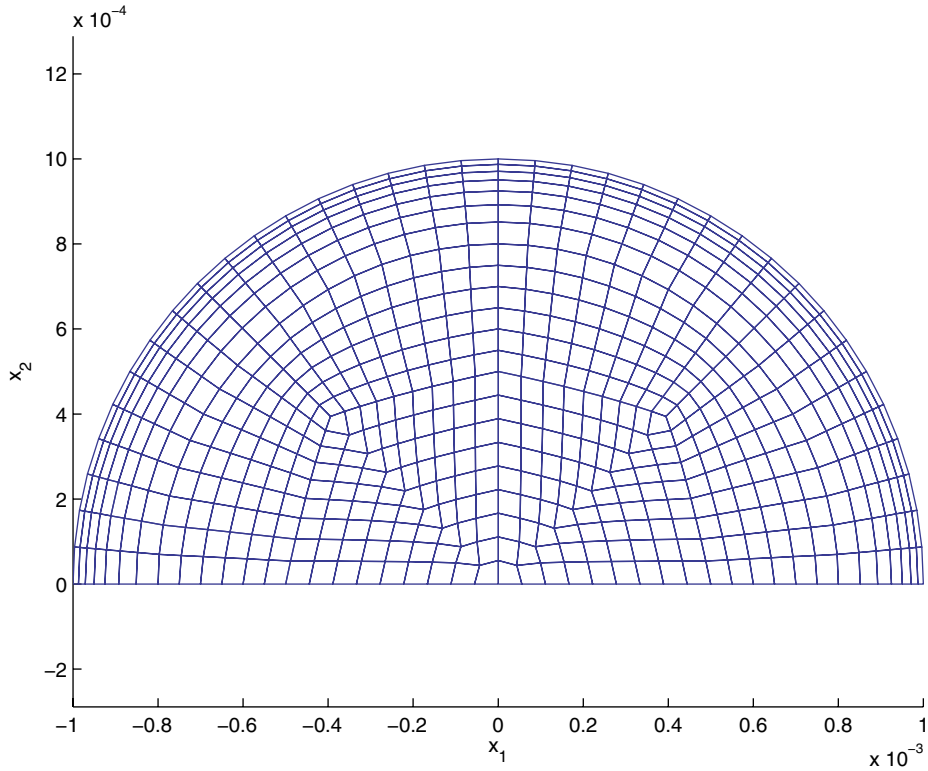


FIG. 5.10. *The intersection of the three-dimensional mesh with the plane* $x_3 = 0$.

TABLE 5.2
The convergence of the nested GCR method in the case of the three-dimensional example.

k	$\|\mathbf{D}_K(\mathbf{F} - \mathbf{K}\mathbf{U}^{(k)})\|/\|\mathbf{D}_K\mathbf{F}\|$
0	$4.259 \cdot 10^{-2}$
1	$3.370 \cdot 10^{-3}$
2	$2.313 \cdot 10^{-4}$
3	$1.710 \cdot 10^{-5}$
4	$1.349 \cdot 10^{-6}$
5	$4.395 \cdot 10^{-7}$

TABLE 5.3
The convergence of the nested GCR method when the axially symmetric problem was solved in three dimensions.

k	$\|\mathbf{D}_K(\mathbf{F} - \mathbf{K}\mathbf{U}^{(k)})\|/\|\mathbf{D}_K\mathbf{F}\|$
0	$1.840 \cdot 10^{-2}$
1	$1.624 \cdot 10^{-4}$
2	$1.219 \cdot 10^{-6}$
3	$1.896 \cdot 10^{-8}$

with the total number of nodes $N = 106802$ (leading to 1068020 real unknowns). The linear system corresponding to the frequency $f = 200$ was solved using $m = 10$, $\delta_{\text{inner}} = 10^{-2}$, and $\epsilon_{\text{inner}} = 10^{-3}$. The results recorded in Table 5.3 show that the outer GCR iteration converges rapidly also in this case.

5.3. Mesh sensitivity. We finally demonstrate an experimental observation that our preconditioner leads to solution algorithms with convergence independent of mesh size. We now consider a straight circular tube having the radius $R = 1.0 \cdot 10^{-3}$ and the length $L = 5.0 \cdot 10^{-3}$. We again exploit axial symmetry, so that the computational domain can be taken to be $\Omega = \{\mathbf{x} = (x_1, x_2) \mid \mathbf{x} \in (0, R) \times (0, L)\}$. In addition, we assume that the medium is air at standard conditions and consider the case of the frequency $f = 10^4$.

The boundary conditions for this example are similar to those used in our first example. On the edge $x_2 = 0$, we prescribe the velocity boundary conditions $V_1 = 0$ and $V_2(x_1) = J_0(\alpha x_1)$, where the maximum value of the Bessel function is scaled to the value $1.0 \cdot 10^{-3}$ and the scaling factor α is defined so that $x_1 = \alpha R$ is the first root of $J_0(x_1)$. In addition, we assume that the fluctuation of the temperature vanishes on this edge. On the edge $x_1 = 0$, the boundary conditions are deduced on the basis of symmetry, while on the remaining part of the boundary all of the components of the velocity vector as well as the disturbance of the temperature are prescribed to vanish.

We solved this problem using four different finite element meshes, for which the numbers of nodes were $N = 1078,\ 2108,\ 4171$, and 8280. The coarsest discretization of the tube into bilinear elements is shown in Figure 5.11. The other three meshes were made so that the fraction of the nodes which are located in the boundary layer to the total number of nodes is approximately the same for all of the meshes. We solved the linear systems using the nested GCR algorithm with the fixed parameters $m = 10$, $\delta_{\text{inner}} = 10^{-2}$, and $\epsilon_{\text{inner}} = 10^{-2}$. The residual after each inner GCR

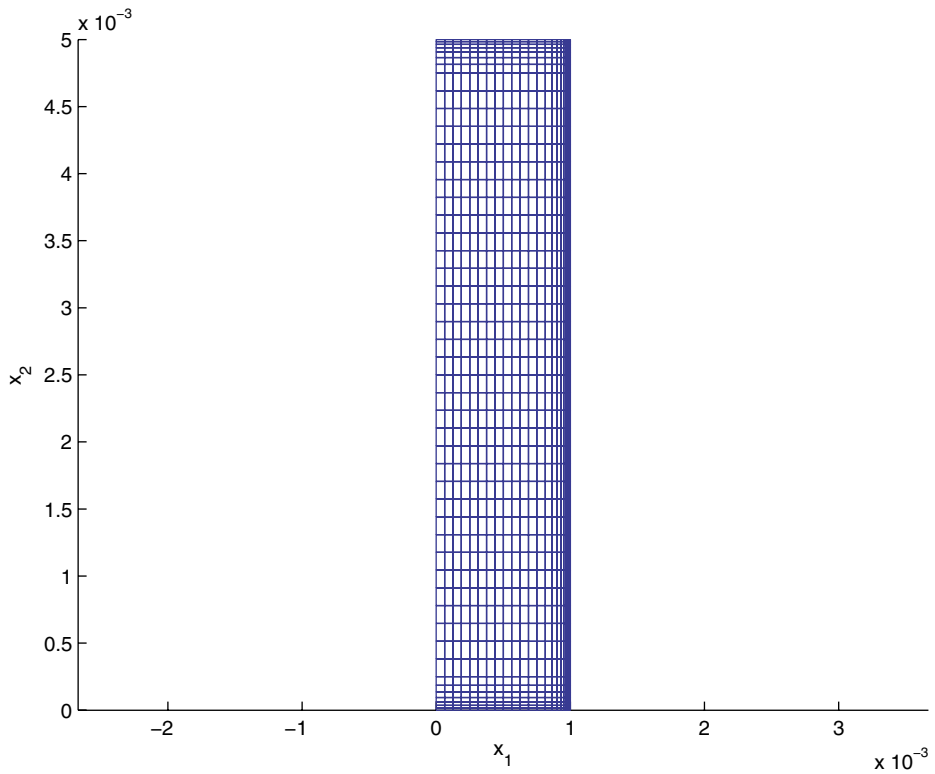


FIG. 5.11. *The coarsest discretization for the mesh sensitivity experiment.*

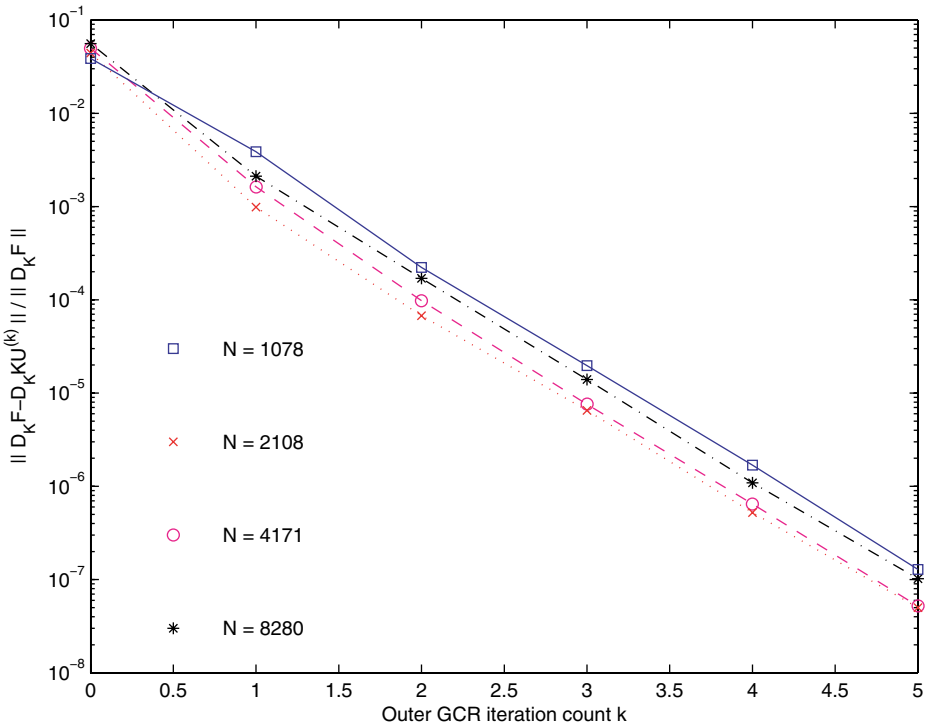


FIG. 5.12. *The residual after each inner GCR iteration when the discretization is refined.*

iteration is displayed in Figure 5.12. It is seen that the performance of the outer GCR iteration does not depend on the discretization; the same outer iteration counts producing residuals of approximately same sizes in all cases. Taking full advantage of this behavior would nevertheless require that this same optimality property would be shared by inner iterative methods. Such inner iteration methods are still to be found.

6. Concluding remarks. The results of this paper show that the performance of Schur complement preconditioners can depend drastically on the boundary conditions of the preconditioner. Here we have derived suitable boundary conditions by relying on a few key principles. The most important of these are the variational characterization of how the unknowns associated with the Schur complement depend on the boundary values of the velocity and the Helmholtz–Hodge decomposition of the velocity solution.

The results of the numerical experiments indicate that the application of the preconditioner we propose leads to robust solution algorithms with convergence independent of discretization mesh size. The construction of an optimal solution method for the dissipative acoustic equations has thus been shown to reduce to developing fast solvers for the subproblems of the preconditioning system. The central issue is the fast solution of the associated Schur complement problem characterized by the coupled system of the heat equation and a wavelike equation. While optimal solvers for this subproblem are still to be found, we have shown that a useful solution algorithm can be obtained when this problem is solved by applying a Krylov method in combination with the incomplete factorization preconditioners.

Acknowledgments. The author thanks Dr. Peter Råback for providing some of the finite element meshes used in the numerical experiments. The author thanks the anonymous referees for their useful comments.

REFERENCES

[1] W. M. Beltman, *Viscothermal wave propagation including acousto-elastic interaction, Part* I: *Theory*, J. Sound Vibration, 227 (1999), pp. 555–586.

[2] P. M. Morse and K. U. Ingard, *Theoretical Acoustics*, Princeton University Press, Princeton, NJ, 1986.

[3] G. H. Golub and H. A. van der Vorst, *Closer to the solution: Iterative linear solvers*, in The State of the Art in Numerical Analysis, I. S. Duff and G. A. Watson, eds., Clarendon Press, Oxford, 1997, pp. 63–92.

[4] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, 1994.

[5] J. H. Bramble and J. E. Pasciak, *Iterative techniques for time dependent Stokes problems*, Comput. Math. Appl., 33 (1997), pp. 13–30.

[6] H. C. Elman, *Preconditioners for saddle point problems arising in computational fluid dynamics*, Appl. Numer. Math., 43 (2002), pp. 75–89.

[7] D. Loghin and A. J. Wathen, *Schur complement preconditioning for elliptic systems of partial differential equations*, Numer. Linear Algebra Appl., 10 (2003), pp. 423–443.

[8] M. Malinen, M. Lyly, P. Råback, A. Kärkkäinen, and L. Kärkkäinen, *A finite element method for the modeling of thermo-viscous effects in acoustics*, in Proceedings of the 4th European Congress on Computational Methods in Applied Sciences and Engineering, P. Neittaanmäki, T. Rossi, K. Majava, O. Pironneau, eds., University of Jyväskylä, 2004.

[9] D. Braess, *Finite Elements. Theory, Fast Solvers, and Applications in Solid Mechanics*, Cambridge University Press, London, 1997.

[10] J. L. Guermond and J. Shen, *A new class of truly consistent splitting schemes for incompressible flows*, J. Comput. Phys., 192 (2003), pp. 262–276.

[11] H. A. van der Vorst and C. Vuik, *GMRESR: A family of nested GMRES methods*, Numer. Linear Algebra Appl., 1 (1994), pp. 369–386.

[12] *Elmer finite element software homepage*, http://www.csc.fi/elmer/.

SIAM J. SCI. COMPUT.
Vol. 29, No. 4, pp. 1593–1612
© 2007 Society for Industrial and Applied Mathematics

ADDITIVE AND MULTIPLICATIVE TWO-LEVEL SPECTRAL PRECONDITIONING FOR GENERAL LINEAR SYSTEMS*

B. CARPENTIERI†, L. GIRAUD‡, AND S. GRATTON§

Abstract. In this paper we introduce new preconditioning techniques for the solution of general symmetric and unsymmetric linear systems $Ax = b$. These approaches borrow some ideas of the multigrid philosophy designed for the solution of linear systems arising from the discretization of elliptic partial differential equations. We attempt to improve the convergence rate of a prescribed preconditioner M_1. In a two-grid framework, this preconditioner is viewed as a smoother and the coarse space is spanned by the eigenvectors associated with the smallest eigenvalues of M_1A. We derive both additive and multiplicative variants of the resulting iterated two-level preconditioners for unsymmetric linear systems that can also be adapted for Hermitian positive definite problems. We show that these two-level preconditioners shift the smallest eigenvalues to one and tend to better cluster around one those eigenvalues that M_1 already succeeded in moving into the neighborhood of one. We illustrate the behavior of our method through extensive numerical experiments on a set of general linear systems. Finally, we show the effectiveness of these approaches on two challenging real applications; the first comes from a nonoverlapping domain decomposition method in semiconductor device modeling, the second from industrial electromagnetism applications.

Key words. iterative methods, Krylov methods, adaptive preconditioning, additive and multiplicative two-grid cycles, spectral preconditioner, deflation techniques, electromagnetic scattering applications, domain decomposition

AMS subject classifications. 65F10, 65F50, 65F15, 65R20, 65N38

DOI. 10.1137/060654906

1. Introduction.

The large linear systems involved in many intensive numerical simulations are solved using preconditioned Krylov solvers. It is well known that the convergence of these linear solvers often depends to a large extent on the eigenvalue distribution of the preconditioned matrix. In many cases, it is observed that "removing" the extreme eigenvalues can greatly improve the convergence. Several techniques have been proposed in the past few years that attempt to tackle this problem. The proposed approaches can be split into two main families, depending on whether the scheme enlarges the generated Krylov space (see, for instance, [22, 25, 30, 33]) or adaptively updates the preconditioner [1, 6, 11, 20, 27, 29]. Many of these preconditioning techniques based on approximate eigenpairs have been proposed in the past few years. The underlying driving idea behind these approaches is to capture in a low-dimensional space the modes that do not quickly converge with a first-level preconditioner. In order to be efficient and keep the dimension of the low-dimensional space reasonably small, these techniques are generally used in combination with a first-level preconditioner that does a good job of clustering most eigenvalues close to one with relatively few outliers near the origin [10, 26, 36]. These spectral preconditioners can also be split into two main families, depending on their effect on the spectrum. They are referred to as deflation preconditioners [12, 14, 28] if they at-

*Received by the editors March 23, 2006; accepted for publication (in revised form) January 24, 2007; published electronically July 18, 2007. The major part of this work was carried out while the authors were working at CERFACS.

http://www.siam.org/journals/sisc/29-4/65490.html

†Karl-Franzens-University Graz, Institute of Mathematics and Scientific Computing, Heinrichstraße 36, A-8010 Graz, Austria (bruno.carpentieri@uni-graz.at).

‡ENSEEIHT-IRIT, 2 Rue C. Camichel, 31 071 Toulouse Cedex, France (giraud@enseeiht.fr).

§CERFACS, 42 Avenue G. Coriolis, 31057 Toulouse Cedex, France (gratton@cerfacs.fr).

tempt to move a subset of eigenvalues to a positive quantity σ; they are referred to as coarse grid preconditioners [6, 14] if they only attempt to shift the subset close to σ. The name of those latter techniques comes from domain decomposition and was first introduced in [5]. For this reason $\sigma = 1$ is often considered in practice.

On the other hand, for the solution of a linear system $Ax = b$ arising from the discretization of partial differential equations (PDE), the multigrid methods are among the fastest techniques. The core of the multigrid algorithms is a two-grid procedure that is applied recursively. A classical two-grid cycle can be briefly described as follows. On the fine grid a few iterations of a smoother are applied that attempt to reduce the high frequencies of the error (i.e., the components of the error in the space spanned by the vectors associated with the largest eigenvalues of A). The residual is then projected on the coarse grid where the low frequencies (i.e., the components associated with the smallest eigenvalues) can be captured and the coarse error equation is solved. The error on the coarse space is prolongated back to the fine grid to update the approximation computed by the presmoothing phase and a few more steps of the smoother are applied. Finally, if the new iterate is not accurate enough, the two-grid cycle is applied iteratively. In classical multigrid, the coarse space is not defined explicitly through the knowledge of the eigencomponents but by the selection of a space that is expected to capture them. The scheme presented above is a multiplicative algorithm [19], but additive variants [2, 13, 34] also exist.

In this work, we apply some underlying ideas from the multigrid approach to design preconditioners for the solution of general symmetric and unsymmetric linear systems (not necessarily coming from PDE applications). We attempt to improve a prescribed preconditioner M_1, which is used to define the smoother involved in the two-grid scheme. In many situations such a preconditioner (that is application-dependent) is able to cluster most of the eigenvalues close to one but still leaves a few close to the origin; this is a fairly common situation for a wide range of problems and preconditioners. In that framework we define the coarse space by the span of the eigenvectors associated with the smallest eigenvalues of M_1A that are computed explicitly (i.e., the components of the error that are not efficiently damped by the smoother). We denote by V the matrix having the eigenvectors of M_1A as its columns. In that context, the prolongation operator is $P = V$, the restriction operator is denoted by R, and the matrix involved in the coarse grid error problem is defined by a Petrov–Galerkin formula $A_c = RAP$. The new approaches handle similar ingredients to those exploited in [6] but use them in a completely different way following another philosophy. This results in particular in a different spectral transformation for the preconditioned matrices. We mention that our multiplicative approach is a generalization of the generalized global basis (GGB) method [36]. For a particular choice of M_1, that is, an algebraic multigrid V-cycle, and a particular choice of V, the eigenvectors associated with the largest eigenvalues in modulus, our multiplicative method coincides with GGB if only one iteration of the smoother is performed. In section 2, we describe the proposed preconditioners and establish their effect on the spectrum of the preconditioned matrices. In section 3, we illustrate the numerical effects related to each parameter defining the new preconditioners on a set of unsymmetric and symmetric positive definite (SPD) linear systems from the Matrix Market [4]. We demonstrate their computational efficiency on very challenging linear systems that arise in two-dimensional unstructured mixed finite element calculation for semiconductor device modeling as well as for the solution of dense linear systems arising from boundary element methods in industrial electromagnetic applications.

2. Two-level spectral preconditioning. We consider the solution of the linear system

$$Ax = b, \tag{1}$$

where A is an $n \times n$ nonsingular matrix, and x and b are vectors of size n. The linear system is solved using a preconditioned Krylov solver, and we denote by M_1 the left preconditioner, meaning that we solve

$$M_1 Ax = M_1 b. \tag{2}$$

Let $\{\lambda_1, \ldots, \lambda_n\}$ be the set of eigenvalues of $M_1 A$ in any order where the multiple eigenvalues are repeated. Let the columns of V_k be the basis of a right invariant subspace of $M_1 A$ of dimension k. Suppose without loss of generality that $M_1 A V_k = V_k J_k$, where the eigenvalues of J_k are $\{\lambda_1, \ldots, \lambda_k\}$. In this section we describe two procedures that attempt to improve the eigenvalue distribution of the preconditioned matrix. We formulate the algorithms in the framework of algebraic multilevel methods; the terminology and the design principles are inherited from classical two-grid algorithms. In the next two sections we present these two variants of the algorithm that are either multiplicative or additive.

2.1. Multiplicative two-grid spectral preconditioning. A generic two-level multigrid cycle is illustrated in Algorithm 1, where M_1 is used to define a weighted stationary method that implements the smoother. The algorithm takes as input a vector r that is the residual vector we want to precondition and returns as output the preconditioned residual vector z. After μ_1 smoothing steps we project, using the restriction $R = W^H$, the residual into the coarse subspace and solve the coarse space error equation involving $A_c = W^H A V_k$. Finally, we prolongate back the error using V_k in the original space and smooth again the new approximation. The preconditioner constructed using this scheme depends on A, M_1, μ_1, μ_2, ε, and $\omega > 0$ and could be denoted by $M_{Mul}(A, M_1, k, \omega, \mu_1, \mu_2)$. For the sake of simplicity of exposure, when no confusion is possible we will simply denote it by M_{Mul} or $M_{Mul}(A, M_1)$.

PROPOSITION 1. [illegible] $A_c = W^H A V_k$ [illegible] $z = M_{Mul}\, r$ [illegible] $iter = 1$ [illegible] M_{Mul} [illegible]

$$M_{Mul} = A^{-1} - (I - \omega M_1 A)^{\mu_2} (I - M_c A)(I - \omega M_1 A)^{\mu_1} A^{-1}, \tag{3}$$

[illegible] $M_c = V_k (W^H A V_k)^{-1} W^H$

This is a fairly standard result; for its proof we refer, for instance, to [8]. □

The following proposition describes the eigenvalue distribution of the preconditioned matrix $M_{Mul} A$. Similarly to multigrid, if we choose V_k to be the invariant space associated with the smallest eigenvalues (i.e., the closest to the origin), this result shows that the smallest eigenvalues are shifted to one and that those that were already close to one are clustered closer to this point. On the assumption that the initial preconditioner M_1 has done a good job of clustering most eigenvalues close to one with relatively few outliers near the origin, the effect of M_{Mul} is expected to be very beneficial.

ALGORITHM 1.

set $z^1 = 0$
for $\ell = 1, iter$ **do**
  %
  $s_1^0 = z^\ell$
  for $j = 1, \mu_1$ **do**
    $s_1^j = s_1^{j-1} + \omega M_1 (r - A s_1^{j-1})$
  end for
  %
  $z_2^\ell = z_1^\ell + V_k A_c^{-1} W^H (r - A z_1^\ell)$
  %
  $s_2^0 = z_2^\ell$
  for $j = 1, \mu_2$ **do**
    $s_2^j = s_2^{j-1} + \omega M_1 (r - A s_2^{j-1})$
  end for
end for
$z = z^{iter}$

PROPOSITION 2. M_{Mul} 1 $M_{Mul}A$

$$\begin{cases} \eta_i = 1 & i < k, \\ \eta_i = 1 - (1 - \omega\lambda_i)^{\mu_1+\mu_2} & i \geq k. \end{cases}$$

We will show that the preconditioned matrix is similar to a matrix whose eigenvalues are those indicated. Let $V_k^\perp$ be an orthogonal complement of V_k. We have

$$M_1 A \left[V_k V_k^\perp\right] = \left[V_k V_k^\perp\right] \begin{pmatrix} J_k & E \\ 0 & F \end{pmatrix},$$

where the eigenvalues of J_k are $\{\lambda_1, \ldots, \lambda_k\}$ and those of F are $\{\lambda_{k+1}, \ldots, \lambda_n\}$.

Thus $M_1 A V_k = V_k J_k$ and $M_c A V_k = V_k$. In addition, it is easy to show by induction on j that $(I - \omega M_1 A)^j V_k = V_k (I - \omega J_k)^j$ and $(I - \omega M_1 A)^j V_k^\perp = V_k^\perp (I - \omega F)^j + V_k P_j$, where $P_j \in \mathbb{R}^{k \times (n-k)}$. We also have $(I - M_c A) V_k^\perp = V_k^\perp - V_k P_c$ with $P_c \in \mathbb{R}^{k \times (n-k)}$. Then the following relations hold:

$$M_{Mul} A \left[V_k V_k^\perp\right] = \left[V_k V_k^\perp\right] \begin{pmatrix} I & * \\ 0 & I - (I - \omega F)^{\mu_1+\mu_2} \end{pmatrix}, \tag{4}$$

where the asterisk denotes a nonzero block whose actual expression is unimportant. The equality (4) concludes the proof. □

1. It should be noted that the spectrum of the preconditioned matrix does not depend on the selection of W^H, which would play the role of the restriction operator in the multiplicative two-grid method.

2.2. Additive two-grid spectral preconditioning. In the additive algorithm, the coarse grid correction and the smoothing operation are decoupled. Each process generates an approximation of the preconditioned residual vector in complementary subspaces. The coarse grid correction computes only components in the space spanned by the eigenvectors associated with the few selected small eigenvalues, while at the end of the smoothing step the preconditioned residual is filtered so that only

components in the complementary subspace are retained. These two contributions are summed together for the solution update. A simple additive two-level multigrid cycle is illustrated in Algorithm 2. In this algorithm we follow [13] and select the procedure advocated in [34]: we define the filtering operators using the grid transfer operators as $(I - V_k W^H)$. This operator is supposed to remove all the components in the V_k directions. A natural choice is to select W^H so that $(I - V_k W^H)V_k = 0$ (i.e., $W^H V_k = I$).

ALGORITHM 2.

1: set $z^1 = 0$
2: **for** $\ell = 1, iter$ **do**
3: %
4: $s^\ell = r - Az^\ell$
5: %
6: % 1-
%
7: $e_1^{\ell,0} = 0$
8: **for** $j = 1, \mu_1 + \mu_2$ **do**
9: $e_1^{\ell,j} = e_1^{\ell,j-1} + \omega M_1(s^\ell - Ae_1^{\ell,j-1})$
10: **end for**
11: %
12: $c_1^\ell = (I - V_k W^H)e_1^{\ell,\mu_1+\mu_2}$
13: % 2-
%
14: $c_2^\ell = V_k A_c^{-1} W^H s^\ell$
15: %
%
16: $z^{\ell+1} = z^\ell + c_1^\ell + c_2^\ell$
17: **end for**
18: $z = z^{iter}$

PROPOSITION 3. W $A_c = W^H A V_k$ $(I - V_k W^H)V_k = 0$ 2 $z = M_{Add}\, r$ $iter = 1$ M_{Add}

$$M_{Add} = V_k A_c^{-1} W^H + (I - V_k W^H)(I - (I - \omega M_1 A)^{\mu_1+\mu_2})A^{-1}. \tag{5}$$

The proof follows arguments similar to those for the proof of Proposition 1. ∎

PROPOSITION 4. M_{Add} 3 $M_{Add}A$

$$\begin{cases} \eta_i = 1 & i \le k, \\ \eta_i = 1 - (1 - \omega\lambda_i)^{\mu_1+\mu_2} & i > k. \end{cases}$$

Using arguments similar to those for the proof of Proposition 2, it can be shown that

$$M_{Add}A\left[V_k V_k^\perp\right] = \left[V_k V_k^\perp\right]\begin{pmatrix} I & * \\ 0 & I - (I - \omega F)^{\mu_1+\mu_2} \end{pmatrix}, \tag{6}$$

where the asterisk denotes a nonzero block whose actual expression is unimportant. The equality (6) concludes the proof. ∎

2.3. The Hermitian positive definite linear systems. For Hermitian positive definite (HPD) linear systems a desirable property for M_1 is to be Hermitian positive definite; we denote by $M_1 = L_1 L_1^H$ its Cholesky decomposition. We note that in many situations the preconditioner M_1 is given in this factorized form as in the incomplete factorization [24], approximate inverse (AINV) [3], or factorized sparse approximate inverse (FSAI) [21]. In the rest of this section we review situations where M_{Mul} and M_{Add} are HPD.

PROPOSITION 5. [illegible] M_{Mul} [illegible] $W = V_k$ [illegible] $\mu_1 + \mu_2$ [illegible]

[illegible] We do not give the details of the calculation, but using simple matrix manipulations it can be shown that

$$M_{Mul}(A, M_1) = L_1 M_{Mul}(L_1^H A L_1, I) L_1^H. \tag{7}$$

We first show that $M_{Mul}(L_1^H A L_1, I)$ is Hermitian. Because $L_1^H A L_1$ is HPD we have $L_1^H A L_1 = \tilde{V} D \tilde{V}^H$, where $D = \mathrm{diag}\,(\lambda_1, \ldots, \lambda_n)$ and $\tilde{V} = [\tilde{V}_k \tilde{V}_k^{\perp}]$. If we denote $D_k = \mathrm{diag}\,(\lambda_1, \ldots, \lambda_k)$ and $D_{\bar{k}} = \mathrm{diag}\,(\lambda_{k+1}, \ldots, \lambda_n)$, we have

$$M_{Mul}(L_1^H A L_1, I) = \tilde{V} \begin{pmatrix} D_k^{-1} & 0 \\ 0 & D_{\bar{k}}^{-1}(I - (I - \omega D_{\bar{k}})^{\mu_1+\mu_2}) \end{pmatrix} \tilde{V}^H = \tilde{V} \hat{D} \tilde{V}^H. \tag{8}$$

Because $\mu_1 + \mu_2$ is odd, all diagonal entries of $\hat{D}$ are positive (the function $f(x) = 1 - (1 - \omega x)^k$ is positive $\forall x \in (0, \infty)$ for k odd; see also Figure 1(b)). Furthermore, (7) leads to $M_{Mul} = (L_1 \tilde{V}) \hat{D} (L_1 \tilde{V})^H$. The Sylvester law of inertia implies that all the eigenvalues of M_{Mul} are positive. ∎

PROPOSITION 6. [illegible] $\mu_1 + \mu_2$ [illegible] M_{Mul} [illegible] $W = V_k$ [illegible] $\omega < 2\lambda_{max}^{-1}(M_1 A)$

[illegible] The proof is similar to that of Proposition 5. ∎

In the HPD situation we have the following properties.

PROPOSITION 7. [illegible] $M_1 = L_1 L_1^H$ [illegible] (1) [illegible] $M_{Mul}(A, M_1)$ [illegible] $L_1^H A L_1 y = L_1^H b$ [illegible] $M_{Mul}(L_1^H A L_1, I)$ [illegible] $x = L_1 y$ [illegible] x_k [illegible] y_k [illegible] M_{Mul} [illegible] 5 [illegible] 6 [illegible] $x_0 = L_1 y_0$ [illegible] k $x_k = L_1 y_k$

[illegible] Because the preconditioners are HPD we consider their Cholesky decompositions, which are $M_{Mul}(A, M_1) = L_M L_M^H$ and $M_{Mul}(L_1^H A L_1, I) = \tilde{L}_M \tilde{L}_M^H$. From (7), we have $L_M L_M^T = L_1 \tilde{L}_M \tilde{L}_M^H L_1^H$, which shows that $L_M = L_1 \tilde{L}_M$ (uniqueness of the Cholesky decomposition). We now use the result on CG iterates from [17, 32] that states that solving $By = g$ with CG preconditioned by LL^T generates iterates y_k that can also be generated by unpreconditioned CG on $L^T B L \tilde{y} = L^T g$, provided that the initial guesses are such that $y_0 = L\tilde{y}_0$. We apply this result in the following sequence: CG on A preconditioned by $L_M L_M^T$ generates the same iterates as unpreconditioned CG on $L_M^T A L_M = \tilde{L}_M^H L_1^H A L_1 \tilde{L}_M$; these iterates are the same as those generated by CG on $L_1^H A L_1$ preconditioned by $\tilde{L}_M \tilde{L}_M^H$. ∎

PROPOSITION 8. [illegible] $M_1 = I$ [illegible] $W = V_k$ [illegible] V_k [illegible] $M_{Mul} = M_{Add}$

[illegible] Because A is HPD, A is diagonalizable in an orthonormal basis of eigenvectors. We then have that the vectors $W = V_k$ comply with the assumption $W^H V_k = I$ of Proposition 3.

With the assumptions that V_k is a set of eigenvectors, we have $M_{Mul}AV_k = V_k = M_{Add}AV_k$ and $M_{Mul}AV_k^{\perp} = M_{Add}AV_k^{\perp}$. □

REMARK 2. A situation where the assumptions of Proposition 8 hold is, for instance, when either $M_{Mul}(L_1^H AL_1, I)$ or $M_{Add}(L_1^H AL_1, I)$ is considered (i.e., the two-grid preconditioners are applied to $L_1^H AL_1 y = L_1^H b$). The additive preconditioner is in general not HPD even when A and M_1 are. One way to still use this preconditioner for an M_1 given in a symmetric factorized form $M_1 = L_1 L_1^H$ is to consider $M_{Add}(L_1^H AL_1, I)$ applied to $L_1^H AL_1 y = L_1^H b$. In that case we have $W = V_k$ and Proposition 8 shows that this preconditioner is the same as $M_{Mul}(L_1^H AL_1, I)$. Furthermore, Proposition 7 shows that CG then generates the same approximation with M_{Add} and M_{Mul}. This shows that for HPD linear systems all of these preconditioners are pretty much the same, and the choice of which to use can be made using floating-point arithmetic complexity as described below.

2.4. Some computational considerations. The spectral properties of M_{Mul} and M_{Add} have been derived from Algorithms 1 and 2, where only one outer iteration is considered. With calculations similar to those used so far, it can be shown that implementing "*iter*" outer steps leads to a spectral transformation for the two preconditioners that is defined by the following proposition.

PROPOSITION 9. [illegible] M_{Mul} [illegible] 1 [illegible] $iter \geq 1$ [illegible] $M_{Mul}A$ [illegible]

$$\begin{cases} \eta_i = 1 & \text{[illegible]} \quad i \leq k, \\ \eta_i = 1 - (1 - \omega\lambda_i)^{iter \times (\mu_1 + \mu_2)} & \text{[illegible]} \quad i > k. \end{cases}$$

Similar proposition holds with M_{Add} defined by Algorithm 2.

From a computational point of view, applying M_{Mul} or M_{Add} does not require the same computational effort. In terms of application of M_1 both need $iter \times (\mu_1 + \mu_2)$ matrix-vector products. The number of matrix-vector products involving A is one less for M_{Add} than for M_{Mul} per cycle. For M_{Add} the number of products by A is

$$(\mu_1 + \mu_2 - 1) + (iter - 1) \times (\mu_1 + \mu_2). \tag{9}$$

Under the assumption that the convergence of the Krylov solver is mainly governed by the eigenvalue distributions (i.e., $iter \times (\mu_1 + \mu_2)$ given), Proposition 9 and the number of matrix-vector products given by (9) indicate that for M_{Mul} it is computationally more efficient to use only one two-grid cycle with $(iter \times (\mu_1 + \mu_2))$ smoothing steps rather than *iter* two-grid cycles with $(\mu_1 + \mu_2)$ smoothing steps per cycle. For M_{Add}, the two variants have the same computational cost.

REMARK 3. In [6], the initial linear system is never considered and the idea is to perform an appropriate low rank correction of the preconditioned matrix to shift part of its spectrum. Here, similarly to the two-grid schemes for PDE solution, we solve exactly the error equation of the unpreconditioned system in the invariant subspace associated with the smallest eigenvalues in modulus. In that respect, even though these approaches exploit the same ingredients, the underlying philosophies are different.

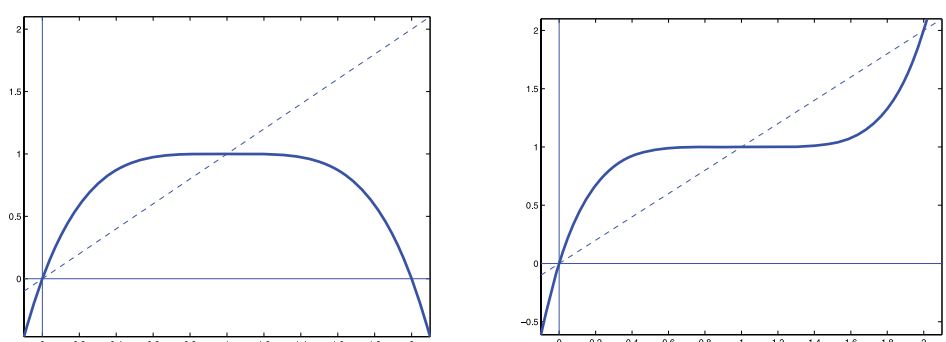
3. Numerical experiments. In this section we illustrate the numerical behavior of the two variants of the spectral algorithm described in the earlier section. In section 3.1 we investigate only the numerical behavior of the proposed schemes and do not address their computational relevance. Their computational efficiency strongly

depends on the relative costs of the smoother, the matrix-vector, and eigenmode calculation, which are application-dependent. Furthermore, the extra cost associated with the setup to compute the eigenvectors might be significant and might not be amortized by the savings in time for the linear system solution. Consequently these computational aspects are investigated on two challenging applications where the preconditioners have been implemented in the real simulation codes and run on front-end computers.

3.1. General test problems. We consider both unsymmetric and SPD systems. The numerical experiments are performed with MATLAB version 7.0. For the sake of simplicity we consider the $ILU(t)$ preconditioner [31] from MATLAB for M_1 in the unsymmetric case and the incomplete Cholesky factorization $IC(t)$ for SPD systems [24]. We report on results using the GMRES and the Bi-CGStab [35] solvers for unsymmetric problems, and the CG method for SPD systems. The experiments are carried out using left preconditioning for the unsymmetric problems. The initial guess for the solution is the zero vector. Because we compare different preconditioners, we choose as a stopping criterion the reduction of the normalized unpreconditioned residual by 10^{-6}, so that the stopping criterion is independent of the preconditioner. We explicitly compute the true unpreconditioned residual at each iteration. For the experiments with M_{Add} we use $W = QR^{-1}$, where $V_k = QR$ to ensure that $W^H V_k = I$. For the experiments with M_{Mul} we consider $W = V_k$. We mention that with these choices of W we have never encountered rank deficiency for A_c. However, it is reasonable to check the nonsingularity of A_c in practice to ensure that the preconditioners are defined; notice that in the HPD situation A_c cannot be singular. The eigenvectors V_k are computed in a preprocessing phase using the MATLAB function `eigs`.

In a first stage we consider both SPD and unsymmetric test matrices from the Matrix Market. In this section, we only report a few results representative of the general trends. For a complete description of the numerical experiments we refer the reader to the appendix of [8], where results on intensive experiments are reported.

For SPD matrices, we display the eigenvalue transformation operated by the preconditioners. These are the curves $f(x) = 1 - (1 - \omega x)^k \ \forall x > 0$ as the eigenvalues of $M_1 A$ are real positive. Figure 1 illustrates that the preconditioned matrices remain SPD for $(\mu_1 + \mu_2)$ odd, while ω should be set such that $\omega < 2\lambda_{max}^{-1}(M_1 A)$ for $(\mu_1 + \mu_2)$ even.



(a) $(\mu_1 + \mu_2)$ even (=4) (b) $(\mu_1 + \mu_2)$ odd (=5)

FIG. 1. *Shape of the polynomial that governs the real eigenvalues distribution of the two-grid spectral preconditioner for* $\omega = 1$.

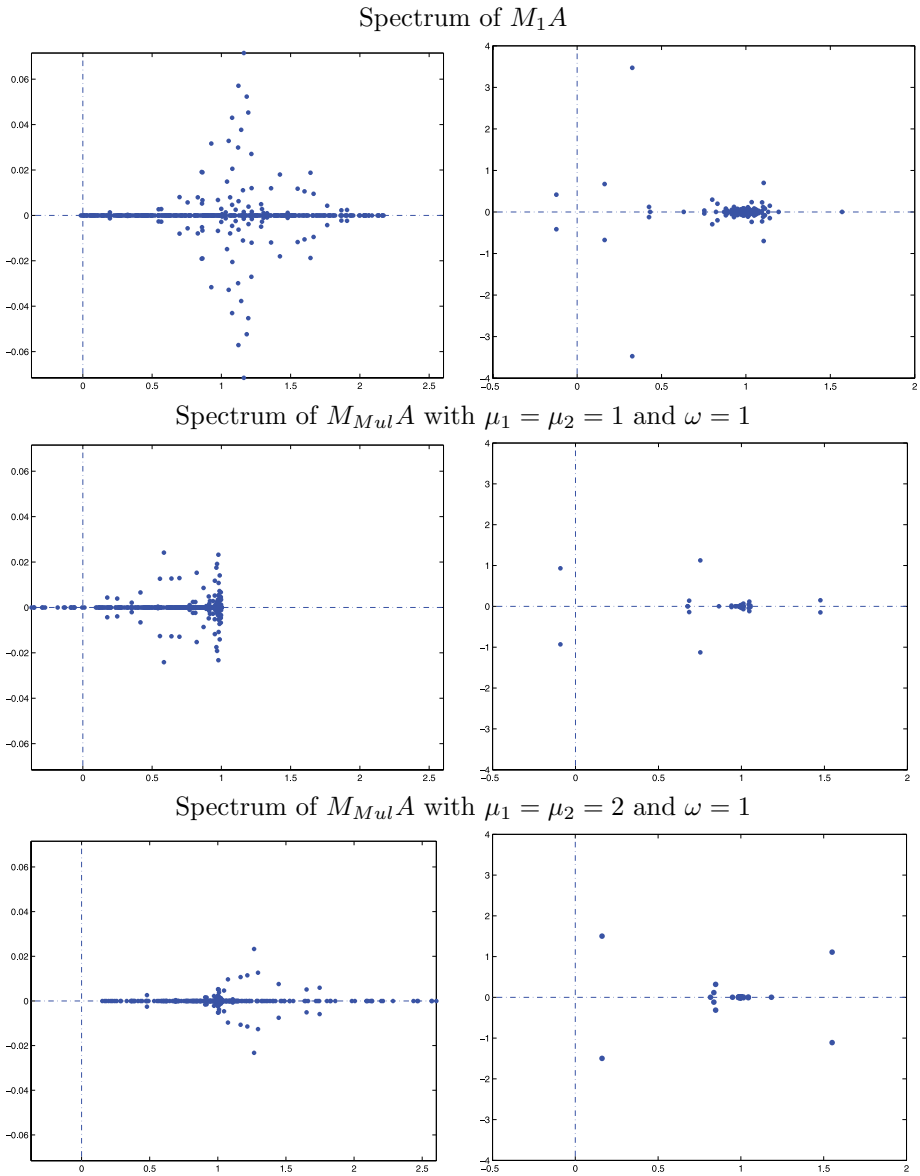


FIG. 2. *Spectrum of the preconditioned matrices (BFW*398*A, left; GRE*1107*, right).*

In Figure 2 we depict the spectrum of preconditioned matrices using only the first-level preconditioner and the two-grid preconditioner with different choices for μ_1 and μ_2. In this figure, it can be seen that the spectral preconditioners do a good job in clustering close to one most of the eigenvalues. More precisely for $\mu_1 + \mu_2 > 1$ all the eigenvalues lying in the open disk of radius one centered in $(1, 0)$ are contracted toward $(1, 0)$; the ones out of this disk are moved away from $(1, 0)$. When $\mu_1 + \mu_2$ is odd, those that are real and larger than 2 become negative, whereas for $\mu_1 + \mu_2$ even they remain positive. This is illustrated by the example of the matrix BFW398A.

In the following subsections we consider the numerical behavior of the proposed

TABLE 1
Number of iterations with M_{Mul} and M_{Add}.

SAYLR1: $t = 3 \cdot 10^{-1}$; $\mu_1 = 2$, $\mu_2 = 1$; $\omega = 1.0$													
		Dimension of the coarse space											
		M_1	0	1	2	3	4	5	6	7	8	9	10
Bi-CGStab	M_{Add}	150	77	64	54	43	27	21	19	18	16	14	14
	M_{Mul}	150	77	63	49	40	28	22	19	17	19	13	14
GMRES(50)	M_{Add}	†	†	388	142	56	47	43	36	36	32	29	26
	M_{Mul}	†	†	441	141	56	47	40	39	36	33	29	26
GMRES(∞)	M_{Add}	116	72	63	56	49	43	36	33	33	30	26	23
	M_{Mul}	116	72	63	56	49	42	36	35	33	30	26	23
BFW398a: $t = 5 \cdot 10^{-1}$; $\mu_1 = 2$, $\mu_2 = 1$; $\omega = 1.0$													
		Dimension of the coarse space											
		M_1	0	1	2	3	4	5	6	7	8	9	10
Bi-CGStab	M_{Add}	112	73	48	44	33	28	23	23	16	14	14	14
	M_{Mul}	112	73	50	43	33	29	23	23	15	14	14	14
GMRES(50)	M_{Add}	286	106	80	52	45	40	34	30	27	25	23	22
	M_{Mul}	286	106	79	52	45	40	34	30	27	25	23	22
GMRES(∞)	M_{Add}	106	66	57	51	45	40	34	30	27	25	23	22
	M_{Mul}	106	66	57	51	45	40	34	30	27	25	23	22

preconditioners. The efficiency in terms of computational cost is very problem-dependent and is addressed in section 3.2, where two real-world applications are presented.

3.1.1. Numerical behavior of M_{Mul} versus M_{add}. The two preconditioners M_{Add} and M_{Mul} give rise to preconditioned systems that have the same eigenvalue distribution but possibly different eigenspaces. In Table 1 we report on the number of iterations of the Krylov solvers for different matrices and different values of μ_1 and μ_2 when the size of the coarse space is varied. The symbol "M_1" for the dimension of the coarse space means that no coarse space correction is needed and only one smoothing is applied; in that case the preconditioner reduces to the standard incomplete factorization. The symbol "†" indicates that the convergence was not observed in less than 1000 iterations. On all the numerical experiments we have performed, M_{Add} and M_{Mul} exhibit a very similar numerical behavior, as illustrated in Table 1. Small differences appear with Bi-CGStab, diminish with restarted GMRES, and vanish with full GMRES. However, from a numerical point of view none of the two preconditioners appears superior to the other. When the size of the coarse space is increased it can be seen that the two-level preconditioners monotonically improve the convergence of full GMRES. This is no longer strictly true for restarted GMRES and Bi-CGStab, but the trend is still to observe fewer iterations when the size of the coarse space is increased. Generally the larger the coarse space, the faster the convergence. One can see on the SAYLR1 matrix that even without coarse space correction few steps of smoothers can significantly improve the convergence of the Krylov solvers. On this latter example, the combination of the smoothing steps and the coarse grid correction is the only way to ensure the convergence of GMRES(50).

Because M_{Add} and M_{Mul} have a similar behavior and because M_{Mul} is naturally defined for SPD problems, we consider only M_{Mul} for the numerical experiments reported in the next three sections.

3.1.2. Effect of the number of smoothing steps. In Table 2 we display the number of iterations when the number of smoothing steps is varied from 1 to 3. It can

TABLE 2
Number of iterations with M_{Mul} when the number of smoothing steps is varied.

S1RMQ4M1: $t = 1 \cdot 10^{-1}$; $\omega = \lambda_{\max}^{-1}(M_1 A)$													
			Dimension of the coarse space										
		M_1	0	1	2	3	4	5	6	7	8	9	10
CG	$\mu_1 + \mu_2 = 1$	204	204	134	114	104	90	89	89	83	82	78	77
	$\mu_1 + \mu_2 = 2$	204	150	97	83	76	65	64	64	60	60	56	56
	$\mu_1 + \mu_2 = 3$	204	123	80	68	62	53	53	53	49	49	46	46
BWM2000: $t = 3 \cdot 10^{-1}$; $\omega = 1.0$													
			Dimension of the coarse space										
		M_1	0	1	2	3	4	5	6	7	8	9	10
GMRES(20)	$\mu_1 + \mu_2 = 1$	66	66	56	36	19	16	12	11	11	11	11	10
	$\mu_1 + \mu_2 = 2$	66	18	16	13	11	9	8	7	7	7	7	7
	$\mu_1 + \mu_2 = 3$	66	15	13	11	9	7	6	6	6	6	6	6
FS-541-4: $t = 8 \cdot 10^{-1}$; $\omega = 1.0$													
			Dimension of the coarse space										
		M_1	0	1	2	3	4	5	6	7	8	9	10
GMRES(100)	$\mu_1 + \mu_2 = 1$	126	126	125	126	124	125	125	99	99	99	99	99
	$\mu_1 + \mu_2 = 2$	126	†	†	†	†	†	†	†	†	†	†	†
	$\mu_1 + \mu_2 = 3$	126	87	86	86	84	84	84	72	72	72	72	72
GMRES(∞)	$\mu_1 + \mu_2 = 1$	106	106	103	106	102	103	103	99	99	99	99	99
	$\mu_1 + \mu_2 = 2$	106	168	163	163	163	163	163	167	167	175	167	167
	$\mu_1 + \mu_2 = 3$	106	78	77	77	77	77	77	71	71	71	70	69
GRE1107: $t = 1 \cdot 10^{-2}$; $\omega = 1.0$													
			Dimension of the coarse space										
		M_1	0	1	2	3	4	5	6	7	8	9	10
GMRES(40)	$\mu_1 + \mu_2 = 1$	†	†	80	40	37	36	32	31	27	26	24	23
	$\mu_1 + \mu_2 = 2$	†	34	31	28	25	24	22	21	19	18	17	16
	$\mu_1 + \mu_2 = 3$	†	28	26	24	23	21	20	20	18	18	16	14

be seen that increasing the number of smoother iterations improves the convergence for all the test examples but the matrix FS5414. For this latter matrix, the convergence using M_{Mul} is worse than using simply M_1. This poor numerical behavior is probably due to the fact that the preconditioned system $M_1 A$ has eigenvalues close to 2. The even number of smoothing iterations moves those eigenvalues near zero, which dramatically affects the numerical behavior of restarted GMRES. Full GMRES succeeds to converge but still using 2 steps of smoothing gives a larger number of iterations than just 1 step. When we move to 3 steps we reduce the number of iterations compared to 2 but also compared to 1 step of smoothing.

Even though no numerical experiments are reported to illustrate this phenomenon, we observed that only the sum $\mu_1 + \mu_2$ plays a role for the convergence of M_{Mul} wherever these smoothing steps are used only as presmoother, postsmoother, or split between the pre- and postsmoothing steps. Furthermore, we observe that the relative improvement of using more smoothing steps tends to be larger for a small dimension of the coarse space.

3.1.3. Effect of the relaxation parameter ω. In Table 3 we display the number of iterations required by CG, and by restarted and full GMRES on a set of matrices when the relaxation parameter ω is varied. For SPD matrices, when $\mu_1 + \mu_2$ is odd, no damping is required for the smoother to ensure that the preconditioner is SPD, as illustrated for the matrices 1138BUS and BCSSTK27. It can be seen that $\omega = 1.0$ leads to worse iteration counts than the incomplete factorization alone, i.e., the two space cycles slow down the convergence of CG. This might be due to the fact

TABLE 3
Number of iterations with M_{Add} when the relaxation parameter is varied.

1138BUS: $t = 4 \cdot 10^{-1}$; $\mu_1 + \mu_2 = 3$													
			Dimension of the coarse space										
	ω	M_1	0	1	2	3	4	5	6	7	8	9	10
CG	1.0	295	502	361	273	224	181	174	156	156	146	132	132
	$2\lambda_{\max}^{-1}(M_1A)$	295	221	161	120	97	81	73	69	69	62	58	58
	$\frac{3}{2}\lambda_{\max}^{-1}(M_1A)$	295	232	166	128	104	86	81	78	77	66	61	58
	$\lambda_{\max}^{-1}(M_1A)$	295	251	185	137	111	91	83	79	77	74	69	66
BCSSTK27: $t = 1 \cdot 10^{-1}$; $\mu_1 + \mu_2 = 2$													
			Dimension of the coarse space										
	ω	M_1	0	1	2	3	4	5	6	7	8	9	10
CG	$\frac{3}{2}\lambda_{\max}^{-1}(M_1A)$	191	139	122	107	94	93	85	75	68	67	67	67
	$\lambda_{\max}^{-1}(M_1A)$	191	159	141	123	108	106	96	85	77	76	76	76
$\mu_1 + \mu_2 = 3$													
CG	1.0	191	1011	896	810	702	688	621	563	505	503	503	501
	$2\lambda_{\max}^{-1}(M_1A)$	191	110	98	85	74	73	65	59	53	53	53	53
	$\frac{3}{2}\lambda_{\max}^{-1}(M_1A)$	191	119	105	91	80	79	71	63	56	56	56	56
	$\lambda_{\max}^{-1}(M_1A)$	191	138	121	106	93	91	82	74	67	67	67	67
FS-541-4: $t = 8 \cdot 10^{-1}$; $\mu_1 + \mu_2 = 2$													
			Dimension of the coarse space										
	ω	M_1	0	1	2	3	4	5	6	7	8	9	10
GMRES(100)	1.0	126	†	†	†	†	†	†	†	†	†	†	†
	$\frac{3}{2}\lambda_{\max}^{-1}(M_1A)$	126	80	80	81	80	80	80	70	70	70	69	69
	$\lambda_{\max}^{-1}(M_1A)$	126	90	88	88	87	87	87	80	80	80	75	75
$\mu_1 + \mu_2 = 3$													
GMRES(∞)	1.0	106	87	86	86	84	84	84	72	72	72	72	72
	$\frac{3}{2}\lambda_{\max}^{-1}(M_1A)$	106	78	77	77	77	77	77	71	71	71	70	69
	$\lambda_{\max}^{-1}(M_1A)$	106	67	67	67	65	65	67	59	59	59	58	58
PORES3: $t = 1 \cdot 10^{-1}$; $\mu_1 + \mu_2 = 3$													
			Dimension of the coarse space										
	ω	M_1	0	1	2	3	4	5	6	7	8	9	10
GMRES(10)	1.0	266	120	33	24	19	16	14	13	10	10	9	9
	$\frac{3}{2}\lambda_{\max}^{-1}(M_1A)$	266	100	28	19	17	13	11	10	9	8	7	7
	$\lambda_{\max}^{-1}(M_1A)$	266	80	32	25	19	16	14	12	10	9	9	8

that when the number of smoothing steps is increased, they tend to spread the right part of the spectrum if the largest eigenvalues are bigger than two. For this reason, using a damping parameter ensures that the smoother is a contraction that better clusters the right part of the spectrum around one. Nevertheless this contraction may not have a positive effect. For instance, if there were a cluster beyond two, it would be spread by the smoothing iterations, which would possibly penalize the convergence of CG. On the other hand an isolated large eigenvalue would not affect CG, but a scaling to move it below two might shrink the complete spectrum and create clusters near the origin, which might have a negative affect on CG convergence.

For unsymmetric problems, the numerical experiments reveal that the damping also plays a role. Often using a damping parameter improves the convergence of GMRES, especially when the number of smoothing steps is odd. However, no general trends on the suitable choice of this parameter have been revealed by the experiments. Its role is clear in the framework of the stationary iterative method, but a better understanding of the convergence of unsymmetric Krylov solver would be necessary to better understand the influence of this parameter.

3.1.4. Sensitivity to the accuracy of the eigencomputation. As mentioned in the previous section, the eigenvalue calculation is performed in a preprocessing phase. In order to investigate the sensitivity of our algorithm to the eigencomputation accuracy we compute the eigenpairs of a slightly perturbed matrix, $(M_1A + E)$, where E is a random matrix that is scaled so that $\frac{||E||}{||M_1A||} = \eta$. We use these computed eigenvectors to build our preconditioners and compute the backward error of these vectors as if they were eigenvectors of M_1A. By varying η, we can monitor the level of the backward error that becomes comparable for each eigenvector. In Table 4, we give the number of iterations of the Krylov solvers when the backward error of the computed eigenvectors is varied. As there is one backward error per eigenvector, we give the average of them in the table. It can be seen that, in many cases, there is no need for very high accuracy in the computation of the eigenvectors. On the 685BUS matrix $\eta = 10^{-4}$ ($\eta = 10^{-3}$ for HOR131) gives similar results to $\eta = 10^{-14}$. The numerical behavior of the preconditioner slightly deteriorates when η grows on the GRE1107 matrix, which is probably due to some ill-conditioning of the eigenvectors. Furthermore, it seems that when the number of smoothing steps is increased the accuracy of the eigencalculation has a smaller impact on the numerical efficiency of the preconditioner.

Finally we mention that a sensitivity analysis of some spectral preconditioners is performed in [15] in the SPD cases. For the studied preconditioners, it is shown that their quality depends in particular on the magnitude and the clustering of the eigenvalues that are involved in their definition (i.e., the part of the spectrum they attempt to move). Similarly for the preconditioners studied here, it can be expected that if small or clustered eigenvalues are targeted, a good accuracy should be requested when computing them.

3.2. Implementation in applications. In the previous sections we have illustrated the possible numerical benefit of using a few smoothing iterations to improve the convergence of the preconditioned Krylov solver. In some situations it is the only way to get convergence and the benefit is clear. In other cases it improves the convergence but we might have some questions:

1. Is there a final benefit in terms of computational time because of the extra cost introduced by the calculation of the residual at each iteration of the smoother?
2. Is there a way to alleviate this residual calculation cost?

The answer to the first question is very problem- and machine-dependent. On parallel distributed computers, the dot product calculation in CG and the orthogonalization process in GMRES are the main bottlenecks for performance; reducing the number of iterations is an obvious way to alleviate this cost. When the preconditioner enables a significant reduction of iterations of the Krylov solvers, the extra cost of the matrix-vector products (that usually involves only neighbor-to-neighbor communication) can eventually be compensated for by the decrease of the cost of the dot products (that involves global communications).

One way to reduce this extra cost is to implement a cheaper but approximated matrix-vector product. Such a possibility exists, for instance, when the fast multipole techniques are implemented to compute the matrix-vector product in electromagnetics application. Another example arises in nonoverlapping domain decomposition and is further investigated in the next section.

TABLE 4
Sensitivity of M_{Mul} efficiency versus the accuracy of the eigencomputation.

685BUS: IC($4 \cdot 10^{-1}$); CG; $\mu_1 + \mu_2 = 1$												
Backward error	Dimension of the small dimensional correction space											
	M_1	0	1	2	3	4	5	6	7	8	9	10
$\approx 1 \cdot 10^{-15}$	123	122	83	83	70	57	51	43	40	36	36	34
$\approx 1 \cdot 10^{-6}$	123	123	86	86	72	57	53	43	41	37	37	35
$\approx 1 \cdot 10^{-4}$	123	122	91	90	76	61	57	46	43	39	38	36
$\approx 1 \cdot 10^{-3}$	123	122	127	127	108	94	81	67	59	49	49	44
$\approx 1 \cdot 10^{-2}$	123	122	134	130	290	298	261	257	250	207	210	209
$\mu_1 + \mu_2 = 3$												
$\approx 1 \cdot 10^{-15}$	123	60	43	44	35	28	28	22	20	19	19	17
$\approx 1 \cdot 10^{-6}$	123	60	46	46	38	32	29	24	22	19	18	18
$\approx 1 \cdot 10^{-4}$	123	60	49	49	41	34	25	25	24	21	20	19
$\approx 1 \cdot 10^{-3}$	123	60	64	59	49	48	39	30	30	24	24	22
$\approx 1 \cdot 10^{-2}$	123	60	81	74	76	75	79	63	71	68	67	65
HOR131: ILU($2 \cdot 10^{-1}$); GMRES(30); $\mu_1 + \mu_2 = 1$												
Backward error	Dimension of the small dimensional correction space											
	M_1	0	1	2	3	4	5	6	7	8	9	10
$\approx 1 \cdot 10^{-14}$	93	93	63	57	55	53	51	47	45	39	39	36
$\approx 1 \cdot 10^{-4}$	93	93	63	57	54	52	52	47	45	40	39	35
$\approx 1 \cdot 10^{-3}$	93	93	57	58	58	54	54	48	47	40	41	37
$\approx 1 \cdot 10^{-2}$	93	93	106	114	134	114	150	103	139	137	143	149
$\mu_1 + \mu_2 = 3$												
$\approx 1 \cdot 10^{-14}$	93	56	37	38	32	32	32	26	25	23	22	21
$\approx 1 \cdot 10^{-4}$	93	56	37	38	35	32	32	28	24	22	23	21
$\approx 1 \cdot 10^{-3}$	93	56	36	35	32	31	33	27	27	24	24	21
$\approx 1 \cdot 10^{-2}$	93	56	53	57	68	63	73	53	79	59	58	82
GRE1107: ILU($1 \cdot 10^{-2}$); GMRES(30); $\mu_1 + \mu_2 = 1$												
Backward error	Dimension of the small dimensional correction space											
	M_1	0	1	2	3	4	5	6	7	8	9	10
$\approx 1 \cdot 10^{-14}$	†	†	80	76	37	34	32	31	27	26	24	23
$\approx 1 \cdot 10^{-8}$	†	†	95	80	40	36	33	34	31	30	29	28
$\approx 1 \cdot 10^{-7}$	†	†	117	105	77	40	38	38	34	33	33	25
$\approx 1 \cdot 10^{-6}$	†	†	440	80	79	78	77	67	40	74	39	38
$\mu_1 + \mu_2 = 3$												
$\approx 1 \cdot 10^{-14}$	†	28	26	25	22	21	19	18	17	16	15	14
$\approx 1 \cdot 10^{-8}$	†	28	27	26	24	23	21	21	19	19	18	18
$\approx 1 \cdot 10^{-7}$	†	28	27	27	26	24	23	24	21	21	21	20
$\approx 1 \cdot 10^{-6}$	†	28	29	27	27	27	26	25	26	27	25	25

3.2.1. A case study in semiconductor device simulation. The numerical simulation of two-dimensional semiconductor devices is extremely demanding in terms of computational time because it involves complex embedded numerical schemes. At the kernel of these schemes is the solution of very ill-conditioned large linear systems. Such problems are challenging because they are large, very ill-conditioned, and extremely badly scaled. In that respect, robust and efficient linear solvers should be selected in order to reduce the elapsed time required to successfully perform a complete simulation. In this section we present some numerical experiments obtained by

TABLE 5

Experiments with M_{Mul} on SPD systems arising in semiconductor device modeling when the density of $\tilde{S}$ is varied; $\omega = \frac{3}{2}\lambda_{\max}^{-1}(M_1 A)$.

$\mu_1 + \mu_2 = 2$											
Density	Dimension of the coarse space										
	M_1	1	2	3	4	5	6	7	8	9	10
29%	52	34	39	38	41	38	37	37	37	37	36
42%	52	30	27	25	22	20	19	18	18	19	18
49%	52	28	25	24	21	19	18	17	17	17	17
58%	52	28	25	24	21	19	18	17	17	17	17
100%	52	28	25	24	21	19	18	17	17	17	17

using the spectral two-level preconditioners for the solution of the Schur complement systems resulting from a nonoverlapping domain decomposition approach. For that example the smoother is defined by an additive Schwarz preconditioner for the Schur complement system; we refer the reader to [16] for more details on this application.

In this implementation of the iterative substructuring approach the Schur complement matrix that is sparse with dense blocks is computed explicitly. In order to reduce the computational cost of the residual calculation involved in each step of the smoother, we consider a sparsified Schur complement to perform this operation. This sparse approximation of the Schur complement is obtained by dropping all the entries that are smaller than a prescribed threshold. Even though we have not considered other variants to get a cheap approximation, we mention that other possibilities exist. For instance, if the Schur complement were not explicitly formed, we might have considered some approximations computed using incomplete factorizations for the local subproblems or using a probing technique [9].

In Table 5 we report numerical experiments observed on one of the SPD linear systems that has to be solved in our semiconductor device simulation. It corresponds to a partitioning of the domain into eight subdomains, as shown in Figure 3. The complete domain is discretized by 155,000 degrees of freedom (dof), and the size of the interface between the subdomains (i.e., the size of the associated Schur complement matrix) is 1607. In this table, we vary the sparsity of $\tilde{S}$, the sparse approximation of the Schur complement matrix, and report the number of CG iterations required to obtain a reduction of the normalized unpreconditioned residual by 10^{-11}. This very small threshold for the stopping criterion of the linear solver was required to ensure the convergence of the nonlinear scheme. It can be seen that when the size of the coarse space is varied, more than half of the entries of S can be dropped for the implementation of the smoother without affecting the numerical behavior of M_{Mul}. This enables us to save a significant amount of floating-point operations involved in the smoother and consequently to reduce its cost substantially. Dropping 70% of the entries doubles the number of iterations while it divides by three the cost of applying the preconditioner. Depending on the target computer this latter situation might possibly lead to saving time.

3.2.2. A case study in electromagnetism applications. Electromagnetic scattering problems give rise to linear systems that are challenging to solve by iterative methods. For three-dimensional problems the boundary integral formulation of Maxwell's equations is often selected because of its nice approximation properties. This approach gives rise to dense complex systems. In recent years, the introduction of fast methods with reduced computational complexity and storage requirement has

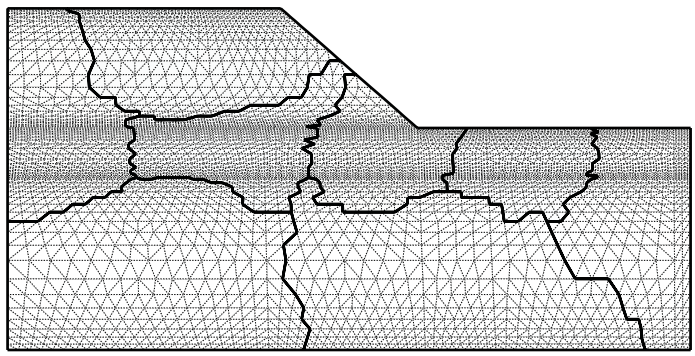
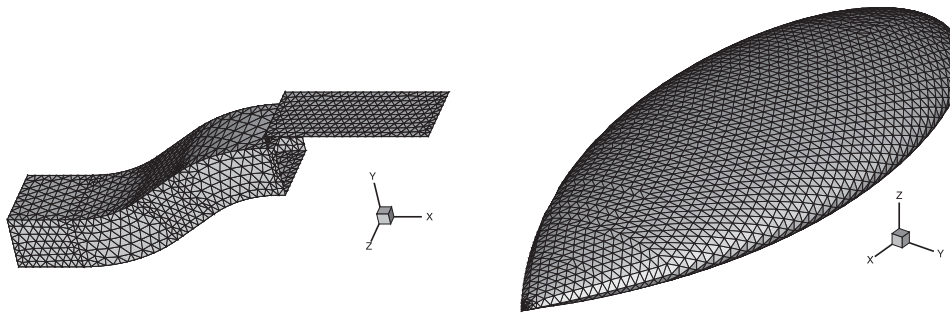


FIG. 3. *Mesh partitioning of the semiconductor discretization.*

attracted an increasing interest in the use of preconditioned Krylov methods for the simulation of real-life electromagnetic problems. In this section, we consider the surface integral formulation of Maxwell's equations modeled via the electric-field integral equation (EFIE), which is the most general but the most difficult formulation to solve for iterative solvers. Thus preconditioning is crucial and approximate inverse methods based on Frobenius-norm minimization have proved to be among the most effective preconditioners for solving these systems efficiently. We refer the reader to [7] and the references therein for a more detailed presentation of this preconditioning technique on this class of problems.

The Frobenius-norm minimization preconditioner M_{FROB} is very effective in clustering most of the eigenvalues near one, but tends to leave a few isolated eigenvalues close to zero. The presence of these very small eigenvalues can slow down the convergence of iterative solvers especially on large problems, where the memory constraints prevent the use of large restarts in GMRES, which is the best-suited solver for those problems. In this section, we apply the two-space spectral preconditioner on top of the Frobenius-norm minimization method. The preconditioner M_{FROB} is actually used to define a stationary iterative scheme implemented as a smoother. In the numerical experiments, the initial guess is the zero vector, we consider a right preconditioned GMRES method, and the threshold for the stopping criterion is set to 10^{-3} on the normwise backward error $\frac{||r||}{||b||}$, where r denotes the residual and b the right-hand side of the linear system. This tolerance is accurate for engineering purposes, as it enables the correct reconstruction of the radar cross-section of the objects. The parallel runs have been performed in single precision complex arithmetic on sixteen processors of an HP-Compaq Alpha server, a cluster of symmetric multiprocessors. We report results on two industrial problems, namely, a cobra (Figure 4(a)) discretized with 60,695 dof and an almond (Figure 4(b)) discretized with 104,793 dof. We are in the suited situation, where the first-level preconditioner M_{FROB} does a good job in clustering most of the eigenvalues near one and leaves only a few outliers close to the origin. Consequently, the eigenvectors are computed in forward mode by ARPACK in a preprocessing phase.

For the matrix-vector products, we use the fast multipole method (FMM) [18], which performs fast matrix-vector products in $O(n\log n)$ arithmetic operations. More precisely, a highly accurate FMM is used for the standard matrix-vector product operation within GMRES, and a less accurate FMM is used for the extra matrix-vector products involved in the smoother of the preconditioning operation. The price



(a) Cobra (b) Almond

FIG. 4. *Mesh associated with test examples.*

to pay for the accuracy is mainly computational time; high accuracy also means more computational time.

In Table 6 we report on the number of iterations and the elapsed time for different sizes of the coarse space and an increasing number of smoothing steps. For the two test examples we show results with restarted and full GMRES. For these experiments we set up $\omega^{-1} = \frac{2}{3}\lambda_{max}(M_{FROB}A)$, which seems to be an overall good choice. On these problems M_{Add} and M_{Mul} give the same number of iterations. For this reason we report only on the M_{Add} preconditioner, which is about 33% faster than M_{Mul} as it requires fewer matrix-vector operations, as indicated in section 2.4. For instance, on the almond test problem using 3 smoothing steps and 50 eigenvectors, both converge in 87 iterations with GMRES(10), M_{Mul} takes 10 minutes, and M_{Add} only 6 minutes.

In this table we also display the number of iterations and the elapsed time of GMRES with only M_{FROB}. It can be seen that the use of M_{Add} is always beneficial both from a number iterations viewpoint and from a computational time viewpoint. The gain in time varies from 14 to infinity with GMRES(10) and between 2 and 3 with GMRES(∞).

From a numerical point of view we observe in these examples the same behavior as before. That is, the larger the coarse space, the better the preconditioner; the number of GMRES iterations decreases when the number of smoothing steps is increased. Furthermore, the gain is larger if restarted GMRES is considered than if full GMRES is used as solver. In particular on the almond with GMRES(10) and less than 50 eigenvectors, the only way to get convergence is to perform a few steps of smoothing; with 50 eigenvectors the gain introduced by the smoothing iterations is still tremendous (i.e., larger than 21). On the Cobra problem, using 15 eigenvectors the gain is far larger than 2 with GMRES(10), and close to 2 for full GMRES.

Not only is the number of iterations significantly reduced, but the solution time is as well. On the almond problem, using 50 eigenvectors and GMRES(10) we gain a factor of 12 in elapsed time when we increase the number of smoothing steps from 1 to 3. We mention that on large electromagnetic problems (of size larger than 0.5 million unknowns) the use of small restarts is recommended in order to save the heavy cost of reorthogonalization and reduce the final solution cost. The choice of a small restart is also dictated by memory constraints [7]. With full GMRES, the number of iterations is significantly decreased but the total solution cost is likely to only slightly decrease

TABLE 6
Experiments with M_{Add} on the electromagnetics problems.

Cobra problem			
With M_{FROB}	GMRES(10) 2719 iterations (1 h 10 min)		
	GMRES(∞) 378 iterations (18 min)		
$\mu_1 + \mu_2 = 1$			
	Dimension of the coarse space		
	5	10	15
GMRES(10)	1458 (42 min)	594 (12 min)	517 (11 min)
GMRES(∞)	262 (9 min)	216 (7 min)	188 (6 min)
$\mu_1 + \mu_2 = 2$			
	Dimension of the coarse space		
	5	10	15
GMRES(10)	471 (18 min)	209 (7 min)	201 (6 min)
GMRES(∞)	161 (8 min)	132 (6 min)	115 (5 min)
$\mu_1 + \mu_2 = 3$			
	Dimension of the coarse space		
	5	10	15
GMRES(10)	281 (12 min)	132 (6 min)	124 (5 min)
GMRES(∞)	120 (7 min)	98 (6 min)	85 (5 min)

Almond problem			
With M_{FROB}	GMRES(10) +3000 iterations		
	GMRES(∞) 242 iterations (14 min)		
$\mu_1 + \mu_2 = 1$			
	Dimension of the coarse space		
	10	30	50
GMRES(10)	+3000	+3000	1867 (1 h 12 min)
GMRES(∞)	229 (13 min)	157 (8 min)	132 (6 min)
$\mu_1 + \mu_2 = 2$			
	Dimension of the coarse space		
	10	30	50
GMRES(10)	552 (29 min)	245 (14 min)	176 (9 min)
GMRES(∞)	134 (9 min)	92 (6 min)	77 (6 min)
$\mu_1 + \mu_2 = 3$			
	Dimension of the coarse space		
	10	30	50
GMRES(10)	216 (16 min)	116 (9 min)	87 (6 min)
GMRES(∞)	97 (9 min)	66 (6 min)	56 (6 min)

when the number of smoothing steps is large as the preconditioner is expensive to apply. The optimal selection of the size of the coarse space and of the number of smoothing steps remains an open question, and the choice mainly depends on the clustering properties of the initial preconditioner. In terms of computational cost the coarse grid correction and the smoothing mechanism are complementary components that have to be suitably combined. On the cobra problem, for instance, using 3 smoothing steps and 5 eigenvectors we obtain a better convergence rate but similar computational time than when using 1 smoothing step but 15 eigenvectors; on the almond problem, the solution cost of GMRES(10) with 3 smoothing steps and 30 eigenvectors is similar to the time with 2 smoothing steps but 50 eigenvectors. Enabling a reduction in the number of eigenvectors by using more smoothing steps is a desirable feature in contexts where computing many eigenvalues can become very expensive, which is, for instance, the case in this application for very large problems.

To illustrate the cost related to the eigenvector calculation, we mention that the computation of 30 eigenvectors in the almond problem requires 1100 matrix-vector products, that is, about 2 hours. In the cobra example, 500 matrix-vector products

are needed to calculate 15 eigenvectors, which corresponds to 1 hour. The extra cost associated with this preprocessing is quickly amortized as many right-hand sides have usually to be solved to compute the so-called radar cross-section. In the cobra example, the eigencalculation is amortized if more than 2 right-hand sides are solved using GMRES(10) or more than 5 if full GMRES can be afforded. For the almond, the gain is unbounded if GMRES(10) is used and 15 right-hand sides should be solved to amortize the eigencalculation if full GMRES can be afforded. To conclude, we mention that numerical experiments are reported in [10] on the same test problems with the preconditioner presented in [6]. The observed behavior is similar to those reported here when $\mu_1 + \mu_2 = 1$, which again illustrates the benefits of the new approach when only a few eigenpairs are available.

4. Concluding remarks. In this work, we exploit some ideas implemented in the multigrid techniques for the solution of PDEs to derive new additive and multiplicative spectral two-level preconditioners for the solution of general linear systems. We propose a scheme that enables us to improve a given preconditioner that leaves only few eigenvalues close to zero. We study the spectrum of the new preconditioned matrix associated with these schemes. The effectiveness and robustness of these preconditioners is mainly due to their ability to shift to one a selected set of eigenvalues and to cluster near one most of the others. We illustrate some features of these techniques in small examples from the Matrix Market and illustrate their attractive numerical behavior on two challenging real-world applications. On electromagnetism industrial problems, we show that the preconditioner enables us to save a significant amount of time for the solution of large problems on parallel platforms.

Acknowledgments. The authors would like to thank Ray Tuminaro for the stimulating discussions he had with the second author when he visited him at Sandia. The authors are grateful to Emeric Martin for his assistance in implementing the preconditioner in the electromagnetism code. Finally, we would like to thank the anonymous referees for their valuable remarks, questions, and comments that enabled us to improve this paper.

REFERENCES

[1] J. Baglama, D. Calvetti, G. H. Golub, and L. Reichel, *Adaptively preconditioned GMRES algorithms*, SIAM J. Sci. Comput., 20 (1998), pp. 243–269.

[2] P. Bastian, W. Hackbush, and G. Wittum, *Additive and multiplicative multigrid: A comparison*, Computing, 60 (1998), pp. 345–364.

[3] M. Benzi, C. D. Meyer, and M. Tůma, *A sparse approximate inverse preconditioner for the conjugate gradient method*, SIAM J. Sci. Comput., 17 (1995), pp. 1135–1149.

[4] R. Boisvert, R. Pozo, K. Remington, R. Barrett, and J. Dongarra, *Matrix Market: A web resource for test matrix collections*, in The Quality of Numerical Software: Assessment and Enhancement, R. Boisvert, ed., Chapman and Hall, London, 1997, pp. 125–137.

[5] J. H. Bramble, J. E. Pasciak, and A. H. Schatz, *The construction of preconditioners for elliptic problems by substructuring* I, Math. Comp., 47 (1986), pp. 103–134.

[6] B. Carpentieri, I. S. Duff, and L. Giraud, *A class of spectral two-level preconditioners*, SIAM J. Sci. Comput., 25 (2003), pp. 749–765.

[7] B. Carpentieri, I. S. Duff, L. Giraud, and G. Sylvand, *Combining fast multipole techniques and an approximate inverse preconditioner for large electromagnetism calculations*, SIAM J. Sci. Comput., 27 (2005), pp. 774–792.

[8] B. Carpentieri, L. Giraud, and S. Gratton, *Additive and Multiplicative Two-Level Spectral Preconditioning for General Linear Systems*, Technical report TR/PA/04/38, CERFACS, Toulouse, France, 2004.

[9] T. F. C. Chan and T. P. Mathew, *The interface probing technique in domain decomposition*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 212–238.

[10] I. S. Duff, L. Giraud, J. Langou, and E. Martin, *Using spectral low rank preconditioners for large electromagnetic calculations*, Internat. J. Numer. Methods Engrg., 62 (2005), pp. 416–434.
[11] J. Erhel, K. Burrage, and B. Pohl, *Restarted GMRES preconditioned by deflation*, J. Comput. Appl. Math., 69 (1996), pp. 303–318.
[12] M. Fisher, *Minimization algorithms for variational data assimilation*, in Proceedings of the ECMWF Seminar "Recent Developments in Numerical Methods for Atmospheric Modelling," 1998, pp. 364–385.
[13] L. Fournier and S. Lanteri, *Multiplicative and additive parallel multigrid algorithms for the acceleration of compressible flow computations on unstructured meshes*, Appl. Numer. Math., 36 (2001), pp. 401–426.
[14] J. Frank and C. Vuik, *On the construction of deflation-based preconditioners*, SIAM J. Sci. Comput., 23 (2001), pp. 442–462.
[15] L. Giraud and S. Gratton, *On the sensitivity of some spectral preconditioners*, SIAM J. Matrix Anal. Appl., 27 (2006), pp. 1089–1105.
[16] L. Giraud, A. Marrocco, and J.-C. Rioual, *Iterative versus direct parallel substructuring methods in semiconductor device modelling*, Numer. Linear Algebra Appl., 12 (2005), pp. 33–53.
[17] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins Studies in the Mathematical Sciences, The Johns Hopkins University Press, Baltimore, MD, 1996.
[18] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comput. Phys., 73 (1987), pp. 325–348.
[19] W. Hackbusch, *Multigrid Methods and Applications*, Springer-Verlag, Berlin, 1985.
[20] S. A. Kharchenko and A. Yu. Yeremin, *Eigenvalue translation based preconditioners for the GMRES(k) method*, Numer. Linear Algebra Appl., 2 (1995), pp. 51–77.
[21] L. Yu. Kolotilina and A. Yu. Yeremin, *Factorized sparse approximate inverse preconditionings* I. *Theory*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 45–58.
[22] C. Le Calvez and B. Molina, *Implicitly restarted and deflated GMRES*, Numer. Algorithms, 21 (1999), pp. 261–285.
[23] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large-Scale Problems with Implicitly Restarted Arnoldi Methods*, SIAM, Philadelphia, 1998.
[24] J. A. Meijerink and H. A. van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comp., 31 (1977), pp. 148–162.
[25] R. B. Morgan, *GMRES with deflated restarting*, SIAM J. Sci. Comput., 24 (2002), pp. 20–37.
[26] R. A. Nabben and C. Vuik, *A comparison of deflation and coarse grid correction applied to porous media flow*, SIAM J. Numer. Anal., 42 (2004), pp. 1631–1647.
[27] R. A. Nicolaides, *Deflation of conjugate gradients with applications to boundary value problems*, SIAM J. Numer. Anal., 24 (1987), pp. 355–365.
[28] A. Padiy, O. Axelsson, and B. Polman, *Generalized augmented matrix preconditioning approach and its application to iterative solution of ill-conditioned algebraic systems*, SIAM J. Matrix Anal. Appl., 22 (2000), pp. 793–818.
[29] Y. Saad, *Projection and deflation methods for partial pole assignment in linear state feedback*, IEEE Trans. Automat. Control, 33 (1988), pp. 290–297.
[30] Y. Saad, *Analysis of augmented Krylov subspace techniques*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 435–449.
[31] Y. Saad, *ILUT: A dual threshold incomplete LU factorization*, Numer. Linear Algebra Appl., 1 (1994), pp. 387–402.
[32] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.
[33] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.
[34] R. S. Tuminaro, *A highly parallel multigrid-like method for the solution of the Euler equations*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 88–100.
[35] H. A. van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 631–644.
[36] H. Waisman, J. Fish, R. S. Tuminaro, and J. Shadid, *The generalized global basis method*, Internat. J. Numer. Methods Engrg., 61 (2004), pp. 1243–1269.

SIAM J. SCI. COMPUT.
Vol. 29, No. 4, pp. 1613–1636
© 2007 Society for Industrial and Applied Mathematics

MULTIGRID FOR HIGH-DIMENSIONAL ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS ON NON-EQUIDISTANT GRIDS*

H. BIN ZUBAIR†, C. W. OOSTERLEE‡, AND R. WIENANDS§

Abstract. This work presents techniques, theory, and numbers for multigrid in a general d-dimensional setting. The main focus of this paper is the multigrid convergence for high-dimensional partial differential equations on non-equidistant grids such as may be encountered in a sparse-grid solution. As a model problem we have chosen the anisotropic stationary diffusion equation on a rectangular hypercube. We present some techniques for building the general d-dimensional adaptations of the multigrid components and propose grid-coarsening strategies to handle anisotropies that are induced due to discretization on a non-equidistant grid. Apart from the practical formulas and techniques, we present—in detail—the smoothing analysis of the point ω-red-black Jacobi method for a general multidimensional case. We show how relaxation parameters may be evaluated efficiently and used for better convergence. This analysis incorporates *full* and *partial doubling* and *quadrupling* coarsening strategies as well as the second- and the fourth-order finite difference operators. Finally we present some results derived from numerical experiments based on the test problem.

Key words. multigrid, high-dimensional PDEs, anisotropic diffusion equation, coarsening strategies, point-smoothing methods, relaxation parameters, Fourier-smoothing analysis

AMS subject classifications. 65N55, 65F10, 65Y20

DOI. 10.1137/060665695

1. Introduction.

Multidimensional partial differential equations have diverse applications in various fields of applied sciences, including financial engineering [8], molecular biology [3], and quantum dynamics [1, 21]. There are quite a few fast and efficient solution techniques for partial differential equations (henceforth PDEs) of which multigrid ranks among the best. Multigrid is a well-known iterative procedure for the solution of large and sparse linear systems that arise from various kinds of PDE discretizations. The existing literature on the multigrid treatment of various problems, however, rarely explores issues that arise out of growth in the dimensionality of the problem. The implications of dimensionality growth include deterioration of the multigrid convergence rate, impractical storage requirements, and huge amounts of CPU time for single grid solution methods. Our main emphasis in this paper lies on the first challenge. We abbreviate [illegible] d [illegible] as [illegible]. In this paper d represents both [illegible] and [illegible]. So, e.g., $3d$ is to be interpreted as 3-dimensional.

The major hinderance in the numerical solution of multidimensional PDEs is the so-called [illegible], which implies that with the growth in dimensions we have an exponential growth in the number of grid points. This increases the

*Received by the editors July 21, 2006; accepted for publication (in revised form) March 27, 2007; published electronically July 18, 2007. This research has been partially supported by the Dutch government through the national program BSIK: knowledge and research capacity, in the ICT project BRICKS (http://www.bsik-bricks.nl), theme MSV1, and partially by the Government of Pakistan through The HEC-Pakistan's research grant, contract Ref: 1-3/PM-OVER/Neth/SPMU/2004.

http://www.siam.org/journals/sisc/29-4/66569.html

†Delft University of Technology, Faculty EEMCS, Delft Institute of Applied Mathematics, 07.030, Mekelweg 4, 2628 CD, Delft, The Netherlands (h.binzubair@tudelft.nl).

‡CWI—Center for Mathematics and Computer Science, Amsterdam, The Netherlands (c.w.oosterlee@cwi.nl).

§Mathematical Institute, University of Cologne, Weyertal 86-90, 50931 Cologne, Germany (wienands@math.uni-koeln.de).

computational cost of many good algorithms. Although we do not address this issue in particular, we would like to stress that a way to handle dimensionality is the sparse-grid method [9, 21]. One of the characteristics of these [illegible] is that these grids are essentially non-equidistant, and therefore efficient solution methods for this type of grid are quite important.

A multigrid treatment of high-dimensional PDEs based on hyperplane relaxation has been proposed by Reisinger and Wittum in [8]. We present a multigrid treatment based on point relaxation and partial coarsening schemes. We demonstrate how the multigrid convergence factor can be brought down for higher d by the suggested grid-coarsening strategies and a proper choice of the relaxation parameters in the smoothing process.

The strategy that we suggest in this paper for good multigrid convergence is that, keeping the point-smoothing method, we coarsen the grid simultaneously along all of the dimensions where the errors are strongly coupled. We call this strategy [illegible]. This way we relax the anisotropy at each successive grid level, until the problem is isotropic to the point where full coarsening is feasible. Full coarsening from this stage onwards brings the grid to the coarsest possible level, where we solve exactly. These strategies therefore consist of two main phases: the [illegible] and the [illegible]. We show that multigrid based on [illegible] ($h \to 4h$) transfers during the partial coarsening phase of the scheme gives a multigrid speed boost if optimal relaxation parameters are used in the point-smoothing process. We would like to point out that partial coarsening schemes have also been put forward by Larsson, Lien, and Yee identifying the conditions for partial coarsening through local Fourier-smoothing analysis (LFA) [7]. Moreover, the utility of adaptive grid-coarsening in multigrid has also been demonstrated by Elias, Stubley, and Raithby and Horton and Vandewalle in [4] and [5], respectively.

Foremost (in section 2) we point out that the Black–Scholes pricing problem PDE can be reduced to a standard -dimensional heat equation, indicating the need for fast solution methods for high-dimensional PDEs of Poisson type. The discretization and subsequent implementation of a d-dimensional PDE is somewhat involved, and in section 3 we show how this can be done with Kronecker products. Section 4 deals with d-multigrid, point-smoothing, coarsening strategies and concludes with a computational complexity analysis. Next we provide the LFA for the ω-RB Jacobi method and show how it can be extended to d dimensions. This is section 5. We also point out here how we incorporate partial doubling and quadrupling in this analysis. This section concludes with a tabular presentation (up to $6d$) of some optimal relaxation parameters. Finally in the last section we present quite a few versatile numerical experiments and demonstrate the excellent multigrid convergence that we get.

2. Multi-*d* equations in finance. The application on which we focus in this section is the pricing of multiasset options through a Black–Scholes model which is a high-dimensional parabolic PDE, reducible to the heat equation.

A generalized d-asset Black–Scholes equation reads

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d}\rho_{ij}\sigma_i\sigma_j S_i S_j \frac{\partial^2 V}{\partial S_i \partial S_j} + r\sum_{i=1}^{d} S_i \frac{\partial V}{\partial S_i} - rV = 0, \tag{2.1}$$
$$(0 < S_1, \ldots, S_d < \infty, \quad 0 \leqslant t < T).$$

V stands for the option price; S_i are the d underlying asset prices; t is the current time; ρ_{ij} are the correlation coefficients between the Wiener processes modeling the

movement of the price of the ith and the jth assets; σ_i is the volatility of the ith asset price; r is the risk-free interest rate. For pricing options on a basket of d assets, it has been shown [6, 20] that (2.1) under the following transform of the asset price S_i:

$$y_i = \frac{1}{\sigma_i}\left(r - \frac{{\sigma_i}^2}{2}\right)\tau + \frac{1}{\sigma_i}\ln S_i, \quad i = 1, 2, \ldots, d, \tag{2.2}$$
$$\tau = T - t$$

(T is a constant and represents the maturity time of the option) yields the equation

$$\frac{\partial V}{\partial \tau} = \frac{1}{2}\sum_{i=1}^{d}\frac{\partial^2 V}{\partial {x_i}^2}, \qquad -\infty < x_i < \infty, \quad 0 < \tau \leqslant T, \tag{2.3}$$

which we identify as the heat-conduction equation in d dimensions. When discretized by the implicit Crank–Nicolson scheme, we get a discrete s.p.d. operator. It is worthwhile to mention that the time discretization adds to the [illegible] of the main diagonal of the discrete operator, and so the stationary diffusion equation serves well as a limiting [illegible] case, in the study of convergence behavior.

3. The discretization. We first recall that discrete operators can be implemented in two different ways. One is the stencil method, and the other is the matrix method. The stencil method saves storage but is inherently difficult to implement due to the visual constraints—imposed by high dimensionality—on the problem. Therefore, to circumvent the complicated implementation issues we use matrices (in sparse storage formats), and here we present some matrix generation formulas based on Kronecker tensor products.

3.1. The continuous problem and its discretization scheme. For analysis and experimentation we choose the -dimensional stationary diffusion equation, with Dirichlet boundary conditions, to serve as our model problem. In what follows $\mathbf{x}$ is a -tuple $\mathbf{x} = (x_1, x_2, \ldots, x_d)$, and $\{a_i, b_i, \varepsilon_i\} \in \mathbb{R}$, with $\varepsilon_i > 0$. The continuous problem reads:

$$-Lu(\mathbf{x}) = -\sum_{i=1}^{d}\varepsilon_i\frac{\partial}{\partial x_i}u(\mathbf{x}) = f^{\Omega}(\mathbf{x}), \quad \mathbf{x} \in \Omega = \prod_{i=1}^{d}(a_i, b_i) \subset \mathbb{R}^d, \tag{3.1}$$
$$u(\mathbf{x}) = f^{\Gamma}(\mathbf{x}), \qquad \mathbf{x} \in \Gamma = \partial\Omega \quad (x_i \in \{a_i, b_i\}).$$

Subsequently, the discrete counterpart reads

$$-L_h u_h(\mathbf{x}) = f_h^{\Omega}(\mathbf{x}), \quad \mathbf{x} \in \Omega_h = \prod_{i=1}^{d}(a_i, b_i) \subset \mathbb{R}^d, \tag{3.2}$$
$$u_h(\mathbf{x}) = f_h^{\Gamma}(\mathbf{x}), \quad \mathbf{x} \in \Gamma_h = \partial\Omega_h \quad (x_i \in \{a_i, b_i\}).$$

The discretization of the Laplacian L_h is chosen to be either $O(h^2)$ accurate, giving a $(2d+1)$-point stencil, or else $O(h^4)$ accurate, with a $(4d+1)$-point long stencil for all [illegible] points. The number of cells in the discretization grid along the ith dimension—represented by N_i—[illegible] be equal to the number of cells along (say)

the jth dimension. So with $h_i = (b_i - a_i)/N_i$—the mesh size along the ith dimension—the 1 variants of these multidimensional stencils are

$$\left(\frac{\partial^2}{\partial x_i^2}\right)_h \triangleq \frac{1}{h_i^2}[1 \quad -\underline{2} \quad 1] + O(h^2), \tag{3.3}$$

$$\left(\frac{\partial^2}{\partial x_i^2}\right)_h \triangleq \frac{1}{12h_i^2}[-1 \quad 16 \quad -\underline{30} \quad 16 \quad -1] + O(h^4). \tag{3.4}$$

It has to be noted that the $O(h^4)$ long stencil uses the so-called [illegible] (points outside the discretization grid) when applied to points [illegible] the boundary. To alleviate this problem we have the option of employing a different stencil, having shorter connections at the boundary. Thus we can either employ the simple $O(h^2)$ operator at the boundaries or else use a different scheme with one-sided differencing. In this work, we use the second-order stencil, for points near the boundaries. The discretization given by (3.2) leads to the following matrix equation:

$$\mathbf{A}_h \mathbf{u}_h = \mathbf{b}_h. \tag{3.5}$$

3.2. The discretization matrix $\mathbf{A_h}$ and implementation in arbitrary dimensions. As we have Dirichlet boundary conditions, we eliminate boundary points from the matrix $\mathbf{A}_h$. This scheme results in a total of M unknowns—$(M \times M)$ being the order of the discretization matrix $\mathbf{A_h}$—with

$$M = \prod_{i=1}^{d} (N_i - 1).$$

We represent the discretization grid $\mathbf{G}$ by

$$\mathbf{G} = [N_1, N_2, \ldots, N_d]. \tag{3.6}$$

The discrete matrix $\mathbf{A}_h$ in (3.5) can be built by the following tensor product formula:

$$\mathbf{A}_h = \sum_{i=1}^{d} \left\{ \bigotimes_{j=i}^{d-1} \mathbf{I}_{(d+i-j)} \otimes \mathbf{L}_i \otimes \bigotimes_{j=1}^{i-1} \mathbf{I}_{(i-j)} \right\}. \tag{3.7}$$

$\otimes$ is the Kronecker tensor product of matrices. Likewise $\bigotimes$ is the cumulative Kronecker tensor product. For example,

$$\bigotimes_{i=1}^{3} \mathbf{P}_i = \mathbf{P}_1 \otimes \mathbf{P}_2 \otimes \mathbf{P}_3.$$

Kronecker tensor products are noncommutative and associative operations (see [10]). The order is determined by the subscripts here, and the associative hierarchy does not matter.

In (3.7), $\mathbf{I}_k$ ($k \in \{1, 2, \ldots, d\}$) is the identity matrix of order $(N_k - 1)$, and $\mathbf{L}_i$ is the one-dimensional discrete-Laplacian matrix, constructed through (3.3) and (3.4) as illustrated by the following example. Suppose that $\mathbf{G} = [8, 6]$ (see (3.6)) is the grid that we have for a certain $2d$ problem; then we construct $\mathbf{L}_i$ by writing down the discrete stencil in (3.4) for each point, including the boundaries. Then we isolate the left and the right boundary vectors (as shown below) and incorporate them in the

right-hand side $\mathbf{b}_h$. For example, $\mathbf{L}_1$ is the following matrix (without the left and the right boundary vectors) according to the choice $O(h^4)$ of the computational accuracy:

$$\frac{\varepsilon_1}{12h_1^2}\begin{bmatrix} & & & & & & & & \\ 12 & -24 & 12 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 16 & -30 & 16 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 16 & -30 & 16 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 16 & -30 & 16 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 16 & -30 & 16 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 16 & -30 & 16 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 12 & -24 & 12 \\ & \multicolumn{7}{c}{\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{\mathbf{L}_1}} & \end{bmatrix}.$$

$1d$ discrete Laplacian matrices constructed in this way for each grid dimension are to be substituted in (3.7) for building up the discrete d-dimensional operator matrix $\mathbf{A}_h$. (The $O(h^2)$ discretization is handled similarly.)

3.3. The right-hand side $\mathbf{b}_h$. The right-hand side $\mathbf{b}_h$ of (3.5) consists of the source function f_h^Ω and the boundary function f_h^Γ.

It is important to define a consistent [illegible] in high dimensions. In our enumeration scheme we represent the entire set of indices by an [illegible] $\mathcal{I}$, where each row (a d-tuple) represents the index of a grid point and counts in descending order, i.e., from right to left, for the ascending order of the dimensions. This formulation of the index set is the natural extension of the so-called [illegible] order in $2d$. As an illustration of this lexicographic order, we have grid coordinates as a pair (i_2, i_1) in $2d$ and as a triple (i_3, i_2, i_1) in $3d$, and thus, generally in d-dimensions, we have the index of a grid point as $(i_d, i_{(d-1)}, \ldots, i_1)$; piling them up in lexicographic order, we get the entire set as

$$\mathcal{I} = [\iota_d \quad \iota_{(d-1)} \ldots \iota_i \ldots \iota_2 \quad \iota_1], \tag{3.8}$$

where each ι_k is a column vector of length M. Also consider the following definitions, which we require to build the index set for the interior and the boundary points:

$$\begin{aligned} &\eta_i; \quad \ni \quad \eta_i = [1, 2, \ldots, (N_i - 1)]^T \quad (i = 1, 2, \ldots, d) \quad \text{(see (3.6))}, \\ &\mathbf{1}_i; \quad \ni \quad \mathbf{1}_i = \underbrace{[1, 1, \ldots, 1]}_{\text{Total } N_i}{}^T. \end{aligned} \tag{3.9}$$

We now formulate the columns of $\mathcal{I}$ as follows:

$$\iota_i = \bigotimes_{j=i}^{d-1} \mathbf{1}_{(d+i-j)} \otimes \eta_i \otimes \bigotimes_{j=1}^{i-1} \mathbf{1}_{(i-j)}. \tag{3.10}$$

At this stage the vector of source function values can be computed as the development of the index set $\mathcal{I}$ is complete. Thus, computing the source function for each row of $\mathcal{I}$ and denoting it by $\mathcal{S}$, we have $\mathcal{S} = f_{\mathcal{I}}^\Omega$.

Now for computing the contribution of boundaries in $\mathbf{b}_h$, recall that we isolated two column vectors, namely, the left and the right boundary-coefficient vectors from the $1d$ Laplacian operators in each dimension. Considering the case of the ith dimension, if we denote these by l_i and r_i, respectively, then we can define the ith

d-dimensional left and right boundary-coefficient vectors, viz., $\mathcal{L}_i$ and $\mathcal{R}_i$ as follows:

$$\mathcal{L}_i = \bigotimes_{j=i}^{d-1} \mathbf{1}_{(d+i-j)} \otimes l_i \otimes \bigotimes_{j=1}^{i-1} \mathbf{1}_{(i-j)}, \tag{3.11}$$
$$\mathcal{R}_i = \bigotimes_{j=i}^{d-1} \mathbf{1}_{(d+i-j)} \otimes r_i \otimes \bigotimes_{j=1}^{i-1} \mathbf{1}_{(i-j)}.$$

The contribution of the boundary values in $\mathbf{b}_h$ has two parts, i.e., values from the left boundary and values from the right boundary. We denote the two by $\mathcal{B}_L$ and $\mathcal{B}_R$, respectively. $\mathcal{B}_L$ is the cumulative sum of the d left boundaries and likewise for the right. At this point a word about the [illegible] is in order. If in (3.8) any ι_i is replaced by a vector of [illegible], we get a left boundary index set, and if we replace it by a vector of [illegible], we get a right boundary index set. If

$$\mathcal{I}_{L_i} = [\iota_d \quad \iota_{(d-1)} \dots \iota_(i-1) \quad \overline{a_i} \quad \iota_(i+1) \dots \iota_2 \quad \iota_1],$$
$$\mathcal{I}_{R_i} = [\iota_d \quad \iota_{(d-1)} \dots \iota_(i-1) \quad \overline{b_i} \quad \iota_(i+1) \dots \iota_2 \quad \iota_1],$$

then

$$\mathcal{B}_L = \sum_{i=1}^{d} (\mathcal{L}_i \diamond f^{\Gamma}_{\mathcal{I}_{L_i}}), \tag{3.12}$$
$$\mathcal{B}_R = \sum_{i=1}^{d} (\mathcal{R}_i \diamond f^{\Gamma}_{\mathcal{I}_{R_i}}).$$

$\diamond$ represents componentwise multiplication of the operand column vectors.

Thus we have from the right-hand side of (3.5)

$$\mathbf{b}_h = \mathcal{S} + \mathcal{B}_L + \mathcal{B}_R, \tag{3.13}$$

and the discretization is complete.

4. d-multigrid based on point smoothing. The core of this work is the coarsening strategies proposed here, which are based on a mixture of doubling ($h \to 2h$) and quadrupling ($h \to 4h$) grid transfers and which—coupled with point-based relaxation—yield very efficient multigrid methods for problems on non-equidistant grids. The aim is to close upon a robust method that applies for general grid-aligned anisotropies in d dimensions.

Like basic multigrid for two- and three-dimensional problems, -multigrid also consists of the essential components, the smoothing method and the coarse grid correction. The well-known algorithm of multigrid as presented in [14] does not change for the higher-dimensional case; however, the components have to be generalized to match this new situation. General multigrid algorithms are presented in the literature [11, 12, 13, 14, 15, 17].

For anisotropic problems it is a choice to keep the point-smoothing method and to coarsen only along a subset of the dimensions, precisely, along those that are strongly coupled. This ensures that coarsening takes place only where the errors are smooth. For [illegible] isotropic problems the best strategy is to combine the point-smoothing method with doubling-based full coarsening and to use the optimal relaxation parameters obtained for d dimensions.

4.1. The relaxation method. Of the many available point-smoothing-based relaxation methods, we choose the ω-red-black Jacobi method due to its excellent smoothing effect for problems of the Poisson type. The red-black Jacobi method is equivalent to the Gauss–Seidel red-black method for the $O(h^2)$ $(2d+1)$-point discretization stencil and in that it is commonly abbreviated as GS-RB in the literature. In this section we assume familiarity with standard GS-RB for $2d$ problems; see [11, 14]. ω-RB Jacobi consists of two partial steps, each an ω-Jacobi sweep; the first one [illegible] only the [illegible] (odd) points and the second one applying to and updating only the [illegible] (even) points in the grid.

From an implementational point of view, this red-black smoothing procedure which is based on partial steps depends upon a partitioning process by which the grid $\boldsymbol{G}$ can be dissected into the [illegible] part ($\boldsymbol{G}_R$) and the [illegible] part ($\boldsymbol{G}_B$). The grid-point enumeration that we employ in our implementation scheme is such that the points are arranged linearly (in a column vector) and counted out in lexicographical order for a d-dimensional grid. The unequal number of cells along different dimensions of the grid makes this partitioning process somewhat nontrivial. In Appendix A we present a way to bring about this segregation of odd and even points from a purely implementational aspect.

We also employ optimal relaxation parameters ω_{opt} in the relaxation process. It is well known that $\omega = 1$ serves as a good choice for the $2d$ isotropic case. In the case of anisotropy and higher dimensions the error-smoothing effect of the relaxation method can be enhanced by the use of optimal relaxation parameters [19]. This implies that a search for ω_{opt} pays off. We employ a d-dimensional LFA for this purpose; see section 5.

4.2. Coarsening strategies to handle anisotropies. We present two [illegible] coarsening procedures as our test cases, both of which have shown excellent convergence results. We call them [illegible] and [illegible]. Both of the strategies are similar in the sense that they consist of two distinct coarsening phases: the partial coarsening phase and the full coarsening phase. The difference is in the partial coarsening phase, where the [illegible] in strategy 1 is doubling (h $\rightarrow$ 2h) and in strategy 2 is quadrupling (h $\rightarrow$ 4h). In the full coarsening phase both of the strategies are identical and consist of doubling transfers only.

In strategy 1 we first identify the dimension(s) having the strongest coupling. This is indicated by the magnitude of the [illegible] $\tilde{c}_i$ defined as

$$\tilde{c}_i = \varepsilon_i \times \left(\frac{N_i}{b_i - a_i} \right)^2;$$

see (3.1) and (3.6). The larger the coupling factor $\tilde{c}_i$, the stronger the coupling. Fourier analysis suggests that all dimensions having a coupling factor $\tilde{c}_i$ within a range of 1.3 times the largest coupling factor identified can be doubled simultaneously. This decision is made (at each successive grid level) and identifies all of those dimensions along which doubling will take place. Employing this strategy recursively makes the discrete problem isotropic inasmuch as all coupling factors are within this range; onwards from here full doubling takes over. Once the coarsest possible grid is reached, an exact solution takes place. In the last section we evaluate this strategy for the $O(h^2)$ $(2d+1)$-point stencil and for the $O(h^4)$ $(4d+1)$-point long stencil.

In strategy 2 the threshold value is 1.0, which means that we quadruple only along the dimension(s) which are identified as having the strongest coupling (having the largest $\tilde{c}_i$). This ensures that quadrupling (in comparison with doubling) takes

place along fewer dimensions. This strategy gives good convergence when employed in conjunction with optimal relaxation parameters and is cheaper than strategy 1 because of quadrupling in the partial phase. Moreover, we suggest that a strategy based on quadrupling in the full coarsening phase should not be employed in a general multidimensional case, as full quadrupling always loses against full doubling and hence is quite apt to hamper multigrid convergence. See, for an example, section 6, Figure 3.

We take the grid size along each dimension always as a power of 2. When the anisotropy stems only from discretization on non-equidistant grids, as encountered in the sparse-grid solution, the sequence of coarse grids generated by the two strategies are as follows.

[illegible] 1. Suppose that a particular discretization grid for a certain $5d$ problem is $\mathbf{G} = [\,32\ \ 8\ \ 8\ \ 128\ \ 32\,]$. $\varepsilon_i = 1\ \forall\ i$ and $\Omega = (0,1)^5$. Then the sequence of grids that we get is the following:

(4.1)

[illegible] 1							[illegible] 2						
Ω_6	= [	32	8	8	128	32	]						
Ω_5	= [	32	8	8	64	32	]	Ω_4	= [	32	8	8	128	32	]
Ω_4	= [	32	8	8	32	32	]	Ω_3	= [	32	8	8	32	32	]
Ω_3	= [	16	8	8	16	16	]	Ω_2	= [	8	8	8	8	8	]
Ω_2	= [	8	8	8	8	8	]	Ω_1	= [	4	4	4	4	4	]
Ω_1	= [	4	4	4	4	4	]	Ω_0	= [	2	2	2	2	2	]
Ω_0	= [	2	2	2	2	2	]								

Results for this particular experiment are available in section 6, Figure 4.

4.3. Coarse-grid discretization. An important component in the coarse-grid correction process is the choice of the coarse-grid operator L_H. In this paper we use the coarse-grid analog of the discrete operator on the fine grid. Once the next coarser grid is decided, we discretize the Laplacian using the same discrete stencils as presented in section 3.

A particularly good choice of the coarse-grid operator for the $O(h^4)$ accuracy is to employ the $O(h^4)$ long stencil only along the noncoarsened dimensions of the grid and to discretize with the $O(h^2)$ stencil on the coarse grids along the dimensions where partial coarsening takes place. This has the marked advantage of saving CPU time as now the coarse-grid operator has increased sparsity. Moreover, on very coarse grids this is advantageous, because at grid points adjacent to boundary points the long stencil cannot be applied since it has entries which lie outside the discrete domain, whereas the $O(h^2)$ discretization can be applied throughout the domain. The overall accuracy remains fourth-order as we have the fourth-order accuracy on the finest grid. This coarse-grid discretization scheme fits very nicely with the numerical experiments, and 2 [illegible] 3 [illegible] (not shown here) confirm this.

We do not use the Galerkin operator because of its disadvantage of being usually more dense than the simple coarse-grid analog of the fine-grid operator (unless special transfer operators are employed to generate the coarse-grid operators). In high-d dimensions this disadvantage becomes more serious and impractical.

4.4. The transfer operators. We employ the d-dimensional analogs of the full-weighting (FW) restriction operator and of the bilinear interpolation operator in two dimensions for the intergrid transfers of the grid functions. In this section we present a tensor formulation to generate the restriction and prolongation operator matrices.

For completeness we first mention [14] that a $2d$ FW restriction operator

$$I_h^{2h} \triangleq \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_h^{2h}$$

is the Kronecker tensor product of the following x_1 and x_2 directional 1-dimensional FW operators:

$$(I_h^{2h})_{x_1} \triangleq \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \end{bmatrix}, \quad (I_h^{2h})_{x_2} \triangleq \frac{1}{4}\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}.$$

A formula based on Kronecker tensor products for building up a FW restriction operator matrix $\mathbf{R}$ reads

$$\mathbf{R} = \prod_{i=1}^{d} (\mathbf{R}_i)^{k_i}, \tag{4.2}$$

$$(\mathbf{R}_i)^{k_i} = \prod_{l=0}^{k_i-1} \left[\bigotimes_{j=i}^{d-1} \mathbf{I}_{N_{(d+i-j)}} \otimes \mathbf{O}_{\left[N_i/2^{(k_i-l-1)}\right]} \otimes \bigotimes_{j=1}^{i-1} \mathbf{I}_{\left[N_{(i-j)}/2^{k_{(i-j)}}\right]} \right].$$

We now define the quantities involved in (4.2) for the dummy subscript a.

$\mathbf{I}_a$ is the identity matrix of order $(a-1)\times(a-1)$.

$\mathbf{O}_a$ is the $1d$ FW restriction operator matrix, order $= \left(\frac{a}{2}-1\right)\times(a-1)$.

$\mathbf{G} = [N_1, N_2, \ldots, N_d]$, as in (3.6).

$\mathbf{T} = [k_1, k_2, \ldots, k_d]$ is the coarsening request; k_i is the count of $(h \to 2h)$ transfers along the ith dimension. We say that [illegible] takes place along the ith dimension if $k_i = 2$. For reasons of space it is not possible to verify (in this manuscript) (4.2) for any realistic example; however, it is trivial to verify the same with a matrix manipulation software package.

Once the FW restriction operator matrix in d dimensions is set, the prolongation (d-linear interpolation) operator matrix can be obtained by the following relation:

$$\mathbf{P} = 2^{(\sum_{i=1}^{d} k_i)}(\mathbf{R}^T). \tag{4.3}$$

A generalized restriction operator (4.2) gives us the freedom to experiment with different types of coarsening strategies depending on the grid. Note that the FW restriction operator given by (4.2) provides the required matrix for any number of coarsenings along any number of dimensions for an abstract d-dimensional problem.

4.5. Computational work for *d*-multigrid. The practical feasibility of a d-multigrid method also has to take into account an estimate of the computational work that it involves. Following the notation from [11, 14], we call this work estimate W_l, assuming that a particular multigrid method is based on a hierarchy of grids $(\Omega_l, \Omega_{l-1}, \ldots, \Omega_1, \Omega_0)$, where Ω_0 is the coarsest grid. The [illegible] W_l per multigrid cycle on Ω_l is modeled by the recursion:

$$W_1 = W_1^0 + W_0, \qquad W_k = W_k^{k-1} + \gamma_{k-1} W_{k-1} \quad (k = 1, 2, \ldots, l). \tag{4.4}$$

W_k^{k-1} is the work estimate for a single 2-grid cycle (h_k, h_{k-1}) excluding the work needed to solve the defect equation on $\Omega_{h_{k-1}}$, and W_0 is the work required to solve

exactly on the coarsest grid. It is reasonable to assume that the multigrid components (relaxation, computation of defect, and the transfer of grid functions) applied to a single unknown require a number of arithmetic operations which—independent of k—is bounded by a small constant C. With M_k the total number of unknowns at grid level k, we have

$$W_k^{k-1} \leqslant CM_k \qquad (k = 1, 2, \ldots, l).$$

For a fixed cycle index γ (the number of times the coarse grids are cycled) and with the work on the coarsest grid neglected, this leads to

$$W_l \leqslant C\left[M_l + \gamma M_{l-1} + \gamma^2 M_{l-2} + \cdots + \gamma^{l-1} M_1\right]. \tag{4.5}$$

Before we proceed further, we consider it important to point out at this stage that the coarsening strategies (algorithms) that we use do not employ the same transfers at all grid levels. Depending on the anisotropy of the system, strategy 1 employs $(h \to 2h)$ transfers (along the strongly coupled dimensions) so that after application of this strategy at each level the grid comes closer to isotropy. When an acceptable isotropy is achieved strategy 1 resorts to standard full coarsening until the grid is brought to the coarsest possible level which the particular discretization allows. Strategy 2 is [illegible] in the sense that it is composed of a mixture of standard and quadrupling transfers; moreover, it is [illegible] just like strategy 1. In strategy 2 we employ full $(h \to 2h)$ transfers when isotropy is achieved, because quadrupling along all dimensions did not prove good even with the use of optimal relaxation parameters. Note that, for the type of grid-induced anisotropy encountered in the sparse-grid solution method, the number of dimensions along which the grid is coarsened increases (or remains constant in the case of an equidistant grid) at each successive level. This gives the following:

$$\begin{aligned} W_l &\leqslant \frac{2^{j\cdot s}}{(2^{j\cdot s} - \gamma)} CM_l, \\ W_l &\leqslant \frac{\tau}{(\tau - \gamma)} CM_l \qquad \text{for} \quad \gamma < \tau = 2^{j\cdot s}. \end{aligned} \tag{4.6}$$

Here j represents the number of dimensions coarsened, $s = 1$ for $(h \to 2h)$ transfers, and $s = 2$ for quadrupling. The worst case would be when the grid is highly stretched along a single dimension, which implies that coarsening takes place only along one dimension, so $j = 1$. With quadrupling, i.e., $s = 2$, this still renders $\tau = 4$, which implies that the complexity of the method is still $O(M_l)$ for $\gamma = 1, 2, 3$. This feature makes quadrupling particularly attractive in higher dimensions for non-equidistant grids. Strategy 1 also gives an $O(M_l)$ algorithm for $\gamma = 1, 2, 3$ as long as $j \geq 2$, whereas for $j = 1$ one has to apply a V-cycle.

If the problem is isotropic with an equidistant grid, we essentially have $s = 1$ and $j = d$, which gives the work estimates that appear in Table 4.1.

The $2d$ and $3d$ results are well known [11], and an estimate for a general d (isotropic problem on an equidistant grid) can be obtained by setting $j = d$ in (4.6).

This leads us now to apply the standard multigrid procedure to our model problem as the multigrid components have been adjusted for a general high-dimensional setting, along with a computational complexity estimate. A LFA of the ω-RB Jacobi method follows, through an implementation of which we derive the relaxation parameters for our experiments.

TABLE 4.1
Multigrid work estimates for the isotropic problem on equidistant grids.

γ	$d=2$	$d=3$	$d=4$	$d=5$	$d=6$
1	$\frac{4}{3}\ CM_l$	$\frac{8}{7}\ CM_l$	$\frac{16}{15}\ CM_l$	$\frac{32}{31}\ CM_l$	$\frac{64}{63}\ CM_l$
2	$2\ CM_l$	$\frac{4}{3}\ CM_l$	$\frac{8}{7}\ CM_l$	$\frac{16}{15}\ CM_l$	$\frac{32}{31}\ CM_l$
4	$O(M_l\ \log_2\ M_l)$	$2\ CM_l$	$\frac{4}{3}\ CM_l$	$\frac{8}{7}\ CM_l$	$\frac{16}{15}\ CM_l$

5. Local Fourier smoothing analysis of the ω-RB Jacobi method. LFA [2] is a tool for analyzing the convergence behavior of multigrid methods. The finer details including the validity of LFA along with related theorems and proofs are presented in [15]. In this section we concentrate on the development of a Fourier representation for the ω-RB Jacobi method applied to the model operator from section 3.1, the definition of the [illegible], the evaluation of relaxation parameters, and extension to a general d-dimensional setting. We assume familiarity with the basics of this tool; see [11, 12, 14, 15, 17] for the same. For the most part in this section we carry on the definitions and notations as present in [14, 17, 18, 19].

Consider the d-dimensional anisotropic diffusion operator

$$-\sum_{i=1}^{d} \varepsilon_i \frac{\partial^2}{\partial x_i^2}, \quad \text{with} \quad \varepsilon_i > 0. \tag{5.1}$$

With respect to the analysis it is more convenient to scale the real positive coefficients ε_i as in [18, 19] and to replace (5.1) by

$$-\sum_{i=1}^{d} c_i \frac{\partial^2}{\partial x_i^2}, \quad \text{with} \quad c_i = \varepsilon_i / \sum_{j=1}^{d} \varepsilon_j \quad \text{and hence} \quad \sum_{i=1}^{d} c_i = 1. \tag{5.2}$$

The partial derivatives are discretized by second- or fourth-order differences (see (3.3), (3.4)) leading to the discrete counterparts of (5.2) denoted by L_h^{2o} and L_h^{4o}, respectively. In contrast to section 3.1 we assume that we are dealing with the same mesh size in each space direction, i.e., $h = h_1 = \cdots = h_d$. Here possible anisotropies caused by different mesh sizes are modeled by varying the coefficients c_i, which is more appropriate for the analysis.

LFA takes into account only the [illegible] nature of the operator. Thus for an effective analysis through this tool one has to have constant coefficients, and one has to neglect the boundary effects. More precisely, all occurring operators are extended to an infinite grid

$$\mathbf{G}_h := \left\{ \boldsymbol{x} = (x_1, \ldots, x_d)^T = \boldsymbol{\kappa} h = h\,(\kappa_1, \ldots, \kappa_d)^T \ :\ \boldsymbol{\kappa} \in \mathbb{Z} \right\}, \tag{5.3}$$

with mesh size h. The discrete eigenfunctions ([illegible]) of the resulting infinite grid operators serve as fundamental quantities in LFA; they are obviously given by

$$\varphi(\boldsymbol{\theta}, \boldsymbol{x}) = e^{i\boldsymbol{\theta}\boldsymbol{x}/h}, \quad \boldsymbol{x} \in \mathbf{G}_h. \tag{5.4}$$

For this analysis we assume that $\boldsymbol{\theta}$ varies continously in $\mathbb{R}^d$. As 2π is the period of φ, we are led to the identity:

$$\varphi(\boldsymbol{\theta}, \boldsymbol{x}) \equiv \varphi(\boldsymbol{\theta}', \boldsymbol{x}) \qquad \text{for } \boldsymbol{x} \in \mathbf{G}_h \quad \text{iff } \boldsymbol{\theta} = \boldsymbol{\theta}' (\text{mod } 2\pi), \tag{5.5}$$

where this difference—of multiples of 2π—is between all of the components of the d-tuples, $(\boldsymbol{\theta} \ \& \ \boldsymbol{\theta}')$; thus it suffices to consider these functions only for $\boldsymbol{\theta} \in \Theta = [-\pi, \pi)^d$.

The corresponding eigenvalues ( ) of L_h^{2o} and L_h^{4o} read

$$\widetilde{L}_h^{2o}(\boldsymbol{\theta}) = \frac{2}{h^2}\left(1 - \sum_{i=1}^{d} c_i \cos(\theta_i)\right) \quad \text{and}$$

$$\widetilde{L}_h^{4o}(\boldsymbol{\theta}) = \frac{1}{6h^2}\left(15 - \sum_{i=1}^{d} c_i \left(16\cos(\theta_i) - \cos(2\theta_i)\right)\right),$$

respectively.

5.1. High and low Fourier frequencies. If full coarsening ($\boldsymbol{H} = 2\boldsymbol{h}$ or $\boldsymbol{H} = 4\boldsymbol{h}$ in the case of quadrupling) is selected, the Fourier components $\varphi_h(\boldsymbol{\theta}, \boldsymbol{x})$, with $|\boldsymbol{\theta}| := \max\{|\theta_1|, \ldots, |\theta_d|\} \le \pi/2$ (or $\le \pi/4$), are also visible on the coarse grid $\mathbf{G}_H$, whereas components with $|\boldsymbol{\theta}| > \pi/2$ ($> \pi/4$) coincide with certain $\varphi_h(\hat{\boldsymbol{\theta}}, \boldsymbol{x})$, where $|\hat{\boldsymbol{\theta}}| \le \pi/2$ ($\le \pi/4$). This observation leads to the distinction between high and low Fourier frequencies.

DEFINITION 1 (high and low frequencies for full coarsening). θ_j $(j \in \{1, \ldots, d\})$ low

$$-\pi/2 \le \theta_j < \pi/2 \quad (\ -\pi/4 \le \theta_j < \pi/4 \ ) \quad \boldsymbol{\theta} \in \Theta.$$

high low Fourier frequency $\boldsymbol{\theta} \in \Theta$ high frequency

The distinction obviously depends on the coarsening as, for different coarsening strategies, different sets of Fourier frequencies are visible on the coarse grid. We define the set of coordinate indices by $\mathcal{J} := \{1, \ldots, d\}$. In the case of partial coarsening (compare with section 4.2), the grid is coarsened only in a subset $\{x_j \mid j \in \mathcal{J}^c \subset \mathcal{J}\}$ of the coordinate directions and remains fixed in the other coordinates x_j, with $j \in \mathcal{J}^f = \mathcal{J} \setminus \mathcal{J}^c$. Especially for two-dimensional problems, this is often called semicoarsening as well. In this case, we have for the coarse grid mesh size $\boldsymbol{H}$ that $H_j = 2h_j$ (or $H_j = 4h_j$ in the case of quadrupling) for $j \in \mathcal{J}^c$ and $H_j = h_j$ for $j \in \mathcal{J}^f$. Then the definition of high and low frequencies has to be adapted accordingly.

DEFINITION 2 (high and low frequencies for partial coarsening). $\boldsymbol{\theta} \in \Theta$ low $\theta_j \in \mathcal{I}^c$ 1 high frequency

5.2. Fourier representation of ω-RB Jacobi. ω-RB Jacobi relaxation consists of two partial steps of ω-Jacobi-type; compare with section 4.1. The iteration matrix of classical damped Jacobi relaxation is given by $\mathbf{S}_h = \mathbf{I}_h - \omega \mathbf{D}_h^{-1} \mathbf{A}_h$, where $\mathbf{I}_h$ denotes the $(M \times M)$-identity matrix and $\mathbf{D}_h$ the diagonal part of $\mathbf{A}_h$. Hence, the Fourier components remain eigenfunctions of ω-Jacobi relaxation. The Fourier symbols of ω-Jacobi relaxation applied to L_h^{2o} and L_h^{4o} read

$$A^{2o}(\boldsymbol{\theta}, \omega) = 1 - \omega \frac{h^2}{2} \widetilde{L}_h(\boldsymbol{\theta}) = 1 - \omega\left(1 - \sum_{i=1}^{d} c_i \cos(\theta_i)\right) \quad \text{and}$$

$$A^{4o}(\boldsymbol{\theta}, \omega) = 1 - \omega \frac{6h^2}{15} \widetilde{L}_h(\boldsymbol{\theta}) = 1 - \omega\left(1 - \frac{1}{15}\sum_{i=1}^{d} c_i \left(16\cos(\theta_i) - \cos(2\theta_i)\right)\right),$$

respectively; see [16, 18, 19].

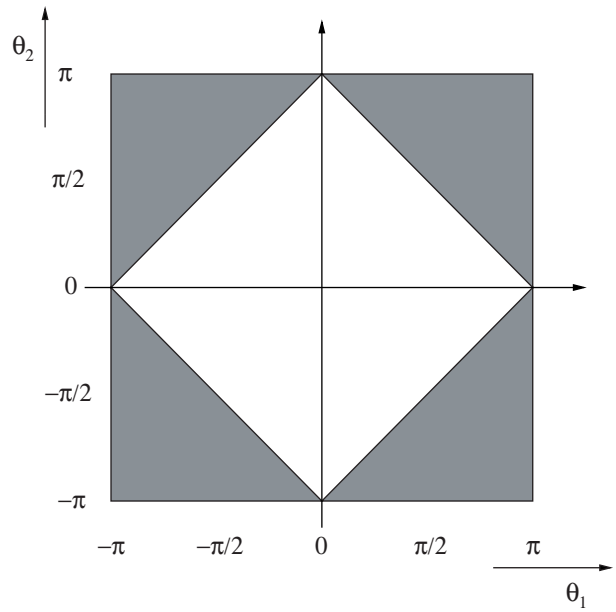


FIG. 1. *Low (white region) and high (shaded region) frequencies for red-black coarsening in two dimensions.*

Note that for pattern relaxations such as ω-RB Jacobi the Fourier components are no longer eigenfunctions. However, the minimal invariant subspaces for ω-RB Jacobi are two-dimensional; see [18]. Each component $\varphi_h(\boldsymbol{\theta}, \boldsymbol{x})$ is coupled only with $\varphi_h(\widehat{\boldsymbol{\theta}}, \boldsymbol{x})$, where

$$\widehat{\theta}_j := \theta_j - \operatorname{sign}(\theta_j)\,\pi, \quad \text{with} \quad j \in \mathcal{J}.$$

Note that ω-RB Jacobi couples those Fourier components that alias on the coarse grid in the case of red-black coarsening. Considering red-black coarsening, the set of low frequencies (compare with Definitions 1 and 2) is given by

$$\Theta_{RB} := \left\{ \boldsymbol{\theta} \in \Theta \,:\, |\boldsymbol{\theta}| := \sum_{i=1}^{d} |\theta_i| \le \frac{d}{2}\pi \right\}. \tag{5.6}$$

For $d = 2$, the set of low frequencies Θ_{RB} is illustrated in Figure 1.

We decompose the set of Fourier frequencies Θ into a direct sum of such two-dimensional subspaces that are coupled by ω-RB Jacobi:

$$\Theta = \bigoplus_{\boldsymbol{\theta} \in \Theta_{RB}} \left\{ \boldsymbol{\theta}, \widehat{\boldsymbol{\theta}} \right\}.$$

The Fourier representations of the half-sweep operators which represent the smoothing steps over the red points (R) and the black points (B) w.r.t. the two-dimensional

minimal invariant subspaces are given in [19]. They read

$$\widetilde{S}_h^R(\boldsymbol{\theta},\omega) = \frac{1}{2}\begin{pmatrix} A(\boldsymbol{\theta},\omega)+1 & A(\widehat{\boldsymbol{\theta}},\omega)-1 \\ A(\boldsymbol{\theta},\omega)-1 & A(\widehat{\boldsymbol{\theta}},\omega)+1 \end{pmatrix}$$

$$\text{and} \qquad \widetilde{S}_h^B(\boldsymbol{\theta},\omega) = \frac{1}{2}\begin{pmatrix} A(\boldsymbol{\theta},\omega)+1 & -A(\widehat{\boldsymbol{\theta}},\omega)+1 \\ -A(\boldsymbol{\theta},\omega)+1 & A(\widehat{\boldsymbol{\theta}},\omega)+1 \end{pmatrix},$$

$$\text{with} \quad A(\boldsymbol{\theta},\omega) = A^{2o}(\boldsymbol{\theta},\omega), A^{4o}(\boldsymbol{\theta},\omega) \quad \text{and} \quad A(\widehat{\boldsymbol{\theta}},\omega) = A^{2o}(\widehat{\boldsymbol{\theta}},\omega), A^{4o}(\widehat{\boldsymbol{\theta}},\omega).$$

The transformation matrix $\widetilde{S}_h(\boldsymbol{\theta},\omega)$ for $(\phi_h(\boldsymbol{\theta},\,.\,), \phi_h(\widehat{\boldsymbol{\theta}},\,.\,))^T$ after one relaxation step is then given by

$$\widetilde{S}_h(\boldsymbol{\theta},\omega) = \widetilde{S}_h^B(\boldsymbol{\theta},\omega)\,\widetilde{S}_h^R(\boldsymbol{\theta},\omega). \tag{5.7}$$

The four components of the (2×2)-matrix $\widetilde{S}_h(\boldsymbol{\theta},\omega) = (s_{ij})_{i,j=1,2}$ read

$$\begin{aligned} s_{11} &= \frac{1}{4}\left[(A(\boldsymbol{\theta},\omega)+1)^2 + (A(\widehat{\boldsymbol{\theta}},\omega)-1)(1-A(\boldsymbol{\theta},\omega))\right], \\ s_{12} &= \frac{1}{4}\left[(A(\boldsymbol{\theta},\omega)+1)(1-A(\widehat{\boldsymbol{\theta}},\omega)) + A(\widehat{\boldsymbol{\theta}},\omega)^2 - 1\right], \\ s_{21} &= \frac{1}{4}\left[(A(\widehat{\boldsymbol{\theta}},\omega)+1)(1-A(\boldsymbol{\theta},\omega)) + A(\boldsymbol{\theta},\omega)^2 - 1\right], \\ s_{22} &= \frac{1}{4}\left[(A(\widehat{\boldsymbol{\theta}},\omega)+1)^2 + (A(\widehat{\boldsymbol{\theta}},\omega)-1)(1-A(\boldsymbol{\theta},\omega))\right]. \end{aligned} \tag{5.8}$$

5.3. Smoothing factor. In order to measure the smoothing properties of pattern relaxation methods, we adopt the general definition of the smoothing factor from [11]. Here the "real" coarse grid correction for a two-grid method is replaced by an "ideal" coarse grid correction operator Q_h^H which annihilates the low-frequency error components and leaves the high-frequency components unchanged. Q_h^H is a projection operator onto the space of high-frequency components. The Fourier representation of the ideal coarse grid correction operator w.r.t. the subspaces $\{\phi_{\boldsymbol{h}}(\boldsymbol{\theta},\,.\,), \phi_{\boldsymbol{h}}(\widehat{\boldsymbol{\theta}},\,.\,)\}$ $(\boldsymbol{\theta}\in\Theta_{RB})$ is then given by

$$\widetilde{Q}_h^H(\boldsymbol{\theta}) = \begin{pmatrix} q(\boldsymbol{\theta}) & 0 \\ 0 & q(\widehat{\boldsymbol{\theta}}) \end{pmatrix}, \tag{5.9}$$

where $q=0$ for a low argument and 1 otherwise; compare with Definitions 1 and 2.

The smoothing factor μ of the ω-RB Jacobi operator is defined as the worst factor by which the high-frequency errors are reduced per iteration step. So with ν denoting the number of relaxation sweeps, and ρ the matrix spectral radius, we have

$$\mu(\omega) := \sup_{\boldsymbol{\theta}\in\Theta_{RB}} \left\{ \sqrt[\nu]{\rho\left(\widetilde{Q}_h^H(\boldsymbol{\theta})\widetilde{S}_h^{\nu}(\boldsymbol{\theta},\omega)\right)} \right\}. \tag{5.10}$$

The smoothing factor can easily be calculated by a computer program which allows for a numerical determination of optimal relaxation parameters for different coarsening

TABLE 5.1
$\mu(1)$, ω_{opt}, and associated $\mu(\omega_{opt})$ for $\nu = 1$ relaxation sweep.

Left: *Doubling along all dimensions, equidistant grid, isotropy,* $O(h^2)$.
Right: *Quadrupling along all dimensions, equidistant grid, isotropy,* $O(h^2)$.

Doubling ($h \to 2h$)					Quadrupling ($h \to 4h$)			
d	$\mu(1)$	ω_{opt}	$\mu(\omega_{opt})$	ω_{ub}	$\mu(\omega) = \mu(1)$	ω_{opt}	$\mu(\omega_{opt})$	ω_{ub}
2	0.25	1.049	0.16	1.072	0.73	1.315	0.31	1.316
3	0.44	1.133	0.23	1.144	0.81	1.398	0.40	1.393
4	0.56	1.195	0.28	1.202	0.86	1.454	0.45	1.455
5	0.64	1.243	0.31	1.250	0.89	1.496	0.50	1.502
6	0.69	1.283	0.35	1.285	0.90	1.528	0.53	1.519

Left: *Doubling along 1 dimension, non-equidistant grid,* $128 \times 32^{(d-1)}$, $O(h^2)$.
Right: *Quadrupling along 1 dimension, non-equidistant grid,* $128 \times 32^{(d-1)}$, $O(h^2)$.

Doubling ($h \to 2h$)					Quadrupling ($h \to 4h$)			
d	$\mu(1)$	ω_{opt}	$\mu(\omega_{opt})$	ω_{ub}	$\mu(\omega) = \mu(1)$	ω_{opt}	$\mu(\omega_{opt})$	ω_{ub}
2	0.125	0.946	0.08	1.033	0.52	1.182	0.22	1.181
3	0.125	0.963	0.10	1.033	0.55	1.190	0.25	1.197
4	0.125	0.980	0.11	1.033	0.57	1.199	0.27	1.207
5	0.125	0.997	0.122	1.033	0.59	1.209	0.29	1.219
6	0.15	1.013	0.13	1.04	0.60	1.219	0.30	1.225

strategies and both types of discretization under consideration. We would like to emphasize that the optimal relaxation parameter depends on the number of relaxation steps ν as can be seen from (5.10).

Note that there are explicit (but very lengthy) analytical formulas for the optimal relaxation parameters ω_{opt} in the case of full coarsening applied to the second-order discretization [16, 19]. Moreover, there is a close to optimal upper bound ω_{ub} [19] for the optimal relaxation parameters which is given by the following handy expression:

$$\omega_{opt} < \omega_{ub} = \frac{2}{1 + \sqrt{1 - \mu(\omega = 1)}}.$$

For partial coarsening and especially for the fourth-order discretization, it seems to be very difficult to derive analytic expressions for ω_{opt}. However, for the second-order discretization it turned out that ω_{ub} is a satisfactory approximation for ω_{opt} even in the case of partial coarsening and quadrupling (but not necessarily an upper bound anymore); see Table 5.1, which presents ω_{opt} (optimized for $\nu = 1$) and ω_{ub} for equidistant and non-equidistant grids. This is a nice generalization of the results from [19]. For the fourth-order discretization this is no longer true, and we have to stay with the numerically derived values.

From the values in Table 5.2 (optimized for $\nu = 2$), we see that the [illegible] of using optimal relaxation parameters is more prominent and pronouced with strategy 2. With strategy 1 this enhancement becomes prominent in the case of [illegible] (with grids that are equidistant along $(d-2)$ or more dimensions). With anisotropic problems where anisotropy is induced by discretization on grids highly elongated along a single dimension (and dealt with strategy 1), the choice $\omega = 1$ is more suitable—first—because the optimal values themselves are very close to 1 and therefore do not bring about a substantial enhancement—and second—the cost of relaxation itself is cut down, which saves CPU time.

TABLE 5.2
$\mu(1)$, ω_{opt}, *and associated* $\mu(\omega_{opt})$ *for* $\nu = 2$ *relaxation sweeps.*

Left: *Doubling along all dimensions, equidistant grid, isotropy,* $O(h^2)$.
Right: *Quadrupling along all dimensions, equidistant grid, isotropy,* $O(h^2)$.

Doubling ($h \to 2h$)				Quadrupling ($h \to 4h$)		
d	$\mu(\omega) = \mu(1)$	ω_{opt}	$\mu(\omega_{opt})$	$\mu(\omega) = \mu(1)$	ω_{opt}	$\mu(\omega_{opt})$
2	0.25	1.0107	0.23	0.73	1.3062	0.39
3	0.44	1.1136	0.28	0.81	1.3928	0.46
4	0.56	1.1832	0.31	0.86	1.4507	0.50
5	0.64	1.2356	0.35	0.89	1.4934	0.53
6	0.69	1.2771	0.37	0.90	1.5266	0.56

Left: *Doubling along 1 dimension, non-equidistant grid,* $128 \times 32^{(d-1)}$, $O(h^2)$.
Right: *Quadrupling along 1 dimension, non-equidistant grid,* $128 \times 32^{(d-1)}$, $O(h^2)$.

Doubling ($h \to 2h$)				Quadrupling ($h \to 4h$)		
d	$\mu(\omega) = \mu(1)$	ω_{opt}	$\mu(\omega_{opt})$	$\mu(\omega) = \mu(1)$	ω_{opt}	$\mu(\omega_{opt})$
2	0.23	0.8490	0.18	0.52	1.1598	0.30
3	0.23	0.8682	0.19	0.55	1.1735	0.31
4	0.23	0.8858	0.19	0.57	1.1864	0.32
5	0.23	0.9022	0.20	0.59	1.1986	0.32
6	0.23	0.9174	0.20	0.60	1.2100	0.33

Left: *Doubling along all dimensions, equidistant grid, isotropy,* $O(h^4)$.
Right: *Quadrupling along all dimensions, equidistant grid, isotropy,* $O(h^4)$.

Doubling ($h \to 2h$)				Quadrupling ($h \to 4h$)		
d	$\mu(\omega) = \mu(1)$	ω_{opt}	$\mu(\omega_{opt})$	$\mu(\omega) = \mu(1)$	ω_{opt}	$\mu(\omega_{opt})$
2	0.28	1.0260	0.25	0.76	1.3110	0.40
3	0.46	1.1108	0.29	0.84	1.3782	0.47
4	0.57	1.1683	0.33	0.87	1.4238	0.52
5	0.65	1.2128	0.36	0.90	1.4579	0.55
6	0.70	1.2492	0.38	0.91	1.4847	0.58

Left: *Doubling along 1 dimension, non-equidistant grid,* $128 \times 32^{(d-1)}$, $O(h^4)$.
Right: *Quadrupling along 1 dimension, non-equidistant grid,* $128 \times 32^{(d-1)}$, $O(h^4)$.

Doubling ($h \to 2h$)				Quadrupling ($h \to 4h$)		
d	$\mu(\omega) = \mu(1)$	ω_{opt}	$\mu(\omega_{opt})$	$\mu(\omega) = \mu(1)$	ω_{opt}	$\mu(\omega_{opt})$
2	0.24	0.8859	0.20	0.59	1.1900	0.35
3	0.26	0.8957	0.21	0.61	1.1942	0.37
4	0.25	0.9122	0.22	0.62	1.2046	0.38
5	0.25	0.9268	0.22	0.64	1.2146	0.38
6	0.25	0.9399	0.23	0.65	1.2243	0.39

6. Numerical experiments. We now present the results of some of our numerical experiments. The emphasis is on grid-induced anisotropies. The PDE is discretized on various equidistant and non-equidistant grids, and the spectral radius ρ of the multigrid iteration operator M_h is presented, which represents the asymptotic convergence factor. During the course of an experiment the only quantity available to estimate this factor is the defect d_h^i after the ith multigrid cycle. The numerical results—presented in Tables 6.1–6.4—depict a close match between the theoretical smoothing factors μ (computed for coarsening based on the finest grid) and an emperical estimate of the multigrid convergence factor $\rho(M_h)$, which we denote by q^m and define as

$$q^m := \frac{\| d_h^m \|}{\| d_h^{m-1} \|},$$

where m represents the number of iterations or multigrid cycles that the discrete problem takes to converge to the numerical solution.

All of the experiments employ one pre- and one postsmoothing, and so the smoothing factor is displayed as a square for a correspondence with q^m; compare with (5.10). The optimal relaxation parameters ω_{opt} that we employ are computed for $\nu = 2$. For each order of accuracy and for each dimension, we have chosen two kinds of grid, one equidistant and one non-equidistant (highly stretched in one dimension). q^m is displayed against the number of multigrid cycles that the experiment took to converge to the tolerance value, which for all of the experiments is 10^{-10}, i.e.,

$$\frac{||\mathbf{b}_h - \mathbf{A}_h \mathbf{u}_h^m||}{||\mathbf{b}_h||} \mid \leq tol.$$

This termination criterion is equivalent to the one based on relative residual reduction because our starting soluton in these experiments is always an all zero vector. Moreover this criterion corresponds to a residual reduction by ≈ 7 orders of magnitude. Through the numerical solution of the model problem we approximate the following test function:

$$u(\mathbf{x}) = \frac{\sum_{i=1}^{d} \sin(d\pi^2 x_i)}{d\pi + \sum_{i=1}^{d} x_i}, \tag{6.1}$$

$\varepsilon_i = 1 \ \forall \ i$, and $\Omega = (0,1)^d$ for all of the problems. The values of this function at the boundary are taken as Dirichlet boundary conditions, and its analytically computed Laplacian forms the source function for these experiments. The experiments include the V- and the W-cycles. In some of the experiments the grids used for the $O(h^2)$ operator are different from the ones for the $O(h^4)$ operator. This serves only the purpose of accumulating results for slightly [illegible] experiments. C_2^2 indicates the use of the second-order stencil along all coarsened and noncoarsened dimensions; likewise for C_4^4. C_2^4 indicates the use of the $O(h^4)$ long stencil along all noncoarsened dimensions and the use of the $O(h^2)$ stencil on the coarse grids along the dimensions where coarsening takes place. This [illegible] coarse grid discretization gives fourth-order accuracy and converges [illegible] than the conventional fourth-order long stencil.

Table 6.1 presents experimental results for equidistant grids. A comparison of the convergence results with and without optimal relaxation parameters indicates the benefits of using them for high-dimensional problems. The cutdown in the multigrid convergence factor as well as in the number of multigrid cycles (required to converge to tol) is quite significant for $d \geq 3$.

Table 6.2 presents experimental results for non-equidistant grids which we have chosen to be highly stretched in only one dimension. Because of this characteristic these experiments are computationally more expensive than any other as coarsening takes place only along the elongated dimension. This is exactly the opposite of the previous case, where coarsening took place along all dimensions, and hence the cutdown in the number of unknowns at each level was optimal. In this table we display the results that we get from strategy 1, which is based purely on $h \rightarrow 2h$ transfers. Optimal relaxation parameters in this case pay off only with V-cycles, with $\omega = 1$ serving as a perfect compromise with W-cycles.

In Table 6.3 we have reworked the experiments of Table 6.2 but with strategy 2 this time. Partial quadrupling (strategy 2) ensures an $O(M_l)$ algorithm even with grids of this type. Note that the computational complexity of the experiments in Table 6.2 is $O(M_l \ \log_2 \ M_l)$, even though the multigrid convergence factor is quite impressive

TABLE 6.1

Results of numerical experiments for $V(1,1)$ and $W(1,1)$ on equidistant grids. The observed convergence rate with the number of iterations, i.e., $(q^m/\#\ it.)$, is presented. For a correspondence comparison the smoothing factors $[\mu(1)]^2$ and $[\mu(\omega_{opt})]^2$ (computed for coarsening based on the finest grid) are also displayed. Results presented include experiments with $\omega = 1$ as well as $\omega = \omega_{opt}$.

$G = 128^2$	V	W	V	W
	$[\mu(1)]^2 = 0.06$		$[\mu(1.011)]^2 = 0.05$	
$O(h^2)C_2^2$	0.10/8	0.06/7	0.09/8	0.05/7
$G = 128^2$	$[\mu(1)]^2 = 0.08$		$[\mu(1.026)]^2 = 0.06$	
$O(h^4)C_4^4$	0.13/9	0.10/8	0.12/9	0.08/7
$O(h^4)C_2^4$	0.10/8	0.07/7	0.09/8	0.05/7
$G = 128^3$	V	W	V	W
	$[\mu(1)]^2 = 0.20$		$[\mu(1.114)]^2 = 0.08$	
$O(h^2)C_2^2$	0.22/11	0.18/10	0.12/9	0.07/7
$G = 64^3$	$[\mu(1)]^2 = 0.21$		$[\mu(1.111)]^2 = 0.08$	
$O(h^4)C_4^4$	0.26/12	0.22/11	0.16/10	0.09/8
$O(h^4)C_2^4$	0.24/12	0.21/11	0.13/9	0.07/7
$G = 64^4$	V	W	V	W
	$[\mu(1)]^2 = 0.32$		$[\mu(1.183)]^2 = 0.10$	
$O(h^2)C_2^2$	0.33/14	0.30/12	0.16/10	0.08/7
$G = 32^4$	$[\mu(1)]^2 = 0.33$		$[\mu(0.168)]^2 = 0.11$	
$O(h^4)C_4^4$	0.39/16	0.34/14	0.20/10	0.11/9
$O(h^4)C_2^4$	0.35/15	0.34/14	0.15/9	0.11/8
$G = 16^5$	V	W	V	W
	$[\mu(1)]^2 = 0.41$		$[\mu(1.236)]^2 = 0.12$	
$O(h^2)C_2^2$	0.38/16	0.38/15	0.18/10	0.09/8
$G = 8^6$	V	W	V	W
	$[\mu(1)]^2 = 0.48$		$[\mu(1.277)]^2 = 0.14$	
$O(h^2)C_2^2$	0.35/15	0.34/15	0.12/9	0.11/9

there. These results show the important role of optimal relaxation parameters in enhancing convergence of the multigrid with quadrupling transfers.

To make the discussion complete we have included some more experiments on a [illegible]. The results are reported in Figure 2. The convergence report is depicted against the iteration scale as well as against the CPU time scale. A comparison of the results of strategies 1 and 2 suggests that, for these kind of grids, a combination of strategy 2 with V-cycles and optimal relaxation parameters works very nicely.

A graphical presentation of the convergence behavior of an isotropic 4-dimensional multigrid experiment appears in Figure 3. The defect reduction is displayed against the iteration and the CPU time scale. To emphasize the inefficiency of full quadrupling with completely equidistant grids, we have included the results against full doubling. Clearly enough, here standard coarsening is the strategy of choice. This is the main reason why we do not keep quadrupling transfers in strategy 2, once grid equidistance has been achieved, but rather resort to doubling from this stage onwards. Quadrupling along all dimensions at the same time hampers the multigrid convergence rate as well as increasing the overall computation time.

Finally, we have performed two general 5-dimensional anisotropic experiments. For the first experiment the anisotropy comes only from discretization on a non-equidistant grid; and the results appear in Figure 4, the sequence of grids that we get

TABLE 6.2

Results of numerical experiments for $V(1,1)$ and $W(1,1)$ with strategy 1 on grids stretched along 1 dimension. The observed convergence rate with the number of iterations, i.e., (q^m/# it.), is presented. For a correspondence comparison the smoothing factors $[\mu(1)]^2$ and $[\mu(\omega_{opt})]^2$ (computed for coarsening based on the finest grid) are also displayed. Results presented include experiments with $\omega = 1$ as well as $\omega = \omega_{opt}$.

$G = 512 \times 32$	V	W	V	W
	$[\mu(1)]^2 = 0.05$		$[\mu(0.835)]^2 = 0.03$	
$O(h^2)C_2^2$	0.06/8	0.003/4	0.06/8	0.03/6
$G = 512 \times 32$	$[\mu(1)]^2 = 0.06$		$[\mu(0.879)]^2 = 0.04$	
$O(h^4)C_4^4$	0.10/8	0.03/6	0.09/8	0.07/7
$O(h^4)C_2^4$	0.10/7	0.02/6	0.05/7	0.04/6
$G = 512 \times 32^2$	V	W	V	W
	$[\mu(1)]^2 = 0.05$		$[\mu(0.835)]^2 = 0.03$	
$O(h^2)C_2^2$	0.20/11	0.005/4	0.12/9	0.03/6
$G = 128 \times 32^2$	$[\mu(1)]^2 = 0.06$		$[\mu(0.978)]^2 = 0.06$	
$O(h^4)C_4^4$	0.24/11	0.04/6	0.17/9	0.05/6
$O(h^4)C_2^4$	0.17/10	0.04/6	0.11/8	0.04/6
$G = 128 \times 8^3$	V	W	V	W
	$[\mu(1)]^2 = 0.05$		$[\mu(0.836)]^2 = 0.03$	
$O(h^2)C_2^2$	0.20/11	0.007/4	0.11/8	0.04/5
$G = 128 \times 32^3$	$[\mu(1)]^2 = 0.06$		$[\mu(1.9121)]^2 = 0.05$	
$O(h^4)C_4^4$	0.33/13	0.04/6	0.21/9	0.06/6
$O(h^4)C_2^4$	0.27/12	0.05/6	0.15/9	0.03/6
$G = 128 \times 8^4$	V	W	V	W
	$[\mu(1)]^2 = 0.05$		$[\mu(0.837)]^2 = 0.03$	
$O(h^2)C_2^2$	0.24/12	0.009/4	0.11/8	0.04/6
$G = 128 \times 8^5$	V	W	V	W
	$[\mu(1)]^2 = 0.05$		$[\mu(0.837)]^2 = 0.03$	
$O(h^2)C_2^2$	0.27/13	0.009/5	0.12/9	0.04/6

with both coarsening strategies are displayed in section 4.2, (4.1). It is quite apparent that the use of optimal relaxation parameters pays off significantly well with V-cycles in combination with strategy 1 ($\omega = 1$ serving as a perfect choice for W-cycles) and with both V- and W-cycles with strategy 2. The second experiment is more general in that it has anisotropic coefficients ε_i in the PDE, it is discretized on the same non-equidistant finest grid, and the domain dimensions are not exactly unit. The results are displayed in Table 6.4. Clearly, V-cycles with optimal weighting in the relaxation process coupled with either coarsening strategy give excellent results. We would like to point out that we expect strategy 1 to be more dependable than strategy 2 if the number of dimensions coarsened on the finest grid is greater than $d/2$.

We have used Matlab as our testing platform; the same experiments implemented in C (in more optimized computer programs) are expected to render smaller CPU times than those presented.

. . . . 1. The multigrid convergence factors displayed in the tables are mostly under 0.1, implying that a full multigrid algorithm starting on the coarsest grid is expected to reach an approximate solution up to the discretization accuracy in just one or two cycles.

. . . . 2. It is also important to point out that, for very coarse discretization grids (say, 8 points along all dimensions), the asymptotic convergence of the relaxation

TABLE 6.3

Results of numerical experiments for $V(1,1)$ and $W(1,1)$ with strategy 2 on grids stretched along 1 dimension. The observed convergence rate with the number of iterations, i.e., $(q^m/\#\text{ it.})$, is presented. For a correspondence comparison the smoothing factors $[\mu(1)]^2$ and $[\mu(\omega_{opt})]^2$ (computed for coarsening based on the finest grid) are also displayed. Results presented include experiments with $\omega = 1$ as well as $\omega = \omega_{opt}$.

$G = 512 \times 32$	V	W	V	W
	$[\mu(1)]^2 = 0.25$		$[\mu(1.147)]^2 = 0.09$	
$O(h^2)C_2^2$	0.27/13	0.24/11	0.14/10	0.08/9
$G = 512 \times 32$	$[\mu(1)]^2 = 0.32$		$[\mu(1.190)]^2 = 0.12$	
$O(h^4)C_4^4$	0.34/14	0.31/13	0.20/10	0.13/8
$O(h^4)C_2^4$	0.31/13	0.30/13	0.12/9	0.09/7
$G = 512 \times 32^2$	V	W	V	W
	$[\mu(1)]^2 = 0.25$		$[\mu(1.147)]^2 = 0.09$	
$O(h^2)C_2^2$	0.30/13	0.24/11	0.17/13	0.08/10
$G = 128 \times 32^2$	$[\mu(1)]^2 = 0.37$		$[\mu(1.194)]^2 = 0.14$	
$O(h^4)C_4^4$	0.34/14	0.35/14	0.22/10	0.13/9
$O(h^4)C_2^4$	0.34/14	0.35/14	0.16/9	0.12/8
$G = 128 \times 8^3$	V	W	V	W
	$[\mu(1)]^2 = 0.25$		$[\mu(1.148)]^2 = 0.09$	
$O(h^2)C_2^2$	0.32	0.24/11	0.16/13	0.08/10
$G = 128 \times 32^3$	$[\mu(1)]^2 = 0.39$		$[\mu(1.205)]^2 = 0.14$	
$O(h^4)C_4^4$	0.38/16	0.37/15	0.24/11	0.14/9
$O(h^4)C_2^4$	0.38/16	0.36/14	0.22/11	0.12/8
$G = 128 \times 8^4$	V	W	V	W
	$[\mu(1)]^2 = 0.26$		$[\mu(1.149)]^2 = 0.09$	
$O(h^2)C_2^2$	0.35/15	0.24/11	0.17/13	0.08/10
$G = 128 \times 8^5$	V	W	V	W
	$[\mu(1)]^2 = 0.26$		$[\mu(1.150]^2 = 0.09$	
$O(h^2)C_2^2$	0.38/16	0.24/11	0.18/13	0.08/10

TABLE 6.4

A general 5d experiment on a non-equidistant grid $G = [32\ \ 8\ \ 8\ \ 128\ \ 32]$. The domain $\Omega = (0.11, 1.21) \times (0.5, 1.51) \times (0.25, 1.26) \times (0, 1) \times (0, 1.11)$, and constant coefficients $\boldsymbol{c} = [200.33, 10^{-7}, 2, 1.5, 200.49]$.

Strategy 1	V	W	V	W
	$[\mu(1)]^2 = 0.08$		$[\mu(1.031)]^2 = 0.058$	
$O(h^2)C_2^2$	0.13/11	0.07/9	0.08/9	0.058/8
Strategy 2	V	W	V	W
	$[\mu(1)]^2 = 0.55$		$[\mu(1.319)]^2 = 0.16$	
$O(h^2)C_2^2$	0.53/31	0.53/31	0.16/11	0.14/11

method ω-RB Jacobi is also quite satisfactory. To confirm this, we conducted a 6-dimensional asymptotic convergence experiment without a hierarchy of multiple grids and measured the number of iterations and the time taken to reach the tolerance (10^{-10}). Though the use of multigrid cuts down the number of iterations by around a factor of 2.5, the time taken to meet the tolerance level is the same.

7. Conclusion. The central emphasis of this paper is on multigrid techniques for high-dimensional elliptic PDEs. To alleviate the implementation issues we have

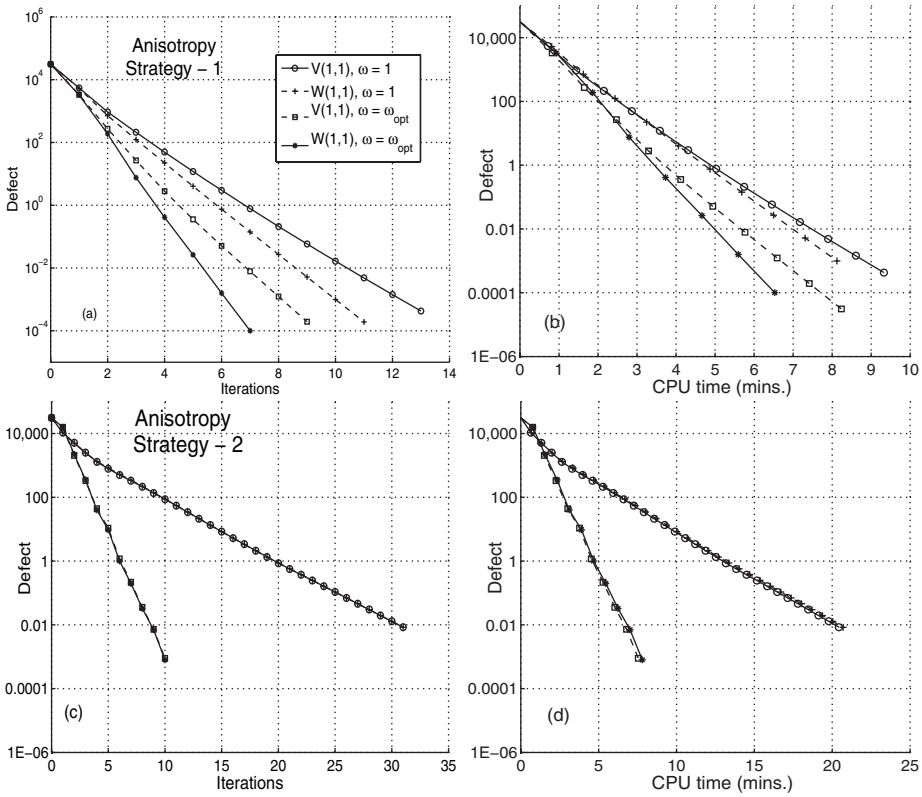


FIG. 2. *Convergence behavior of different multigrid cycles for a* 4*d problem, on a non-equidistant grid* [64 64 64 16] *(anisotropy with* 3,750,705 *unknowns), left side defect vs. iterations, right side defect vs. CPU time.*

shown how difference operator matrices can be put together through tensor methods. This renders testing in abstract higher d dimensions easy. Of course, tailored routines (for specific applications) can be programmed and tuned as per requirement. The main idea has been to demonstrate how the multigrid convergence factor can be reduced efficiently with appropriate coarsening strategies combined with the use of relaxation parameters in the smoothing process. Through numerical results supported by the LFA we have shown that partial quadrupling is a strategy of choice in higher dimensions and ensures a computational complexity of $O(M_l)$ even in the worst case, i.e., coarsening in one dimension only. This we have confirmed through a complexity analysis. Results of the numerical experiments display the excellent multigrid convergence that can be brought about with the presented strategies.

Appendix A. Implementational aspects of ω-RB Jacobi. ω-RB Jacobi can be implemented in several ways depending on the test platform. We have here a method that is suitable for iterations over a solution vector as opposed to explicit updates in a loop. The scheme that we present in this appendix is developed to bring about the odd-even (red-black) partitioning with minimal manipulation of the grid points.

We store three vectors in this scheme (two of which have just half the storage requirement as the first one). The first vector is the unmanipulated vector (henceforth

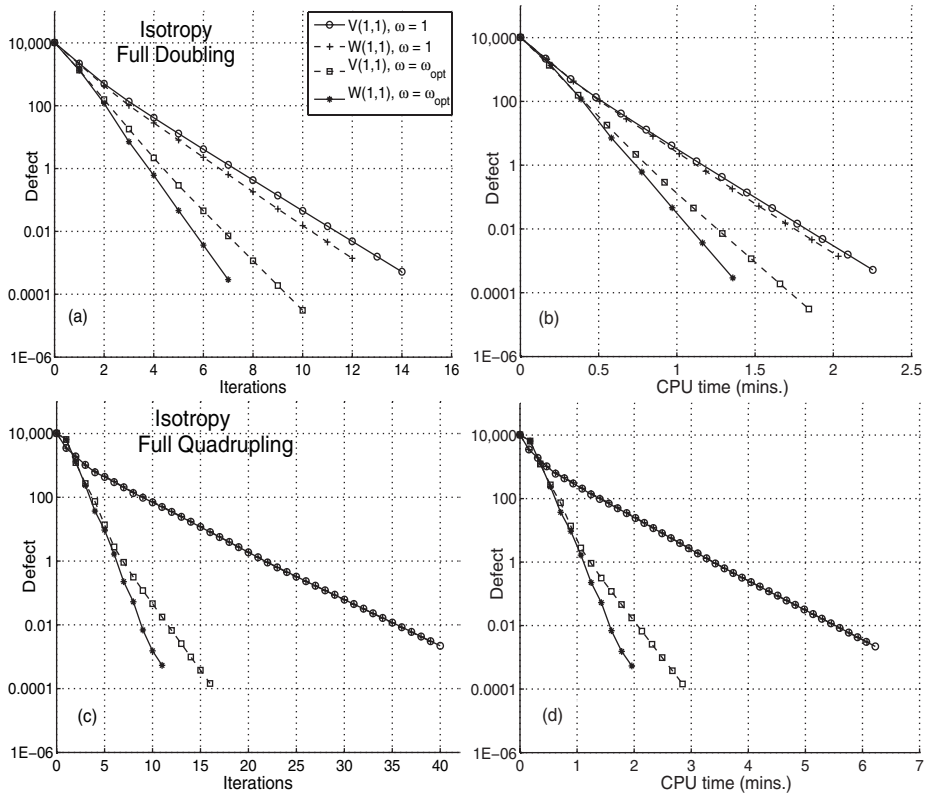


FIG. 3. *Convergence behavior of different multigrid cycles for a 4d problem, on an equidistant grid* 32^4 *(isotropy with* 923,521 *unknowns), left side defect vs. iterations, right side defect vs. CPU time.*

called the , , , ,) containing values for all grid points (red as well as black); the second vector has the values at the black points ejected out and replaced with zeros. Symmetrically, the third has the values at the red positions ejected out and replaced by zeros. This ejection process is actually where our injection operators (henceforth called , , ,) fit in.

First we construct the partition of the main vector storing the red and the black parts; then we carry out the first partial ω-Jacobi sweep by updating only the red part. This new red part along with the previously stored black part represents the main vector after the first partial sweep. Carrying out the second partial sweep in exactly the same manner, now for the black part instead, gives one ω-RB Jacobi iteration.

We present two , , , operators, one for points of each color (even/odd category). We denote these , , , by $\mathcal{E}_R$ and $\mathcal{E}_B$, with

$$\mathcal{E}_R = \left(\bigoplus_{i=1}^{d} \eta_{(d-i+1)} \right) \mod 2,$$

$$\mathcal{E}_B = (\mathcal{E}_R + 1) \mod 2.$$

$\bigoplus$ is the cumulative tensor sum of η_i, which counts the interior points along the ith dimension (see (3.9)); i has the reverse order (from d to 1) to match the lexicographic

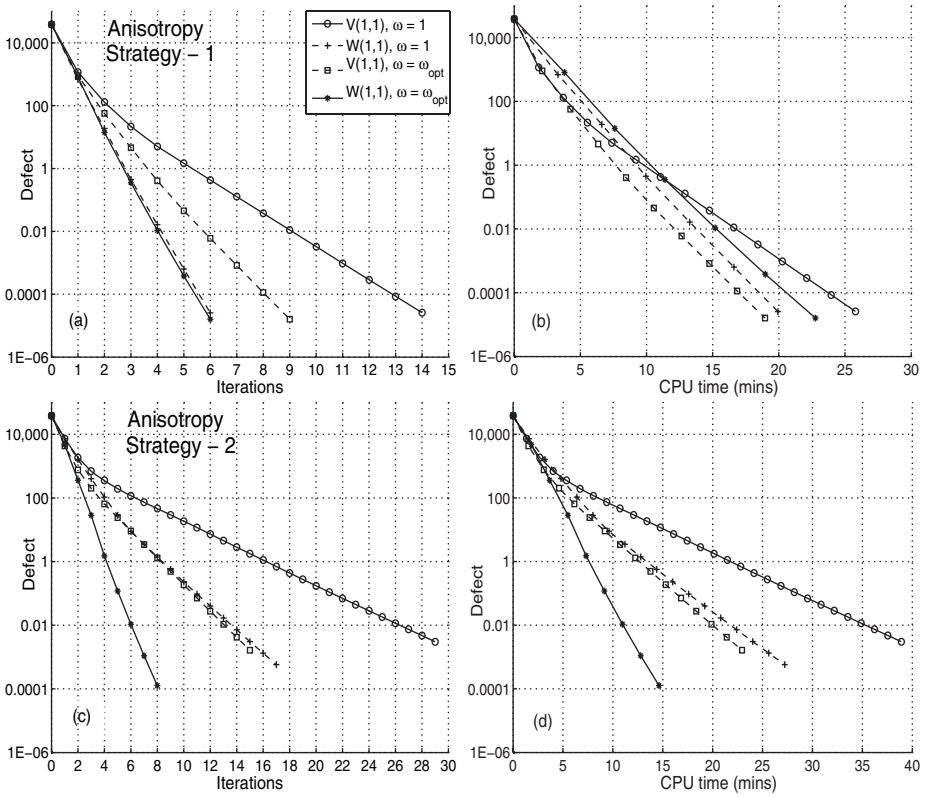


FIG. 4. *Convergence behavior of different multigrid cycles for a* 5*d problem, on a non-equidistant grid* [32 8 8 128 32] *(anisotropy with* 5,980,303 *unknowns), left side defect vs. iterations, right side defect vs. CPU time.*

layout of the grid points. Due to space limitations we can provide only a $2d$ example, although this formulation is true in general for an abstract higher dimension d.

2. Consider a $2d$ grid $\mathbf{G} = [4, 5]$. In all we have 12 interior points, which, when counted in the lexicographic order, appear as follows:

$$\mathbf{u} = [u_{11}\ \ u_{12}\ \ u_{13}\ \ u_{14}\ \ u_{21}\ \ u_{22}\ \ u_{23}\ \ u_{24}\ \ u_{31}\ \ u_{32}\ \ u_{33}\ \ u_{34}]^T.$$

Evidently, $\boldsymbol{G}$, the collection of all of the points, has the following partition:

$$\begin{aligned} \boldsymbol{G}_R &= \{u_{11}, u_{13}, u_{22}, u_{24}, u_{31}, u_{33}\}, \\ \boldsymbol{G}_B &= \{u_{12}, u_{14}, u_{21}, u_{23}, u_{32}, u_{34}\}. \end{aligned}$$

Therefore according to our scheme

$$\begin{aligned} \mathbf{u}_R &= [\ \ u_{11}\ \ \ 0\ \ \ u_{13}\ \ \ 0\ \ \ 0\ \ \ u_{22}\ \ \ 0\ \ \ u_{24}\ \ \ u_{31}\ \ \ 0\ \ \ u_{33}\ \ \ 0\ \ \ ]^T, \\ \mathbf{u}_B &= [\ \ 0\ \ \ u_{12}\ \ \ 0\ \ \ u_{14}\ \ \ u_{21}\ \ \ 0\ \ \ u_{23}\ \ \ 0\ \ \ 0\ \ \ u_{32}\ \ \ 0\ \ \ u_{34}\ \ ]^T, \end{aligned}$$

and η_i in this case would be

$$\eta_1 = [1\ \ 2\ \ 3\ \ 4]^T \quad \text{and} \quad \eta_2 = [1\ \ 2\ \ 3]^T,$$

which leads to

$$\begin{aligned} \mathcal{E}_R &= (\eta_2 \oplus \eta_1) \mod 2 \\ &= [2\ 3\ 4\ 5\ 3\ 4\ 5\ 6\ 4\ 5\ 6\ 7]^T \mod 2 \\ &= [0\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 1]^T \\ \therefore \quad \mathcal{E}_B &= [1\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0]^T. \end{aligned}$$

These red and black point ejectors now can be used to partition the grid as described. Once this partition is obtained, ω-RB Jacobi relaxation sweeps are trivial to perform, in that they are no different than their $2d$ counterparts.

REFERENCES

[1] G. Beylkin and M.J. Mohlenkamp, *Algorithms for numerical analysis in high dimensions*, SIAM J. Sci. Comput., 26 (2005), pp. 2133–2159.

[2] A. Brandt, *Multi-level adaptive solutions to boundary-value problems*, Math. Comp., 31 (1977), pp. 333–390.

[3] J. Elf, P. Lötstedt, and P. Sjöberg, *Problems of high dimension in molecular biology*, in Proceedings of the 19th GAMM-Seminar Leipzig, Max-Planck-Institute, Leipzig, 2003, pp. 21–30.

[4] S.R. Elias, G.D. Stubley, and G.D. Raithby, *An adaptive agglomeration method for additive correction multigrid.*, Internat. J. Numer. Methods Engrg., 40 (1997), pp. 887–903.

[5] G. Horton and S. Vandewalle, *A Space-Time Multigrid Method for Parabolic PDEs*, Technical report 6/93, IMMD 3, Universitaet Erlangen, 1993.

[6] Y.K. Kwok, *Mathematical Models of Financial Derivatives*, 2nd ed., Springer Finance, Springer, Singapore, 1998.

[7] J. Larsson, F.S. Lien, and E. Yee, *Conditional semicoarsening multigrid algorithm for the poisson equation on anisotropic grids*, J. Comput. Phys., 208 (2005), pp. 368–383.

[8] C. Reisinger and G. Wittum, *On multigrid for anisotropic equations and variational inequalities*, Comput. Vis. Sci., (2004).

[9] C. Reisinger and G. Wittum, *Efficient Hierarchical Approximation of High-Dimensional Option Pricing Problems*, SIAM J. Sci. Comput., 29 (2007), pp. 440–458.

[10] H.W. Steeb, *Kronecker Product of Matrices and Applications*, Wissenschaftsverlag, 1991.

[11] K. Stüben and U. Trottenberg, *Multigrid Methods: Fundamental Algorithms, Model Problem Analysis and Applications*, Springer, Berlin, 1982.

[12] C. Reisinger and G. Wittum, *On the Construction of Fast Solvers for Elliptic Equations*, Technical report, von Karman Institute for Fluid Dynamics, Rhode-Saint-Genese, Belgium, 1982.

[13] C.A. Thole and U. Trottenberg, *Basic smoothing procedures for the multigrid treatment of elliptic 3-d operators.*, Appl. Math. Comput., 19 (1986), pp. 333–345.

[14] U. Trottenberg, C.W. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, New York, 2001.

[15] P. Wesseling, *An Introduction to Multigrid Methods*, Pure Appl. Math., John Wiley & Sons, New York, 1992.

[16] R. Wienands, *Extended Local Fourier Analysis for Multigrid: Optimal Smoothing, Coarse Grid Correction, and Preconditioning*, Ph.D. thesis, University of Cologne, 2001.

[17] R. Wienands and W. Joppich, *Practical Fourier Analysis for Multigrid Methods*, Numerical Insights, Chapman and Hall/CRC, Boca Raton, FL, 2005.

[18] I. Yavneh, *Multigrid smoothing factors for red-black Gauss–Seidel relaxation applied to a class of elliptic operators*, SIAM J. Numer. Anal., 32 (1995), pp. 1126–1138.

[19] I. Yavneh, *On red-black SOR smoothing in multigrid*, SIAM J. Sci. Comput., 17 (1996), pp. 180–192.

[20] Z. You-lan, W. Xiaonan, and C. I-Liang, *Derivative Securities and Difference Methods*, Springer Finance, Springer Science+Business Media Inc. USA, New York, 2004.

[21] H. Yserentant, *Sparse grid spaces for the numerical solution of the electronic Schrödinger equation*, Numer. Math., 101 (2005), pp. 381–389.

SIAM J. SCI. COMPUT.
Vol. 29, No. 4, pp. 1637–1649
© 2007 Society for Industrial and Applied Mathematics

A NEW IMPLEMENTATION OF SYMPLECTIC RUNGE–KUTTA METHODS*

ROBERT I. MCLACHLAN†

Abstract. We propose a "Newton–Taylor" iteration for solving the implicit equations of symplectic Runge–Kutta methods, using the Jacobian of the vector field and matrix-vector multiplications whose extra cost for certain structured problems is negligible. The structure of Hamiltonian ODEs allows this very simple iteration to be effective. The iteration reduces the number of vector field evaluations almost to that of Newton's method, often only one or two per time step, making symplectic Runge–Kutta methods more efficient even at relatively large time steps.

Key words. Runge–Kutta, implicit, symplectic integrators, inexact Newton methods

AMS subject classifications. 65L06, 65P10

DOI. 10.1137/06065338X

1. Introduction. Symplectic integrators based on splitting, such as the leapfrog method for separable Hamiltonian systems, are fast and simple and give very good results for long simulations [8]. They are widely, almost universally, used in applications from accelerator physics to molecular dynamics to celestial mechanics [16]. However, the only symplectic integrators for general Hamiltonian systems are implicit, such as the Gaussian Runge–Kutta methods [11], which has tended to limit their popularity. (The implicit equations must be solved to within round-off error, to preserve symplecticity.) Hairer, Lubich, and Wanner [8] have studied the implementation issues and found the simple ("standard") iteration, (4) below, superior to Newton's method; for very small error tolerances it can even compete with high-order methods based on splitting and composition. However, for the relatively low orders and large time steps often used in long symplectic integrations, the convergence of the standard iteration deteriorates, making the method more expensive, which defeats the purpose of using a large time step.

For other applications of implicit methods, there are methods in which the linear equations defining the Newton step are solved only approximately. In the inexact Newton method an (e.g., Krylov or Chebyshev) iterative method is applied [2, 4, 5]; in the Newton-chord method [18] the Jacobian itself is approximated to simplify the solution step. In the Jacobian-free-Newton–Krylov method [12] the Jacobian-vector multiplications are approximated by finite differences, so the Jacobian itself is never formed.

The application to Hamiltonian systems has a number of special features that lead us to propose a very simple inexact Newton method that, for many structured problems, costs only a constant factor close to 1 times the cost of the standard iteration, yet has a convergence rate very close to Newton's method. For the midpoint rule, for example, the method may require only one or two evaluations of the vector field per time step.

*Received by the editors March 1, 2006; accepted for publication (in revised form) January 9, 2007; published electronically August 1, 2007. This work was supported by the Marsden Fund and the New Zealand Institute of Mathematics and Its Applications.
http://www.siam.org/journals/sisc/29-4/65338.html
†Institute of Fundamental Sciences, Massey University, Palmerston North, New Zealand (r.mclachlan@massey.ac.nz).

The method uses the Jacobian of the vector field and performs matrix-vector multiplications. It is efficient when those two operations are cheap compared to the cost of evaluating the vector field itself. Implementations of implicit methods using Newton's method and direct linear solves try to balance the number of Jacobian factorizations against the number of vector field evaluations [19]. In contrast, our goal is to minimize the number of vector field evaluations without performing any operations that cost substantially (e.g., asymptotically, for large systems) more than a vector field evaluation.

First, consider N-body problems with Hamiltonians of the form

$$H = \sum_{i,j=1}^{N} p_i G(\|q_i - q_j\|) p_j + V(\|q_i - q_j\|). \tag{1}$$

The cost of evaluating the vector field directly is $\mathcal{O}(N^2)$. If G, V, and their derivatives are cheap, then the cost is dominated by the cost of computing $\|q_i - q_j\|$, and the Jacobian is available essentially for free. This has been exploited for separable systems ($G = 1$) by designing efficient high-order symplectic integrators using the Jacobian and matrix-vector multiplications [3, 16]. On the other hand, if G and V are special functions, then it often happens that it is relatively cheap to compute their derivatives at the same time. (This happens in the first example below, where G is an exponential or a Bessel function.) The matrix-vector multiplications, on the other hand, are also $\mathcal{O}(N^2)$ and hence carry a constant work overhead compared to the cost of evaluating the vector field.

Second, consider lattice systems with independent variables indexed by a lattice L containing N points and a Hamiltonian of the form

$$H = \sum_{i \in L} F(x_{i+\Delta}), \tag{2}$$

where $\Delta \subset L$ is a fixed template specifying which sites are coupled. The cost of computing the vector field is $\mathcal{O}(N)$. The Jacobian matrix has only $\mathcal{O}(N)$ nonzero entries, and hence matrix-vector multiplications cost $\mathcal{O}(N)$. Again, the cost of forming and multiplying by the Jacobian is negligible for certain functions F.

2. The Newton–Taylor iteration. We describe the algorithm for the implicit midpoint rule. (Its extension to any implicit Runge–Kutta method is straightforward.) For the ODE $\dot{x} = f(x)$ we are required to solve [11]

$$g(z) := z - \frac{1}{2}\Delta t f(x_0 + z) = 0 \tag{3}$$

for z, where Δt is the time step. (The final update is $x_0 \mapsto x_1 := x_0 + 2z$.) The standard fixed-point iteration is

$$z^{k+1} = \frac{1}{2}\Delta t f(x_0 + z^k) = z^k - g(z^k), \tag{4}$$

and Newton's method is

$$z^{k+1} = z^k - g'(z^k)^{-1} g(z^k) = z^k - (I - A_k)^{-1} g(z^k), \tag{5}$$

where $A_k = \frac{1}{2}\Delta t f'(x_0 + z^k)$.

We replace the iteration matrix $(I - A_k)^{-1}$ by its Taylor series $\sum_{i=0}^{M} A_k^i$ of order M to give the following (Newton–Taylor) outer iteration.

Outer iteration:

$$z^{k+1} = h(z^k) := z^k - \left(\sum_{i=0}^{M} A_k^i \right) g(z^k), \tag{6}$$

which can be evaluated using one vector field, Jacobian evaluation at z^k, and M matrix-vector multiplications.

The analysis of the algorithm is standard in the context of inexact Newton methods [6, 7]. However, it is simpler here because of the simple iteration. Let z^* be the desired solution so that $g(z^*) = 0$, and let $e^k = z^k - z^*$ be the error in the current approximation. Then by Taylor's theorem we have for some $\theta \in [0, 1]$,

$$\begin{aligned} e^{k+1} &= h'(z^k)e^k + \frac{1}{2}h''(z^k + \theta(z^{k+1} - z^k))(e^k, e^k) \\ &= \left(I - \left(\sum_{i=0}^{M} A_k^i \right) (I - A_k) \right) e^k + \frac{1}{2}h''(z^k + \theta(z^{k+1} - z^k))(e^k, e^k) \\ &= A_k^{M+1} e^k + P(e^k, e^k) + \mathcal{O}(\|e^k\|^3), \end{aligned} \tag{7}$$

where $P = \frac{1}{2}h''(z^*) = \mathcal{O}((\Delta t)^2)$. The convergence rate of the modified iteration (6) is therefore $\rho(A_k^{M+1}) = \rho(A_k)^{M+1} = \mathcal{O}((\Delta t)^{M+1})$, where $\rho(A_k)$ is the convergence rate of the standard iteration. Not only is the modified iteration higher-order in the time step, but (at the linear level, i.e., for sufficiently small e^k) one step of the modified iteration is [illegible] to $M + 1$ steps of the standard iteration.[1] It is therefore worth increasing M by one precisely when a Jacobian-vector multiplication is cheaper than evaluating the vector field.

We also conclude from (7) that the modified iteration (6) converges for sufficiently small $\|e^0\|$ if and only if the simple iteration converges. We therefore henceforth assume that $\rho(A_k) < 1$.

If M is taken too large, however, then e^{k+1} will be dominated by the quadratic term in (7). Note that as $M \to \infty$, $A_k^{M+1} \to 0$ and $P \to P^\infty$ (say), the iteration tensor for Newton's method. In fact, $P = P^\infty + \mathcal{O}((\Delta t)^{M+1})$, so as $M \to \infty$ the behavior of the modified iteration tends to that of Newton's method. One should therefore adapt M depending on the current iteration, so that $\|A_k^{M+1} e^k\| \approx \|P(e^k, e^k)\|$. This gives the following algorithm, in which (6) is implemented.

Inner iteration:

1. Given z^k, set $w^0 = g(z^k)$;
2. while $\|w^{m+1} - w^m\| > \max(c\|w^0\|^2, \text{[illegible]})$, set $w^{m+1} = w^0 + A_k w^m$;
3. set $z^{k+1} = z^k - w^m$.

Here [illegible] is the tolerance, usually close to machine precision, to which the Runge–Kutta equations are to be solved. The ("forcing") parameter c depends on the problem and is related to the norm of P, i.e., to the second derivatives of the vector field [7]. Note that w^0 is the update to z^k in the standard iteration and hence, as long as $\rho(A_k)$ is bounded away from 1, w^0 can be used as an approximation of e^k.

[1] Often, higher-order methods have large error constants, rendering them useless unless the time step is sufficiently small. This does not happen here.

This algorithm will lead to the number M of inner iterations roughly doubling each outer iteration, just as the error decreases in Newton's method.

It is theoretically possible, as in the automatic detection of stiffness [11] and the traditional convergence tests for implicit methods [19], to maintain running estimates of the slowly varying parameters $\rho(A_k)$ and the norm of P, and hence adapt the choice of M more precisely. However, this seems to be more difficult in the present context than in those just cited, because (i) we do relatively few inner iterations and outer iterations, so there is little data on which to base the estimates; (ii) the last iterations may be polluted by round-off error; (iii) the absolute values of the eigenvalues of the Hamiltonian matrix A_k occur with multiplicity 2 (if the eigenvalue is real or imaginary) or 4 (if the eigenvalue is complex), which means that even in the best case, when A_k is not severely nonnormal, an estimate of the convergence rate of the inner iteration can be obtained only from four consecutive iterations; (iv) if the time step is large, as we hope to achieve, then the parameters vary quite quickly (see the first example below).

A more positive viewpoint is to consider these factors as advantages, as they support the use of a very simple algorithm! Most of the experiments reported below use $c = 1$, $\epsilon_1 = 10^{-15}$, and $\|\cdot\|_\infty$ error norms.

To further reduce the number of evaluations of f we terminate the outer iteration (6) at z^{k+1} when $\|g(z^k)\| < \epsilon_2$, where $\epsilon_2 > \epsilon_1$. Under the model used for terminating the inner iterations, one reasonable choice is to take $\epsilon_2 = \sqrt{\epsilon_1 / c}$, as the final iteration will then take the error down to about $2\epsilon_1$. This saves one evaluation of f and is the termination criterion used in our experiments. If long-term growth of round-off error is a concern, these criteria may need to be tightened.

A good initial guess is essential for the performance of this method, as one wants the underlying Newton's method to stay close to the root where convergence is quickest. For the midpoint rule, we have used polynomial extrapolation from the previous time steps [8]. The order s of extrapolation is adapted based on the decrease of the backward differences, as follows.

Initialization iteration:

1. Set $w^0 = z_n$;
2. while $\|\nabla^{s+1} z_n\| < \|\nabla^s z_n\|$, set $w^{s+1} = w^s + \nabla^{s+1} z_n$;
3. set $z_{n+1}^0 = w^s$.

The modified iteration also works successfully for higher-order Runge–Kutta methods. The tensor product structure of the linear equations means that only s matrix-vector products are needed per inner iteration for an s-stage method. For systems in $\mathbb{R}^N$, $2N^2 s + 2Ns^2$ operations are required per inner iteration, so the overhead does grow with increasing s, but not at leading ($\mathcal{O}(N^2)$) order.

3. Other inexact Newton iterations (see [10, 13, 14]). We are approximating the solution of $(I - B)\delta z = g$ by the Taylor polynomial $\sum_{i=0}^{M} B^i g$, where for a general Runge–Kutta method with coefficients a_{lj} the matrix B is a block matrix whose (l, j) block is given by $\Delta t a_{lj} f'(Z_j)$, where Z_j are the stage values and $f'(z) = JH''(z)$. We have seen that this is a reasonable approximation, but aren't there better approximations? Krylov methods form the best (in some sense) approximation to δz of the form $p(B)g$ over all polynomials p of degree M; Faber methods use the Faber polynomial, which is the best (in some sense) approximation of $(1-t)^{-1}$ on $\Omega \subset \mathbb{C}$, where Ω contains the spectrum of B. These satisfy an (expensive) long-term recurrence; hence one often chooses Ω to be an ellipse, in which case the Faber polynomials are Chebyshev polynomials which satisfy a 2-term recurrence. In this context,

the Taylor polynomials are themselves the special case of the Chebyshev polynomials, obtained when the ellipse is a circle centered on the origin.

In this case the best approximation property is easily seen. Let $a(t)$ be a complex function analytic on a disk of radius r centered (now for convenience only) on the origin. The powers t^n are orthogonal with respect to the L_2-inner product $(a, b) = \int_{|t|\le r} a(t)\overline{b}(t)\,|dt|^2$, and hence, from the general theory of orthogonal functions, the expansion of $a(t)$ in powers of t—the Taylor polynomial—is the best L_2 approximation to $a(t)$ by a polynomial.

The role of two key features of Hamiltonian systems and of symplectic integrators is now apparent. They both stem from the fact that the Jacobian f' is a Hamiltonian matrix whose eigenvalues exhibit "Hamiltonian symmetry"; namely, they occur in λ, $-\lambda$, $\overline{\lambda}$, $-\overline{\lambda}$ quadruplets. For the midpoint rule, $B = \frac{1}{2}\Delta t f'(Z_1)$ is also Hamiltonian.

First, for the implicit equations to have a solution we need to assume $\mathrm{Re}(\lambda(B)) < 1$, which implies $\mathrm{Re}(\lambda(B)) > -1$. We can also assume $|\mathrm{Im}(\lambda(B))| < 1$, for otherwise the fast oscillations are very poorly represented by the integrator and can even destabilize the calculation. (The currently known integrators that are effective when the fast modes are not resolved are quite different; they are explicit and nonsymplectic [8].) Therefore, the assumption $\rho(B) < 1$ that is required for the Taylor series to converge is not overly restrictive. (Applications where Krylov and Chebyshev iterations are effective typically have $\mathrm{Re}(\lambda(B))$ very large and negative [2, 12].)

Second, the eigenvalue symmetry means that a disk centered on 0, enclosing $\lambda(B)$, will tend to be a much better approximation of $\lambda(B)$ than in the general, non-Hamiltonian, case. Further, the Taylor polynomial requires no knowledge of the spectrum and is the best approximation on all disks centered on the origin (i.e., the best regardless of $\rho(B)$).

The eigenvalue symmetry does mean that an ellipse centered on 0 with real and imaginary axes is never a worse approximation to $\lambda(B)$ than the disk. If this ellipse were known, then a Chebyshev polynomial would be a better approximation to $(1-t)^{-1}$, and would lead to faster convergence, than the Taylor polynomial. Two factors suggest that this advantage will be difficult to exploit in our case. First, for Hamiltonian systems the focal interval of the bounding ellipse can be either real or imaginary, typically alternating between the two, making an accurate estimate of it more difficult; see the first example below. (Existing applications of Chebyshev polynomials use far more iterations than we are proposing here, because they are for $\rho(B)$ large, and hence they can afford to do a few secondary iterations to estimate the ellipse that bounds $\lambda(B)$.) Second, when $\rho(B) < 1$ the convergence advantage of Chebyshev over Taylor polynomials is not that great. For small $\rho(B)$ the Chebyshev convergence factor (see, e.g., [13]) is at best $\frac{1}{2}\rho(B)$ and achieves that only when $\lambda(B)$ is entirely real or entirely imaginary. We leave a fuller comparison of the two approaches for the future.

For the midpoint rule—or indeed for any 1-stage Runge–Kutta method—the iteration matrix B is Hamiltonian. This still holds for almost all 2-stage Runge–Kutta methods, as we now show. If $a_{12}a_{21} \neq 0$, then B is a Hamiltonian matrix because it can be written as a product of an antisymmetric and a symmetric matrix:

$$
\begin{aligned}
B &= \Delta t \begin{pmatrix} a_{11}JH''(Z_1) & a_{12}JH''(Z_2) \\ a_{21}JH''(Z_1) & a_{22}JH''(Z_2) \end{pmatrix} \\
&= \Delta t \begin{pmatrix} \frac{a_{11}}{a_{21}}J & J \\ J & \frac{a_{22}}{a_{12}}J \end{pmatrix} \begin{pmatrix} a_{21}H''(Z_1) & 0 \\ 0 & a_{12}H''(Z_2) \end{pmatrix}.
\end{aligned}
$$

(The symplectic structure, given by the first matrix factor, is noncanonical, but this does not affect the eigenvalue symmetry.) On the other hand, if $a_{12} = 0$, say, then B is block lower triangular and not Hamiltonian. However, its eigenvalues are $\Delta t a_{11}\lambda(JH''(Z_1)) \cup \Delta t a_{22}\lambda(JH''(Z_2))$, which do have Hamiltonian symmetry.

For Runge–Kutta methods with ≥ 3 stages the iteration matrix is not in general Hamiltonian, and its eigenvalues need not have Hamiltonian symmetry, numerical examples being easy to construct. In fact, attempting to express the iteration matrix for a 3-stage Runge–Kutta method in the form of an antisymmetric matrix $\alpha_{lj} \otimes J$ times the symmetric matrix $\sum_k \beta_{ljk} H''(Z_k)$ (where $\alpha_{lj} = \alpha_{jl}$, $\beta_{ljk} = \beta_{jlk}$) yields, on eliminating the variables α_{lj}, β_{ljk}, the result that such a factorization exists only if $a_{13}a_{21}a_{32} = a_{12}a_{23}a_{31}$, which does not hold for the 3-stage Gaussian Runge–Kutta. Nevertheless, the iteration matrix will always be [illegible] Hamiltonian, because the matrices $H''(Z_k)$ are all equal to within $\mathcal{O}(\Delta t)$, and in the case of equal matrices such a factorization always exists, namely $a_{lj}JH''(Z) = (a_{lj} \otimes J)(I \otimes H''(Z))$. Therefore the eigenvalues of the iteration matrix will always be close to obeying the same symmetry as those of Hamiltonian matrices.

In other applications, the use of preconditioners is vital [5, 12]. Preconditioning will work here too—decreasing $\rho(B)$, but also destroying, in general, the Hamiltonian structure of B—although not generally as well as in the stiff, dissipative case. We do so few inner iterations anyway that the preconditioner has to be very cheap or extremely effective. An example of a use of a preconditioner, or equivalently a Newton-chord iteration, for a Hamiltonian system is given in [1] for the Korteweg–de Vries equation $u_t = uu_x + u_{xxx}$. The Jacobian is frozen at the stiff term ∂_{xxx}. A second example is given in the solar system study below.

4. Examples. The following example arose in a study of the dynamics of certain curves called momentum sheets in a generalized Camassa–Holm equation that is thought to govern certain classes of solitary waves in shallow water [15]. The curve motion is governed by a PDE which is discretized by placing particles at positions $q_i \in \mathbb{R}^2$ on the curve. Each particle carries momentum $p_i \in \mathbb{R}^2$, and the Hamiltonian is

$$H = \sum_{i=1}^{n} (p_i \cdot p_j) G(\|q_i - q_j\|), \tag{8}$$

which describes geodesics on a Riemannian manifold with metric the inverse of the matrix of G's. The three kernels $G(r)$ of most interest in the application were the Gaussian $\exp(-r^2)$, the modified Bessel functions $K_0(r)$ and $rK_1(r)$ (with the singularity at $r = 0$ appropriately removed). Note that we have, e.g., $K_0'(r) = -K_1(r)$ and $K_0''(r) = K_0(r) + K_1(r)/r$. Since K_0 and K_0' already occur in the vector field, no new Bessel functions have to be computed for the Jacobian.

The motion of the sheet for the Gaussian kernel is shown in Figure 1 for time 50, during which it undergoes a large-scale evolution. There are 50 particles, and the total dimension of the system is 200. To give an idea of the time scale of this problem, both the standard and modified iterations converge at $\Delta t = 6$ but not at $\Delta t = 7$. The final position error is 0.7% at $\Delta t = 1$ and 15.3% at $\Delta t = 5$. The maximum relative energy error during the simulation is 0.2% at $\Delta t = 1$. The spectrum is also shown in Figure 1. The focal interval of the bounding ellipse changes from imaginary to real, to imaginary, and back to real, although the ellipse never becomes very eccentric, and a circle is always a reasonable approximation to the spectrum.

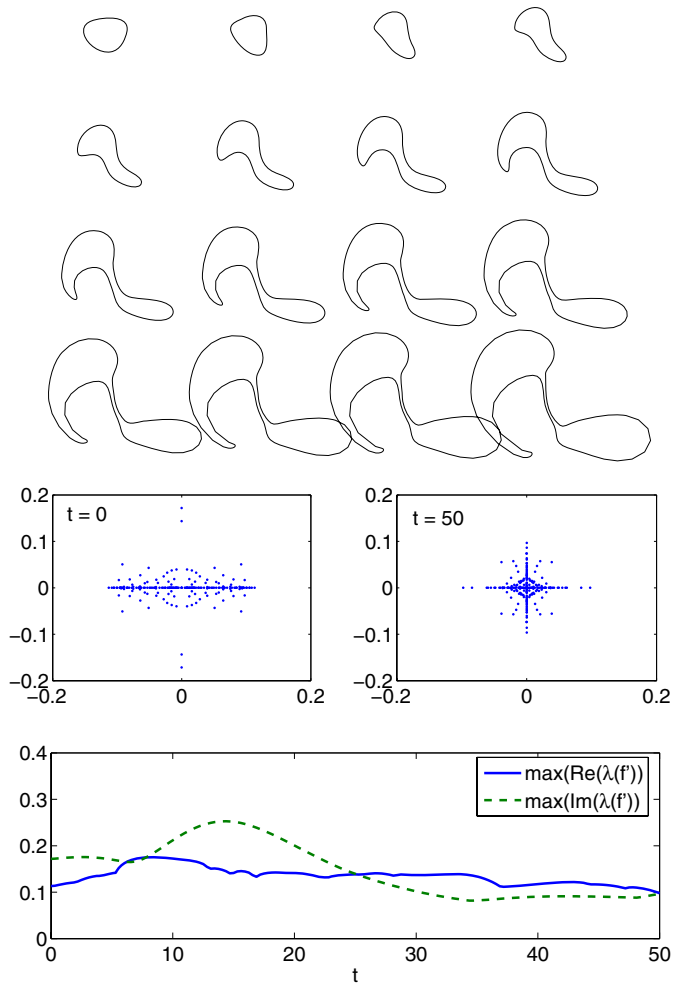


FIG. 1. *Top: Snapshots of the motion of the geodesic curve motion, for* $t \in [0, 50]$. *Middle: Spectrum of the Jacobian at the initial and final times. Bottom: Evolution of the eigenvalues of maximum real and imaginary parts.*

The results are shown in Table 1 and Figure 2. The number of outer iterations (equivalently, number of vector field evaluations) per time step is very close to that used by Newton's method. For the Gaussian kernel with the midpoint rule, one could use $\Delta t = 0.4$ and do about 1 evaluation of f per step (the standard iteration needs 6), achieving a final position error of 0.0011; or one could use $\Delta t = 0.8$ and do about 2 evaluations of f per step (the standard iteration needs 10), achieving a final position error of 0.0045. Clearly, in this problem there is no advantage in the larger time step. But even $\Delta t = 0.4$ is relatively large for this problem and is analogous to the large time steps often used in molecular dynamics problems.

A second observation is that even with the simplest implementation possible, namely using $c = 1$ in the termination criterion for the inner iteration, we have $\overline{N}_{\text{outer}} + \overline{N}_{\text{inner}} \approx \overline{N}_{\text{std}}$ across the whole range of time steps. The algorithm is roughly equivalent to exchanging most iterates of the standard iteration (4) with a matrix-vector multiplication.

TABLE 1

Results for the curve geodesic problem (8) *shown in Figure* 1 *with* 50 *points and dimension* 200. *The integration time is* 50 *and the time step is* $\Delta t = 50/\text{steps}$. $\overline{N}_{\text{outer}}$ *is the mean number of outer iterations (= number of vector field evaluations) per time step used by the modified iteration equation* (6) *(including the initial steps, when the order of the initialization is necessarily lower and more iterations are needed).* $\overline{N}_{\text{inner}}$ *is the mean number of inner iterations (= number of matrix-vector multiplications) per time step.* $\overline{N}_{\text{std}}$ *is the mean number of vector field evaluations per time step used by the standard iteration* (4). *CPU times in seconds are shown for the two methods in a MATLAB implementation. The same extrapolation method is used for initialization for all cases. Its order ranges from* 4 *(for large time steps) to* 14 *(for small time steps).*

$G(r)$	Order	Steps	$\overline{N}_{\text{std}}$	$\overline{N}_{\text{outer}}$	$\overline{N}_{\text{inner}}$	CPU_{std}	CPU
e^{-r^2}	2	10	42.10	4.60	38.50	6.30	1.77
		14	28.07	3.79	24.50	6.03	1.99
		20	20.65	3.30	17.85	6.18	2.37
		28	16.39	3.04	13.75	6.87	3.12
		40	13.13	2.53	10.95	7.87	3.52
		57	10.49	2.07	8.56	8.82	4.10
		80	8.26	1.90	6.50	9.80	5.34
		113	6.09	1.34	4.89	10.29	5.36
		160	4.32	1.06	3.24	10.36	6.02
		226	3.27	1.03	2.24	11.19	8.19
		320	2.98	1.02	1.96	14.53	11.36
e^{-r^2}	4	6	35.83	5.17	37.17	5.20	2.00
		10	23.00	4.10	22.30	5.30	2.52
		20	15.30	3.10	13.40	7.11	3.74
		57	9.32	2.05	7.63	12.38	6.99
		160	4.35	1.07	3.36	17.04	10.73
$K_0(r)$	2	20	19.55	2.45	17.20	12.74	2.82
		40	8.95	1.50	7.48	11.88	3.04
		80	4.90	1.16	3.85	11.90	4.64
		160	3.27	1.05	2.19	15.87	8.25

For the Gaussian kernel and fourth-order 2-stage Gaussian Runge–Kutta, the position errors are now only 0.2% at $\Delta t = 5$ and 4×10^{-6} at $\Delta t = 1$, and the maximum relative energy errors 0.3% at $\Delta t = 5$ and 3×10^{-5} at $\Delta t = 1$. The improvements to the fourth-order method are very similar to those enjoyed by the second-order method. The number of outer iterations is reduced slightly, and the number of inner iterations is significantly reduced at large time steps, because the spectral radius of (a_{ij}) is reduced by a factor 0.58. For the same reason, the fourth-order method converges for larger time steps, failing to converge only at $\Delta t = 9$. For this problem, the fourth-order method always gives a smaller error for a given work (see Figure 2). Both methods can be used satisfactorily at large time steps.

The speedup in CPU time is not as great as the reduction in function evaluations. It is generally a factor of about 2 in this MATLAB implementation—more for larger time steps, a bit less if the order is increased to 4.

The following refinement of the algorithm yields further improvement in the CPU times: On outer iterations after the first, update the Jacobian only if the norm of the previous update of z is greater than some chosen tolerance. This use of inexact Jacobians slows the convergence of both the inner and outer iterations, but (depending on the complexity of the Jacobian) can nevertheless lead to an overall speedup. In the first example of Table 1 (Gaussian kernel with the midpoint rule), with tolerance 0.1 the speedup factor was in the range 1.3–1.38 for the critical range of 20–80 time steps ($0.625 \le \Delta t \le 2.5$). Clearly, the choice of the tolerance is crucial and is problem-dependent.

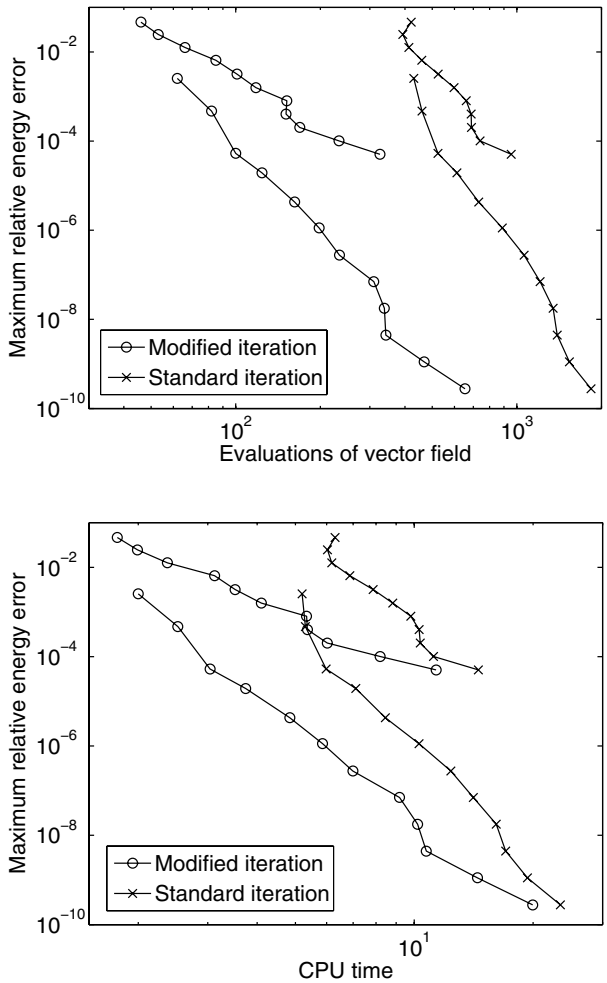


FIG. 2. *Curve geodesics problem. The data from Table* 1 *compared for efficiency* ($G(r) = \mathrm{e}^{-r^2}$, *orders* 2 *and* 4*).*

Sample results are also shown for the more expensive Bessel function kernel $K_0(r)$ with the midpoint rule. $K_0(r)$ can be evaluated in about 78 multiplications, and $K_0(r)$ and $K_1(r)$ together in about 90 multiplications, in MATLAB, which calls a Fortran subroutine. Now the vector field evaluations occupy a larger fraction of the CPU time, and hence the observed speedups are greater.

Two further examples are taken from the results in [8] in a direct comparison with the Fortran code `gnicode`.[2] They are both for separable Hamiltonians of the form $H = \left(\sum_{i=1}^{n} p_i^2/m_i\right) + V(q)$, and in [8] the standard implementation of Gaussian Runge–Kutta methods is compared to explicit composition methods. For these Hamiltonians the momenta at the internal stages can be eliminated, leaving a nonlinear equation for the positions whose Jacobian is now a block matrix with (l, j) block given by $\delta_{lj} I - ((\Delta t A)^2)_{lj} V''(Q_j)$, where A is the matrix of the Butcher tableau and the Q_j are the stage values of the positions. This halves the number of standard iterations,

[2]The results for `gnicode` shown in Table 2 are better than those given in [8], because the code was improved further since the book was published [9].

TABLE 2

Kepler problem. One period of the 2D Kepler problem with period 2π and eccentricity 0.6 is integrated using the standard iteration in the `gnicode` *Fortran implementation and using the modified iteration in an adaptation of the same code. CPU times are reported for the GNU Fortran optimizing compiler 3.4 on a 1GHz G4 Mac. The same initialization is used in each case (extrapolation for order 2, and an order $s+2$ Runge–Kutta-type method for order $2s \geq 4$). N_f is the total number of vector field evaluations including those needed for initialization.*

Order	Δt	N_f Std	N_f Mod.	N_{inner}	$\overline{N}_{\text{outer}}$ Std	$\overline{N}_{\text{outer}}$ Mod.	$\overline{N}_{\text{inner}}$	CPU time Std	μs Mod.
2	$2\pi/50$	383	110	242	7.68	2.20	4.84	349	185
	$2\pi/100$	399	142	229	3.99	1.42	2.29	439	281
	$2\pi/200$	511	225	284	2.56	1.12	1.42	653	494
	$2\pi/400$	590	407	438	1.47	1.02	1.09	988	925
4	$2\pi/25$	375	143	110	7.50	1.86	4.40	261	133
	$2\pi/50$	547	235	142	5.47	1.35	2.84	379	203
	$2\pi/100$	881	427	194	4.41	1.13	1.94	611	352
	$2\pi/200$	1405	805	279	3.51	1.01	1.40	989	647
	$2\pi/400$	2521	1601	447	3.15	1.00	1.12	1764	1252
8	$2\pi/25$	525	213	75	5.25	1.63	3.00	327	207
	$2\pi/50$	791	343	99	3.96	1.22	1.98	494	316
	$2\pi/100$	1215	627	141	3.04	1.07	1.41	771	557
	$2\pi/200$	1707	1203	222	2.13	1.00	1.11	1143	1039
	$2\pi/400$	2779	2403	401	1.74	1.00	1.00	1913	2045
12	$2\pi/25$	679	265	70	4.53	1.43	2.80	414	274
	$2\pi/50$	951	447	83	3.17	1.16	1.66	591	424
	$2\pi/100$	1255	805	123	2.09	1.01	1.23	829	745
	$2\pi/200$	1923	1605	206	1.60	1.00	1.03	1328	1444
	$2\pi/400$	3367	3205	401	1.40	1.00	1.00	2371	2864

somewhat reduces the number of outer iterations, and halves the number of inner iterations and the size of the matrices involved. We present results for the relative energy errors; results for the global errors are similar.

The first example is the two-dimensional Kepler problem with $H = \frac{1}{2}(p_1^2 + p_2^2) + 1/\sqrt{q_1^2 + q_2^2}$, for which results are given in Table 2. Despite the low dimension, the completely different implementation, and the fact that the vector field evaluations do not dominate the CPU time (the single square root costs only about 7 multiplications) the results are strikingly similar to those for the curve geodesic problem.

The second example is the outer solar system problem. The sun, four outer planets, and Pluto are evolved in standard coordinates for 500,000 days. Again, the problem is not very large, and the $n(n-1)/2 = 15$ square roots actually number less than the $3n = 18$ degrees of freedom. Still, the results are similar to those of the previous two examples. For this problem due to the scaling of the variables it was advantageous to adapt the forcing parameter c that controls the termination of the inner iteration. The results are shown for $c = 100$ in Figure 3. The order 4 implicit and explicit methods now have equal performance for time steps less than about 120 days (indicated by an arrow). The order 8 and 12 methods beat the explicit methods for relative energy errors less than about 10^{-7}.

Because the system is nearly integrable, a good preconditioner is available in the form of the product of the exact inverses of the Jacobians of the sun-planet two-body systems. These can be calculated explicitly in terms of scalar products. The convergence factor $\rho(A)$ is reduced dramatically, so that only one inner iteration per Newton step is required. However, the preconditioning is expensive, and it is not yet clear whether the extra level of complexity is justified. (Besides, explicit methods can also exploit the near integrability.)

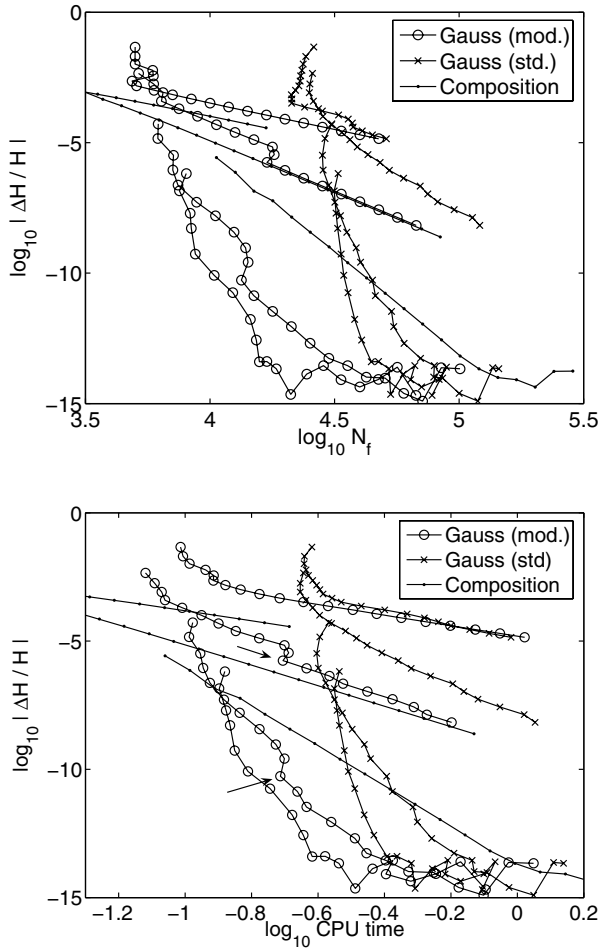


FIG. 3. *Outer solar system problem. Gaussian Runge–Kutta methods of orders* 2, 4, 8, *and* 12 *compared in the standard and modified Fortran implementations. Results for three explicit composition methods are also shown, of orders* 2 *(*1 *stage leapfrog),* 4 *(*5 *stages), and* 8 *(*17 *stages),* `gnicode`*'s methods* 21, 45, *and* 817, *respectively. The performance of the implicit methods is best when the mean number of outer iterations drops to* 1, *which occurs at the points marked by arrows. These correspond to a time step of* 120 *days for order* 4 *and* 240 *days for order* 8. *The order* 12 *results are influenced by round-off for time steps less than* 300 *days.*

5. Conclusions. Even if the vector field has no particular structure, the modified iteration (6) is likely to be an improvement over the standard iteration (4). The only unavoidable requirement is that we need $\rho(A) < 1$; i.e., the problem must not be stiff. However, the case $\rho(A) > 1$ takes us into the entirely different realm of highly oscillatory systems [8]. Therefore the present algorithm is most suited to nonstiff problems in which implicit methods are needed for their geometric properties such as symplecticity.

Many large physical systems have an appropriate structure which reduces the cost of the integration considerably, down to one or two vector field evaluations per time step. Of course, some systems have so much structure that it is possible to exploit it directly in solving the implicit Runge–Kutta equations. For example, for N-body systems one could use a fast multipole method, and for smooth lattice systems such as

those arising as semidiscretizations of PDEs one could use multigrid. However, these approaches are not only vastly more complicated than the present proposal, they are also much more problem-specific.

It is tempting to ask whether the information contained in the Jacobians can be used for purposes other than just speeding up the convergence. Unfortunately, there are no symplectic multiderivative Runge–Kutta methods [8], nor are there any "elementary differential" Runge–Kutta methods using just the Jacobian—the simplest is the midpoint rule applied to the modified Hamiltonian $H + \frac{1}{24}(\Delta t)^2 f^T H'' f$, which requires the second derivative of the vector field f. There are Runge–Kutta-like methods using f' [17], but these are not very developed.

The algorithm presented here is a reasonable one because of the special (Hamiltonian) structure of the problem. It is tempting to consider further specializations of this structure for which the algorithm can be improved further. Directions for future research include automatic selection of the tolerances and exploring cases in which inexact Jacobians, Chebyshev iteration, and/or preconditioners should be used.

Acknowledgements. I would like to thank Stephen Marsland for discussions on the momentum sheet example, and Ander Murua for discussions on the use of Chebyshev iteration.

REFERENCES

[1] U. M. Ascher and R. I. McLachlan, *Multisymplectic box schemes and the Korteweg–de Vries equation*, Appl. Numer. Math., 48 (2004), pp. 255–269.

[2] L. Bergamaschi, M. Caliari, and M. Vianello, *Efficient approximation of the exponential operator for discrete 2D advection-diffusion problems*, Numer. Linear Algebra Appl., 10 (2003), pp. 271–289.

[3] S. Blanes and P. C. Moan, *Practical symplectic partitioned Runge–Kutta and Runge–Kutta–Nyström methods*, J. Comput. Appl. Math., 142 (2002), pp. 313–330.

[4] P. N. Brown and A. C. Hindmarsh, *Matrix-free methods for stiff systems of ODE's*, SIAM J. Numer. Anal., 23 (1986), pp. 610–638.

[5] P. N. Brown and A. C. Hindmarsh, *Reduced storage matrix methods in stiff ODE systems*, J. Appl. Math. Comput., 31 (1989), pp. 40–91.

[6] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *Inexact Newton methods*, SIAM J. Numer. Anal., 19 (1982), pp. 400–408.

[7] S. C. Eisenstat and H. F. Walker, *Choosing the forcing terms in an inexact Newton method*, SIAM J. Sci. Comput., 17 (1996), pp. 16–32.

[8] E. Hairer, C. Lubich, and G. Wanner, *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*, Springer, Berlin, 2002.

[9] E. Hairer, *personal communication*, University of Geneva, 2006.

[10] M. H. Gutknecht and S. Röllin, *The Chebyshev iteration revisited*, Parallel Computing, 28 (2002), pp. 263–283.

[11] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations*. II. *Stiff and Differential-Algebraic Problems*, 2nd ed., Springer, Berlin, 1996.

[12] D. A. Knoll and D. E. Keyes, *Jacobian-free Newton–Krylov methods: A survey of approaches and applications*, J. Comput. Phys., 193 (2004), pp. 357–397.

[13] X. Li, *The convergence rate of the Chebyshev SIM under a perturbation of a complex line segment spectrum*, Linear Algebra Appl., 230 (1995), pp. 47–60.

[14] T. A. Manteuffel, *The Tchebyshev iteration for nonsymmetric systems*, Numer. Math., 28 (1977), pp. 307–327.

[15] R. I. McLachlan and S. R. Marsland, *The Kelvin–Helmholtz instability of momentum sheets in the Euler equations for planar diffeomorphisms*, SIAM J. Appl. Dyn. Sys., 5 (2006), pp. 726–758.

[16] R. I. McLachlan and G. R. W. Quispel, *Geometric integrators for ODEs*, J. Phys. A, 39 (2006), pp. 5251–5285.

[17] S. Miesbach and H. J. Pesch, *Symplectic phase flow approximation for the numerical integration of canonical systems*, Numer. Math., 61 (1992), pp. 501–521.

[18] R. Schreiber and H. B. Keller, *Driven cavity flows by efficient numerical techniques*, J. Comput. Phys., 49 (1983), pp. 310–333.
[19] L. F. Shampine, *Implementation of implicit formulas for the solution of ODEs*, SIAM J. Sci. Stat. Comput., 1 (1980), pp. 103–118.

SIAM J. SCI. COMPUT.
Vol. 29, No. 4, pp. 1650–1667
© 2007 Society for Industrial and Applied Mathematics

SENSITIVITIES IN LARGE EDDY SIMULATION AND IMPROVED ESTIMATES OF TURBULENT FLOW FUNCTIONALS*

MIHAI ANITESCU† AND WILLIAM J. LAYTON‡

Abstract. We consider a prototypical problem: simulate velocity and pressure in a turbulent flow using large eddy simulation (LES) and then use the results to estimate the force that the underlying turbulent flow exerts on an immersed body. For eddy viscosity-type LES models we develop the appropriate continuous sensitivity equation and show how it can improve the functional estimate by using the sensitivity with respect to the user-selected length scale to incorporate the effects of the underlying unresolved small-scale fluctuations on the functionals sought.

Key words. large eddy simulations, sensitivity, turbulence

AMS subject classifications. 76D05, 35Q30, 90C31

DOI. 10.1137/050631161

1. Introduction.

For the foreseeable future, computational resources will be insufficient for the direct numerical solution of most turbulent flow problems. On the other hand, important decisions are made and significant designs produced based on estimates of quantities in turbulent flow that are simulated by using various models of turbulence. Even when using the model that current practice considers best for a particular application, often the reliability of the model's predictions for the specific application is not assessed. This situation is particularly troublesome because solutions of turbulence models can display sensitivity to the user-elected model parameters, in addition to sensitivity to the upstream flow, subgrid model, and numerical realization of it (reported by Sagaut and Lê [19]).

For example, calculating the force that a fluid exerts on an immersed body, such as lift or drag, involves first solving the Navier–Stokes equations

$$\begin{aligned} &u_t + \nabla \cdot (u\ u) + \nabla \cdot \sigma(u,p) = f \qquad \text{in } \Omega \times (0,T), \\ &\sigma(u,p) := pI - 2\nu\nabla^s u, \qquad \nabla^s u := \frac{1}{2}(\nabla u + \nabla u^t), \\ &\nabla \cdot u = 0 \text{ in } \Omega \times (0,T), \quad u = 0 \text{ on } \partial\Omega, \quad \text{and} \quad u(x,0) = u_0(x) \text{ in } \Omega. \end{aligned} \tag{1.1}$$

*Received by the editors May 10, 2005; accepted for publication (in revised form) January 16, 2007; published electronically August 1, 2007. This paper is developed from a 2003 research note with a similar title. The submitted manuscript has been created by UChicago Argonne, LLC, Operator of Argonne National Laboratory ("Argonne"). Argonne, a U.S. Department of Energy Office of Science laboratory, is operated under contract DE-AC02-06CH11357. The U.S. Government retains for itself, and others acting on its behalf, a paid-up nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government.

http://www.siam.org/journals/sisc/29-4/63116.html

†Argonne National Laboratory, Mathematics and Computer Science Division, Argonne, IL 60439 (anitescu@math.pitt.edu). This author was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under contract W-31-109-ENG-38, and NSF grant DMS 0112239.

‡University of Pittsburgh, Department of Mathematics, Pittsburgh, PA 15260 (wjl@pitt.edu). This author was partially supported by NSF grants DMS 9972622, INT 9814115, DMS 0207627, and DMS 0508260.

If B denotes the boundary of the immersed body, next the force must be calculated on B:

$$\text{force on } B \;= \int_B \hat{n} \cdot \sigma \, ds = \int_B \hat{n} \cdot [pI - 2\nu \nabla^s u] \; ds, \tag{1.2}$$

$$\hat{n} = \text{outward unit normal to } B.$$

This calculation requires accurate estimation of p and derivatives of u on the flow boundary—a problem harder than accurately predicting the turbulent velocity itself.

The basic approach used for turbulent flows has been to replace the Navier–Stokes equations (1.1) by a turbulence model and then insert the velocity-pressure predicted by the turbulence model into the right-hand side of the functional, such as (1.2). Uncertainties arise immediately, however, because of the typical sensitivity to the model's input parameters. Perhaps more important, turbulence models approximate flow averages. Thus, all the information on fluctuations is lost in them. However, the small-scale fluctuations can have a determining role in functionals such as (1.2). Mathematically, the reason for this is that the derivatives occurring in (1.2) outweigh velocity changes occurring across small distances. One concrete example is drag in the flow over a dimpled versus a smooth golf ball; a related example is riblets. Both change the geometry (and flow) on scales below typical feasible meshes. Indeed, drag reduction strategies of injection of small amounts of microbubbles or polymers near a body are, in part, based on the expectation that a little power input to alter the flow's small scales can have a large effect on the global drag.

This paper considers functional estimation and solution sensitivity for models arising in large eddy simulation (LES). In section 2 we apply the continuous sensitivity idea to a common class of LES models. The sensitivity equation approach has the great advantage of giving computable quantitative estimates of the local sensitivity of the model's predicted flow field to variations in the input parameters. Thus, a sensitivity calculation will show in which regions the predicted velocities are reliable and hence believable and in which regions those predicted velocities are highly sensitive and hence should be viewed with greater suspicion. We focus on the case of sensitivity with respect to the user-selected length scale δ. The reason for this focus is (section 3) that the sensitivity of the flow to variations in δ can be used to improve the estimate of flow functionals, such as lift and drag, which can depend strongly on the unknown and unresolved turbulent fluctuations.

2. The sensitivity equation of eddy viscosity models. The continuous sensitivity equation approach is becoming increasingly important in computational fluid dynamics but is not yet a common tool in LES of turbulence. In this section we apply the sensitivity idea to find the continuous sensitivity equation with respect to variations in the cutoff length scale delta. For a general treatment of sensitivities and applications to other flow problems, see (among many interesting works) [3, 4, 10, 9, 20].

Suppose that a local spatial filter has been selected (denoted by an overbar) associated with the length scale δ. Filtering the Navier–Stokes equations leads to the problem of closure, and one common class of closure models is based on the Boussinesq, or eddy viscosity, hypothesis. The model we consider takes the general form of finding the approximate large-scale velocity $w(x,t)$ and pressure $q(x,t)$ satisfying $\nabla \cdot w = 0$ and

$$w_t + \nabla \cdot (w \; w) - \nu \Delta w + \nabla q - \nabla \cdot (\nu_T(\delta, w) \nabla^s w) = \overline{f}, \tag{2.1}$$

where ν is the kinematic viscosity, $\overline{f}$ is the space filtered body force, and ν_T is the eddy viscosity coefficient that must be specified to select the model. As an example, the Smagorinsky model, while not considered the best, is perhaps the most commonly used model and is given by the eddy viscosity choice

$$\nu_T = \nu_{\text{Smag}}(\delta, w) := (C_s\delta)^2|\nabla^s w|, \quad C_s \sim 0.1. \tag{2.2}$$

For other eddy viscosity models, see the presentation in [18, 13, 2, 16, 12]. Once the Smagorinsky constant, C_s, and the initial and boundary conditions are specified, for a given δ the equations (2.1), (2.2) uniquely determine a solution w, q implicitly as a function of δ.[1]

DEFINITION 2.1. [illegible] w, q [illegible] (2.1). [illegible] w, q [illegible] δ [illegible] w, q [illegible] δ

$$w_\delta := \frac{\partial w}{\partial \delta}, \qquad q_\delta := \frac{\partial q}{\partial \delta}.$$

One can easily derive continuous equations for the sensitivities by implicit differentiation of (2.1) with respect to δ. Doing so gives the equations

$$w_{\delta,t} + \nabla\cdot(ww_\delta + w_\delta w) - \nu\Delta w_\delta + \nabla q_\delta \tag{2.3}$$

$$-\nabla\cdot\left(\left[\frac{\partial}{\partial\delta}\nu_T(\delta,w) + \frac{\partial}{\partial w}\nu_T(\delta,w)\cdot w_\delta\right]\nabla^s w + \nu_T(\delta,w)\nabla^s w_\delta\right) = \frac{\partial}{\partial\delta}(\overline{f}) \quad \text{and}$$

$$\nabla\cdot w_\delta = 0. \tag{2.4}$$

Thus, once the large eddy velocity and pressure, w, q, are calculated, the corresponding sensitivities can be found by solving a linear problem for w_δ, q_δ that is precisely the nonlinear LES model linearized about (w, q). Thus, sensitivities can be quickly and economically calculated by modifying the program used to calculate (w, q). On the other hand, one can simply use an unmodified program to test whether sensitivity information will be useful for a given application before beginning program modifications. This approach involves solving the model for $w(\delta_1)$ and $w(\delta_2)$ for $|\delta_1 - \delta_2|$ small and computing sensitivities by means of the divided difference

$$s = \frac{w(\delta_2) - w(\delta_1)}{\delta_2 - \delta_1}.$$

In our numerical simulations in section 5 we use the latter approach.

For the Smagorinsky model we have $\nu_T = (C_s\delta)^2|\nabla^s w|$. Thus the bracketed term is

$$[\nu_{T,\delta} + \nu_{T,w}\cdot w_\delta] = 2C_s^2\delta|\nabla^s w| + (C_s\delta)^2\left(\frac{\nabla^s w}{|\nabla^s w|}\right) : \nabla^s w_\delta. \tag{2.5}$$

It is interesting to note that the Smagorinsky model enters the fluid flow equation (2.1) only through the term δ^2. It may therefore be useful to use as a natural variable $\gamma = \delta^2$ and to investigate the sensitivity w_γ with respect to γ.

[1] Fundamental theoretical questions arise even at this first step. For example, it is not known whether the solution of the Smagorinsky model is a C^1 function of δ for $\delta > 0$ or whether the branch is continuous down to $\delta = 0$.

In that case the equations for the sensitivity with respect to γ become

$$w_{\gamma,t} + \nabla \cdot (w w_\gamma + w_\gamma w) - \nu \Delta w_\gamma + \nabla q_\gamma \tag{2.6}$$

$$- \nabla \cdot \left(\left[\frac{\partial}{\partial \gamma} \nu_T(\gamma, w) + \frac{\partial}{\partial w} \nu_T(\gamma, w) \cdot w_\gamma \right] \nabla^s w + \nu_T(\gamma, w) \nabla^s w_\gamma \right) = \frac{\partial}{\partial \gamma}(\overline{f}) \text{ and}$$

$$\nabla \cdot w_\gamma = 0, \tag{2.7}$$

where the bracketed term is

$$[\nu_{T,\gamma} + \nu_{T,w} \cdot w_\gamma] = 2C_s^2 |\nabla^s w| + (C_s)^2 \gamma \left(\frac{\nabla^s w}{|\nabla^s w|} \right) : \nabla^s w_\gamma. \tag{2.8}$$

Since from the chain rule we have that $w_\delta = 2\delta w_\gamma$, it follows that all the conclusions that we draw for w_δ extend immediately to w_γ, with one exception outlined in section 5.2.2. Therefore, we will investigate only the sensitivity with respect to δ, the conclusions extending immediately to the sensitivity with respect to γ.

2.1. Calculating $\overline{f}_\delta = \frac{\partial}{\partial \delta} \overline{f}$. If the body forces acting on the flow are smooth enough, $\overline{f}_\delta$ is negligible. Otherwise, in the right-hand side of (2.3), $\overline{f}_\delta$ can play an important role in the sensitivity equation because it incorporates information about body force fluctuations. When f is not smooth, the exact value of $\overline{f}_\delta$ depends on the precise filter specified. When $\overline{f}$ is defined by convolutions, extending f by zero off the flow domain and then defining

$$\overline{f} := \int_{\mathbf{R}^d} \delta^{-d} g\left(\frac{x'}{\delta} \right) f(x - x') dx, \qquad g(x) := \text{ chosen filter kernel,}$$

then $\overline{f}_\delta$ can be calculated explicitly.

When $\overline{f}$ is defined by using differential filters [7, 16, 13, 2], a small modification is needed that depends on the exact differential filter specified. Two interesting differential filters are defined by solving the shifted Poisson problem for $\overline{f}$,

$$-\delta^2 \Delta \overline{f} + \overline{f} = f \text{ in } \Omega, \qquad \overline{f} = 0 \text{ on } \partial\Omega, \tag{2.9}$$

and by solving the shifted Stokes problem for $\overline{f}$,

$$-\delta^2 \Delta \overline{f} + \overline{f} + \nabla \lambda = f \quad \text{and} \quad \nabla \cdot \overline{f} = 0 \text{ in } \Omega, \quad \text{and} \quad \overline{f} = 0 \text{ on } \partial\Omega. \tag{2.10}$$

The second differential filter (2.10) preserves incompressibility and is thus interesting despite costing more than the first.

With the first filter $\overline{f} = (-\delta^2 \Delta + 1)^{-1} f$, we differentiate implicitly with respect to δ to derive an equation for $\overline{f}_\delta$:

$$(-\delta^2 \Delta + 1) \overline{f}_\delta = 2\delta \Delta \overline{f} = (\text{via } (2.9)) = \left(-\frac{2}{\delta} \right) (f - \overline{f}).$$

Thus,

$$\overline{f}_\delta = \left(-\frac{2}{\delta} \right) (\overline{f'}). \tag{2.11}$$

Analogously, we obtain the initial condition for w_δ,

$$w_\delta(x,0) = \overline{u}_{0,\delta}(x) = \left(-\frac{2}{\delta}\right)(\overline{u_0'}). \tag{2.12}$$

Here, we denote by h' the fluctuation of a function $h(x)$, that is, $h' = h - \overline{h}$. Neither (2.11) nor (2.12) introduce singularities as $\delta \to 0$, provided $\nabla f \in L^2(\Omega)$ and $\nabla u_0 \in L^2(\Omega)$. On the other hand, a large value of $\overline{f}_\delta$ or $\overline{u}_{0,\delta}$ for rough data reflects an essential sensitivity of the data.

2.2. Boundary conditions for the sensitivities. Boundary conditions for sensitivities must be specified. The most interesting and important cases are sensitivity with respect to the (modeled) upstream conditions and the (modeled) outflow conditions [19]. At this point, a mathematical formulation of the former is unclear, whereas there exist many options for the latter. Thus we consider here only boundary conditions for the sensitivities at solid walls.

With a differential filter like (2.9) and (2.10), the boundary conditions for the sensitivities are clear because there exist no error and no variability with respect to δ in the conditions for w on the wall: $w_\delta = 0$ on the boundary.

Some slight modifications of the wall models are necessary to compute sensitivities when near-wall models are used. These are usually associated with averaging; see [6] for some mathematical issues involved. Many near-wall models or numerical boundary conditions are possible. For specificity and clarity we treat the simplest ones considered in, for example, [14], in which the large structure's action on the boundary is modeled by no penetration and slip with friction conditions:

$$w \cdot \hat{n} = 0 \quad \text{and} \quad \beta(\delta,\nu) w \cdot \hat{\tau}_j + \hat{n} \cdot (\nu \nabla^s w) \cdot \hat{\tau}_j = 0 \quad \text{on } \partial\Omega,$$

where $\hat{n}$ is the unit normal to the boundary and $\hat{\tau}_1, \hat{\tau}_2$ are a system of tangent vectors on the boundary. This is the original boundary condition of Navier (Navier's slip law). In large eddy simulation it is motivated by the general observation that large structures (e.g., hurricanes) do not stick but rather slip and lose energy as they do. Loosely interpreting an 1869 analysis of Maxwell suggests a scaling for $\beta \sim \frac{\nu L}{\delta}$, explored in [14].

Implicit differentiation with respect to δ gives the boundary conditions for the sensitivities:

$$w_\delta \cdot \hat{n} = 0 \quad \text{and} \quad \beta(\delta,\nu) w_\delta \cdot \hat{\tau}_j + \hat{n} \cdot (2\nu \nabla^s w_\delta) \cdot \hat{\tau}_j = -\beta_\delta(\delta,\nu) w \cdot \hat{\tau}_j \quad \text{on} \quad \partial\Omega.$$

Since $\beta(\delta)$ increases monotonically to infinity as $\delta \to 0$ (see [14]), $\beta_\delta > 0$. Thus, slippage in the flow velocity w acts to decrease the slippage in the sensitivities when they are aligned and increase it when they are opposed.

3. Improving estimates of functionals of turbulent quantities. The usual procedure for estimating a flow functional is to compute the LES velocity w approximating the fluid motion on scales larger than the cutoff length scale and then to approximate the sought solution functional $J(u)$ by $J(w)$. This can be a reasonable approximation when the small scales do not influence the functional and when w is a good approximation to the large scales of u. When the functionals involve derivatives of u, however, those derivatives tend to outweigh the small scales in the functional. We show in this section how the approximation of $J(u)$ can be improved to incorporate some of the unresolved scale effects through sensitivity information. Numerical

tests of the improvement of flow functionals by using sensitivity information (with accompanying mathematical support) have yielded positive results for a different eddy viscosity model in [17].

We suppose that the LES solution w implicitly defines a smooth function of $\delta, w = w(\delta)$, with the property that

$$w(\delta) \to u \quad \text{as} \quad \delta \to 0. \tag{3.1}$$

This is a minimal analytic consistency condition for an LES, which nevertheless has so far been proven to hold for only a few large eddy models; see, for example, [5, 15]. If the functional is C^1, then the composition

$$\delta \mapsto J(w(\delta)) =: J(\delta)$$

defines a smooth map. The value $J(\delta) = J(w(\delta))$ is computable while $J(u) = J(w(0)) = J(0)$ is sought. Since δ is small, the linear approximation to $J(0)$ is justified. This yields the following approximation to $J(u)$:

$$J(u) \doteq J(w(\delta)) - \delta J'(w(\delta)) \cdot w_\delta. \tag{3.2}$$

The increment $\delta J'(w(\delta))$ incorporates effects of unresolved scales on $J(u)$ and is computable once the solution sensitivities w_δ are calculated.

If the functional itself is regularized, $J(u)$ is approximated by a δ-dependent approximation $J(\delta, w(\delta))$. Then (3.2) is modified by

$$J(u) \doteq J(\delta, w(\delta)) - \delta(J_w(\delta, w(\delta)) \cdot w_\delta + J_\delta(\delta, w)). \tag{3.3}$$

3.1. Example: Lift, drag, and other fluid-boundary interactions in turbulent flows. In many applications, forces exerted by the fluid on its boundaries must be estimated. In this case the functional is given by

$$J(u, p) := \oint_B \hat{n} \cdot [pI - 2\nu\nabla^s u] \cdot \hat{a} \, ds, \tag{3.4}$$

where $\hat{a}$ is a unit vector and B is the boundary of the immersed body. If $\hat{a}$ points in the direction of motion, $J(u)$ represents drag, whereas if $\hat{a}$ points in the direction of gravity, $J(u)$ represents lift.

For general surfaces B, the straightforward approximation of $J(u, p)$ by $J(w, q)$ is not so clear because (see, for example, [18]) while $w(\delta) \to u$ as $\delta \to 0$, $q(\delta) \to p + 1/3 \cdot \text{trace}\,(R)$, where R is the large eddy subgrid stress tensor

$$R_{ij} := \overline{u}_i \overline{u}_j - \overline{u_i u_j}.$$

Thus, for a general surface B, $q(\delta)$ is not a direct approximation to p, and its use in (3.4) could skew the estimate of the force on that surface B. For a wall B, the situation is clearer. Indeed, let $k(v) := 1/2|v|^2(x, t)$ denote the kinetic energy distribution of a velocity field v. Then we have

$$q(\delta) = p + (2/3)(k(\overline{u}) - \overline{k(u)}). \tag{3.5}$$

Since w approximates $\overline{u}$, the excess pressure contribution to q from $k(\overline{u})$ is computable and thus correctable by using $k(w)$, but that contributed by $\overline{k(u)}$ is not easily calculable for general surfaces. If B is a solid wall and the averaging operator, such as

(2.9) and (2.10), preserves the no-slip condition, then $k(\overline{u}) = \overline{k(u)} = 0$ on B, and p can be estimated on B by $q(\delta)$.

Since the boundary-force functional $J(w, q)$, given by (3.4), is a linear functional $J' = J$, we obtain the corrected approximation to the force on B:

$$J(u,p) \sim J(w,q) - \delta J(w_\delta, q_\delta) = J(w - \delta w_\delta, q - \delta q_\delta)$$

$$= \oint_B \hat{n} \cdot [(q - \delta q_\delta)I - 2\nu\nabla^s(w - \delta w_\delta)] \cdot \hat{a}\ ds. \tag{3.6}$$

3.2. Example: Flow matching. In flow matching, a desired velocity field u^* is specified, the flow is simulated, a functional such as

$$J(u) := \frac{1}{2}\int_{\Omega\times(0,T)} |u - u^*|^2 dx\ dt$$

is calculated, and the design or control parameters are used to drive $J(\cdot)$ to its minimum value. Thus, one aspect of flow matching involves getting the best estimate of $J(\cdot)$; this is challenging in the case of a turbulence flow.

Abstractly, given the LES velocity w and its sensitivity w_δ, (3.2) provides an estimate of $J(u)$, improving the estimate given by $J(w)$. Since $J(\cdot)$ is quadratic, it is straightforward to calculate

$$J'(w)w_\delta = \int_{\Omega\times(0,T)} (w - w^*) \cdot w_\delta dx\ dt,$$

giving the approximation

$$J(u) \doteq \int_{\Omega\times(0,T)} \frac{1}{2}|w - u^*|^2 - \delta(w - u^*) \cdot w_\delta dx\ dt. \tag{3.7}$$

4. Approximating sensitivities requires a finer mesh. If the selected LES model is accurate, its solution will not vary over scales smaller than $O(\delta)$.[2] On the other hand, if the sensitivity w_δ captures some information in fluctuations through the expansion

$$u \doteq w - \delta w_\delta + O(\delta^2),$$

then w_δ will vary on scales smaller than $O(\delta)$. Formal asymptotics can give some preliminary insight into the width of transition regions of w_δ (and thus into mesh design for the sensitivity equation). The correct beginning point for this is an underresolved laminar flow with a simple Prandtl boundary layer along a flat plate. This is far from the case of turbulence, of course, but it is the case where asymptotic analysis of transition regions is best understood. (We stress that this section is only a beginning point in understanding meshes needed to resolve sensitivity equations.)

The simplest interesting eddy viscosity is a constant $\nu_T = O(\delta^2)$, chosen so that the $O(\nu_T^{\frac{1}{2}})$ boundary layer in w is spread over an $O(\delta)$ transition region. We thus suppose that $\nu \ll \nu_T = O(\delta^2)$, and the flow domain in the upper half plain and the

[2] A general proof of this is open for almost all LES models. Even for homogeneous isotropic turbulence, estimates are not available for the smallest persistent scale in the solution of many LES models.

model velocity approach a fixed value $(v, 0)$. Here v increases from $v = 0$ to $v = O(1)$ rapidly as y increases above the plate.

Consider thus the equilibrium problems for w,

$$w \cdot \nabla w - (\nu + \nu_T)\delta w + \nabla q = 0,$$

and its sensitivity equation for $s = w_\delta$,

$$s \cdot \nabla w + w \cdot \nabla s - (\nu + \nu_T)\delta s - \nu_T'(\delta)\delta w + \nabla q_\delta = 0.$$

We expect the x-sensitivity s_1 to go from 0 to $O(1)$ back to near 0 as y increases from the wall. Supposing that this result occurs over an $O(\alpha)$ transition region (and following the analysis of laminar boundary layers, e.g., [8]), we consider the x-component of the sensitivity equation:

$$w_1 s_{1,x} + w_2 s_{1,y} + s_1 w_{1,x} + s_2 w_{1,y} - \nu_T(s_{1,xx} + s_{1,yy}) - \nu_T'(\delta)(w_{1,xx} + w_{1,yy}) + q_{\delta,x} = 0.$$

The relative orders of magnitude in the transition region of each term in the above equation are

$$O(1)O(1) + O(\delta)O(\alpha^{-1}) + O(1)O(1)$$
$$+O(\alpha)O(\delta^{-1}) + O(\delta^2)(O(1) + O(\alpha^{-2})) + O(\delta)(O(1) + O(\delta^{-2})) = 0.$$

If we keep in mind the expectation that $\alpha < \delta$, the dominant terms are then $\left(\frac{\delta}{\alpha}\right)^2$ and $\frac{1}{\delta}$. Equating this suggests that $\alpha \sim \delta^{\frac{3}{2}} \cong \delta^{\frac{1}{2}}\nu_T^{\frac{1}{2}}$, suggesting that [illegible] w [illegible] $\delta^{\frac{1}{2}}$. This prediction is confirmed in the numerical tests in section 5; see especially Figure 5.5.

5. Numerical simulations. We use the FreeFem package [11] to validate the analytical findings concerning the width of the boundary layer of the sensitivity w_δ and to assess the estimation potential of the sensitivity-based method.

For validation, we used a fine mesh to calculate the "true" Navier–Stokes solution corresponding to $\delta = 0$, which subsequently serves as a starting point for the sensitivity calculations. We will investigate the behavior of the sensitivity calculations by either divided differences, as outlined in section 2, or by numerically integrating (2.6).

When divided differences are used, they are computed on the finer mesh used to calculate the "true" Navier-Stokes solution. In this way we uncouple the important issue of continuous versus discrete sensitivities from evaluating either's potential for improved estimation of flow functionals. The "true" mesh is displayed in Figure 5.1. Of course, normally the LES+sensitivity calculations are performed on a mesh coarser than that required for the Navier–Stokes equation, and numerical errors in the sensitivity calculation must be studied. For first steps in the numerical analysis of sensitivity equations in computational fluid dynamics, see [17]. Moreover, we investigate the dependence on the mesh size of the error in the sensitivity calculations in section 5.2.3.

5.1. Computational setup. For our numerical simulations, we consider the rectangular domain from Figures 5.2 and 5.3. The domain is eight units long and two units wide. A disk of radius 0.15 is placed at $(3, 1)$. We impose no-slip boundary conditions on the horizontal walls and on the boundary of the disk, a do-nothing

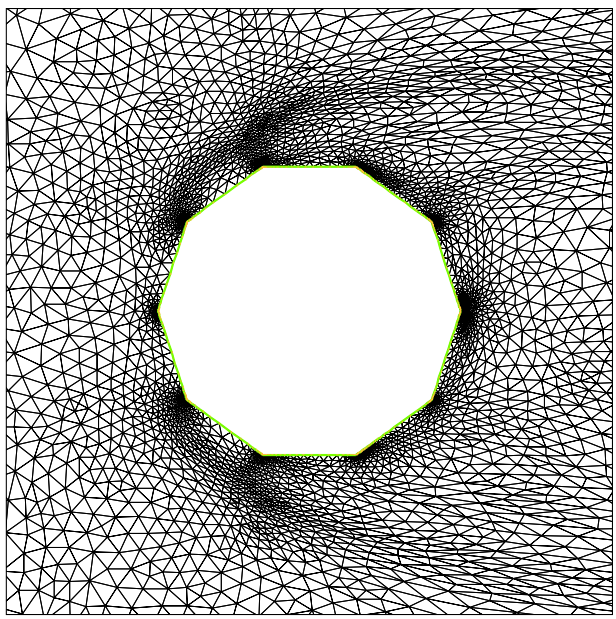


FIG. 5.1. *Final mesh, zoomed around the obstacle.*

condition at the outflow (right) boundary, and a time-independent parabolic profile at the inflow boundary, with the rule $3y(2-y)$. As initial conditions we use the solution of the steady-state Stokes problem with the same boundary conditions.

We use an implicit-explicit time-stepping scheme to approximate the solution of (2.1):

$$\frac{w^{k+1}-w^k}{\Delta t} + \nabla\cdot(w^k\ w^k) - \nu\Delta w^{k+1} + \nabla q^{k+1} - \nabla\cdot(\nu_T(\delta, w^k)\nabla^s w^{k+1}) = 0, \tag{5.1}$$

$$\nabla\cdot w^{k+1} = 0,$$

where we denote the velocity at time step k by w^k. The problem is subsequently discretized in space by a Taylor–Hood finite element from FreeFem [11] and is implemented as a FreeFem script, where, given w^k, we compute w^{k+1}. We choose a time step $\Delta t = 0.015$ and an inverse of the Reynolds number $\nu = \frac{1}{200}$, and we run the computation for 500 steps. The approximation of a circular cylinder by one with a polygonal cross section with 10 faces originates in our need to keep the initial mesh of a reasonable size. The shape of the obstacle is subsequently preserved, even if the mesh is refined. The mesh is initialized to have around 10,000 degrees of freedom on the first pass and is subsequently refined every 50 time steps. The densest portion of the final mesh is presented in Figure 5.1. We use the Smagorinsky model (2.2) with an appropriate choice of δ, specified below.

In Figures 5.2 and 5.3, respectively, we have presented the contour plot of the Euclidean norm of the velocity and the square root of the absolute value of the vorticity at four time steps of the simulation for $\delta = 0$. In the latter case, we chose the square root of the vorticity as a function to plot because it has the same contour plot as the vorticity function but is more sensitive in the range of interest. The results show a time-periodic behavior where vortices are generated behind the disk and shed by the flow which is what was observed in other computational studies. Animations of the

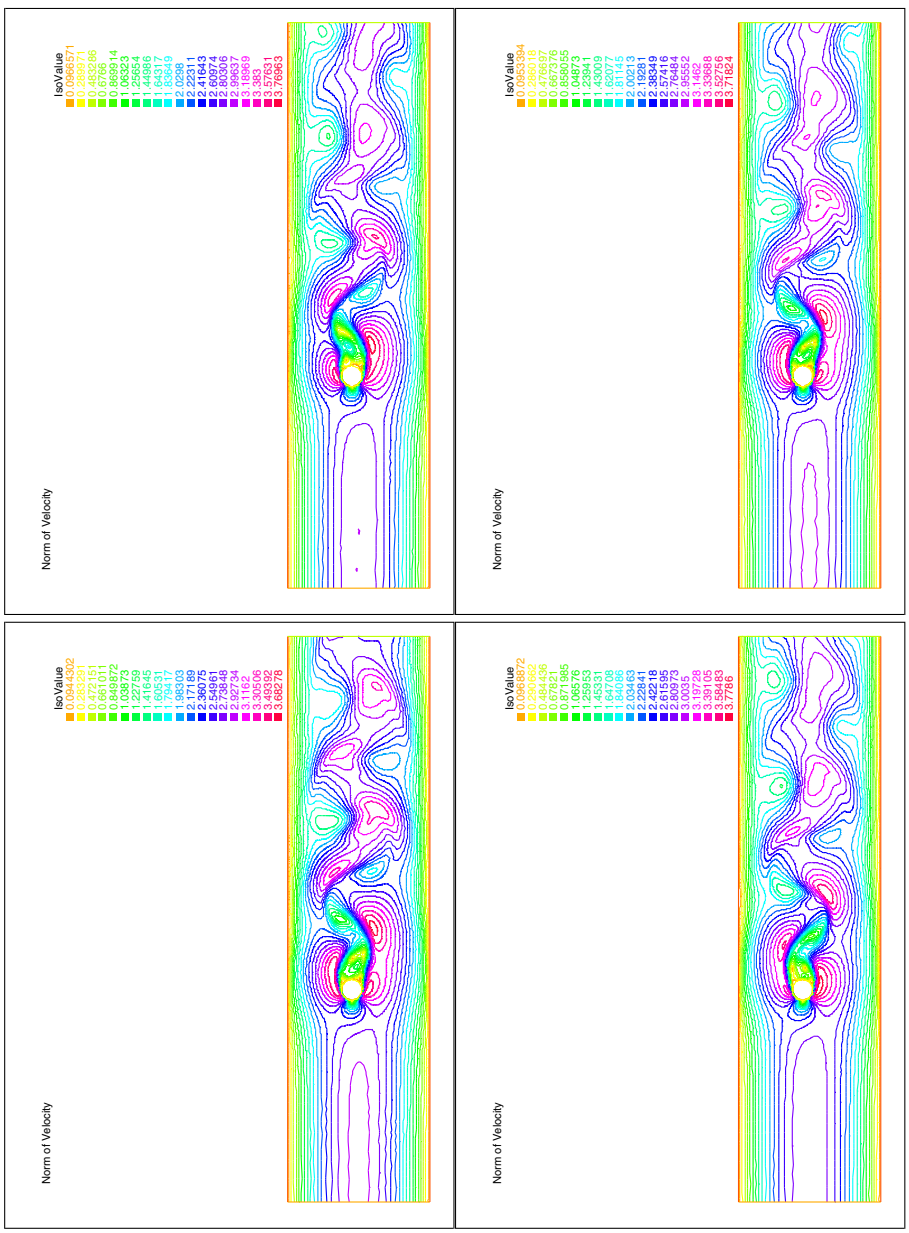


FIG. 5.2. *Contour plot of the velocity at* 1.65 *second time intervals, ending with the snapshot at* 7.5 *seconds.*

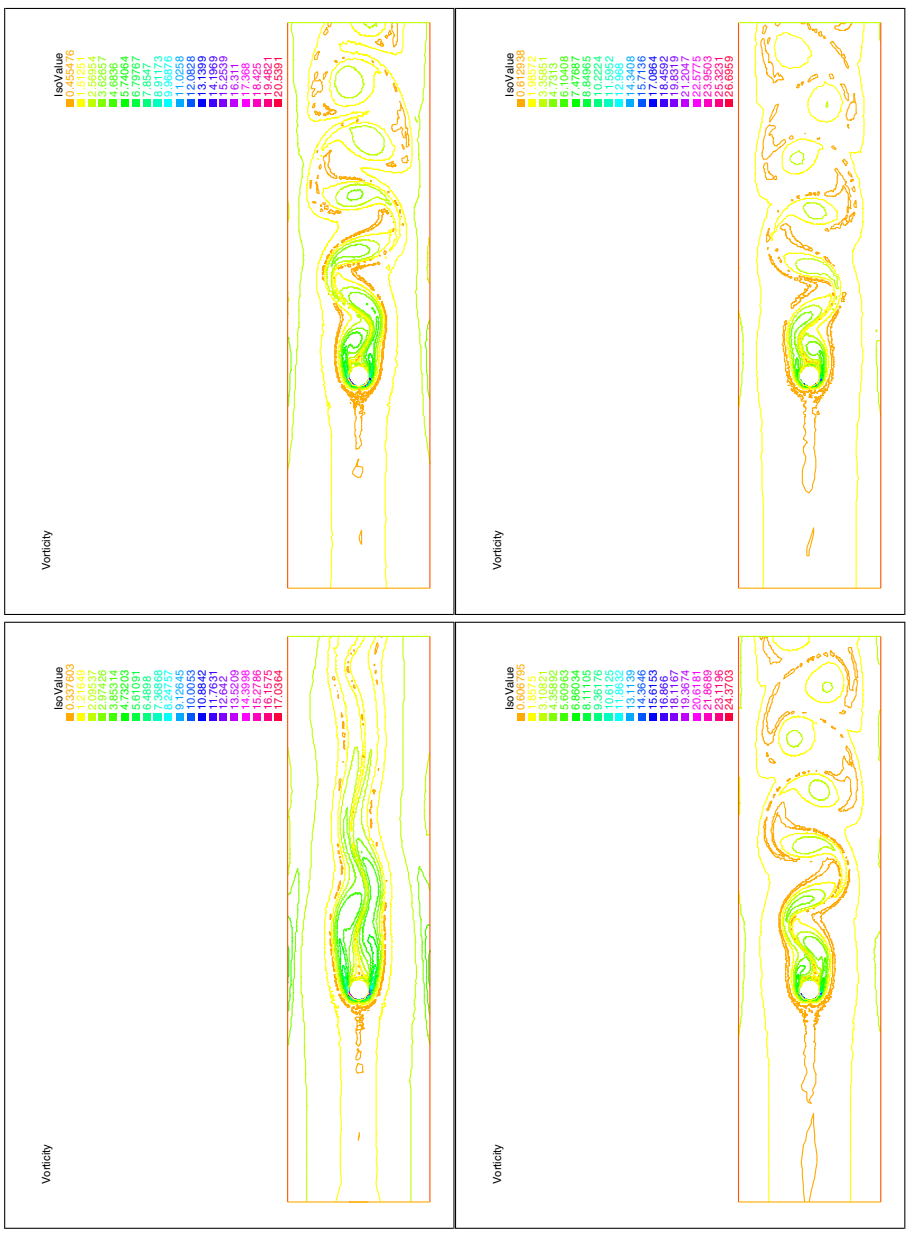


FIG. 5.3. *Contour plot of the square root of the vorticity at* 1.65 *second time intervals, ending with the snapshot at* 7.5 *seconds.*

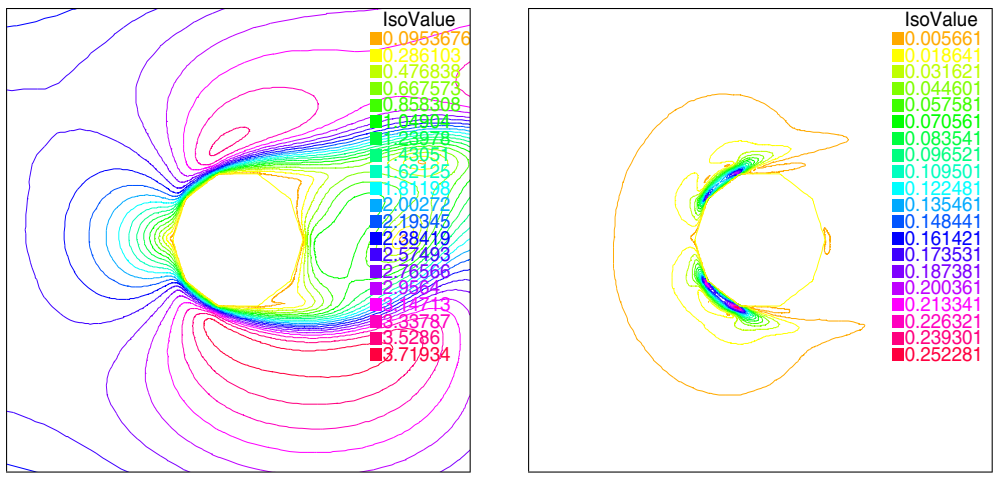


FIG. 5.4. *Velocity and sensitivity at* 7.5 *seconds.*

simulations that more clearly support this observation can be found at the website [1]. The result of the Navier–Stokes simulation at 7.5 seconds, which thus represents a realistic velocity distribution, is the starting point for our sensitivity estimations.

5.2. Results and discussions. In this subsection, we validate the various conclusions that we have drawn in sections 2 and 3.

5.2.1. Evaluation of the boundary layer of the sensitivity. We computed the sensitivity at time 7.5 seconds by running the numerical scheme (5.1) for one time step with $\delta = 0.05$, denoting the result by $\hat{w}^k$, and one time step with $\delta = 0$, denoting the result by w^k and approximating the sensitivity as

$$w_\delta^k = \frac{\hat{w}^k - w^k}{\delta}.$$

In Figure 5.4 we plot w^k and $|w^k - \hat{w}^k|$ side-by-side (sensitivity scaled by δ), where the latter is measured in the local 2-norm. The results show that the largest sensitivity occurs roughly at $\pm\frac{3\pi}{4}$, which is also the place where the boundary layer appears to be the thinnest for the velocity w^k.

Subsequently, we computed the w^k and w_δ^k profile on a direction that originates on the center of the disk at the angle -0.79π. In Figure 5.5 we plot the scaled versions of the sensitivity $|w_\delta^k|$, the velocity $|w^k|$, and the sensitivity $|w_\delta^k|$ stretched by $\delta^{-0.5}$, which we denote by $|w_\delta^k(\delta^{-0.5}\cdot)|$. The scaling is done by dividing the data by the largest values of the respective quantities along the specified direction. We see that the position of the maximum of the stretched sensitivity $|w_\delta^k(\delta^{-0.5}\cdot)|$ coincides with the maximum of the velocity $|w^k|$, which validates our theoretical finding that the boundary layer of the sensitivity is $\delta^{0.5}$ thinner than the boundary layer of the velocity.

5.2.2. Improving functional estimation. We now turn to the problem of estimating functionals of the Navier–Stokes flow $\delta = 0$ using computations with $\delta \neq 0$ followed by sensitivity-based correction.

When doing sensitivity-based correction, it is better to use the sensitivity with respect to $\gamma = \delta^2$, for reasons that will become immediately clear. If we denote the

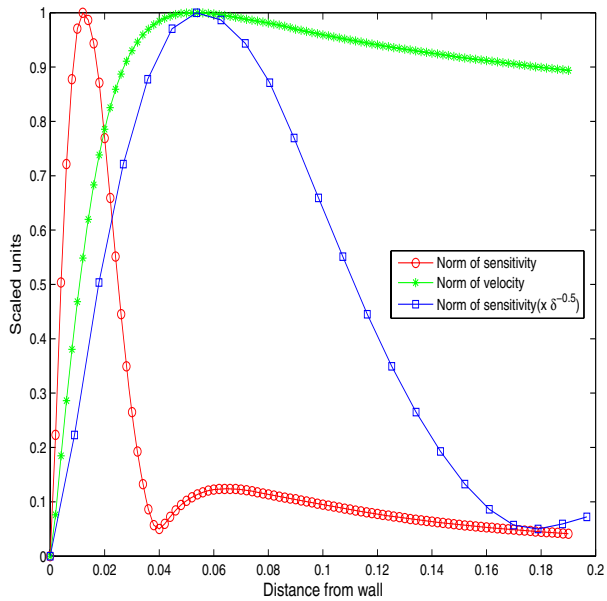


FIG. 5.5. *Comparison of the scaled sensitivity and velocity on a direction at* -0.79π *from the center of the disk.*

dependence of a functional J on γ by $\hat{J}(\gamma)$ and on δ by $J(\gamma)$, the chain rule gives us

$$J'(\delta) = \frac{dJ(\delta)}{d\delta} = 2\delta \frac{d\hat{J}(\gamma)}{d\gamma} = 2\delta \hat{J}'(\gamma).$$

We therefore have that $J'(0) = 0$. Since the correction procedure outlined in (3.6) is tantamount to the use of linear extrapolation at δ to determine $J(0)$, it will perform better if variables are used which reflect the natural ones in the model (δ^2 for the Smagorinsky model) and for which $J'(0) \neq 0$.

We therefore used the sensitivity with respect to $\gamma = \delta^2$ as the vehicle to improve a functional J. As an initial test, we compute the values of $\|w_\delta\|$ and $\|w_\gamma\|$, and we display the results in Figure 5.6. The former is computed by using divided differences between velocity values at two successive δ values, whereas the latter is computed as $\frac{w_\delta}{2\delta}$. The computation is done for one time step of simulation using (5.1) for a range of time steps and parameters δ. We see that $w_\delta \to 0$ and $w_\gamma \to 0$ as $\Delta t \to 0$, as expected (since both sensitivity equations have zero initial conditions). In addition, $w_\delta \to 0$ as $\delta \to 0$, as predicted in the preceding paragraph.

In our functional improvement computations, we consider the computation of the drag functional (3.4) using the correction (3.6), where δ is replaced with γ, and where the sensitivities are either computed by divided differences or simulated using (2.6). Since we use a mesh that is finer than the one needed to resolve the sensitivity w_δ, and since the correction of the functional (3.4) is a linear function of w_δ, determining w_δ by divided differences and then using it in the correction is equivalent to computing the drag functional for $\delta \neq 0$ and then using divided differences for drag estimation at 0.

We computed the drag functional on the disk in the x direction, from the pressure and velocity derivative data, obtained from running the Smagorinsky model for one

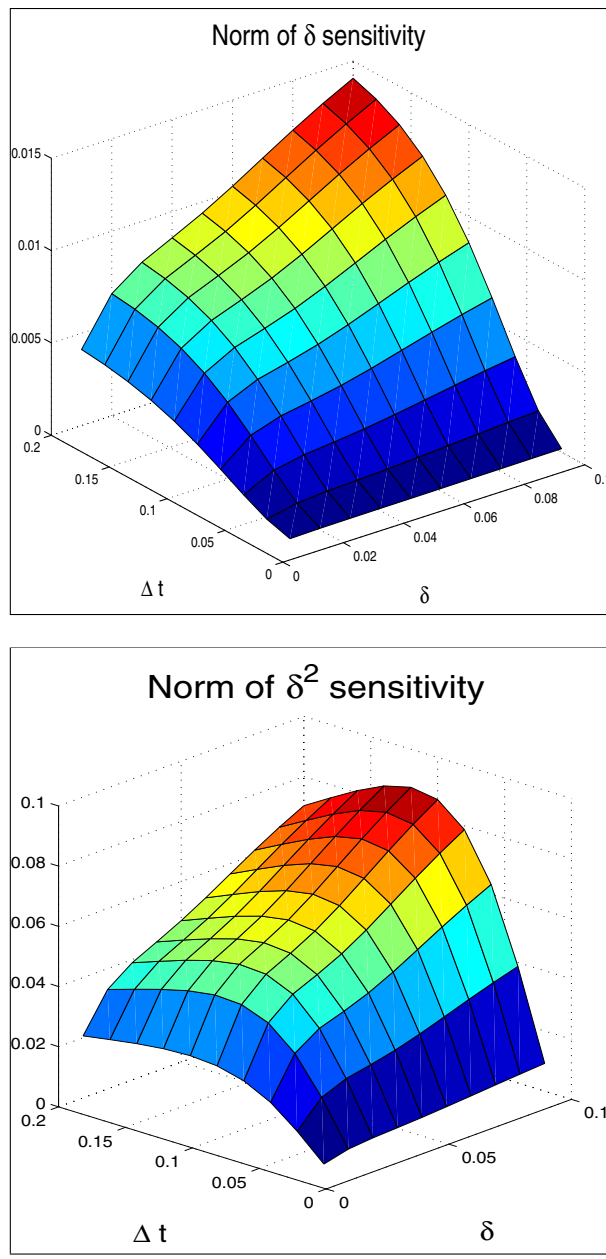


FIG. 5.6. *Sensitivity behavior for two models.*

0.015 second time step for the δ in Table 5.1. The computed values are presented in Table 5.1, with entry "Direct computation" in the "Method" column. We see that the correction using first-order sensitivities computed by numerically integrating the sensitivity equations (2.6) and using (3.6) results in a 77% reduction of the relative error, correction using first-order sensitivities computed by using divided differences using (3.6) results in a 91% reduction of the relative error, whereas correction using second-order sensitivities (quadratic correction) results in a 98% reduction of the

TABLE 5.1
Drag values.

δ	Drag	Method
0.190	-3.183	Direct computation
0.195	-3.224	Direct computation
0.200	-3.266	Direct computation
0.0	-1.455	Direct computation
0.0	-1.882	Linear correction from numerical sensitivity equation
0.0	-1.620	Linear correction from divided differences
0.0	-1.479	Quadratic correction

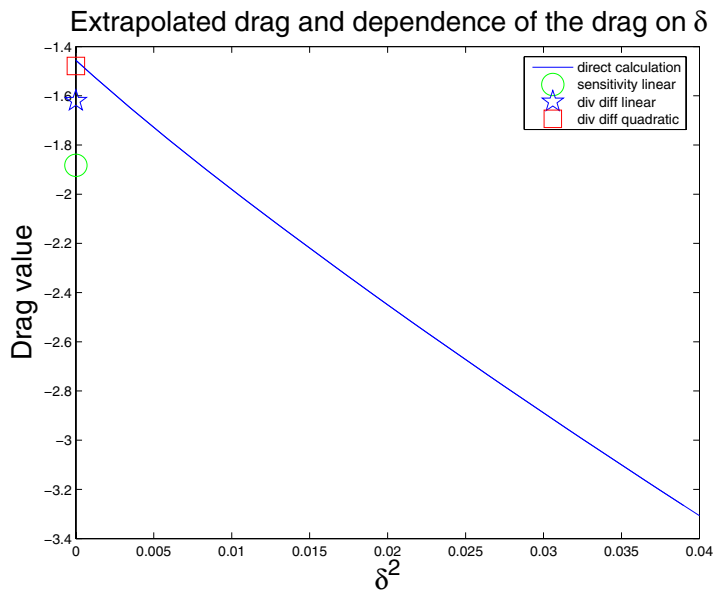


FIG. 5.7. *Computed drag as a function of δ and corrected drag.*

relative error, both compared to the case where the drag is estimated by using only its value at $\delta = 0.2$. We represent these results graphically in Figure 5.7, together with the drag values for δ in the $[0, 0.2]$ range. Both the corrected values and their methods are listed in Table 5.1. The difference in the sensitivities computed by divided differences and the ones computed by numerical integration is reasonably small in size, not surprisingly, since we also incur a time discretization error. The 91% error reduction is asymptotically achievable for increasingly accurate sensitivity computation.

While we did not explicitly describe a higher-order sensitivity development, that extension is straightforward, and we interpret the above results as evidence of the excellent potential of the sensitivity-based estimation method described in this work.

5.2.3. Mesh size-dependence of the sensitivity error. We now numerically investigate our statement about the sensitivity needing to be a result on a finer mesh than the velocity. Since "resolved flow" cannot be easily quantified, we turn our focus to the dependence of the sensitivity error and velocity error on the mesh size.

A comprehensive numerical test for the asymptotic behavior of the error as a function of the mesh size is difficult for the following reason. Since we do not have a

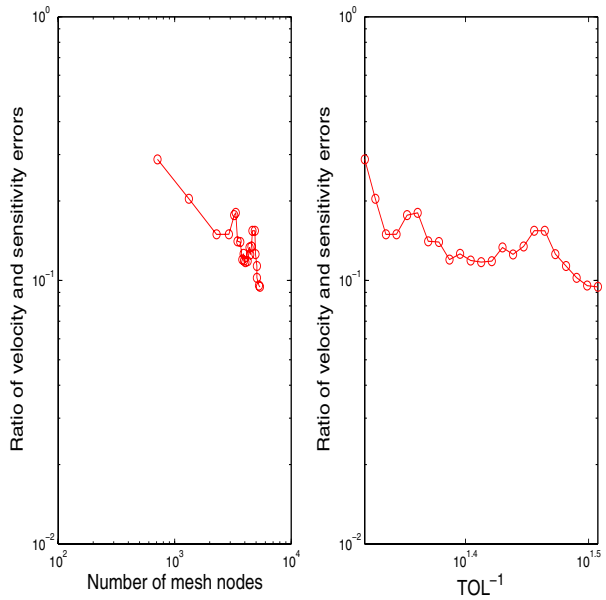


FIG. 5.8. *Mesh-dependency of velocity and velocity sensitivity error on the obstacle, plotted with respect to the number of mesh nodes and with respect to the size of the tolerance parameter that was used in the routine* `adaptmesh` *to generate that mesh.*

rectangular domain (because of the circular inclusion) we cannot use a structured grid of an arbitrary size h. We must use an unstructured mesh, for which an asymptotic analysis requires the regular division with the same factor of all elements of the mesh. If we use a factor of two, after three such divisions, the size of the linear system that is created by one step of the problem (5.1) cannot be solved on our computer. Therefore such an analysis will generate preciously few points.

We will therefore carry out two experiments. The first experiment creates quite a few data points for an increasing number of mesh nodes, though the meshes are not in the same asymptotic family. The second example will generate only three data points, but the meshes are obtained by a regular division procedure by a factor of two and is suitable for proper asymptotic analysis. In both these experiments we integrate numerically the partial differential equation of the sensitivity (2.6) with the same type step but on different meshes.

In the first experiment, the sensitivity is computed at time $t = 1.125$ on a circle of radius 0.151, cocentric with the obstacle, by computing one step of the linear partial differential equation (2.6). The velocity profile at $t = 1.125$ is obtained by using the numerical scheme (5.1) with a time step of Δt and by refining the mesh every three time steps, using the routine `adaptmesh` of FreeFem. When calling this routine, the tolerance parameter is initialized at 0.5 and is decreased every five time steps by a factor of 1.02. This generates 25 meshes, for each of which we compute the velocity v and its sensitivity w_γ for $\delta = 0.2$ (and $\gamma = \delta^2 = 0.04$), for one time step of length 1.5×10^{-4}. We estimate the error (heuristically) by subtracting the values obtained for the sensitivity and the velocity on the finest mesh from all the other sensitivity and velocity values. We plot the ratio of the velocity and sensitivity errors in doubly logarithmic scale in Figure 5.8.

In the second experiment, we use the numerical scheme (5.1) with a time step of

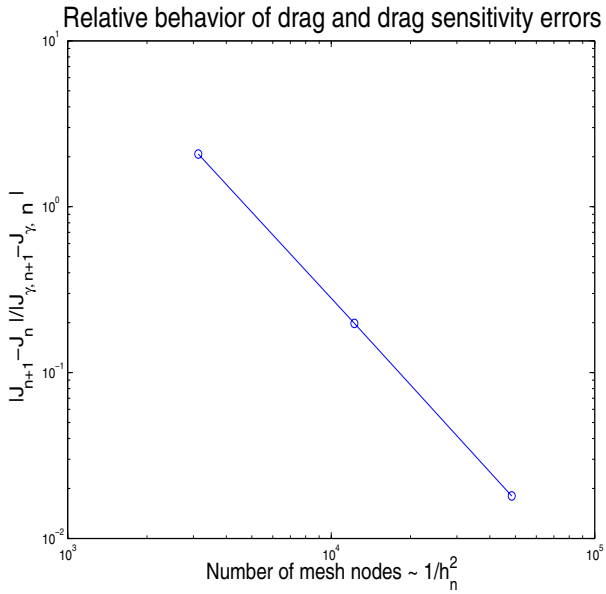


FIG. 5.9. *Mesh-dependency of drag and drag sensitivity error.*

$\Delta = 0.015$ to compute the velocity at the time $t = 7.5$ (as in the numerical experiments of the preceding sections), and the sensitivity w_γ and, respectively, the velocity v are also computed from the time discretization of (2.6) and from the scheme (5.1), both for one time step of length 1.5×10^{-4}, on four nested meshes. Each of these meshes, for which we denote the diameter of the typical element by h_n, $n = 1, 2, 3, 4$, is obtained by uniformly refining the preceding mesh by a factor of two. Therefore $h_{n+1} = \frac{h_n}{2}$, $n = 1, 2, 3$. Once the numerical values for w_γ and v are obtained, the drag J_n and the drag sensitivity $J_{\gamma,n}$, for $n = 1, 2, 3, 4$, are computed on the same circle as in the preceding experiment, using formulas (3.4) and (3.6), where we substitute γ for δ. In Figure 5.9 we plot on a doubly logarithmic scale the factor

$$u_{n+1} = \frac{J_{n+1} - J_n}{J_{\gamma,n+1} - J_{\gamma,n}}, \quad n = 1, 2, 3.$$

We see that both Figures 5.8 and 5.9 are strong evidence for the fact that the error in sensitivity improved more slowly than did the error in the velocity for increasing the finer meshes. This finding supports our statement that, for the same level of error, a finer mesh is needed for sensitivity. In addition, the almost perfect linear behavior (although from only three points) in Figure 5.9 suggests that both the error in drag and drag sensitivity behave like a power of the mesh size h, with smaller exponent for the sensitivity.

6. Conclusions and future prospects. Many fundamental theoretical questions in LES are still open, and many of these are important for the validation of the sensitivity approach to estimating model uncertainty. Nevertheless, we have seen that the continuous sensitivity equation approach in these tests has given a remarkably accurate estimate of the solution change as the length scale parameter varies, [illegible] itself. Formal asymptotics in an oversimplified setting gives preliminary estimates of the mesh-resolution requirements of the continuous sensitivity equation that are, again, remarkably accurate in our test.

The success of the sensitivity equation approach as shown here (and in other tests) suggests its broader usefulness and hints at the possibility of establishing a general analytic foundation for the method in LES.

Acknowledgments. We are grateful to the referees for their helpful comments that have improved the quality of the paper. We are also grateful to Paul Fischer for helpful comments that have improved the quality of the paper.

REFERENCES

[1] M. Anitescu, *Fluid simulations*, http://www.mcs.anl.gov/∼anitescu/FLUIDS/fluids.htm.
[2] L. C. Berselli, T. Iliescu, and W. J. Layton, *Mathematics of Large Eddy Simulation of Turbulent Flows*, Springer-Verlag, Berlin, 2006.
[3] J. Borggaard and J. Burns, *A PDE sensitivity equation method for optimal aerodynamic design*, J. Comput. Phys., 136 (1997), pp. 366–384.
[4] J. Borggaard and A. Verma, On efficiently *solutions to the continuous sensitivity equation using automatic differentiation*, SIAM J. Sci. Comput., 22 (2000), pp. 39–62.
[5] Q. Du and M. Gunzburger, *Analysis of a Ladyzhenskaya model for incompressible viscous flow*, J. Math. Anal. Appl., 155 (1991), pp. 21–45.
[6] A. Dunca, V. John, and W. Layton, *The commutation error of the space averaged Navier Stokes equations on a bounded domain*, in Contributions to Current Challenges in Mathematical Fluid Mechanics, Adv. Math. Fluid Mech. 3, G. Galdi, J. Heywood, and R. Rannacher, eds., Birkhäuser-Verlag, Basel, Switzerland, 2004, pp. 53–78.
[7] M. Germano, *Differential filters for the large eddy simulation of turbulent flows*, Phys. Fluids, 29 (1986), pp. 1755–1757.
[8] S. Goldstein, *Modern Developments in Fluid Dynamics*, Vol. 1, Dover, New York, 1965.
[9] M. Gunzburger, *Sensitivities, adjoints, and flow optimization*, Int. J. Numer. Methods Fluids, 31 (1999), pp. 53–78.
[10] M. D. Gunzburger, *Perspectives in Flow Control and Optimization*, Adv. Design Controls 5, SIAM, Philadelphia, 2002.
[11] F. Hecht, O. Pironneau, and K. Ohtsuka, *FreeFem++ Manual*, http://www.freefem.org.
[12] T. Iliescu and W. Layton, *Approximating the larger eddies in fluid motion* III: *The Boussinesq model for turbulent fluctuations*, Analele Stinfice ale Universitatii "Al. I. Cuza" Iasi, 44 (1998), pp. 245–261.
[13] V. John, *Large Eddy Simulation of Turbulent Incompressible Flows. Analytical and Numerical Results for a Class of LES Models*, Lecture Notes in Comput. Sci. Engrg. 34, Springer-Verlag, Berlin, 2004.
[14] V. John, W. Layton, and N. Sahin, *Derivation and analysis of near wall models for channel and recirculating flows*, Comput. Math. Appl., 48 (2004), pp. 1135–1151.
[15] M. Kaya and W. Layton, *On verifiability of models of the motion of large eddies in turbulent flows*, Differential and Integral Equations, 15 (2002), pp. 1395–1407.
[16] W. Layton and R. Lewandowski, *Analysis of an eddy viscosity model for large eddy simulation of turbulent flows*, J. Math. Fluid Mech., 2 (2002), pp. 374–399.
[17] F. Pahlevani, *Sensitivity Analysis of Eddy Viscosity Models*, Ph.D. thesis, Department of Mathematics, University of Pittsburgh, Pittsburgh, PA, 2004.
[18] P. Sagaut, *Large Eddy Simulation for Incompressible Flows*, Springer, New York, 2001.
[19] P. Sagaut and T. Lê, *Some Investigations of the Sensitivity of Large Eddy Simulation*, Technical report 1997-12, ONERA, Toulouse, France, 1997.
[20] L. G. Stanley and D. L. Stewart, *Design Sensitivity Analysis: Computational Issues of Sensitivity Equation Methods*, Frontiers in Appl. Math. 25, SIAM, Philadelphia, 2002.

SIAM J. SCI. COMPUT.
Vol. 29, No. 4, pp. 1668–1682
© 2007 Society for Industrial and Applied Mathematics

A CORNER CUTTING ALGORITHM FOR EVALUATING RATIONAL BÉZIER SURFACES AND THE OPTIMAL STABILITY OF THE BASIS*

JORGE DELGADO† AND J. M. PEÑA‡

Abstract. The usual method for evaluating rational Bézier surfaces uses the projection operator and the representation provided by the Bernstein basis. We prove the optimal stability of the basis used in this representation. We also propose an alternative method for evaluating rational surfaces through that representation. We show the stability properties of this last method and prove that it has better properties than other known methods from the point of view of avoiding underflow and overflow.

Key words. rational surfaces, optimal stability, underflow, overflow, error analysis

AMS subject classifications. 65G50, 65D17, 41A20

DOI. 10.1137/060649148

1. Introduction.

Given a matrix of positive weights $(w_{ij})_{0\le i\le m;0\le j\le n}$ and a matrix of control points $(P_{ij})_{0\le i\le m;0\le j\le n}$ in $\mathbf{R}^3$, we define the rational Bézier surface

$$F(x,y)=\sum_{i=0}^{m}\sum_{j=0}^{n}P_{ij}\,\frac{w_{ij}\,b_i^m(x)\,b_j^n(y)}{\sum_{i=0}^{m}\sum_{j=0}^{n}w_{ij}\,b_i^m(x)\,b_j^n(y)},\quad (x,y)\in[0,1]\times[0,1], \tag{1.1}$$

where for any positive integer r and $0\le i\le r$, b_i^r denotes the Bernstein polynomial

$$b_i^r(x)=\binom{r}{i}x^i(1-x)^{r-i},\quad i\in\{0,1,\dots,r\}.$$

The evaluation of rational Bézier surfaces is an important task in the field of geometric modeling (cf. [2], [5], and [12]) as well as in other fields such as finite elements (cf. [1, 16]). The usual method for evaluating rational Bézier surfaces uses the projection operator Π and the representation (1.1) through the Bernstein basis. Evaluation algorithms of nested type and with lower complexity have also been considered (cf. [15]). In the case of evaluating bivariate polynomials defined on triangulations, such an algorithm has been called VS in [14], so that here we also refer to the corresponding algorithm for rational surfaces as the VS algorithm. Finally, the possibility of obtaining corner cutting algorithms for the evaluation of rational surfaces has also been suggested in [2, p. 299]. In this paper we present a corner cutting algorithm for the evaluation of rational Bézier surfaces, we carry out its error analysis, and we show its better behavior with respect to the problem of avoiding underflow and overflow. Thus, the algorithm proposed in this paper is more robust than the other two alternative algorithms. In section 2 we recall the mentioned evaluation algorithms for

*Received by the editors January 5, 2006; accepted for publication (in revised form) January 31, 2007; published electronically August 1, 2007. This research was partially supported by the Spanish Research grant MTM2006-03388 and by Gobierno de Aragón and Fondo Social Europeo.
http://www.siam.org/journals/sisc/29-4/64914.html
†Departamento de Matemáticas (Sección Departamental de Gijón), Universidad de Oviedo, Spain (delgadojorge@uniovi.es).
‡Departamento de Matemática Aplicada, Universidad de Zaragoza, 50009 Zaragoza, Spain (jmpena@unizar.es).

rational Bézier surfaces and also present the new evaluation algorithm that also uses the representation (1.1) provided through the Bernstein basis. We call it a corner cutting rational surface algorithm because [illegible] steps are given by convex combinations of previously computed numbers. It is well known that corner cutting algorithms present good stability properties (cf. [7]). In section 3 we perform an error analysis of the new algorithm and show its backward stability.

Let us recall that the bases convenient in computer aided design are bases of nonnegative functions. Bases optimally stable among all bases formed by nonnegative functions have been identified for several spaces of univariate functions (see [3], [9], and [12]) and for some spaces of multivariate functions [6]. In section 4 we prove that the basis used in the representation (1.1) has optimal stability properties.

The numerical examples of section 5 illustrate the nice behavior, in general, of the three algorithms mentioned in the paper even for functions generalizing Wilkinson polynomials (cf. [17], [18], [19]). The ill-conditioning of these polynomials is well known, but the three algorithms considered in this paper present good behavior even for these polynomials. Analogously to the generalization of the Wilkinson polynomials, we have considered another very ill-conditioned function. In this case, the VS algorithm does not present good behavior, in contrast to the other two algorithms. Moreover, we show with very simple examples that the usual rational evaluation algorithm and the VS algorithm can present problems of underflow and overflow, in contrast to the corner cutting rational surface algorithm presented in this paper. In fact, we prove in section 5 that this algorithm avoids overflow and underflow and has optimal growth factor. We finish in the last section by summarizing our main conclusions.

2. Evaluation algorithms. Now, let us consider the following Bernstein tensor product:

$$\widetilde{F}(x,y)=\sum_{i=0}^{m}\sum_{j=0}^{n}\widetilde{P}_{ij}\,b_i^m(x)\,b_j^n(y), \tag{2.1}$$

where $\widetilde{P}_{ij}:=w_{ij}\,(P_{ij},1)^T$. The usual method of evaluating a rational Bézier surface (1.1) at a point $(x,y)\in[0,1]\times[0,1]$ uses the projection operator Π defined by $\Pi(x,y,z,w)=(x/w,y/w,z/w)$, taking into account that, by formulas (1.1) and (2.1),

$$\Pi(\widetilde{F}(x,y))=F(x,y). \tag{2.2}$$

Thus the algorithm can be described as follows: let us denote $f_{ij}^{00}:=w_{ij}\,(P_{ij},1)^T$ for all $i\in\{0,1,\dots,m\}$ and $j\in\{0,1\dots,n\}$; then we obtain the coefficients f_{ij}^{0r} ($i=0,1,\dots,m$, $r=1,\dots,n$, and $j=0,1,\dots,n-r$) performing

$$f_{ij}^{0r}=(1-y)\,f_{ij}^{0,r-1}+y\,f_{i,j+1}^{0,r-1}.$$

Finally we obtain the coefficients f_{i0}^{rn} ($r=1,\dots,m$ and $i=0,1,\dots,m-r$) performing

$$f_{i0}^{rn}=(1-x)\,f_{i0}^{r-1,n}+x\,f_{i+1,0}^{r-1,n}.$$

Then $f_{00}^{mn}=\widetilde{F}(x,y)$ and, by (2.2), $\Pi(f_{00}^{mn})=F(x,y)$. Due to the use of the projection operator, from now on we refer to this algorithm as the [illegible]. We can easily check that the previous algorithm consists of $2(m+1)n(n+1)+2m(m+1)$

sums, $4(m+1)n(n+1)+4m(m+1)+3(m+1)(n+1)$ multiplications, and three quotients.

On the other hand, in [14] Schumaker and Volk introduced a new algorithm for the evaluation of bivariate polynomials on triangulations which was called the VS algorithm. This algorithm presents lower complexity than the de Casteljau algorithm and has a nested nature, like the Horner algorithm. In [10] an error analysis of this polynomial evaluation algorithm was performed. The VS algorithm for bivariate polynomials on triangulations can be adapted for the evaluation of univariate polynomials. Taking into account this adaptation, in [8] a nested VS based algorithm for the evaluation of Bézier tensor products with less computational cost than the usual evaluation algorithm was presented. Analogously, in [15], using the projection operator Π, a nested VS based algorithm for the evaluation of tensor product rational Bézier surfaces with less computational cost than the usual was presented.

Now, let us consider the rational VS surface given by

$$G(x,y)=\sum_{i=0}^{m}\sum_{j=0}^{n}P_{ij}\,\frac{w_{ij}\,z_i^m(x)\,z_j^n(y)}{\sum_{i=0}^{m}\sum_{j=0}^{n}w_{ij}\,z_i^m(x)\,z_j^n(y)},\quad (x,y)\in[0,1]\times[0,1],$$

where $(P_{ij})_{0\le i\le m;0\le j\le n}$ is a matrix of control points, $(w_{ij})_{0\le i\le m;0\le j\le n}$ is a matrix of positive weights, and, for any positive integer r,

$$z_i^r(x)=x^i(1-x)^{r-i},\quad i\in\{0,1,\dots,r\}.$$

Analogously to the Bernstein case, denoting $\widetilde{P}_{ij}:=w_{ij}\,(P_{ij},1)^T$, we can deduce that the tensor product

$$\widetilde{G}(x,y)=\sum_{i=0}^{m}\sum_{j=0}^{n}\widetilde{P}_{ij}\,z_i^m(x)\,z_j^n(y)$$

satisfies $\Pi(\widetilde{G}(x,y))=G(x,y)$. So, analogously to the above projective algorithm, it is straightforward to deduce from the VS algorithm for univariate polynomials another projective algorithm for evaluating $G(x,y)$. From now on we refer to this algorithm as the [illegible].

We can check that the VS algorithm consists of $4(m+1)n+4m$ sums, $8(m+1)n+8m+3(m+1)(n+1)$ multiplications, and five quotients.

The two previous algorithms are not corner cutting algorithms. Now we propose an alternative algorithm for evaluating rational Bézier surfaces as in (1.1), which is a corner cutting algorithm.

Considering a componentwise evaluation of the rational Bézier surface (1.1), in order to perform the error analysis we shall consider from now on rational Bernstein–Bézier surfaces given by

$$F(x,y)=\sum_{i=0}^{m}\sum_{j=0}^{n}f_{ij}\,\frac{w_{ij}\,b_i^m(x)\,b_j^n(y)}{\sum_{i=0}^{m}\sum_{j=0}^{n}w_{ij}\,b_i^m(x)\,b_j^n(y)},\quad (x,y)\in[0,1]\times[0,1],\tag{2.3}$$

where $(f_{ij})_{0\le i\le m;0\le j\le n}$ is a matrix of real numbers and $(w_{ij})_{0\le i\le m;0\le j\le n}$ is a matrix of positive weights.

Observing that formula (2.3) can be written as

$$F(x,y)=\frac{\sum_{i=0}^{m}\left[\left(\sum_{j=0}^{n}\frac{f_{ij}\,w_{ij}\,b_j^n(y)}{\sum_{j=0}^{n}w_{ij}\,b_j^n(y)}\right)\left(\sum_{j=0}^{n}w_{ij}\,b_j^n(y)\right)\,b_i^m(x)\right]}{\sum_{i=0}^{m}\left(\sum_{j=0}^{n}w_{ij}\,b_j^n(y)\right)\,b_i^m(x)}$$

and denoting

$$w_i := \sum_{j=0}^{n} w_{ij}\, b_j^n(y), \tag{2.4}$$

$$f_i := \sum_{j=0}^{n} f_{ij}\, \frac{w_{ij}\, b_j^n(y)}{\sum_{j=0}^{n} w_{ij}\, b_j^n(y)} \tag{2.5}$$

for all $i \in \{0, 1, \ldots, m\}$, we can deduce that

$$F(x, y) = \sum_{i=0}^{m} f_i\, \frac{w_i\, b_i^m(x)}{\sum_{i=0}^{m} w_i\, b_i^m(x)}. \tag{2.6}$$

So, taking into account that in [2] was showed a corner cutting algorithm which evaluates rational Bézier curves (for instance, the rational Bézier curves in formulas (2.5) and (2.6)) and simultaneously provides the evaluation of the denominator of the corresponding rational curve (for instance, the values w_i in formula (2.4) corresponding to the denominators of the rational Bézier curves in (2.5)), we can deduce a corner cutting algorithm to evaluate surfaces $F(x, y)$ as in formula (2.3) at a point $(x, y) \in [0, 1] \times [0, 1]$ through the evaluation of $(m + 2)$ rational Bézier curves (the $(m + 1)$ of degree n in (2.5) and the other one of degree m in (2.6)). Then the algorithm can be described as follows: let us denote $f_{ij}^{00} := f_{ij}$ and $w_{ij}^{00} := w_{ij}$ for all $i \in \{0, 1, \ldots, m\}$ and $j \in \{0, 1 \ldots, n\}$; then we obtain the coefficients f_{ij}^{0r} $(i = 0, 1, \ldots, m,\ r = 1, \ldots, n,$ and $j = 0, 1, \ldots, n - r)$ performing

$$f_{ij}^{0r} = (1 - y)\, \frac{w_{ij}^{0,r-1}}{w_{ij}^{0r}}\, f_{ij}^{0,r-1} + y\, \frac{w_{i,j+1}^{0,r-1}}{w_{ij}^{0r}}\, f_{i,j+1}^{0,r-1} \tag{2.7}$$

with

$$w_{ij}^{0r} = (1 - y)\, w_{ij}^{0,r-1} + y\, w_{i,j+1}^{0,r-1}; \tag{2.8}$$

finally we obtain the coefficients f_{i0}^{rn} $(r = 1, \ldots, m$ and $i = 0, 1, \ldots, m - r)$ performing

$$f_{i0}^{rn} = (1 - x)\, \frac{w_{i0}^{r-1,n}}{w_{i0}^{rn}}\, f_{i0}^{r-1,n} + x\, \frac{w_{i+1,0}^{r-1,n}}{w_{i0}^{rn}}\, f_{i+1,0}^{r-1,n} \tag{2.9}$$

with

$$w_{i0}^{rn} = (1 - x)\, w_{i0}^{r-1,n} + x\, w_{i+1,0}^{r-1,n}; \tag{2.10}$$

so $f_{00}^{mn} = F(x, y)$. This algorithm will be called the [illegible]. Taking into account that the control points of a parametric rational function represented in the Bernstein basis belong to the set $\mathbf{R}^3$, we can easily check that the previous algorithm consists of $3(m + 1)n(n + 1) + 2m(m + 1)$ sums, $6(m + 1)n(n + 1) + 6m(m + 1)$ multiplications, and $3(m + 1)n(n + 1) + 3m(m + 1)$ quotients. Although the other two previous algorithms required only three and five quotients (respectively) and the division operation for floating point numbers has higher complexity than the other elementary operations, let us take into account that m and n are not very great in the context of geometric modeling and that the complexity order of this last algorithm is not higher than that of the projective algorithm

($\mathcal{O}(p^3)$ with $p = \max\{m, n\}$). Moreover, as we show in this paper, this last algorithm is more robust than the other two algorithms presented in this paper: it has good stability properties and even has clear advantages with respect to avoiding overflow and underflow.

3. Error analysis of the corner cutting rational surface algorithm. Let us now introduce some standard notation in error analysis. Given $a \in \mathbf{R}$, the computed element in floating point arithmetic will be denoted by either $fl(a)$ or by $\widehat{a}$. As usual, to investigate the effect of rounding errors we use the model

$$fl(a\ op\ b) = (a\ op\ b)\,(1+\delta), \quad |\delta| \le u, \tag{3.1}$$

with u the unit roundoff and op any of the elementary operations $+, -, \times, /$ (see [4, pp. 44–45] for more details). Given $k \in \mathbf{N}_0$ such that $ku < 1$, let us define

$$\gamma_k := \frac{ku}{1-ku} = ku + \mathcal{O}(u^2). \tag{3.2}$$

In our error analysis we shall deal with quantities satisfying that their absolute value is bounded above by γ_k. Following [4], we denote by θ_k such quantities, and let us take into account that, by Lemmas 3.3 and 3.1 of [4], the following properties hold:

$$(1+\theta_k)\,(1+\theta_j) = 1+\theta_{k+j}, \tag{3.3}$$

$$\frac{1+\theta_k}{1+\theta_j} = \begin{cases} 1+\theta_{k+j}, & j \le k, \\ 1+\theta_{k+2j}, & j > k. \end{cases} \tag{3.4}$$

As in [6], for any $x \in \Omega \subseteq \mathbf{R}^s$, basis $u = (u_0, \dots, u_n)$, and $f = \sum_{i=0}^n c_i u_i$, let us consider the relative condition number for the evaluation of f at x:

$$\operatorname{cond}(u; f, x) := \frac{\sum_{i=0}^n |c_i u_i(x)|}{\|\sum_{i=0}^n c_i u_i\|_\infty}.$$

The numerator $\sum_{i=0}^n |c_i u_i(x)|$ of the previous formula is the absolute condition number for the evaluation of the function f at x. Given $v = (v_0, \dots, v_n)$, let us denote $|v| := (|v_0|, \dots, |v_n|)$. The following result provides a forward error bound for the corner cutting rational surface algorithm.

THEOREM 3.1. [illegible] $F(x,y)$ [illegible] (2.3), [illegible] $(f_{ij})_{0\le i\le m; 0\le j\le n}$ [illegible] $(w_{ij})_{0\le i\le m; 0\le j\le n}$ [illegible] $\widehat{F}(x,y)$ [illegible]

$$|F(x,y) - \widehat{F}(x,y)| \le \gamma_k \sum_{i=0}^m \sum_{j=0}^n |f_{ij}| \frac{w_{ij}\, b_i^m(x)\, b_j^n(y)}{\sum_{i=0}^m \sum_{j=0}^n w_{ij}\, b_i^m(x)\, b_j^n(y)}, \tag{3.5}$$

[illegible] $k = (3m^2+5m) + (3n^2+5n) + 6mn$ [illegible] γ_k [illegible] (3.2).

[illegible]. We deduce from (2.7) that for all $r \in \{1, \dots, n\}$ and $i \in \{0, 1, \dots, m\}$,

$$f_i^{(r)} = V_{i,r}\, f_i^{(r-1)},$$

where $f_i^{(r)} := (f_{i0}^{0r}, \ldots, f_{i,n-r}^{0r})^T$ and $V_{i,r}$ is the $(n-r+1) \times (n-r+2)$ matrix of the form

$$V_{i,r} = \begin{pmatrix} \overline{x}_{i,0}^r & \overline{y}_{i,0}^r & 0 & \cdots & \cdots & 0 \\ 0 & \overline{x}_{i,1}^r & \overline{y}_{i,1}^r & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & \overline{x}_{i,n-r}^r & \overline{y}_{i,n-r}^r \end{pmatrix}$$

with

$$\overline{x}_{i,j}^r := (1-y)\,\frac{w_{ij}^{0,r-1}}{w_{ij}^{0r}}, \quad \overline{y}_{i,j}^r := y\,\frac{w_{i,j+1}^{0,r-1}}{w_{ij}^{0r}}, \quad j = 0, 1, \ldots, n-r.$$

Therefore, for all $k \in \{1, \ldots, n\}$ and $i \in \{0, 1, \ldots, m\}$, we have

$$f_i^{(k)} = V_{i,k} \cdots V_{i,1}\, f_i^{(0)}. \tag{3.6}$$

Analogously, we can also deduce from formula (2.8) that, for all $r \in \{1, \ldots, n\}$ and $i \in \{0, 1, \ldots, m\}$,

$$v_i^{(r)} = U_r\, v_i^{(r-1)},$$

where $v_i^{(r)} := (w_{i0}^{0r}, \ldots, w_{i,n-r}^{0r})^T$ and U_r is the $(n-r+1) \times (n-r+2)$ matrix of the form

$$U_r = \begin{pmatrix} 1-y & y & 0 & \cdots & \cdots & 0 \\ 0 & 1-y & y & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & 1-y & y \end{pmatrix}.$$

Therefore, for all $k \in \{1, \ldots, n\}$ and $i \in \{0, 1, \ldots, m\}$, we can write

$$v_i^{(k)} = U_k \cdots U_1\, v_i^{(0)}. \tag{3.7}$$

From the recursive definition of w_{ij}^{0k} (see (2.8)) and (3.1), we can deduce that, for all $k \in \{1, \ldots, n\}$, $i \in \{0, 1, \ldots, m\}$, and $j \in \{0, 1, \ldots, n-k\}$,

$$\widehat{w}_{ij}^{0k} = ((1-y)\,\widehat{w}_{ij}^{0,k-1}\,(1+\delta_{ij}^{0,k-1}) + y\,\widehat{w}_{i,j+1}^{0,k-1}\,(1+\delta_{i,j+1}^{0,k-1}))\,(1+\overline{\delta}_{ij}^{0k}),$$

where $|\delta_{ij}^{0,k-1}|$, $|\delta_{i,j+1}^{0,k-1}|$, and $|\overline{\delta}_{ij}^{0k}|$ are numbers less than or equal to the roundoff unit. Then, by (3.3), we have

$$\widehat{w}_{ij}^{0k} = (1-y)\,\widehat{w}_{ij}^{0,k-1}\,(1+\theta_2) + y\,\widehat{w}_{i,j+1}^{0,k-1}\,(1+\theta_2)$$

for all $j = 0, 1, \ldots, n-k$. Iterating the previous arguments and using (3.3) and (3.7), we can get

$$\widehat{v}_i^{(k)} = U_k \cdots U_1 \begin{pmatrix} w_{i0}^{00}\,(1+\theta_{2k}) \\ \vdots \\ w_{in}^{00}\,(1+\theta_{2k}) \end{pmatrix} = U_k \cdots U_1 \left(v_i^{(0)} + \begin{pmatrix} w_{i0}^{00}\,\theta_{2k} \\ \vdots \\ w_{in}^{00}\,\theta_{2k} \end{pmatrix} \right)$$

for all $k \in \{1,\dots,n\}$ and $i \in \{0,1,\dots,m\}$. So, again by (3.7), we can write

$$\widehat{v}_i^{(k)} = v_i^{(k)} + U_k \cdots U_1 \begin{pmatrix} w_{i0}^{00}\,\theta_{2k} \\ \vdots \\ w_{in}^{00}\,\theta_{2k} \end{pmatrix}.$$

Hence, taking into account the nonnegativity of the matrices U_j and of the weights, the previous formula, and (3.7), we deduce that

$$|\widehat{v}_i^{(k)} - v_i^{(k)}| \le \gamma_{2k}\,|v_i^{(k)}| = \gamma_{2k}\,v_i^{(k)}$$

for all $k \in \{1,\dots,n\}$ and $i \in \{0,1,\dots,m\}$. Thus we can write

$$\widehat{w}_{ij}^{0k} = w_{ij}^{0k}\,(1+\theta_{2k}) \tag{3.8}$$

for all $i \in \{0,1,\dots,m\}$, $k \in \{1,\dots,n\}$, and $j \in \{0,1,\dots,n-k\}$.

Let us now perform the calculation of the coefficients f_{ij}^{0r} for all $j \in \{0,1,\dots,n-r\}$ and $r \in \{1,\dots,n\}$. From the recursive definition of f_{ij}^{0r} (see formula (2.7)), (3.1), and (3.3), we can deduce that, for all $r \in \{1,\dots,n\}$ and $i \in \{0,1,\dots,m\}$,

$$\widehat{f}_{ij}^{0r} = \left((1-y)\,\frac{\widehat{w}_{ij}^{0,r-1}}{\widehat{w}_{ij}^{0r}}\,\widehat{f}_{ij}^{0,r-1} + y\,\frac{\widehat{w}_{i,j+1}^{0,r-1}}{\widehat{w}_{ij}^{0r}}\,\widehat{f}_{i,j+1}^{0,r-1}\right)(1+\theta_4), \quad j \in \{0,1,\dots,n-r\}.$$

Applying (3.8) in the previous formula, we obtain

$$\widehat{f}_{ij}^{0r} = \left((1-y)\,\frac{w_{ij}^{0,r-1}\,(1+\theta_{2(r-1)})}{w_{ij}^{0r}\,(1+\theta_{2r})}\,\widehat{f}_{ij}^{0,r-1} + y\,\frac{w_{i,j+1}^{0,r-1}\,(1+\theta_{2(r-1)})}{w_{ij}^{0r}\,(1+\theta_{2r})}\,\widehat{f}_{i,j+1}^{0,r-1}\right)(1+\theta_4)$$

for all $j \in \{0,1,\dots,n-r\}$. Then, using (3.4) and (3.3), we derive

$$\begin{aligned}\widehat{f}_{ij}^{0r} &= (1-y)\,\frac{w_{ij}^{0,r-1}}{w_{ij}^{0r}}\,\widehat{f}_{ij}^{0,r-1}\,(1+\theta_{6r+2}) \\ &\quad + y\,\frac{w_{i,j+1}^{0,r-1}}{w_{ij}^{0r}}\,\widehat{f}_{i,j+1}^{0,r-1}\,(1+\theta_{6r+2}), \quad j \in \{0,1,\dots,n-r\}.\end{aligned}$$

Iterating the previous arguments, using (3.3) and (3.6), and taking into account that $\sum_{r=1}^{n}(6r+2) = 3n^2+5n$, analogously as it has been done above for the weights, we can deduce that

$$|\widehat{f}_i^{(n)} - f_i^{(n)}| \le \gamma_{3n^2+5n}\,|f_i^{(n)}|$$

for all $i \in \{0,1,\dots,m\}$. Thus we can write

$$\widehat{f}_i^{(n)} = f_i^{(n)}\,(1+\theta_{3n^2+5n}) \tag{3.9}$$

for all $i \in \{0,1,\dots,m\}$.

In a similar way as has been done above for the points f_{ij}^{0r} and the weights w_{ij}^{0r}, considering now the $(m-r+1)\times(m-r+2)$ matrices of the form

$$M_r := \begin{pmatrix} x_0^r & y_0^r & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & x_{m-r}^r & y_{m-r}^r \end{pmatrix}, \quad N_r = \begin{pmatrix} 1-x & x & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1-x & x \end{pmatrix}$$

with

$$x_i^r := (1-x)\,\frac{w_{i0}^{r-1,n}}{w_{i0}^{rn}}, \quad y_i^r := x\,\frac{w_{i+1,0}^{r-1,n}}{w_{i0}^{rn}}, \quad i = 0, 1, \ldots, m-r,$$

and the vectors $g^{(r)} := (f_{00}^{rn}, \ldots, f_{m-r,0}^{r,n})^T$ (in particular, $g^{(0)} = (f_0^{(n)}, \ldots, f_m^{(n)})^T$) and $w^{(r)} := (w_{00}^{rn}, \ldots, w_{m-r,0}^{rn})^T$, by formulas (2.9) and (2.10) we have $g^{(r)} = M_r \cdots M_1 g^{(0)}$ and $w^{(r)} = N_r \cdots N_1 w^{(0)}$ for all $r \in \{1, \ldots, m\}$. Taking again into account (3.1), (3.3), and (3.4), the nonnegativity of the weights and of the matrices M_r and N_r, that $\sum_{r=1}^{m}(6(n+r)+2) = 3m^2 + 6mn + 5m$, and, in addition, formulas (3.8) and (3.9), we can deduce

$$\widehat{w}_{i0}^{kn} = w_{i0}^{kn}\,(1 + \theta_{2(n+k)})$$

for all $k \in \{1, \ldots, m\}$ and $i \in \{0, 1, \ldots, m-k\}$, and

$$\widehat{f}_{00}^{mn} = f_{00}^{mn}\,(1 + \theta_{(3m^2+5m)+(3n^2+5n)+6mn}).$$

Hence we can conclude

$$\begin{aligned}|\widehat{f}_{00}^{mn} - f_{00}^{mn}| &\le \gamma_{(3m^2+5m)+(3n^2+5n)+6mn}\,|f_{00}^{mn}| \\ &\le \gamma_{(3m^2+5m)+(3n^2+5n)+6mn} \sum_{i=0}^{m}\sum_{j=0}^{n} |f_{ij}|\,\frac{w_{ij}\,b_i^m(x)\,b_j^n(y)}{\sum_{i=0}^{m}\sum_{j=0}^{n} w_{ij}\,b_i^m(x)\,b_j^n(y)},\end{aligned}$$

and the result follows. □

The previous bound (3.5) for the absolute forward error can be interpreted in the following classical way. The first factor γ_k corresponds to the backward error, and the second factor is the absolute condition number. So Theorem 3.1 guarantees the backward stability of the corner cutting rational surface algorithm. The next section will show that the contribution of the second factor (due to the conditioning of the basis) is as small as possible.

4. Optimal stability of the basis. Consider now the rational Bézier surface (2.3) defined on $\Omega = [0,1]\times[0,1]$. Given the double-index $\alpha = \alpha_1\alpha_2$, with $0 \le \alpha_1 \le m$, $0 \le \alpha_2 \le n$, we can define the corresponding basis function

$$r_\alpha(x,y) := \frac{w_\alpha\,b_{\alpha_1}^m(x)\,b_{\alpha_2}^n(y)}{\sum_{\alpha_1=0}^{m}\sum_{\alpha_2=0}^{n} w_\alpha\,b_{\alpha_1}^m(x)\,b_{\alpha_2}^n(y)}. \tag{4.1}$$

Let

$$U = \left\{\sum_\alpha c_\alpha r_\alpha : c_\alpha \in \mathbf{R}\right\} \tag{4.2}$$

be the corresponding space of rational functions on Ω. We arrange the elements of $r = (r_\alpha)$ in lexicographical order.

The following result proves that the basis given by (4.1) is optimally stable.

THEOREM 4.1. [illegible] $r = (r_\alpha)$ [illegible] (4.1) [illegible] U [illegible] (4.2). [illegible] $u = (u_\alpha)$ [illegible] U [illegible]

$$(u; f, (x,y)) \le \quad (r; f, (x,y)) \tag{4.3}$$

[illegible] $f \in U$ [illegible] $(x,y) \in \Omega$ [illegible] $u = r$ [illegible]
[illegible]. By Lemma 3.1 of [6], there exists a nonnegative matrix A such that

$$r = uA$$

and, by Lemma 3.3 of [6], it is enough to show that the first and last nonzero element in each column of A^{-1} is positive. Fix a column β of A^{-1}, and let (c_α) be the elements of A^{-1} in this column. Then

$$u_\beta(x,y) = \sum_\alpha c_\alpha r_\alpha(x,y). \tag{4.4}$$

Let $\alpha = l_1 l_2$ be the double-index of the first nonzero coefficient c_α in (4.4). The function u_β of (4.4) can be written as

$$u_\beta(x,y) = \sum_{\alpha_1=l_1}^{m} b_{\alpha_1}^m(x) \left(\sum_{\alpha_2=0}^{n} \frac{c_\alpha\, w_\alpha\, b_{\alpha_2}^n(y)}{\sum_{\alpha_1=0}^m \sum_{\alpha_2=0}^n w_\alpha\, b_{\alpha_1}^m(x)\, b_{\alpha_2}^n(y)} \right). \tag{4.5}$$

Using the nonnegativity of u_β and $b_{l_1}^m$ and the fact that

$$0 = \lim_{x\to 0^+} \frac{b_{\alpha_1}^m(x)}{b_{l_1}^m(x)}, \quad \alpha_1 > l_1, \tag{4.6}$$

we deduce that

$$0 \le \lim_{x\to 0^+} \frac{u_\beta(x,y)}{b_{l_1}^m(x)} = g(y), \tag{4.7}$$

where

$$g(y) := \sum_{\alpha_2=l_2}^{n} \frac{c_{l_1\alpha_2}\, w_{l_1\alpha_2}\, b_{\alpha_2}^n(y)}{\sum_{\alpha_2=0}^n w_{0\alpha_2}\, b_{\alpha_2}^n(y)}. \tag{4.8}$$

Taking into account (4.5), (4.7), (4.8), and

$$0 = \lim_{y\to 0^+} \frac{b_{\alpha_2}^n(y)}{b_{l_2}^n(y)}, \quad \alpha_2 > l_2, \tag{4.9}$$

we deduce that

$$0 \le \lim_{y\to 0^+} \frac{g(y)}{b_{l_2}^n(y)} = \frac{w_{l_1 l_2}}{w_{00}}\, c_{l_1 l_2}.$$

Hence, taking into account that $w_{l_1 l_2}$ and w_{00} are positive weights, we conclude that the nonzero coefficient $c_{l_1 l_2}$ is in fact positive.

Let $\alpha = m_1 m_2$ be the double-index of the last nonzero c_α in (4.4). Applying a reasoning analogous to the proof of the positivity of the first nonzero coefficient in (4.4), taking limits as $x, y \to 1^-$, and using

$$0 = \lim_{x\to 1^-} \frac{b_{\alpha_1}^m(x)}{b_{m_1}^m(x)}, \quad \alpha_1 < m_1, \tag{4.10}$$

and

$$0 = \lim_{y \to 1^-} \frac{b_{\alpha_2}^n(y)}{b_{m_2}^n(y)}, \quad \alpha_2 < m_2, \tag{4.11}$$

instead of (4.6) and (4.9) (respectively), we may deduce that the last nonzero coefficient in (4.4) must be also positive, that is, the last nonzero entry of each column of A^{-1} is positive, and the result follows. □

Let us recall that the bases convenient in computer aided design are bases of nonnegative functions. Theorem 4.1 has proved that the basis used by the projective and by the corner cutting algorithms is optimally stable, in the sense that there is not (up to permutation and positive scaling) a better conditioned basis formed by nonnegative functions. Numerical experiments of the next section will confirm these good stability properties as well as the advantages of the corner cutting algorithm.

5. Numerical experiments. In this section we carry out numerical experiments when evaluating rational surfaces through the projective algorithm, the VS algorithm, and the corner cutting rational surface algorithm introduced in this paper. The numerical tests were performed on a Windows XP machine with the C compiler gcc of Mingw. For gcc the difference between 1.0 and the smaller number greater than 1.0 representable by the computer is $1.19209290e-07$ in simple precision and $1.1102230246251568e-16$ in double precision. In addition, gcc uses as a rounding strategy choosing the closest representable number. For more details on the compiler, see http://www.mingw.org.

The polynomials $\prod_{k=1}^{20}(x-\frac{2}{2^k})$ and $\prod_{k=1}^{20}(x-\frac{1}{k})$ with roots $\left\{\left(\frac{2}{2^k}\right) \mid k=1,\dots,20\right\}$ and $\left\{\left(\frac{1}{k}\right) \mid k=1,\dots,20\right\}$, respectively, were originally studied by Wilkinson (see [17], [18], and [19], where he showed the ill-conditioning properties of the roots). So we consider here the following bivariate polynomials:

$$\begin{aligned} I(x,y) &= \textstyle\prod_{i=1}^{12}\left(x-\frac{2}{2^i}\right)\prod_{j=1}^{12}\left(y-\frac{2}{2^j}\right), \\ J(x,y) &= \textstyle\prod_{i=1}^{12}\left(x-\frac{i}{12}\right)\prod_{j=1}^{12}\left(y-\frac{j}{12}\right), \end{aligned} \qquad (x,y)\in[0,1]\times[0,1],$$

which can be seen as a generalization of the univariate polynomials above considered by Wilkinson. Now, taking the strictly positive sequence $(w_{ij})_{0\le i,j\le 12}$ given by $w_{ij} = |i-j|+1$ for all i,j, we can introduce a rational Bézier surface $F(x,y)$ as follows:

$$F(x,y) = \frac{I(x,y)}{\sum_{i=0}^{12}\sum_{j=0}^{12} w_{ij}\, b_i^{12}(x)\, b_j^{12}(y)}, \quad (x,y)\in[0,1]\times[0,1].$$

We evaluated the function $F(x,y)$ at a mesh $(x_i,x_j)_{0\le i,j\le 24}$ such that $x_i = 0.04\,i$ for $i\in\{0,1,\dots,24\}$ using the compiler gcc with double precision arithmetic by the evaluation algorithms considered in this paper, i.e., $\widehat{F}(x_i,x_j)$. Using the algebraic manipulator MATHEMATICA, we computed the exact value of $F(x_i,x_j)$ and the corresponding relative errors $E_{rel}(x_i,x_j) = \frac{|F(x_i,x_j)-\widehat{F}(x_i,x_j)|}{|F(x_i,x_j)|}$ for each evaluation algorithm considered. In the first example we have also measured the time spent by the CPU of the computer (in seconds) for the evaluation of $F(x,y)$ at all points of the mesh $(x_i,x_j)_{0\le i,j\le 24}$ for each of the evaluation algorithms considered for comparing their computational costs. Let us remark that, in contrast to the remaining examples, we have used in the three initial examples double precision due to the well known ill-conditioning properties of these polynomials.

TABLE 5.1
Evaluating $F(x, y)$.

	Projective alg.	Corner cutting alg.	VS alg.
Maximum relative error	2.19×10^{-11}	2.19×10^{-11}	2.19×10^{-11}
Mean relative error	1.56×10^{-13}	1.43×10^{-13}	1.28×10^{-13}
Seconds of CPU time	0.025406	0.052438	0.014000

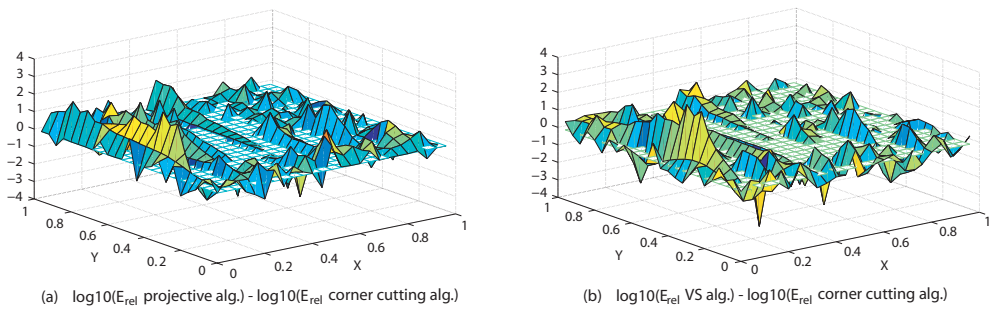


FIG. 5.1. *Comparison of the relative errors in logarithmic scale when evaluating $F(x, y)$.*

We have seen that the three algorithms have very good behavior with respect to the error (in spite of the ill-conditioning of these polynomials) and, in addition, that they produce similar relative errors (see Table 5.1 and Figure 5.1). We can also observe in Table 5.1 that the VS algorithm clocked the fastest time of the three evaluation algorithms, which agrees with section 2. In Figure 5.1(a) is plotted with MATLAB the surface given by the differences between the points of the bilinear surfaces formed by the points $(x_i, x_j, log_{10}(E_{rel}(x_i, x_j)))_{0 \leq i,j \leq 24}$ obtained through the projective and the corner cutting rational surface algorithms. At the points where $z > 0$ the corner cutting rational surface algorithm performs better than the projective algorithm, while at the points where $z < 0$ the projective algorithm performs better. Analogously, in Figure 5.1(b) is plotted the surface given by the differences between the points of the bilinear surfaces formed by the logarithms of the relative errors obtained through the VS and the corner cutting rational surface algorithms. We have also performed the same process for the rational surface $G(x, y)$ given by

$$G(x, y) = \frac{J(x, y)}{\sum_{i=0}^{12} \sum_{j=0}^{12} w_{ij}\, b_i^{12}(x)\, b_j^{12}(y)}, \quad (x, y) \in [0, 1] \times [0, 1],$$

with $w_{ij} = |i - j| + 1$ for all i, j, checking that, in this case, the three algorithms considered also produce very similar and accurate results.

The following example shows that the projective and the corner cutting rational surface algorithms behave better than the VS algorithm with respect to error for some extremely ill-conditioned functions. Let us consider the surface

$$H(x, y) = \left(x - \frac{1}{2}\right)^{12} \left(y - \frac{1}{2}\right)^{12}, \quad (x, y) \in [0, 1] \times [0, 1],$$

which can be expressed as a degenerated rational Bézier surface taking as weights $w_{ij} = 1$ for all $i, j \in \{0, 1, \ldots, 12\}$. Since the polynomial $H(x, y)$ has roots of very

TABLE 5.2
Relative errors when evaluating $H(x, y)$.

	Projective alg.	Corner cutting alg.	VS alg.
Maximum relative error	2.17×10^{-14}	2.17×10^{-14}	1.93×10^{16}
Mean relative error	3.55×10^{-15}	3.55×10^{-15}	1.23×10^{14}

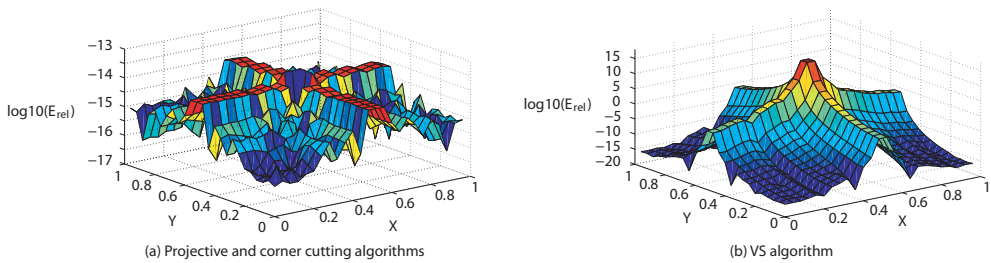


FIG. 5.2. *Relative errors in logarithmic scale.*

high multiplicity on the lines $x = 1/2$ and $y = 1/2$ it is very ill-conditioned. Analogously to the function $F(x, y)$, we evaluated $H(x, y)$ at a mesh $(x_i, x_j)_{0 \le i,j \le 24}$ with $x_i = 0.04\, i$ for $i \in \{0, 1, \ldots, 24\}$ using the compiler gcc with double precision arithmetic by the evaluation algorithms considered in this paper, and with the support of MATHEMATICA we obtained the corresponding relative errors for each evaluation algorithm considered.

On the one hand, we have seen that the projective and the corner cutting rational surface algorithms are very accurate (in spite of the extreme ill-conditioning of polynomial H) and, in addition, both algorithms produce exactly the same relative errors (see Table 5.2 and Figure 5.2(a)) since in this case the algorithms coincide because all the weights are equal to one. On the other hand, the VS algorithm presents catastrophical behavior at some points of the mesh (see Table 5.2 and Figure 5.2(b)), for instance, at every point of the mesh close to either the line $x = 1/3$ or to the line $y = 1/2$.

The following examples illustrate the fact that the corner cutting rational surface algorithm cannot present underflow or overflow, in contrast to the other two algorithms. It is not surprising that the corner cutting rational surface algorithm presents better behavior with respect to avoiding underflow and overflow than the projective and VS algorithms, since all its steps are corner cutting steps. In fact, since the corner cutting rational surface algorithm satisfies, by (2.8) and (2.7), that

$$\min_{0 \le i \le m; 0 \le j \le n} \{w_{ij}\} \le w_{ij}^{0r} \le \max_{0 \le i \le m; 0 \le j \le n} \{w_{ij}\},$$

$$\min_{0 \le i \le m; 0 \le j \le n} \{|f_{ij}|\} \le |f_{ij}^{0r}| \le \max_{0 \le i \le m; 0 \le j \le n} \{|f_{ij}|\},$$

for all $i \in \{0, 1, \ldots, m\}$, $r \in \{1, \ldots, n\}$, and $j \in \{0, 1, \ldots, n - r\}$ and, by (2.10) and (2.9), that

$$\min_{0 \le i \le m; 0 \le j \le n} \{w_{ij}\} \le w_{i0}^{sn} \le \max_{0 \le i \le m; 0 \le j \le n} \{w_{ij}\},$$

$$\min_{0 \le i \le m; 0 \le j \le n} \{|f_{ij}|\} \le |f_{i0}^{sn}| \le \max_{0 \le i \le m; 0 \le j \le n} \{|f_{ij}|\},$$

TABLE 5.3
Relative errors.

	Corner cutting alg.
Maximum relative error	4.11×10^{-5}
Mean relative error	4.21×10^{-7}

for all $s \in \{1, \dots, m\}$ and $i \in \{0, 1, \dots, m-s\}$, underflow and overflow are avoided. In fact, we have proved that this algorithm has optimal growth factor. Let us recall that the growth factor of a numerical algorithm is the quotient between the ∞-norm of the elements calculated during the algorithm and the ∞-norm of the initial data. In general, the control of growth factor avoids overflow and is an indicator of stability.

The following two examples illustrate the good behavior of the corner cutting algorithm with respect to overflow, in contrast to the other two algorithms. We have considered a rational function $K(x, y)$ given by

$$K(x,y) = \sum_{i=0}^{10} \sum_{j=0}^{10} f_{ij} \, \frac{w_{ij} \, b_i^{10}(x) \, b_j^{10}(x)}{\sum_{i=0}^{n} \sum_{j=0}^{n} w_{ij} \, b_i^{10}(x) \, b_j^{10}(x)}, \quad (x,y) \in [0,1] \times [0,1]. \tag{5.1}$$

First we have evaluated the surface $K(x, y)$ given by $f_{ij} = (-1)^{i+j} \, 10^{20+i-j}$ and $w_{ij} = 10^{10+i-j}$ for all $i, j \in \{0, 1, \dots, 10\}$ at a mesh $(x_i, x_j)_{0 \le i,j \le 20}$, where $x_i = 0.05\, i$ for $i \in \{0, 1, \dots, 20\}$, using the C compiler Mingw with simple precision through the three evaluation algorithms considered. On the one hand, for the VS and the projective algorithms the unique result obtained at all points of the mesh $(x_i, x_j)_{0 \le i,j \le 20}$ is $-1.\#IND0 \dots 0$ (out of the range of representation) because some of the products $w_{ij} \, f_{ij}$ in the projective case and some of the products $\binom{10}{i} \binom{10}{j} w_{ij} \, f_{ij}$ in the VS case present overflow. On the other hand, the results obtained with the corner cutting rational surface algorithm were satisfactory and accurate at all points of the mesh, as shown in Table 5.3.

Although the selection of the weights w_{ij} can seem nonstandard, let us take into account that changing weights is a very useful tool for interactive shape design (as remarked in [11, p. 120]). In section 11.3.3 of [11] the effect of modifying a single weight in a rational surface is analyzed. The following example considers a more standard case, where the weights vary no more than three orders of magnitude.

Let $K(x, y)$ be the surface given by $f_{ij} = (-1)^{i+j} \cdot 5 \cdot 10^{26+i-j}$ and $w_{ij} = 10^{(i+j)\%4}$, where $a\%b$ denotes the remainder of dividing a by b. We have evaluated this surface at a mesh $(x_i, x_j)_{0 \le i,j \le 20}$, where $x_i = 0.05\, i$ for $i \in \{0, 1, \dots, 20\}$, using the C compiler Mingw with simple precision through the three evaluation algorithms considered. On the one hand, the unique result obtained for the VS and the projective algorithms at all points of the mesh $(x_i, x_j)_{0 \le i,j \le 20}$ is $-1.\#IND0 \dots 0$ (out of the range of representation) because some of the products $w_{ij} \, f_{ij}$ in the projective case and some of the products $\binom{10}{i} \binom{10}{j} w_{ij} \, f_{ij}$ in the VS case present overflow. On the other hand, the results obtained with the corner cutting rational surface algorithm were satisfactory and accurate at all points of the mesh, as shown in Table 5.4.

Finally we have evaluated the surface $K(x, y)$ in (5.1) given by

$$f_{ij} = (-1)^{i+j} \, 1.2 \, 10^{-33+i-j} \quad \text{and} \quad w_{ij} = \begin{cases} 10^{-10} & \text{if } i+j \text{ is even,} \\ 10^{-7} & \text{if } i+j \text{ is odd,} \end{cases}$$

at a mesh $(x_i, x_j)_{0 \le i,j \le 20}$, where $x_i = 0.05\, i$ for $i \in \{0, 1, \dots, 20\}$, using the C compiler Mingw with simple precision through the three evaluation algorithms considered. On

TABLE 5.4
Relative errors.

	Corner cutting algorithm
Maximum relative error	7.51×10^{-6}
Mean relative error	3.65×10^{-7}

TABLE 5.5
Relative errors.

	Projective alg.	Corner cutting alg.	VS alg.
Maximum relative error	1.34×10^{0}	6.37×10^{-7}	1.36×10^{0}
Mean relative error	5.67×10^{-2}	1.47×10^{-7}	5.9×10^{-2}

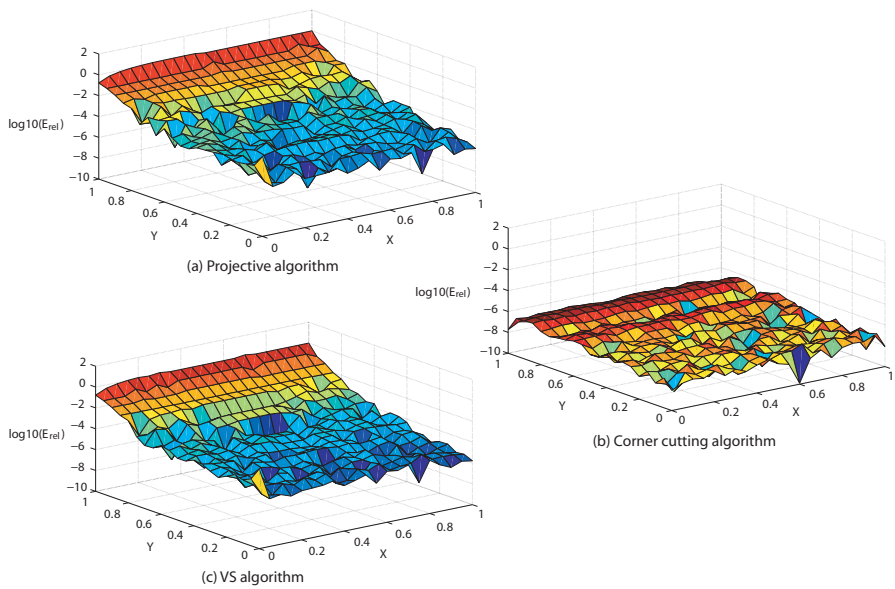


FIG. 5.3. *Relative errors in logarithmic scale.*

the one hand, the results obtained at the points $(1.00, 0.50)$, $(1.00, 0.55)$, $(1.00, 0.60)$, $(1.00, 0.65)$, $(1.00, 0.75)$, $(1.00, 0.80)$, $(1.00, 0.85)$, $(1.00, 0.90)$, and $(1.00, 0.95)$ of the mesh with the VS and the projective algorithms are exactly 0 because some of the products $w_{ij}\, f_{ij}$ in the projective case and some of the products $\binom{10}{i}\binom{10}{j} w_{ij}\, f_{ij}$ in the VS case are taken as 0 due to underflow problems. So these results are completely wrong. But, in addition to this fact, there are more points of the mesh such that the projective and the VS algorithms produce very inaccurate results: for example, both algorithms produce very high relative errors at the points $(1.00, 0.30)$, $(1.00, 0.20)$, $(1.00, 0.10)$, ... For more details on the bad performance of the projective and the VS algorithms, see Table 5.5 and Figure 5.3(a) and (c). On the other hand, the results obtained with the corner cutting rational surface algorithm were satisfactory and accurate at all points of the mesh as the table and Figure 5.3(b) show. The two standard algorithms also suffer from overflow and underflow in double precision.

6. Conclusions. Let us summarize some conclusions on the comparison of the three algorithms considered in this paper. On the one hand, as for the complexity, we have seen in section 2 that the VS algorithm has the lowest computational cost. On the other hand, we have proved in section 4 that the basis (4.1) corresponding to the projective and the corner cutting algorithms is optimally stable. Besides, the stability of the new corner cutting algorithm has been shown in section 3. In section 5, numerical experiments have confirmed the previous results. In spite of the good stability properties of the three algorithms even for ill-conditioned polynomials, we illustrate that the algorithms associated with the basis (4.1) (i.e., the projective and the corner cutting rational surface algorithms) are more stable than the VS algorithm. Moreover, due to the good behavior of the corner cutting rational surface algorithm with respect to avoiding overflow and underflow problems, we can also say that it is more robust than the other algorithms for the evaluation of rational surfaces. In addition, we have proved in section 5 that it has optimal growth factor.

In a future work we will study the possibility of a corner cutting algorithm for evaluating nonuniform rational B-spline (NURBS) surfaces.

Acknowledgment. The authors are thankful to the referees for their valuable comments and suggestions.

REFERENCES

[1] W. Dahmen, H. P. Dikshit, and A. Ojha, *On Wachspress quadrilateral patches*, Computer Aided Geometric Design, 17 (2000), pp. 879–890.
[2] G. Farin, *Curves and Surfaces for Computer Aided Geometric Design*, 5th ed., Academic Press, San Diego, CA, 2002.
[3] R. T. Farouki and T. N. T. Goodman, *On the optimal stability of the Bernstein basis*, Math. Comp., 65 (1996), pp. 1553–1566.
[4] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM, Philadelphia, 2002.
[5] J. Hoschek and D. Lasser, *Fundamentals of Computer Aided Geometric Design*, A K Peters, Wellesley, MA, 1993.
[6] T. Lyche and J. M. Peña, *Optimally stable multivariate bases*, Adv. Comput. Math., 20 (2004), pp. 149–159.
[7] E. Mainar and J. M. Peña, *Error analysis of corner cutting algorithms*, Numer. Algorithms, 22 (1999), pp. 41–52.
[8] S. Mann and T. DeRose, *Computing values and derivatives of Bézier and B-spline tensor products*, Computer Aided Geometric Design, 12 (1995), pp. 107–110.
[9] J. M. Peña, *B-splines and optimal stability*, Math. Comp., 66 (1997), pp. 1555–1560.
[10] J. M. Peña, *Error analysis of algorithms for evaluating Bernstein-Bézier type multivariate polynomials*, in Curves and Surfaces Design, P. J. Laurent, P. Sablonnière, and L. L. Schumaker, eds., Vanderbilt University Press, Nashville, TN, 2000, pp. 315–324.
[11] L. Piegl and W. Tiller, *The NURBS Book*, 2nd ed., Springer-Verlag, New York, 1997.
[12] J. M. Peña, *On the optimal stability of bases of univariate functions*, Numer. Math., 91 (2002), pp. 305–318.
[13] R. F. Riesenfeld, *Homogeneous coordinates and projective planes in computer graphics*, IEEE Comput. Graph. Appl., 1 (1981), pp. 50–55.
[14] L. L. Schumaker and W. Volk, *Efficient evaluation of multivariate polynomials*, Computer Aided Geometric Design, 3 (1986), pp. 149–154.
[15] T. W. Sederberg, *Point and tangent computation of tensor product rational Bézier surfaces*, Computer Aided Geometric Design, 12 (1995), pp. 103–106.
[16] E. L. Wachspress, *A Rational Finite Element Basis*, Academic Press, Boston, 1975.
[17] J. H. Wilkinson, *The evaluation of the zeros of ill-conditioned polynomials, Part* I, Numer. Math., 1 (1959), pp. 150–166.
[18] J. H. Wilkinson, *The evaluation of the zeros of ill-conditioned polynomials, Part* II, Numer. Math., 1 (1959), pp. 167–180.
[19] J. H. Wilkinson, *Rounding Errors in Algebraic Processes*, Notes Appl. Sci. 32, Her Majesty's Stationery Office, London, 1963; reprinted by Dover, New York, 1994.

SIAM J. SCI. COMPUT.
Vol. 29, No. 4, pp. 1683–1709
© 2007 Society for Industrial and Applied Mathematics

PARTITIONING SPARSE MATRICES FOR PARALLEL PRECONDITIONED ITERATIVE METHODS*

BORA UÇAR† AND CEVDET AYKANAT‡

Abstract. This paper addresses the parallelization of the preconditioned iterative methods that use explicit preconditioners such as approximate inverses. Parallelizing a full step of these methods requires the coefficient and preconditioner matrices to be well partitioned. We first show that different methods impose different partitioning requirements for the matrices. Then we develop hypergraph models to meet those requirements. In particular, we develop models that enable us to obtain partitionings on the coefficient and preconditioner matrices simultaneously. Experiments on a set of unsymmetric sparse matrices show that the proposed models yield effective partitioning results. A parallel implementation of the right preconditioned BiCGStab method on a PC cluster verifies that the theoretical gains obtained by the models hold in practice.

Key words. matrix partitioning, preconditioning, iterative method, parallel computing

AMS subject classifications. 05C50, 05C65, 65F10, 65F35, 65F50, 65Y05

DOI. 10.1137/040617431

1. Introduction.

We consider the parallelization of the preconditioned iterative methods that use explicit preconditioners such as approximate inverses or factored approximate inverses. Our objective is to develop methods for obtaining one-dimensional (1D) partitions on a coefficient matrix and a preconditioner matrix or factors of a preconditioner matrix simultaneously to efficiently parallelize a full step of the preconditioned iterative methods. We assume preconditioner matrices or their sparsity patterns are available beforehand. It has been shown that the rates of convergence of iterative methods depend on the partitioning method when the preconditioners are built from partitioned coefficient matrices [26]. With the above assumption in mind, we neither deteriorate nor improve the effects of the selected preconditioners on the rate of convergence. Our assumption is justified in applications where the preconditioner matrices can be reused; see, for example, [12] and a discussion of it in [10]. The assumption is also justified in the preconditioner constructing methods that require a priori sparsity patterns for the preconditioner matrices [44, 45], where techniques to develop effective sparsity patterns already exist in the literature [23, 24, 39].

Approximate inverse preconditioning techniques explicitly compute and store a sparse matrix $M \approx A^{-1}$ to be used as a preconditioner. Application of such preconditioners requires one or two matrix-vector multiply operations. Two types of approximate inverses exist in the literature. In the first type, an approximate inverse is stored as a single matrix, whereas in the second type it is stored as a product of two matrices. The second type of preconditioners are referred to as factored approximate inverses. Among the most notable approximate inverse preconditioners are AINV and its variants by Benzi et al. [5, 6, 7, 8]; SPAI by Grote and Huckle [33]; FSAI by Kolotilina and Yeremin [44, 45]; and MR by Chow and Saad [25]. See [4, 9, 31] for

*Received by the editors October 21, 2004; accepted for publication (in revised form) January 31, 2007; published electronically August 1, 2007. This work was partially supported by The Scientific and Technological Research Council of Turkey (TÜBİTAK) under project 106E069.
http://www.siam.org/journals/sisc/29-4/61743.html
†CERFACS, 42 av. Gaspard Coriolis, 31057, Toulouse Cedex 1, France (ubora@cerfacs.fr).
‡Computer Engineering Department, Bilkent University, Ankara, 06800, Turkey (aykanat@cs.bilkent.edu.tr).

a recent survey and the use of the approximate inverse preconditioning techniques. See [48, 52] for a general treatment of the preconditioning techniques.

Hendrickson and Kolda [36] give a thorough survey of the graph partitioning models used for partitioning sparse matrices. In these models, a K-way partition of the vertices of a given graph or hypergraph is computed. The partitioning constraint is to maintain a balance criterion on the number of vertices in each part; if the vertices are weighted, then the constraint is to maintain a balance criterion on the sum of the vertex weights in each part. The partitioning objective is to minimize the cutsize of the partition defined over the edges or hyperedges. The partitioning constraint and objective relate, respectively, to maintaining computational load balance and minimizing the total communication volume. Among those models, the hypergraph models by Aykanat, Pınar, and Çatalyürek [2], Çatalyürek and Aykanat [17, 18], and Pınar et al. [49] and the bipartite graph model by Hendrickson and Kolda [37, 43] are said to have more expressive power than the other models [18, 34, 36, 37]. These models have the flexibility of producing unsymmetric partitions on the input and output vectors of the sparse matrix-vector multiplies. A distinct advantage of the hypergraph models over both the standard graph and the bipartite graph models is that the partitioning objective in the hypergraph models is an exact measure of the total communication volume, whereas the objective in the graph models is an approximation [18, 34, 36, 37]. As noted in the survey [36] and in [34], all these graph and hypergraph models, except the bipartite graph model, are used to optimize a single sparse matrix-vector multiply operation. However, a single sparse matrix-vector multiply operation is only a piece of a larger computation in the preconditioned iterative methods. Therefore, new partitioning models that optimize a full step of these iterative methods are needed—this was also stated by Hendrickson [34].

Optimizing a full step of the preconditioned iterative methods requires partitioning the coefficient and preconditioner matrices. This problem has been formulated in terms of bipartite graph partitioning [37]. In the bipartite graph model, each vertex in one part represents a row of A and a column of M, and each vertex in the other part represents a column of A and a row of M. There is an edge between two vertices if the corresponding entries in A or M are nonzero. Partitioning this bipartite graph produces unsymmetric partitions on A and M, where the row partition of A and the column partition of M are the same, and the column partition of A and the row partition of M are the same. The formulation is based on the cut edges and hence does not capture the total communication volume exactly. The multiphase mesh partitioning method of Walshaw et al. [47, 64] and multiconstraint/multiobjective graph partitioning methods of Karypis et al. [41, 53] address the partitioning problem in scientific computations whose computational structures are similar to that of preconditioned iterative methods. These models can be used to partition a graph corresponding to the sparsity pattern of the matrix $A+M$ to find a single partition for both of the matrices. Partitioning such a graph will produce symmetric partition on the coefficient and preconditioner matrices.

In this work, we propose methods to build composite hypergraph models for preconditioned iterative methods from simple hypergraph models for a single matrix-vector multiply [18]. We show how to use the composite models to obtain 1D partitions on a matrix and its approximate inverse preconditioners for optimizing a full step of the preconditioned iterative methods. The composite hypergraph models encode the total communication volume exactly, can handle unsymmetric dependencies, and can produce unsymmetric partitions. Furthermore, they are more powerful than the

bipartite graph model in the sense that they can produce a variety of partitions on the matrices. For example, given two matrices A and M, it is possible to obtain a symmetric partition on A and a unsymmetric partition on M while optimizing a full step of a preconditioned iterative method.

We review simple hypergraph models which are the building blocks of composite models in section 2. We discuss a procedure to analyze iterative methods in order to determine partitioning requirements for efficient parallelization and illustrate the procedure on a well-known iterative method in section 3. The partitioning requirements of a number of widely used iterative methods are also given in the same section. We propose methods to build composite hypergraph models for meeting the partitioning requirements in the preconditioned iterative methods in section 4. We discuss the applicability of the composite hypergraph models to a few additional scientific applications in section 5. The proposed methods are evaluated in section 6.

2. Preliminaries. The kernel operations in the iterative methods that use approximate inverse preconditioners are matrix-vector multiplies with both coefficient and preconditioner matrices. These multiply operations are performed on vectors that are dependent on each other through linear vector operations. For example, the computations of the form $y \leftarrow AMz$ are performed as $x \leftarrow Mz$ and then $y \leftarrow Ax$. If the partition on the x vector for the first multiply operation is different from that for the second multiply operation, then the x-vector entries should be reordered in between the two multiplies. Since the reordering requires communication, it should be avoided. In order to avoid this reordering operation, there should be only one partition on the intermediate vector x, and the partition should be helpful for both of the parallel multiply operations. We call the problem of finding partitions on two or more matrices in such a way that the common vectors are not reordered in a parallel implementation the simultaneous partitioning problem.

2.1. Parallel multiplies. Given a K-way rowwise partition on a matrix A, a conformable partition on the output vector, and a possibly different K-way partition on the input vector of a matrix-vector multiply operation, the matrix A can be permuted into a block structure:

$$PAQ = A_{BL} = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1K} \\ A_{21} & A_{22} & \cdots & A_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ A_{K1} & A_{K2} & \cdots & A_{KK} \end{bmatrix}. \tag{2.1}$$

Here, K is the number of processor and P and Q are permutation matrices. The partition on the rows of A is used to define the permutation P by permuting the rows belonging to processor P_k before those belonging to P_ℓ for $k < \ell$. The partition on the input vector is used to define the permutation Q by permuting the columns corresponding to the vector entries belonging to processor P_k before those corresponding to the vector entries belonging to P_ℓ for $k < \ell$. In either of the permutations, the order of the rows or columns associated with a common processor can be arbitrary.

Similarly, given a K-way columnwise partition on A, a conformable partition on the input vector, and a K-way partition on the output vector among K processors, the matrix A again can be permuted into a $K \times K$ block structure $PAQ = A_{BL}$ (2.1). Here, P is defined using the partition on the output vector, and Q is defined using the partition on the matrix columns (and input vector).

Let A be of size $m \times n$. Then the block $A_{k\ell}$ in (2.1) is of size $m_k \times n_\ell$, where

$\sum_k m_k = m$ and $\sum_\ell n_\ell = n$. In a rowwise partitioning, the processor P_k holds the kth row stripe $[A_{k1} \cdots A_{kK}]$ of size $m_k \times n$. In a columnwise partitioning, P_k holds the kth column stripe $[A_{1k}^T \cdots A_{Kk}^T]^T$ of size $m \times n_k$. In the rowwise partitioning, the row stripes should have a nearly equal number of nonzeros for maintaining computational load balance among processors. The same requirement exists for the column stripes in the columnwise partitioning.

2.1.1. Row-parallel multiplies. Consider matrix-vector multiplies of the form $y \leftarrow Ax$, where A is partitioned rowwise. The rowwise partition of matrix A induces a partition on the output vector y, i.e., $y = [y_1^T \cdots y_K^T]^T$, where the processor P_k is set to be responsible for computing the subvector y_k of size m_k. According to the partition on the input vector $x = [x_1^T \cdots x_K^T]^T$, the processor P_k holds the subvector x_k of size n_k. Assume the matrix A has been permuted into the block structure (2.1) using these partitions. Then the [illegible] $y \leftarrow Ax$ executes the following steps at processor P_k [37, 57, 59, 61]:

1. For each nonzero off-diagonal block $A_{\ell k}$, send sparse vector $\hat{x}_k^\ell$ to processor P_ℓ, where $\hat{x}_k^\ell$ contains only those entries of x_k corresponding to the nonzero columns in $A_{\ell k}$.
2. Compute the diagonal block product $y_k^k = A_{kk} \times x_k$, and set $y_k = y_k^k$.
3. For each nonzero off-diagonal block $A_{k\ell}$, receive $\hat{x}_\ell^k$ from processor P_ℓ, then compute $y_k^\ell = A_{k\ell} \times \hat{x}_\ell^k$, and update $y_k = y_k + y_k^\ell$.

In step 1, P_k might be sending the same x_k-vector entry to different processors according to the sparsity pattern of column k of A. This multicast-like operation is referred to here as the [illegible] operation.

2.1.2. Column-parallel multiplies. Consider matrix-vector multiplies of the form $w \leftarrow Az$, where A is partitioned columnwise. The columnwise partition of A induces a partition on the input vector z, i.e., $z = [z_1^T \cdots z_K^T]^T$, where the processor P_k holds the subvector z_k of size n_k. According to the partition on the output vector $w = [w_1^T \cdots w_K^T]^T$, the processor P_k is set to be responsible for computing the subvector w_k of size m_k. Assume the matrix A has been permuted into the block structure (2.1) using these partitions. Then the [illegible] $w \leftarrow Az$ executes the following steps at processor P_k:

1. For each nonzero off-diagonal block $A_{\ell k}$, form sparse vector $\hat{w}_\ell^k$, which contains only those results of $w_\ell^k = A_{\ell k} \times z_k$ corresponding to the nonzero rows in $A_{\ell k}$. Send $\hat{w}_\ell^k$ to processor P_ℓ.
2. Compute the diagonal block product $w_k^k = A_{kk} \times z_k$, and set $w_k = w_k^k$.
3. For each nonzero off-diagonal block $A_{k\ell}$ receive partial-result vector $\hat{w}_k^\ell$ from processor P_ℓ, and update $w_k = w_k + \hat{w}_k^\ell$.

In step 3, the multinode accumulation on the w_k-vector entries is referred to here as the [illegible] operation.

2.2. Hypergraph partitioning. A hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{N})$ is defined as a set of vertices $\mathcal{V}$ and a set of nets $\mathcal{N}$. Every net is a subset of vertices. The vertices of a net are also called its [illegible]. The size of a net n_i is equal to the number of its pins, i.e., $|n_i|$. The set of nets that contain vertex v_j is denoted by [illegible] (v_j). Weights can be associated with vertices.

Given a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{N})$, $\Pi = \{\mathcal{V}_1, \ldots, \mathcal{V}_K\}$ is called a K-way partition of the vertex set $\mathcal{V}$ if each part is nonempty, i.e., $\mathcal{V}_k \neq \emptyset$ for $1 \leq k \leq K$; parts are pairwise disjoint, i.e., $\mathcal{V}_k \cap \mathcal{V}_\ell = \emptyset$ for $1 \leq k < \ell \leq K$; and the union of parts gives $\mathcal{V}$, i.e., $\bigcup_k \mathcal{V}_k = \mathcal{V}$. In Π, a net is said to [illegible] a part if it has at least one pin

in that part. The connectivity set Λ_i of a net n_i is the set of parts connected by n_i. The connectivity $\lambda_i = |\Lambda_i|$ of a net n_i is the number of parts connected by n_i. A net is said to be cut if it connects more than one part and uncut otherwise. The set of cut and uncut nets is also referred to as external and internal nets, respectively. In Π, the weight of a part is the sum of the weights of vertices in that part.

In the hypergraph partitioning problem, the objective is to minimize the cutsize:

$$\text{cutsize}(\Pi) = \sum_{n_i \in \mathcal{N}} (\lambda_i - 1). \tag{2.2}$$

This objective function is widely used in the VLSI community [46] and in the scientific computing community [2, 18, 59], and it is referred to as the connectivity -1 cutsize metric. The partitioning constraint is to satisfy a balancing constraint on part weights:

$$\frac{W_{max} - W_{avg}}{W_{avg}} \leq \epsilon. \tag{2.3}$$

Here W_{max} is the maximum part weight, W_{avg} is the average part weight, and ϵ is a predetermined imbalance ratio. This problem is NP-hard [46].

A recent variant of the above problem is the multiconstraint hypergraph partitioning [16, 20, 42] in which each vertex has a vector of weights associated with it. In this problem, the partitioning objective is the same as in (2.2), and the partitioning constraint is to satisfy a balancing constraint associated with each weight. Another variant is the multiobjective hypergraph partitioning [1, 53, 55], in which there are several objectives to be minimized. Specifically, a given net contributes different costs to different objectives.

2.3. Hypergraph models for row-parallel and column-parallel multiplies. It is inherent in the parallel matrix-vector multiply algorithms given in section 2.1 and existent in the literature [18, 36, 37, 59] that in partitioning a matrix the key is to find permutation matrices P and Q such that most of the nonzeros of the matrix $PAQ = A_{BL}$ (2.1) are in the diagonal blocks. Here, we propose a slight enhancement of the computational hypergraph models [18] to find permutation matrices P and Q. The proposed enhancement is to add a new set of vertices into the column-net and row-net hypergraph models.

In the column-net hypergraph model, an $m \times n$ matrix A is represented as a hypergraph $\mathcal{H} = (\mathcal{V}_\mathcal{R}, \mathcal{N}_\mathcal{C})$ for rowwise partitioning. Vertex and net sets, $\mathcal{V}_\mathcal{R}$ and $\mathcal{N}_\mathcal{C}$, correspond to the rows and columns of A, respectively. There exist one vertex r_i and one net c_j for each row i and column j, respectively. In this model, $r_i \in c_j$ if and only if $a_{ij} \neq 0$. The proposed enhancement is to add n new vertices each representing an input-vector entry of the $y \leftarrow Ax$ multiply. Each new vertex x_j is added to the net c_j, i.e., $c_j = c_j \cup \{x_j\}$ and $\text{vtx}(x_j) = \{c_j\}$. Each vertex r_i corresponds to the task of computing the inner product of the row i with the column vector x. Hence, the computational weight associated with the vertex r_i is equal to the number of nonzeros in row i. Each row vertex r_i also represents the vector entry y_i. Weights can be assigned to the vertices in regard to the vector entries. For example, a unit weight may be assigned to the vertex x_j for the corresponding vector entry, and a unit weight may be assigned to the vertex r_i for the vector entry y_i. These weights can be used in a multiconstraint formulation to balance the linear vector operations.

Figure 2.1 shows a sample matrix and its column-net hypergraph model enhanced with the x vertices. In the figure, the white and black circles represent, respectively,

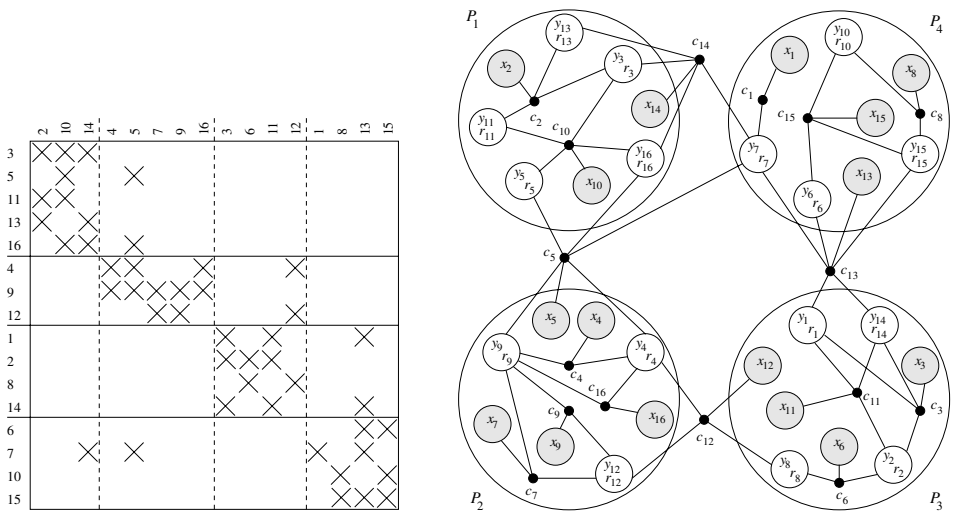


FIG. 2.1. *Matrix A, enhanced column-net hypergraph model for row-parallel $y \leftarrow Ax$, and a four-way rowwise partitioning.*

the vertices and nets of the original column-net hypergraph model [18], and straight lines show pins. The newly added vertices are shown with gray circles. A four-way partition is shown by four big circles encompassing the vertices of the hypergraph.

Given a partition Π on $\mathcal{H}$, the permutations P and Q can be found as follows. The permutation P is partially defined by the partition on the row vertices. The rows corresponding to the row vertices in $\mathcal{V}_k$ are mapped to processor P_k and therefore permuted before the rows corresponding to the row vertices in $\mathcal{V}_\ell$ for $1 \leq k < \ell \leq K$. In Figure 2.1, the permutation of the rows of A is shown by the permuted row indices, where the horizontal solid lines separate row stripes that belong to different processors. The permutation of the rows in the same row stripe can be arbitrary. The permutation Q is partially defined by the partition on the x vertices. The x-vector entries corresponding to the x vertices in $\mathcal{V}_k$ are mapped to processor P_k, and therefore the associated columns are permuted before the columns that are associated with the x vertices in $\mathcal{V}_\ell$ for $1 \leq k < \ell \leq K$. Figure 2.1 shows the permutation on the columns of A, where the vertical dashed lines separate virtual column stripes that are aligned with the x-vector entries belonging to different processors. Again, the permutation of the columns within the same column stripe can be arbitrary. In the figure, the processor P_2 is set to be responsible for computing the inner products of x with the rows in the second row stripe, i.e., the rows 4, 9, and 12. P_2 holds x_4, x_5, x_7, x_9, and x_{16} and thus expands x_5 to the processors P_1 and P_4. Observe that the net c_5 connects the parts P_1, P_2, and P_4. This association between the connectivity of nets and the communication requirements is not accidental, as shown by the following theorem.

THEOREM 2.1. Π A P Q Π Π $y \leftarrow Ax$. Consider an internal net c_i. Since the net is not cut, the row vertices that need x_i should be in the part that contains the vertex x_i. Hence, no communication occurs for the x-vector entries associated with the internal nets. Consider an external

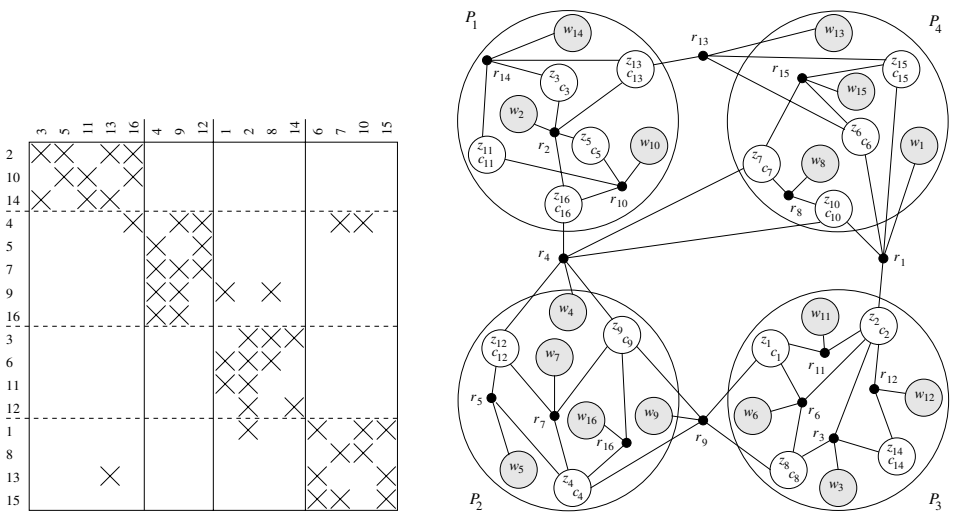


FIG. 2.2. *Matrix A, enhanced row-net hypergraph model for column-parallel $w \leftarrow Az$, and a four-way columnwise partitioning.*

net n_e with the connectivity set Λ_e. The processors corresponding to the parts in the set Λ_e need x_e. One of them owns x_e, since $x_e \in n_e$. The owner should send x_e to the others. That is, for each x_e there are a total of $|\Lambda_e| - 1 = \lambda_e - 1$ messages carrying x_e. The overall sum of these quantities matches the cutsize definition (2.2). □

With similar reasoning, it is concluded in [18] that the hypergraph partitioning objective and constraint correspond, respectively, to minimizing the total communication volume and maintaining the computational load balance. In Figure 2.1, the cutsize and hence the total communication volume is five units, and the part weights and hence the computational loads of the processors are 12, 12, 11, and 11.

The row-net hypergraph model [18] can be enhanced by adding m new vertices, each representing a w-vector entry to find the permutation matrices P and Q for the column-parallel $w \leftarrow Az$ multiplies. Recall that in the row-net model $\mathcal{H} = (\mathcal{V}_\mathcal{C}, \mathcal{N}_\mathcal{R})$, the vertices and nets represent the columns and rows of A, respectively. In this model, $c_i \in r_j$ if and only if $a_{ji} \neq 0$. Each new w_i vertex is added to the net r_i, i.e., $r_i = r_i \cup \{w_i\}$ and $\Lambda_{r_i}(w_i) = \{r_i\}$. Upon partitioning the enhanced row-net hypergraph model, we use the partition on the w vertices and the partition on the column vertices to find the permutation matrices P and Q, respectively, with a procedure similar to that discussed for the column-net model. Figure 2.2 shows a sample matrix and its enhanced row-net hypergraph model. The column stripes determine the computational loads of processors. The virtual row stripes are defined by the partition on the w vertices and therefore designate the processors' responsibilities on folding the w-vector entries. For example, in Figure 2.2, the processor P_2 is set to be responsible for folding the w-vector entries that correspond to the rows in the second virtual row stripe. Therefore, the processors P_1 and P_4 have to send their contributions for w_4 to P_2. Again, there is the same association between the connectivity of the nets and the total communication volume. In the figure, the cutsize and hence the total communication volume is five units.

3. Determining partitioning requirements. In iterative methods, all vectors that participate in a linear vector operation should be partitioned conformally in order

to avoid the communication of the vector entries during the operation. To obtain such conformable partitions, we classify the vectors according to their relations to the inputs and outputs of the matrix-vector multiplies. In particular, we call a vector to be in the [illegible] of a matrix A if it is multiplied by A or it undergoes linear vector operations with other vectors in the input space of A. Accordingly, we call a vector to be in the [illegible] of a matrix A if it is obtained by multiplying A with another vector or it undergoes linear vector operations with other vectors in the output space of A. For example, in the $y \leftarrow Ax$ multiply, the y vector is in the output space of A, whereas x is in the input space of A.

In some iterative methods, e.g., conjugate gradients [30], the input space and output space of the A matrix coincide; i.e., the input-space vectors undergo linear vector operations with the output-space vectors. Such methods require a symmetric partition PAP^T in which all vectors are partitioned conformally with the permutation P. In some other methods, the input space and output space of A differ. Such methods allow unsymmetric partition PAQ in which all output-space vectors are partitioned conformally with P, whereas all input-space vectors are partitioned conformally with Q. If the method involves more than one multiply with different matrices, the output space of one matrix may coincide with the input space of another one. In this case, the output-space permutation for the first one becomes an input-space permutation for the other one.

All vectors in a full step of an iterative algorithm should be analyzed in terms of their relations to the input and output spaces of all matrices to determine the partitioning requirements. We analyze the right preconditioned BiCGStab[1] method [62] given in Figure 3.1 and determine its partitioning requirements as an example. There are ten vectors in the method: r, b, $\tilde{r}$, x, p, v, $\hat{p}$, s, $\hat{s}$, and t. Because of the linear vector operations in lines 1, 2, 4, 7, 10, 14, 15, 19, 20, and 21, the vectors r, b, $\tilde{r}$, p, v, s, t, and x should be partitioned conformally. All these vectors are in the output space of A because of the matrix-vector multiplies in lines 13 and 18. We are left with the vectors $\hat{p}$ and $\hat{s}$. Because of the matrix-vector multiplies in lines 13 and 18, these two are in the input space of A, and thus can have a different partition Q. Since we have completed the classification of vectors, we can determine the partitioning requirements for A and M. The input and output spaces of A differ. Therefore, the partitioning requirement for the A matrix is PAQ. The vectors $\hat{p}$ and $\hat{s}$ are in the output space of M because of the matrix-vector multiplies in lines 12 and 17. Therefore, the output space of M coincides with the input space of A. Similarly, the input space of M coincides with the output space of A because of the vectors p and s. Therefore, the partitioning requirement for the M matrix is $Q^T MP^T$. The overall requirement is thus PAQ and $Q^T MP^T$. We express this requirement as $PAMP^T$ to simplify the notation.

We examined a number of widely used preconditioned iterative methods whose codes are given in the literature. We noticed that different methods have different partitioning requirements, as shown in Table 3.1. Several caveats are necessary for the table to be useful.

1. We analyze the methods in their original form as given in the references; i.e., we do not consider any type of code optimizations for performance gains.

[1]Note that the given code works with preconditioned x vector; i.e., the solution vector x obtained at the termination is a solution to $AMx = b$. That is, in order to get a solution to $Ax = b$ we have to multiply x with the approximate inverse preconditioner M at the termination. However, using $\hat{p}$ and $\hat{s}$ instead of p and s in lines 20 and 16 would yield the solution to $Ax = b$ as given in [3]. In this case, the analysis would yield different classification of vectors.

BiCGStab(A, M, x, b)#Solve $Ax = b$ using the right preconditioner M
begin
(1) $r^{(0)} = b - AMx^{(0)}$ for some initial $x^{(0)} = x$
(2) $\tilde{r} = r^{(0)}$
(3) **for** $i = 1, 2, \ldots$ **do**
(4) $\rho_{i-1} = \tilde{r}^T r^{(i-1)}$
(5) **if** $\rho_{i-1} = 0$ method fails
(6) **if** $i = 1$
(7) $p^{(i)} = r^{(i-1)}$
(8) **else**
(9) $\beta_{i-1} = (\rho_{i-1}/\rho_{i-2})(\alpha_{i-1}/\omega_{i-1})$
(10) $p^{(i)} = r^{(i-1)} + \beta_{i-1}\left(p^{(i-1)} - \omega_{i-1}v^{(i-1)}\right)$
(11) **endif**
(12) $\hat{p} = Mp^{(i)}$
(13) $v^{(i)} = A\hat{p}$
(14) $\alpha_i = \rho_{i-1}/\tilde{r}^T v^{(i)}$
(15) $s = r^{(i-1)} - \alpha_i v^{(i)}$
(16) check norm of s; if small enough; set $x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$ and stop
(17) $\hat{s} = Ms$
(18) $t = A\hat{s}$
(19) $\omega_i = t^T s/t^T t$
(20) $x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)} + \omega_i s$
(21) $r^{(i)} = s - \omega_i t$
(22) check convergence; continue if necessary
(23) for continuation it is necessary that $\omega_i \neq 0$
(24) **endfor**
end

FIG. 3.1. *Preconditioned BiCGStab using the approximate inverse M as a right preconditioner.*

TABLE 3.1
Iterative methods and partitioning requirements.

Method	Partitioning requirement	Number of distinct vector partitions
BiCGStab right precond. [3, 33]	$PAMP^T$	2
BiCGStab right factor. precond. [3]	$PAM_1M_2P^T$	3
symmetric GMRES right precond. [21]	PAP^T–PMP^T	1
GMRES right precond. [52]	$PAMP^T$	2
GMRES left precond. [52]	$PMAP^T$	2
TFQMR symmetric precond [29]	PAP^T–$PM_2M_1P^T$	2
TFQMR original form [28]	$PM_1AM_2P^T$	3
CGNE [52]	PAQ–PMP^T	2
CGNR [52]	QAP^T–PMP^T	2
CGS right precond. [3]	$PAMP^T$	2
PCG [3, 30, 45]	PAP^T–PMM^TP^T	2
	PAP^T–PMP^T	2

2. If two matrices are written consecutively in a partitioning requirement, then the two matrix-vector multiplies involving these matrices follow each other without any interleaving synchronization. For example, in the $PAMP^T$ case, the input space of A and the output space of M coincide. In other words, there is an arbitrary permutation matrix and its transpose in between the two matrices which designates a distinct vector partition. Therefore, $PAMP^T$ means unsymmetric partitions PAQ for A and Q^TMP^T for M. We write the number of distinct vector partitions for each method in the rightmost column of Table 3.1.

3. Listing two partitioning requirements separated by "–" means that there is at least one synchronization point between the two matrix-vector multiplies. Therefore, we distinguish the partitioning requirement PAP^T and PMP^T from PAP^T–PMP^T. The first one states only that the output spaces of the matrices coincide with the input spaces of the matrices. The second partitioning requirement further states that the two matrix-vector multiplies are interleaved with synchronizations.

4. Factored approximate inverse M_1M_2 can be used (the table contains such an example for BiCGStab) instead of M by just writing the factors consecutively in place of M to determine their partitioning requirement. For example, the use of a factored approximate inverse in the right preconditioned CGNE necessitates PAQ–$PM_1M_2P^T$, which in turn gives the requirements PAQ, PM_1R, and $R^TM_2P^T$.

5. The given requirements are independent of the dimensions along which the matrices are partitioned. That is, matrices can be partitioned rowwise or columnwise, whichever is preferable.

In choosing a partitioning dimension, three issues should be considered. The first issue is the individual matrix characteristics, i.e., the number of nonzeros per row and column. If, for example, a matrix has dense rows but no dense columns, then it is advisable to partition it along the columns [37]. This choice will more likely lead to reduced communication volume and better computational load balance compared to partitioning along the rows [37].

The second issue in choosing a partitioning dimension is the relation between the partitioning requirement and the set of concurrent tasks to be partitioned. For example, in the $PAMP^T$ case, we have four partitioning choices for the pair of A (of size $m \times n$) and M (of size $n \times m$) matrices: (RR), (RC), (CR), and (CC). In the RR scheme, the partitioning determines the output-space permutation for the two matrices. Since the output spaces of the two matrices differ, there are a total of $m+n$ tasks, each representing computations involving either A or M. In the CC scheme, the partitioning determines the input-space permutation. Since the input spaces differ, there are a total of $n + m$ tasks, each representing computations involving either A or M. In the RC and CR schemes, the partition determines the permutation for the coinciding input and output spaces. Therefore, in the RC and CR schemes, the number of tasks reduces to m and n, respectively, each representing computations involving both A and M. In any reasonable task partitioning for the RC and CR schemes, each processor is guaranteed to take part in both of the multiplies. Therefore, these two schemes may lead to more efficient parallel algorithms.

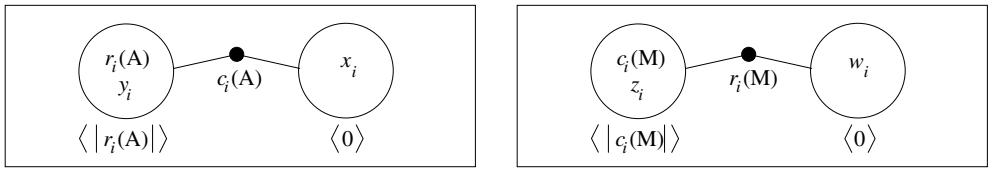
The third issue is the arrangement of computations and communications. Consider the partitioning requirement of $PAMP^T$ for the multiplies of the form $y \leftarrow AMz$, which are performed as $x \leftarrow Mz$ and then $y \leftarrow Ax$. For each multiply, there exists an expand or a fold communication operation. The partitioning dimension determines whether these communications take place before or after the local com-

putations. In the RC partitioning scheme, the fold and expand operations take place successively in between the two multiplies. There are dependencies between these two communication operations; before expanding a particular x-vector entry it should be folded. Because of these dependencies, the successive fold and expand operations are likely to incur a local synchronization point which separates the two multiplies. Consequently, processor loads should be balanced for the individual matrix-vector multiplies in the RC scheme. The RR and CC schemes have either an expand or a fold in between the two multiply operations. Although such communications do not incur synchronization points under the assumption that each processor has enough local computation which overlaps with the incoming messages, it is advisable to balance processors' loads for the individual matrix-vector multiplies when the matrices have a comparable number of nonzeros. The CR scheme is unique in that the two multiply operations are performed successively without any interleaving communication. Therefore, processor loads may be balanced for the overall computation $y \leftarrow AMz$.

4. Building composite hypergraph models. We combine enhanced hypergraph models of the matrix-vector multiplies with the coefficient matrix A and the preconditioner matrix M or its factors M_1 and M_2 into a composite hypergraph whose partitioning meets the requirements given in section 3. We define two operations to combine the enhanced hypergraph models. These are called vertex amalgamation and vertex weighting. The first operation is used to enforce identical partitions on the vertices of the individual hypergraphs. The second operation is used to enable load balancing. The key point in combining a number of hypergraphs is to preserve the identities of the nets of the individual hypergraphs.

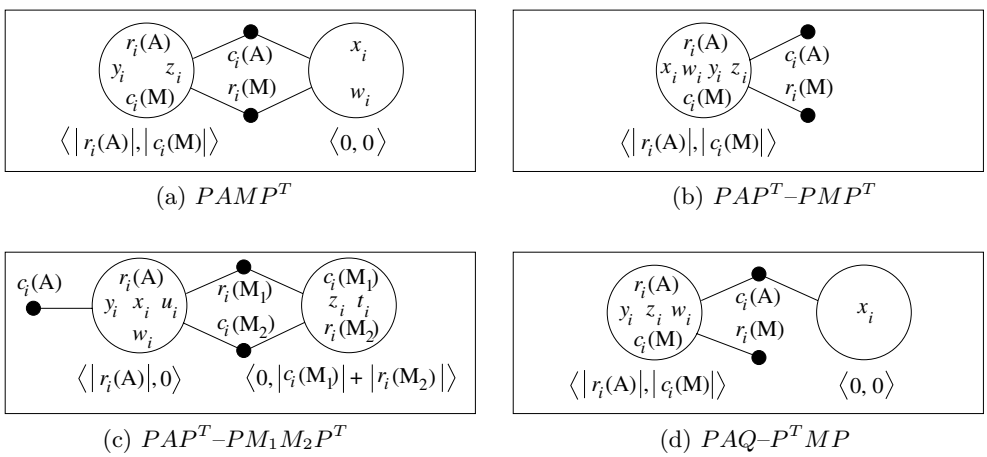
In the following discussion, we assume that A is to be partitioned rowwise and M is assumed to be partitioned columnwise. In the factored case, where $M = M_1M_2$, M_1 is to be partitioned columnwise, and M_2 is to be partitioned rowwise. That is, we have the enhanced column-net hypergraphs for A and M_2 and the enhanced row-net hypergraph for M_1. As discussed in section 2.3, weights can be associated with the vertices in reference to the vector entries. However, we do not use weights for the vector entries and show only the weights corresponding to the matrix rows or columns to simplify the discussion. Figure 4.1 shows portions of the enhanced column-net hypergraph model for $y \leftarrow Ax$ and the enhanced row-net hypergraph model for $w \leftarrow Mz$. These portions will be used while building composite hypergraph models.

Vertex amalgamation. In this operation, the vertices of the individual hypergraphs are combined into a single vertex. The net set of the resulting composite vertex is set to the union of the nets of the constituent vertices. For example, in the $PAMP^T$ case for the operations $w \leftarrow Mz$ and $y \leftarrow Ax$, we amalgamate the row-vertex $r_i(A)$ with the column-vertex $c_i(M)$ into v_i so that $n(v_i) = n(r_i(A)) \cup n(c_i(M))$. Furthermore, we amalgamate the vertex x_i with the vertex w_i to avoid a vector reordering operation. Figure 4.2(a) shows a portion of the resulting composite hypergraph. In a partition of the composite hypergraph, a row or a column vertex v_i being in a part $\mathcal{V}_k$ shows that the processor P_k is responsible for performing multiplications with the ith row or ith column of the matrices. Similarly, a vector-entry vertex x_i being in a part $\mathcal{V}_k$ shows that the processor P_k is responsible for folding or expanding x_i. Figures 4.2(b) to 4.2(d) show portions of the composite hypergraph models built using a vertex amalgamation operation for some other partitioning requirements from Table 3.1.



(a) (b)

FIG. 4.1. *Portions of enhanced hypergraph models for* (a) *row-parallel* $y \leftarrow Ax$, (b) *column-parallel* $w \leftarrow Mz$. *We use* $c_i(\cdot)$ *and* $r_i(\cdot)$ *to represent, respectively, the* i*th column and* i*th row of the matrices;* $|c_i(\cdot)|$ *and* $|r_i(\cdot)|$ *to represent the number of nonzero elements in the* i*th column and row, respectively; and* $\langle\cdot,\cdot\rangle$ *to represent the weight(s) of a vertex. The nets are labeled with a single* $r_i(\cdot)$ *or a single* $c_i(\cdot)$.



(a) $PAMP^T$ (b) PAP^T–PMP^T

(c) PAP^T–$PM_1M_2P^T$ (d) PAQ–P^TMP

FIG. 4.2. *Composite hypergraph models for different partitioning requirements. The computations to be carried out are* $y \leftarrow Ax$, $w \leftarrow Mz$, $w \leftarrow M_1z$, *and* $t \leftarrow M_2u$.

Vertex weighting. This operation is used to enable load balancing. Remember that in some of the iterative methods there are synchronization points between different matrix-vector multiplies. That is, computations occur in phases. Therefore, we define multiple weights for vertices—one for each computation phase—and use a multiconstraint formulation to obtain load balance for each computation phase. For a certain phase, the weight of a vertex is set to the weight of the constituent vertex in the hypergraph of the matrix associated with that phase. In case the vertices of the simple hypergraphs bear weights for the vector entries, another weight component should be defined to account for those weights. Consider right preconditioned symmetric-GMRES [21] and its partitioning requirement PAP^T–PMP^T which designates $w \leftarrow Mz$ and $y \leftarrow Ax$, where all vectors are to be partitioned conformally. As seen in Figure 4.2(b), we apply vertex amalgamation to the vertices x_i and w_i. Since symmetric partitions are required for both of the multiplies, we also amalgamate the row-vertex $r_i(A)$ and the column-vertex $c_i(M)$ with the vertex composed of x_i and w_i. The vertex in Figure 4.2(b) has two weights, since the multiplies with A and M occur in different phases. The first weight represents the computational load associated with the ith row of A, i.e., $|r_i(A)|$. The second weight represents the computational load associated with the ith column of M, i.e., $|c_i(M)|$. In some cases, different matrix-vector multiplies occur successively

without any interleaving synchronization. In these cases, the weight of a composite vertex is set to the sum of the weights of its constituent vertices. Consider the TFQMR method using symmetric preconditioning and its partitioning requirement PAP^T–$PM_1M_2P^T$. Since M_1 and M_2 are partitioned columnwise and rowwise, respectively, there is no synchronization between the respective matrix-vector multiplies. Therefore, the weights of $c_i(M_1)$ and $r_i(M_2)$ are added up as seen in Figure 4.2(c).

Note that partitioning a composite hypergraph results in partitioning the coefficient and preconditioner matrices simultaneously, since the vertices of the composite hypergraph cover the rows or columns of each matrix. Given a partition on the composite hypergraph, we obtain a rowwise or columnwise partition and row and column permutations for each matrix by using the partitions on the vertices pertaining to the respective matrix-vector multiply operation. For example, when a matrix A is to be partitioned rowwise, we use the partition on the composite vertices that contain the rows of A to obtain both a rowwise partition on A and a row permutation, and we use the partition on the composite vertices that contain the input vector entries to obtain a column permutation. Having defined the row and column permutations for each matrix, we obtain the following theorem.

THEOREM 4.1. [illegible]. The proof follows easily by using the same arguments stated in the proof of Theorem 2.1 under the following observations. First, the identities of the nets are preserved while building the composite models. Second, each vertex of a net contains at least one vertex of the original enhanced hypergraph model. Third, the connectivity of a net can be calculated by using only the vertices of the respective enhanced hypergraph model. □

Illustration. Consider the right preconditioned BiCGStab method and its partitioning requirement $PAMP^T$ obtained in section 3. Let A and M be the matrices shown in Figures 2.2 and 2.1, respectively. Suppose that A is to be partitioned columnwise and M is to be partitioned rowwise. We generate the enhanced row- and column-net hypergraph models of A and M, respectively, as shown in Figures 2.2 and 2.1. The partitioning requirement imposes identical partitions on the columns of A and rows of M. Hence, we amalgamate the vertices $c_i(A)$ and $r_i(M)$. The method has no synchronization point between the two multiplies, and the separated expand and fold operations do not cause synchronization. Therefore, we add the weights of the vertices $c_i(A)$ and $r_i(M)$. Since the partitioning requirement imposes identical partitions on the output vector y of the first multiply ($y \leftarrow Ax$) and the input vector z of the second multiply ($w \leftarrow Mz$), we apply vertex amalgamation to the vertices y_i and z_i. A 4-way partition of the resulting hypergraph is shown in Figure 4.3(a). In order to distinguish the nets, the source matrix names are written next to them. The pins of the internal nets are not shown for the sake of clarity. The permutations on the matrices induced by the composite hypergraph partitioning are shown in Figure 4.3(b). Note that the resulting partitions and permutations are identical to those shown in Figures 2.2 and 2.1. As seen from Figure 4.3(a), the cutsize is 10, and hence the total volume of communication is 10 units, where each multiply contributes 5.

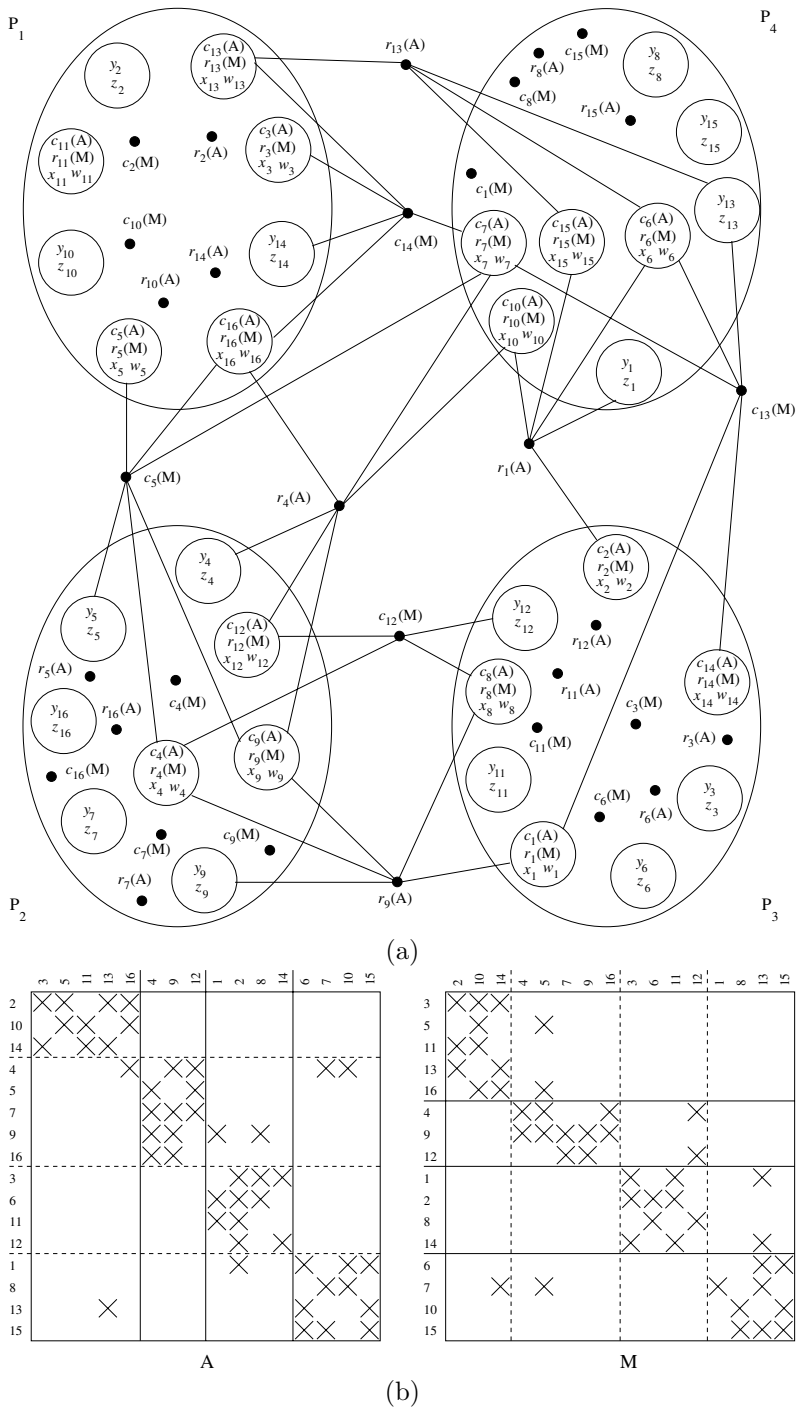


FIG. 4.3. (a) *A composite hypergraph formed by combining the enhanced row-net hypergraph of A and the enhanced column-net hypergraph of M for the computations $y \leftarrow Ax$ and $w \leftarrow Mz$ and a partition which meets the requirement $PAMP^T$. The pins of the internal nets and the weights of the vertices are not shown.* (b) *A columnwise partition of A and a rowwise partition of M induced by the partition on the vertices of the composite hypergraph.*

5. Further notes.

5.1. Investigations on composite models. In the enhanced hypergraph model, the partition on the vertices corresponding to the vector entries are used to obtain permutation matrices. In the previous computational hypergraph models [18], each net is mapped to a part in its connectivity set to obtain permutation matrices. The freedom in mapping a net to one of the parts in its connectivity set has been exploited to minimize communication cost metrics such as the total number of messages and the maximum volume per processor defined in terms of sends [58, 59], and maximum volume per processor [13, 63] defined in terms of both sends and receives. These works achieve their goals by assigning the vector entries to processors upon partitioning the matrix with the computational hypergraph models. In the enhanced hypergraph model and the composite hypergraph model, the vertices that do not contain row or column vertices can be reassigned with the same freedom. For example, the vertex formed by x_i and w_i in Figure 4.2(a) and the vertex x_i in Figure 4.2(d) can be reassigned to optimize the aforementioned communication cost metrics.

5.2. Generalizations. The computational structure of the preconditioned iterative methods is similar to that of a more general class of scientific computations, including multiphase, multiphysics, and multimesh simulations.

In multiphase simulations, there are a number of computational phases separated by global synchronization points. The existence of the global synchronizations necessitates each phase to be load balanced individually. In our model, the multiple weights associated with vertices can be used to achieve this goal as described in [41, 64].

In multiphysics simulations, a variety of materials and processes are analyzed using different physics procedures. In these types of simulations, computational as well as memory requirements are not uniform across the mesh [54]. For scalability issues, processor loads should be balanced in terms of these two components. The multiconstraint partitioning framework also addresses these problems [54].

In multimesh simulations, a number of grids with different discretization schemes and with arbitrary overlaps are used. The existence of overlapping grid points necessitates the simultaneous partitioning of the grids [54]. This simultaneous partitioning should balance the computational loads of the processors and minimize the communication cost due to interactions within a grid as well as interactions among different grids. The vertex amalgamation operation used in our models can be applied to overlapped grid points to build a composite hypergraph. With the use of vertex weighting operations, our models can be used to address the partitioning problem in the multimesh computations. Although simultaneous partitioning seems to be more adequate for these types of problems, independent partitioning is also possible (see, for example, [51] for independent partitionings on a two-grid system).

In some contact/impact problems, there is a priori knowledge about the to-be-contacting surfaces. The implementation in [38] uses this information to decompose the underlying mesh among processors. The implementation uses the graph model and adds edges between the to-be-contacting surface elements. Partitioning such a graph using two constraints balances the loads of processors, both for the finite element analysis and for the contact detection phases. The partitioning algorithm tries to minimize the communication cost by minimizing the edge cut. By modeling the interactions among the to-be-contacting surface elements with hypergraphs, we can build a composite hypergraph to address these problems. However, the implementation in [38] is reported to be suffering from load imbalances and to be limited to a small number of processors; see the comments on it in [50].

In obtaining partitions for two or more computation phases interleaved with synchronization points, our models lead to the minimization of the overall sum of the total volume of communication in all phases. For the preconditioned iterative methods, this approach will likely minimize the communication cost in one full step. However, in more sophisticated simulations, the magnitude of the interactions in one phase may be different from that of the interactions in another one. In such settings, minimizing the total volume of communication in each phase separately may be advantageous [53]. This problem can be formulated as a multiobjective hypergraph partitioning problem [1, 53, 55] on the composite hypergraphs.

As discussed above, our models can be applied to the multiphase, multiphysics, and multimesh computations but with certain limitations. The dependencies must remain the same throughout the computations; our methods cannot be used, for example, in adaptive mesh refinement. The weights assigned to elements, for load balancing issues, should be static and available prior to the partitioning; our methods cannot be used for applications whose computational requirements vary in time [35]. If, however, the computational requirements change gradually in time, then our methods can be used to repartition the load at certain time intervals. Some problems are more suitable to geometric partitioning methods; contact detection without a priori knowledge of the contacting surfaces, for example, should be performed on geometrically close elements [14, 40, 50]. Our methods, in their current forms, will probably be of little help in those problems. To be useful, the models should be enriched with some geometric constructs, as is done in [40].

6. Experiments. We chose the right preconditioned BiCGStab method to evaluate the effectiveness of the proposed composite hypergraph partitioning approach. We used a set of unsymmetric sparse matrices from the University of Florida Sparse Matrix Collection [27]. Approximate inverse preconditioners were obtained using SPAI version 3.0 [32]. Factored approximate inverses were obtained using AINV [11]. These two programs have parameters that affect the quality of the preconditioner matrices. However, we set the parameters in such a way that the number of nonzeros of the approximate inverse or the total number of nonzeros of the factors of the approximate inverse is at most twice and at least half the number of nonzeros of the coefficient matrix. We adjusted the tolerance parameter *eps*, the number of nonzero entries allowed per step *mn*, and the number of steps *ns* in SPAI. In AINV, we adjusted the drop tolerance parameter τ. The properties of the matrices, approximate inverses, and factors of the approximate inverses are given in Table 6.1. In the table, the coefficient matrices are listed with a suffix of A; the approximate inverse matrices are listed with a suffix of M; the factors of the approximate inverse matrices are listed with suffixes of Z and W, where the approximate inverse is equivalent to ZW. The hypergraphs were partitioned using PaToH [19] with default parameters. The imbalance among processor loads is kept below 10% in all partitioning instances. Throughout this section, we use the term "SPAI-matrices" to refer to a pair consisting of a coefficient matrix and its approximate inverse preconditioner. Similarly, we use "AINV-matrices" to refer to a triplet of coefficient matrix and the factors of its approximate inverse preconditioner.

Since the partitioning tool PaToH incorporates randomized algorithms, it was run 20 times starting from different random seeds for partitioning the hypergraphs. Averages of the resulting communication volumes of these runs are displayed in the following tables. PaToH is a fairly stable toolkit; the standard deviation of the total communication volumes of the 20 runs is less than 4% of the mean for all hypergraphs

TABLE 6.1
Properties of test matrices.

Matrix	Number of rows/cols	Number of nonzeros					
		Total	Average	Row		Column	
			row/col	min	max	min	max
Zhao1-A	33861	166453	4.9	3	6	2	7
big-A	13209	91465	6.9	3	12	3	12
cage11-A	39082	559722	14.3	3	31	3	31
cage12-A	130228	2032536	15.6	5	33	5	33
epb2-A	25228	175027	6.9	3	87	3	87
epb3-A	84617	463625	5.5	3	6	3	7
mark3jac060-A	27449	170695	6.2	2	44	2	47
olafu-A	16146	1015156	62.9	24	89	24	89
stomach-A	213360	3021648	14.2	7	19	6	22
xenon1-A	48600	1181120	24.3	1	27	1	27
SPAI							
Zhao1-M	33861	180988	5.3	1	11	1	16
big-M	13209	109088	8.3	2	22	1	21
cage11-M	39082	424708	10.9	2	51	2	21
cage12-M	130228	1444650	11.1	1	62	2	21
epb2-M	25228	244453	9.7	2	177	2	21
epb3-M	84617	532851	6.3	2	20	2	20
mark3jac060-M	27449	276586	10.1	1	37	1	21
olafu-M	16146	719873	44.6	5	114	4	46
stomach-M	213360	2910283	13.6	2	120	2	46
xenon1-M	48600	878143	18.1	1	35	1	21
AINV; $M = ZW$							
Zhao1-Z	33861	179803	5.3	1	13	1	28
Zhao1-W	33861	57832	1.7	1	5	1	6
big-Z	13209	56302	4.3	1	11	1	13
big-W	13209	56314	4.3	1	13	1	11
cage11-Z	39082	302775	7.7	1	26	1	110
cage11-W	39082	299939	7.7	1	26	1	32
epb2-Z	25228	116161	4.6	1	13	1	22
epb2-W	25228	107620	4.3	1	36	1	19

(except `Zhao1-W` in Table 6.3, which has a negligible total communication volume). Details of the communication patterns can be found in the accompanying report [60].

6.1. Composite versus simple hypergraph partitioning. Here we evaluate two alternative approaches to partitioning composite hypergraphs. These alternative approaches are based on partitioning simple hypergraphs, i.e., partitioning hypergraphs of a single matrix. The first alternative is to obtain independent partitions on the matrices by partitioning the simple hypergraph models of the coefficient and the preconditioner matrices independently. This approach requires vector reordering in between the two matrix-vector multiplies. We discuss this alternative in section 6.1.1. The second alternative is to obtain the same partition for the coefficient and preconditioner matrices. For this purpose, we partition the coefficient matrices symmetrically by rows or columns using hypergraph models and apply the resulting partitions to the preconditioner matrices as well. This alternative avoids the vector reordering op-

TABLE 6.2
Total communication volume for 64-way simple hypergraph (C/R and R/C) and composite hypergraph (CR and RC) partitionings for SPAI-matrices. In simple hypergraph partitionings, A and M are partitioned independently, and therefore vector entries are reordered in between the two multiplies to meet the partitioning requirement.

	C/R (simple)		CR (composite)	R/C (simple)		RC (composite)
	Volume		Volume	Volume		Volume
Matrix	SpMxV	Reorder	SpMxV	SpMxV	Reorder	SpMxV
Zhao1-A	11421	134307	13026	10811	135348	13734
Zhao1-M	9857		10809	11756		14551
big-A	3210	52150	3666	3215	52804	5453
big-M	3054		3562	5447		7610
cage11-A	58177	153366	63779	58272	151840	79052
cage11-M	32937		38714	42512		61304
cage12-A	185531	504816	207077	185191	510824	248253
cage12-M	98493		119287	122513		180128
epb2-A	5984	98523	6731	5527	100912	10610
epb2-M	5967		7215	7411		11694
epb3-A	5713	332064	8167	7250	333320	17911
epb3-M	6846		9759	7057		18512
mark3jac060-A	13331	108209	17447	12676	109780	19503
mark3jac060-M	15567		18970	16563		21096
olafu-A	14012	63743	16743	13912	57372	23870
olafu-M	25137		29348	24735		36427
stomach-A	36800	837087	47689	37219	853440	77080
stomach-M	44232		57755	47965		89479
xenon1-A	26710	189847	29644	26669	178388	36044
xenon1-M	33597		40270	33241		46970

eration by partitioning all vectors conformally. Note that since we partition a single matrix, a graph model could also be used. We discuss this alternative in section 6.1.2.

6.1.1. Simple hypergraph partitioning: Independent partitions on the matrices. We have conducted experiments with the CR and RC partitionings of SPAI-matrices where the first partition dimension corresponds to the matrix A and the second to the matrix M. In the independent partitioning approach, we partition the simple hypergraph models of the matrices A and M independently.

Table 6.2 displays the average communication volumes in the sparse matrix-vector multiply (SpMxV) operations resulting from the composite and simple hypergraph partitionings for 64-way partitioning of SPAI-matrices. The table also shows the volume of communication required to reorder the vector entries—in an iteration of the BiCGStab method—when the matrices are partitioned independently. Suppose that symmetric partitions PAP^T and QMQ^T were obtained on the A and M matrices. Then at each iteration we have to reorder $\hat{p}$ and $\hat{s}$ from Q to P after the matrix-vector multiplies at lines 12 and 17 of the BiCGStab method (see Figure 3.1), respectively. We also have to reorder v and t from P to Q before the vector update at line 15 and the inner product at line 19, respectively. The volume of communication in the reordering operation is given as the average of 20 different partitions of SPAI-matrices. In all of the partitioning instances, the volume of communication in the reordering operation

TABLE 6.3

Total communication volume for 64*-way simple hypergraph (C/R/C and R/C/R) and composite hypergraph (CRC and RCR) partitionings for AINV-matrices. In simple hypergraph partitionings, A, Z, and W are partitioned independently, and therefore vector entries are reordered in between the successive multiplies to meet the partitioning requirement.*

	C/R/C (simple)		CRC (composite)	R/C/R (simple)		RCR (composite)
	Volume		Volume	Volume		Volume
Matrix	SpMxV	Reorder	SpMxV	SpMxV	Reorder	SpMxV
Zhao1-A	11421	199423	15258	10811	199881	14977
Zhao1-Z	10808		12811	13580		17376
Zhao1-W	170		2494	377		4380
big-A	3210	77975	3730	3215	78077	3633
big-Z	1861		2262	1947		2314
big-W	1859		2305	1948		2325
cage11-A	58177	225147	65430	58272	226743	65052
cage11-Z	22023		28640	40428		48783
cage11-W	21676		27397	39731		48453
epb2-A	5984	148944	10313	5527	148830	10490
epb2-Z	2058		3967	2478		5672
epb2-W	1431		3786	1174		4935

itself is higher than that obtained by the simultaneous partitioning method. These high volumes of communication and the associated message start-up overheads prohibit the use of the independent partitioning method. For example, the independent partitioning method incurs higher total communication volume than the proposed simultaneous partitioning method by an average ratio of 8.4 for 32-way CR partitioning (see [60]). The average ratio in 64-way CR partitioning is 6.5. For RC partitioning, the average ratios are 5.6 and 4.2 for 32- and 64-way partitionings, respectively.

Table 6.3 displays the averages of the communication volumes of the 64-way simultaneous and independent partitionings for AINV-matrices. In this table CRC corresponds to the case where the A, Z, and W matrices are partitioned columnwise, rowwise, and columnwise, respectively. Similarly, RCR corresponds to the case where those matrices are partitioned rowwise, columnwise, and rowwise. With AINV preconditioning, the independent partitioning method requires two additional vector-reordering operations (due to the chains of matrix-vector multiplies at lines 12 and 17 of the BiCGStab method). For 64-way partitioning of AINV-matrices, the average ratios of the communication volumes in the independent partitionings (including the reordering cost) to those in the simultaneous partitionings is 7.2 for the CRC case and 6.5 for the RCR case. The average ratios for 32-way partitionings are 10.1 and 9.0 for the CRC and RCR cases, respectively (see [60]). As is clear from these numbers, the independent partitioning approach is not feasible for the AINV-matrices as well.

Consider the difference between the total communication volumes of the composite and simple hypergraph partitionings (without the reordering cost). The increases in the total communication volumes in the composite partitionings remain below 26% of those in the simple partitionings, on average, for the 32-way CR partitioning instances. The minimum and the maximum of these increases are 13% (`Zhao1`) and 61% (`epb3`). The 64-way CR partitionings give better ratios. The average increase is 20% with the minimum and the maximum being 12% and 43%, which are obtained for the same matrices. In fact, for each matrix the increase in the 64-way CR partitioning

TABLE 6.4
Relation between the sparsity patterns of the coefficient matrices and the approximate inverses. We use A and M to represent the set of the positions of the nonzeros in the corresponding matrices.

Matrix	Number of nonzeros			
	$A \cup M$	$A \setminus M$	$M \setminus A$	$\frac{A \cap M}{M}$
Zhao1	234205	67752	53217	0.63
big	147632	56167	38544	0.49
cage11	780776	221054	356068	0.48
cage12	2784199	751663	1339549	0.48
epb2	333794	158767	89341	0.35
epb3	773107	309482	240256	0.42
mark3jac060	397706	227011	121120	0.18
olafu	1357370	342214	637497	0.52
stomach	5182305	2160657	2272022	0.26
xenon1	1520936	339816	642793	0.61

is smaller than the increase in the 32-way CR partitioning. We investigated the 8- and 16-way CR partitionings as well [60] and observed that for each matrix in our data set the larger the number of parts, the smaller the increases. The same relation holds for most of the RC partitioning cases and for the CRC and RCR partitioning of AINV-matrices [60]. The reason behind this may be the following. The cutsize function almost always increases monotonically with the increasing K. In other words, the flexibility of finding better partitions reduces with the increasing K. At the limit, where $K = |\mathcal{V}|$ and all the nets are in cut, the cutsize of a partition on the composite hypergraph will be equivalent to the sum of the cutsizes of partitions on the simple hypergraphs (i.e., $nnz(A) + nnz(M) - 2m$) that forms it. Therefore, the difference between the total communication volumes should decrease as K increases and has to be zero at the limit.

6.1.2. Simple hypergraph partitioning: The same partition on the matrices. Obtaining a symmetric partition on A and then applying the resulting partition to M results in CC or RR partitionings on the A and M matrices. Recall that in CC and RR partitioning schemes, there is a communication phase in between the two matrix-vector multiplies. Since the A and M matrices have a comparable number of nonzeros (see Table 6.1), processor loads for the two matrix-vector multiplies should be balanced separately, i.e., a two-constraint formulation is necessary.

Observe that the approach evaluated here disregards the sparsity pattern of the preconditioner matrices. However, the sparsity patterns of the approximate inverse preconditioners are usually related to the sparsity patterns of the coefficient matrices [23, 39]. Therefore, the partitions on the coefficient matrices are expected to be effective for the preconditioners. To justify this reasoning, we show the relation between the sparsity patterns of the coefficient matrices and the approximate inverses in Table 6.4. As seen in the table, the relation between the sparsity patterns of the coefficient and preconditioner matrices varies; 63% of the nonzeros of `Zhao1-M` are covered by the nonzeros of `Zhao1-A`, and only 18% of the nonzeros of `mark3jac060-M` are covered by the nonzeros of `mark3jac060-A`. Another reason for using the same partition on the coefficient and preconditioner matrices is the following. Parallel construction of the approximate inverse preconditioners produces preconditioners in such a way that the initial partitions on the coefficient matrices become partitions on the preconditioner matrices. For example, the left approximate inverse preconditioners

TABLE 6.5

Total communication volume for 64*-way simple hypergraph and composite hypergraph partitionings for CC and RR partitioning of SPAI-matrices. In simple hypergraph partitionings, A and M have the same symmetric partition which is obtained by partitioning A.*

	CC			RR		
	Volume		%	Volume		%
Matrix	Simple	Composite	gain	Simple	Composite	gain
Zhao1-A	12982	12892	8	12502	12131	8
Zhao1-M	15406	13116		12523	10896	
big-A	4089	3849	18	4068	3783	20
big-M	8082	6114		5548	3904	
cage11-A	67198	64928	18	68374	63612	19
cage11-M	72710	49361		60385	41030	
cage12-A	213360	208209	18	216501	203197	18
cage12-M	218227	146726		183548	125836	
epb2-A	9820	9357	16	9611	8545	19
epb2-M	12913	9766		11649	8590	
epb3-A	11764	11293	26	12441	11607	22
epb3-M	21010	13063		18036	12201	
mark3jac060-A	15676	18090	34	18107	15891	36
mark3jac060-M	42608	20183		34929	18005	
olafu-A	19802	16733	26	20273	16173	26
olafu-M	38318	26470		39026	27911	
stomach-A	50980	57777	19	52582	58230	13
stomach-M	107817	71060		90605	66561	
xenon1-A	30510	27683	21	29597	27615	17
xenon1-M	49568	35520		47909	36437	

can be efficiently constructed rowwise when the coefficient matrix A is partitioned rowwise [24]. The construction yields the same rowwise partition on the approximate inverse M. Equivalently, a right approximate inverse preconditioner can be efficiently constructed columnwise when the coefficient matrix A is partitioned columnwise.

The row-net hypergraph model of A can be used to obtain a CC partition on A and M. In order to obtain load balance for the two multiplies, the vertices of A are assigned two weights which correspond to the number of nonzeros in the respective columns of A and M matrices. That is, v_i has weights $\langle |c_i(A)|, |c_i(M)| \rangle$. Similarly, the column-net hypergraph model of A, with two weights on the vertices, can be used to obtain an RR partition on A and M.

The composite hypergraph model for the CC partitioning scheme is built in three steps as follows. First, the enhanced row-net hypergraph models of $y \leftarrow Ax$ and $w \leftarrow Mz$ are created. Second, vertex amalgamation operations are applied to the vertices y_i and $c_i(M)/z_i$, and also to the vertices $c_i(A)/x_i$ and w_i for each i. Third, vertex weighting operations are applied to the vertices in such a way that the vertex $y_i/c_i(M)/z_i$ has weights $\langle 0, |c_i(M)| \rangle$, and the vertex $w_i/c_i(A)/x_i$ has weights $\langle |c_i(A), 0| \rangle$ for each i. Theoretically, a three-constraint formulation is necessary to balance the vector operations; however, we use a two-constraint formulation in order to ease the job of the hypergraph partitioning tool. The composite hypergraph model for the RR partitioning scheme is built similarly.

Table 6.5 displays the averages of the communication volumes of the 64-way partitioning of the SPAI-matrices with the composite hypergraphs and simple hypergraph models of the coefficient matrices. The "% gain" columns in this table show the

improvements achieved by the composite hypergraph partitioning as the percentage of the total communication volumes found by partitioning the simple hypergraph of A. The minimum improvements are obtained for the `Zhao1` matrices in both CC and RR cases. The maximum improvements are obtained for the `mark3jac060` matrices in both CC and RR cases. As seen in Table 6.4, the `Zhao1-A` and `Zhao1-M` matrix pair have the highest ratio of common nonzeros, and the `mark3jac060-A` and `mark3jac060-M` matrix pair have the lowest ratio of common nonzeros. Although `xenon1-A` covers 61% of `xenon1-M` (second maximum), the improvements achieved for these matrices are quite satisfactory. The average of the improvements is 20% in the 32- and 64-way CC and RR partitioning choices (see also [60]).

We have also experimented with the 32- and 64-way CC and RR partitionings using single constraint formulation. In the single constraint formulation, the weight of a vertex v_i is set to the sum of the number of nonzeros in the ith columns (rows) of A and M for the CC (RR) partitioning. Both the composite hypergraph and the simple hypergraph formulations were able to obtain balance on the total loads of the processors. Since these formulations ignore the fact that there is a local synchronization, both formulations could not obtain balance on the loads of the processors for the individual matrix-vector multiplies. The composite hypergraph partitioning approach obtained (on average 17%) better solutions than the simple hypergraph partitioning [60]. The best and worst improvements are again obtained for the `mark3jac060` and `Zhao1` matrix pairs, respectively.

6.2. Effects of partitioning dimensions on the composite hypergraph partitioning. Comparing the left and right halves of Table 6.2, we see that the CR partitioning yields, on average, 26% better total communication volume than the RC one. This ratio remains the same for 8-, 16-, and 32-way partitionings (see Table 6.6 and [60]). The matrices in our data set do not have dense rows or dense columns. Therefore, the rowwise and columnwise partitionings of the matrices are expected to be comparable in terms of the total communication volume. This theoretical expectation is verified by the communication volume values listed in the SpMxV columns in Table 6.2 for the simple hypergraph partitioning. In light of this observation, we can deduce that the performance difference between the CR and RC partitioning schemes is due to the two-constraint formulation in the RC scheme. This degradation in the multiconstraint formulation is in concordance with the previously reported results [41, 64]. The degradation in our case stems from two facts. First, the additional balance constraints shrink the search space. Second, the heuristics in PaToH are not very well tailored toward handling the multiple vertex weights.

6.3. Parallelization results. It is important to see whether the theoretical improvements obtained by the proposed simultaneous partitioning method hold in practice. For this purpose, we have implemented a parallel program for the BiCGStab method. The program uses the LAM/MPI 6.5.6 [15] message passing library. The tests were carried out on a Beowulf class [56] PC cluster with 24 nodes. Each node has a 400MHz Pentium-II processor and 128 MB memory. The interconnection network comprises a 3COM Superstack II 3900 managed switch connected to Intel Ethernet Pro 100 Fast Ethernet network interface cards at each node. The system runs Linux kernel 2.4.20 and the Debian GNU/Linux 3.0 distribution.

We are not concerned with the numerics of the preconditioners and the BiCGStab method. Therefore, for each matrix, we let the BiCGStab run for 100 iterations and measure the average running time of a single iteration. In order to guarantee 100 iterations, we set ρ and ω of the BiCGStab method (see Figure 3.1) to 1.0 after computing

TABLE 6.6
Communication patterns for 8- *and* 16-*way composite hypergraph partitionings for SPAI-matrices and the respective speedup values. Matrix name* `mark3jac060` *is shortened to* `mark3_060`.

	8-way					16-way				
	Volume		Message		Sp.	Volume		Message		Sp.
Matrix	tot	max	tot	max	up	tot	max	tot	max	up
	CR									
Zhao1-A	4098	746	32.2	5.7	6.2	6444	586	96.7	9.3	8.7
Zhao1-M	3514	694	32.2	5.6		5478	551	97.0	9.3	
big-A	1032	201	31.4	5.7	5.7	1581	156	73.2	7.5	7.3
big-M	989	191	31.9	5.6		1527	150	75.3	7.5	
cage11-A	24424	4144	54.6	7.0	5.5	34835	3314	201.2	14.8	8.1
cage11-M	14663	2439	55.1	7.0		21010	1917	208.9	15.0	
cage12-A	87542	14306	56.0	7.0	5.9	122878	11925	230.3	15.0	9.4
cage12-M	50962	7839	56.0	7.0		71066	6136	233.1	15.0	
epb2-A	2326	429	39.0	6.4	6.4	3357	371	102.8	9.6	8.6
epb2-M	2242	438	35.0	6.5		3335	335	84.7	8.4	
epb3-A	2354	442	23.9	4.3	7.3	3971	393	66.0	6.5	12.4
epb3-M	3003	536	23.9	4.3		5023	496	66.3	6.5	
mark3_060-A	5249	960	35.2	6.3	5.8	9370	786	115.0	11.3	8.7
mark3_060-M	6323	1182	32.2	6.0		10287	964	105.3	11.0	
olafu-A	3908	960	25.8	5.0	6.7	6489	781	66.2	6.8	10.6
olafu-M	6749	1449	28.0	5.4		11258	1285	77.8	7.8	
stomach-A	14614	2815	21.1	4.0	7.1	24436	2351	67.2	7.0	14.1
stomach-M	16193	3206	21.4	4.0		28014	2652	67.8	7.1	
xenon1-A	10848	2037	36.2	6.5	6.7	15998	1496	113.2	11.3	11.2
xenon1-M	14437	2523	37.7	6.7		21459	2032	117.8	11.8	
	RC									
Zhao1-A	4146	905	33.2	5.8	5.9	6728	734	95.5	9.3	8.1
Zhao1-M	4362	771	33.1	5.6		7141	716	95.5	9.3	
big-A	1467	320	33.9	5.8	5.2	2408	280	84.7	8.8	6.4
big-M	2084	433	34.0	5.8		3326	366	85.9	8.4	
cage11-A	30373	6054	55.9	7.0	4.2	43335	4599	227.0	15.0	5.5
cage11-M	24447	4302	55.9	7.0		34339	3151	227.4	15.0	
cage12-A	107187	17621	56.0	7.0	4.4	147504	12516	239.8	15.0	6.2
cage12-M	80949	15047	56.0	7.0		109472	10885	239.7	15.0	
epb2-A	3074	646	46.7	6.8	6.1	4555	560	132.1	12.9	8.6
epb2-M	3789	814	43.8	6.5		5388	633	109.5	10.6	
epb3-A	6347	1907	39.8	6.8	7.2	8490	1244	108.3	11.7	11.7
epb3-M	6766	1937	39.5	6.8		8991	1480	108.0	11.8	
mark3_060-A	5568	981	40.7	6.7	5.7	9792	906	137.4	13.0	7.4
mark3_060-M	6417	1213	38.6	6.8		11099	1027	126.3	12.2	
olafu-A	5605	1088	30.5	5.8	6.0	9325	1012	86.0	9.1	8.6
olafu-M	9075	1947	33.5	6.2		14636	1636	99.8	10.2	
stomach-A	21139	4354	27.4	5.3	7.1	35856	4255	81.5	8.8	12.7
stomach-M	24538	5221	27.6	5.7		41254	4660	82.8	8.9	
xenon1-A	12654	2322	39.6	6.9	6.1	18800	1797	126.2	12.2	9.2
xenon1-M	16486	2974	40.8	7.0		24477	2286	131.6	12.9	

their actual values. The speedup values corresponding to these running times are given in Table 6.6 under the column *Sp./up*. The given speedup values are the averages of 20 runs corresponding to different partitions. In order to show how the improvements obtained by the proposed method relate to parallel running times, we give the average communication patterns induced by the partitions in the same table as well.

As seen from Table 6.6, CR gives better speedup values than RC for all matrices (the reasons are discussed in section 6.2). On average, CR obtains speedup values of

6.3 and 9.9 for 8- and 16-way partitionings, respectively, where the highest speedups are 7.3 (epb3) and 14.1 (stomach). Meanwhile, RC obtains speedups of 5.8 and 8.4, on average, for 8-way and 16-way partitionings, respectively, where the highest speedups are 7.2 (epb3) and 12.7 (stomach). The lowest speedups for 8-way partitioning are obtained for the cage11 matrix by both of the partitioning schemes. The lowest speedups for 16-way partitioning are obtained for the big and cage11 matrices by the CR and RC schemes, respectively. As seen from Table 6.6, the cage11 matrix pair has a communication pattern inferior to all but the cage12 matrix pair in terms of the total and maximum number of message metrics. Therefore, we were already expecting to have the lowest speedups with the cage matrices. The big matrix has the smallest number of nonzeros. The low granularity of computations may be the reason behind having the lowest speedup with 16-way CR partitioning of the big matrix. The same reasoning may also explain why we obtain better speedups in cage12 than those in cage11.

We have also experimented with the CRC and RCR partitioning schemes for AINV-matrices (see [60]). As expected, the speedup values are not as good as those given in Table 6.6 because of an additional load balancing constraint for the third matrix-vector multiply, and because of an additional communication phase. The highest speedups for 8- and 16-way CRC partitionings are 6.7 and 8.9, respectively. The highest speedups for 8- and 16-way RCR partitionings are 6.4 and 8.9, respectively.

6.4. Partitioning timings. Finally, we comment on the additional partitioning overhead introduced by simultaneous partitioning instead of individual partitionings. Let the sum of the times elapsed in individual partitionings (of the SPAI- and AINV-matrices) be 1.0. Then the average running times of the simultaneous partitioning of the SPAI-matrices with the CR and RC schemes are 1.4 and 1.2, respectively, for all $K = 8$, 16, 32, and 64. The average running times of the simultaneous partitioning of the AINV-matrices with both the CRC and RCR schemes are close to 1.4. These increases are acceptable because the simultaneous partitioning method obtains much smaller total communication volume than the individual partitioning method combined with the reordering cost. See [60] for a list of the partitioning times.

7. Conclusion. We demonstrated that hypergraph models are able to capture the application of multiple matrices. In particular, we developed models that allow simultaneous partitioning of a matrix and its explicit preconditioner or factors of the preconditioner. These points were raised by Hendrickson and Kolda [37].

The computational structure of preconditioned iterative methods abounds in scientific computing applications and data aggregation/reduction applications in large distributed data sets. We discussed the applicability of the proposed work in certain scientific computations. We think that the proposed work is applicable in distributed data-set applications where hypergraph models have already been used [22].

Acknowledgments. We thank Bruce Hendrickson for introducing us to the simultaneous partitioning problem; Michele Benzi, Miroslav Tůma, and Marcus Grote for their research software; Levent Gürel and his computational electromagnetics research group at Bilkent University for granting us access to their computing machinery. We are indebted to anonymous referees for insightful comments.

REFERENCES

[1] C. Ababei, N. Selvakkumaran, K. Bazargan, and G. Karypis, *Multi-objective circuit partitioning for cutsize and path-based delay minimization*, in Proceedings of the 2002

IEEE/ACM International Conference on Computer-aided Design (ICCAD '02), San Jose, CA, 2002, pp. 181–185.

[2] C. Aykanat, A. Pinar, and Ü. V. Çatalyürek, *Permuting sparse rectangular matrices into block-diagonal form*, SIAM J. Sci. Comput., 25 (2004), pp. 1860–1879.

[3] R. Barrett, M. W. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. A. Van Der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, 1994.

[4] M. Benzi, *Preconditioning techniques for large linear systems: A survey*, J. Comput. Phys., 182 (2002), pp. 418–477.

[5] M. Benzi, J. K. Cullum, and M. Tůma, *Robust approximate inverse preconditioning for the conjugate gradient method*, SIAM J. Sci. Comput., 22 (2000), pp. 1318–1332.

[6] M. Benzi, J. C. Haws, and M. Tůma, *Preconditioning highly indefinite and nonsymmetric matrices*, SIAM J. Sci. Comput., 22 (2000), pp. 1333–1353.

[7] M. Benzi, C. D. Meyer, and M. Tůma, *A sparse approximate inverse preconditioner for the conjugate gradient method*, SIAM J. Sci. Comput., 17 (1996), pp. 1135–1149.

[8] M. Benzi and M. Tůma, *A sparse approximate inverse preconditioner for nonsymmetric linear systems*, SIAM J. Sci. Comput., 19 (1998), pp. 968–994.

[9] M. Benzi and M. Tůma, *A comparative study of sparse approximate inverse preconditioners*, Appl. Numer. Math., 30 (1999), pp. 305–340.

[10] M. Benzi and M. Tůma, *A parallel solver for large-scale Markov chains*, Appl. Numer. Math., 41 (2002), pp. 135–153.

[11] M. Benzi and M. Tůma, *Private communication*, 2003.

[12] L. Bergamaschi, G. Pini, and F. Sartoretto, *Approximate inverse preconditioning in the solution of sparse eigenproblems*, Numer. Linear Algebra Appl., 7 (2000), pp. 99–116.

[13] R. H. Bisseling and W. Meesen, *Communication balancing in parallel sparse matrix-vector multiplication*, Electron. Trans. Numer. Anal., 21 (2005), pp. 47–65.

[14] K. Brown, S. Attaway, S. J. Plimpton, and B. Hendrickson, *Parallel strategies for crash and impact simulations*, Comput. Methods Appl. Mech. Engrg., 184 (2000), pp. 373–390.

[15] G. Burns, R. Daoud, and J. Vaigl, *LAM: An open cluster environment for MPI*, in Proceedings of Supercomputing Symposium '94, J. W. Ross, ed., University of Toronto, 1994, pp. 379–386.

[16] Ü. V. Çatalyürek, *Hypergraph Models for Sparse Matrix Partitioning and Reordering*, Ph.D. thesis, Computer Engineering and Information Science, Bilkent University, Ankara, Turkey, 1999.

[17] Ü. V. Çatalyürek and C. Aykanat, *Decomposing irregularly sparse matrices for parallel matrix-vector multiplications*, in Parallel Algorithms for Irregularly Structured Problems, Lecture Notes in Comput. Sci. 1117, Springer, Berlin, 1996, pp. 75–86.

[18] Ü. V. Çatalyürek and C. Aykanat, *Hypergraph-partitioning based decomposition for parallel sparse-matrix vector multiplication*, IEEE Trans. Parallel Distrib. Systems, 10 (1999), pp. 673–693.

[19] Ü. V. Çatalyürek and C. Aykanat, *PaToH: A Multilevel Hypergraph Partitioning Tool*, Version 3.0, Tech. Report BU-CE-9915, Computer Engineering Department, Bilkent University, Ankara, Turkey, 1999.

[20] Ü. V. Çatalyürek and C. Aykanat, *A hypergraph-partitioning approach for coarse-grain decomposition*, in Proceedings of Scientific Computing 2001 (SC2001), Denver, CO, 2001, pp. 10–16.

[21] T. F. Chan, E. Chow, Y. Saad, and M. C. Yeung, *Preserving symmetry in preconditioned Krylov subspace methods*, SIAM J. Sci. Comput., 20 (1998), pp. 568–581.

[22] C. Chang, T. Kurc, A. Sussman, Ü. V. Çatalyürek, and J. Saltz, *A hypergraph-based workload partitioning strategy for parallel data aggregation*, in Proceedings of the Tenth SIAM Conference on Parallel Processing for Scientific Computing, Portsmouth, VA, CD-ROM, SIAM, Philadelphia, 2001.

[23] E. Chow, *A priori sparsity patterns for parallel sparse approximate inverse preconditioners*, SIAM J. Sci. Comput., 21 (2000), pp. 1804–1822.

[24] E. Chow, *Parallel implementation and practical use of sparse approximate inverse preconditioners with a priori sparsity patterns*, Internat. J. High Perform. Comput. Appl., 15 (2001), pp. 56–74.

[25] E. Chow and Y. Saad, *Approximate inverse preconditioners via sparse-sparse iterations*, SIAM J. Sci. Comput., 19 (1998), pp. 995–1023.

[26] J. K. Cullum, K. Johnson, and M. Tůma, *Effects of problem decomposition on the convergence behavior of parallel numerical algorithms*, Numer. Linear Algebra Appl., 10 (2003), pp. 445–465.

[27] T. Davis, *University of Florida Sparse Matrix Collection*, http://www.cise.ufl.edu/research/sparse/matrices/.

[28] R. W. Freund, *A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems*, SIAM J. Sci. Comput., 14 (1993), pp. 470–482.

[29] R. W. Freund and N. M. Nachtigal, *A New Krylov-Subspace Method for Symmetric Indefinite Linear Systems*, Tech. Report ORNL/TM-12754, Oak Ridge National Laboratories, 1994.

[30] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, MD, 1996.

[31] N. I. M. Gould and J. A. Scott, *Sparse approximate-inverse preconditioners using norm-minimization techniques*, SIAM J. Sci. Comput., 19 (1998), pp. 605–625.

[32] M. J. Grote, *SPAI: SParse Approximate Inverse Preconditioner*, http://www.sam.math.ethz.ch/~grote/spai/.

[33] M. J. Grote and T. Huckle, *Parallel preconditioning with sparse approximate inverses*, SIAM J. Sci. Comput., 18 (1997), pp. 838–853.

[34] B. Hendrickson, *Graph partitioning and parallel solvers: Has the emperor no clothes?*, in Workshop on Parallel Algorithms for Irregularly Structured Problems, Lecture Notes in Comput. Sci. 1457, Springer, Berlin, 1998, pp. 218–225.

[35] B. Hendrickson and K. Devine, *Dynamic load balancing in computational mechanics*, Comput. Methods Appl. Mech. Engrg., 184 (2000), pp. 485–500.

[36] B. Hendrickson and T. G. Kolda, *Graph partitioning models for parallel computing*, Parallel Comput., 26 (2000), pp. 1519–1534.

[37] B. Hendrickson and T. G. Kolda, *Partitioning rectangular and structurally unsymmetric sparse matrices for parallel processing*, SIAM J. Sci. Comput., 21 (2000), pp. 2048–2072.

[38] C. Hoover, A. DeGroot, J. Maltby, and R. Procassini, *Paradyn: Dyna3d for Massively Parallel Computers*, Presentation at Tri-Laboratory Engineering Conference on Computational Modeling, Pleasanton, CA, 1995.

[39] T. Huckle, *Approximate Sparsity Patterns for the Inverse of a Matrix and Preconditioning*, Tech. Report 342/12/98 A, Technische Universität München, Germany, 1998.

[40] G. Karypis, *Multi-constraint Mesh Partitioning for Contact/Impact Computations*, Tech. Report 03-022, Department of Computer Science and Engineering/Army HPC Research Center, University of Minnesota, Minneapolis, MN, 2003.

[41] G. Karypis and V. Kumar, *Multilevel Algorithms for Multi-constraint Graph Partitioning*, Tech. Report 98-019, Department of Computer Science/Army HPC Research Center, University of Minnesota, Minneapolis, MN, 1998.

[42] G. Karypis and V. Kumar, *Multilevel Algorithms for Multi-constraint Hypergraph Partitioning*, Tech. Report 99-034, Department of Computer Science/Army HPC Research Center, University of Minnesota, Minneapolis, MN, 1998.

[43] T. G. Kolda, *Partitioning sparse rectangular matrices for parallel processing*, in Solving Irregularly Structured Problems in Parallel, Lecture Notes in Comput. Sci. 1457, Springer, Berlin, 1998, pp. 68–79.

[44] L. Yu. Kolotilina and A. Yu. Yeremin, *Factorized sparse approximate inverse preconditionings* I. *Theory*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 45–58.

[45] L. Yu. Kolotilina and A. Yu. Yeremin, *Factorized sparse approximate inverse preconditioning* II: *Solution of* 3*D FE systems on massively parallel computers*, Internat. J. High Speed Comput., 7 (1995), pp. 191–216.

[46] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout*, Wiley–Teubner, Chichester, UK, 1990.

[47] K. McManus, M. Cross, C. Walshaw, S. Johnson, and P. Leggett, *A scalable strategy for the parallelization of multiphysics unstructured mesh-iterative codes on distributed-memory systems*, Internat. J. High Perform. Comput. Appl., 14 (2000), pp. 137–174.

[48] G. Meurant, *Computer Solution of Large Linear Systems*, Elsevier, Amsterdam, The Netherlands, 1999.

[49] A. Pinar, Ü. V. Çatalyürek, C. Aykanat, and M. Pinar, *Decomposing linear programs for parallel solution*, in Proceedings of the Second International Workshop on Applied Parallel Computing in Physics, Chemistry and Engineering Science, Lecture Notes in Comput. Sci. 1041, Springer, Berlin, 1996, pp. 473–482.

[50] S. Plimpton, S. Attaway, B. Hendrickson, J. Swegle, C. Vaughan, and D. Gardner, *Transient dynamics simulations: Parallel algorithms for contact detection and smoothed particle hydrodynamics*, J. Parallel Distrib. Comput., 50 (1998), pp. 104–122.

[51] S. Plimpton, B. Hendrickson, and J. Stewart, *A parallel rendezvous algorithm for interpolation between multiple grids*, in Proceedings of the 1998 ACM/IEEE Conference on

Supercomputing, San Jose, CA, 1998, CD-ROM, pp. 1–8.
[52] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing, Boston, MA, 1996.
[53] K. Schloegel, G. Karypis, and V. Kumar, *A New Algorithm for Multi-objective Graph Partitioning*, Tech. Report 99-033, Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN, 1999.
[54] K. Schloegel, G. Karypis, and V. Kumar, *Graph partitioning for high-performance scientific simulations*, in CRPC Parallel Computing Handbook, J. Dongarra, I. Foster, G. Fox, W. Gropp, K. Kennedy, L. Torczon, and A. White, eds., Morgan Kaufmann, San Francisco, CA, 2002, Chap. 18, pp. 491–541.
[55] N. Selvakkumaran and G. Karypis, *Multi-objective hypergraph partitioning algorithms for cut and maximum subdomain degree minimization*, in Proceedings of the 2003 IEEE/ACM International Conference on Computer-Aided Design (ICCAD '03), San Jose, CA, 2003, pp. 726–733.
[56] T. Sterling, D. Savarese, D. J. Becker, J. E. Dorband, U. A. Ranaweke, and C. V. Packer, *BEOWULF: A parallel workstation for scientific computation*, in Proceedings of the 24th International Conference on Parallel Processing, Oconomowoc, WI, 1995, pp. 11–34.
[57] R. S. Tuminaro, J. N. Shadid, and S. A. Hutchinson, *Parallel sparse matrix vector multiply software for matrices with data locality*, Concurrency: Practice and Experience, 10 (1998), pp. 229–247.
[58] B. Uçar and C. Aykanat, *Minimizing communication cost in fine-grain partitioning of sparse matrices*, in Proceedings of the 18th International Symposium on Computer and Information Sciences (ISCIS XVIII), A. Yazıcı and C. Şener, eds., Antalya, Turkey, 2003.
[59] B. Uçar and C. Aykanat, *Encapsulating multiple communication-cost metrics in partitioning sparse rectangular matrices for parallel matrix-vector multiplies*, SIAM J. Sci. Comput., 25 (2004), pp. 1837–1859.
[60] B. Uçar and C. Aykanat, *Experiments on Hypergraph Models for Parallelizing Preconditioned Iterative Methods*, Tech. Report BU-CE-0413, Department of Computer Engineering, Bilkent University, Ankara, Turkey, 2004.
[61] B. Uçar and C. Aykanat, *A Library for Parallel Sparse Matrix-Vector Multiplies*, Tech. Report BU-CE-0506, Department of Computer Engineering, Bilkent University, Ankara, Turkey, 2005.
[62] H. A. van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 631–644.
[63] B. Vastenhouw and R. H. Bisseling, *A two-dimensional data distribution method for parallel sparse matrix-vector multiplication*, SIAM Rev., 47 (2005), pp. 67–95.
[64] C. Walshaw, M. Cross, and K. McManus, *Multiphase mesh partitioning*, Appl. Math. Modeling, 25 (2000), pp. 123–140.

SIAM J. SCI. COMPUT.
Vol. 29, No. 4, pp. 1710–1737
© 2007 Society for Industrial and Applied Mathematics

FAST DIRECTIONAL MULTILEVEL ALGORITHMS FOR OSCILLATORY KERNELS*

BJÖRN ENGQUIST† AND LEXING YING†

Abstract. This paper introduces a new directional multilevel algorithm for solving N-body or N-point problems with highly oscillatory kernels. These systems often result from the boundary integral formulations of scattering problems and are difficult due to the oscillatory nature of the kernel and the non-uniformity of the particle distribution. We address the problem by first proving that the interaction between a ball of radius r and a well-separated region has an approximate low rank representation, as long as the well-separated region belongs to a cone with a spanning angle of $O(1/r)$ and is at a distance which is at least $O(r^2)$ away from from the ball. We then propose an efficient and accurate procedure which utilizes random sampling to generate such a separated, low rank representation. Based on the resulting representations, our new algorithm organizes the high frequency far field computation by a multidirectional and multiscale strategy to achieve maximum efficiency. The algorithm performs well on a large group of highly oscillatory kernels. Our algorithm is proved to have $O(N \log N)$ computational complexity for any given accuracy when the points are sampled from a two dimensional surface. We also provide numerical results to demonstrate these properties.

Key words. N-body problems, scattering problems, Helmholtz equation, oscillatory kernels, fast multipole methods, separated representations, random sampling, operator compression, multidirectional computation, multiscale methods

AMS subject classifications. 65C99, 65R99, 78A45

DOI. 10.1137/07068583X

1. Introduction.

This paper is concerned with the rapid solutions to a class of N-body problems. Let $\{f_i, 1 \le i \le N\}$ be a set of N densities located at points $\{p_i, 1 \le i \le N\}$ in $\mathbb{R}^3$ with $|p_i| \le K/2$, where $|\cdot|$ is the Euclidean norm and K is a fixed constant K. Our goal is to compute the potentials $\{u_i, 1 \le i \le N\}$ defined by

$$u_i = \sum_{j=1}^{N} G(p_i, p_j) \cdot f_j \tag{1}$$

where $G(x, y) = e^{2\pi\imath|x-y|}/|x-y|$ is the Green's function of the Helmholtz equation and is usually called the Helmholtz kernel. Throughout this paper, we use $\imath$ to denote the complex number $\sqrt{-1}$. We have scaled the problem such that the wave length equals one and thus high frequencies correspond to problems with large computational domains.

Such a computation comes mainly from applications in acoustic and electromagnetic scattering, where the usual partial differential equation (PDE) formulations are transformed into boundary integral equation (BIE) formulations. The advantages of the BIE formulations are the convenience of handling the boundary conditions at infinity and the decrease in the dimensionality of the problem. The integral equation formulations are often discretized using appropriate quadrature methods. The resulting linear systems are always dense due to the fact that an integral formulation

*Received by the editors March 20, 2007; accepted for publication (in revised form) April 30, 2007; published electronically August 10, 2007.
http://www.siam.org/journals/sisc/29-4/68583.html
†Department of Mathematics, University of Texas at Austin, 1 University Station C1200, Austin, TX 78712 (engquist@math.utexas.edu, lexing@math.utexas.edu).

involves interaction between any two points or elements on the boundary. These dense linear systems are further solved using iterative methods, at each step of which the evaluation in (1) or closely related equations must be carried out.

Two observations are here in order. First, in scattering applications, the complexity of a problem scales with the size of its boundary in terms of the wavelength. For a prescribed accuracy, the complexity of (1) depends on the value of K because the wavelength is equal to one in our setup. In many practical applications, K is usually equal to a few hundred or even thousand. Assuming that the boundary surface in $\mathbb{R}^3$ is discretized with a fixed number of points per wavelength, the number of samples N is of order $O(K^2)$. Second, since the unknown field is supported only on the boundary in a BIE formulation, the points $\{p_i\}$ are samples of a two dimensional manifold. Consequently, the distribution of $\{p_i\}$ is highly nonuniform.

We would like to point out that, though the discussion in this paper mainly focuses on the Helmholtz kernel, our approach is quite general. Since many PDE kernels (e.g. the Green's functions of the time-harmonic Maxwell equations [21] and the linear elasticity equation [36]) are basically derivatives of the Helmholtz kernel, our algorithm can be readily extended to address these cases.

1.1. Previous work. Direct computation of (1) requires $O(N^2)$ operations, which can be intolerably slow for large values of N. During the last few decades, much attention has been devoted to the development of algorithms which evaluate (1) efficiently without compromising accuracy. An early class of algorithms (for example, [7, 8]) use the fast Fourier transform (FFT) by exploiting the fact that the kernel is translation invariant, and hence can be diagonalized through Fourier transform. Though quite efficient for uniformly distributed point sets, they require $O(N^{3/2} \log N)$ complexity in both computation time and storage space for a point set sampled from a two dimensional manifold. This poses a difficulty for large scattering problems. A recent improvement in [12] addresses this issue by using plane densities to reduce the sizes of the required FFTs.

The second class of algorithms (see for example [1, 3, 6, 10, 16]) discretize the boundary integral equations using the Galerkin approximation with either local Fourier bases or wavelet bases as the approximation space. The resulting discrete systems often become approximately sparse in the sense that a majority of the entries are close to zero with these bases. Such approaches have been shown to offer good theoretical estimates. However, constructing the remaining non-negligible entries both efficiently and accurately is non-trivial.

The third, and probably most popular, class of algorithms are the fast multipole methods (FMMs). The original FMM (see for example [17, 32, 34]) was developed for the kernel of the Laplace equation, and it evaluates the N-body problem in $O(N)$ operations for any fixed accuracy. Its success hinges on the observation that the interaction via the kernel between well-separated sets of points is approximately of low rank. Many other related developments, such as the panel clustering method [35], the H-matrix framework [9] and the interpolation techniques [11], are based on the same observation. In the low frequency regime where K is close to 1, this low rank property still holds for the Helmholtz kernel and the FMM for the Laplace equation can be applied to our problem with slight modifications (see [33]). In the high frequency regime where K is much larger than 1, the situation is however drastically different as the low rank assumption is not valid any more. In fact, the approximate rank of the interaction grows linearly with the size of the point sets (in terms of the wavelength). However, Rokhlin [42, 43] observed that the interaction between well-

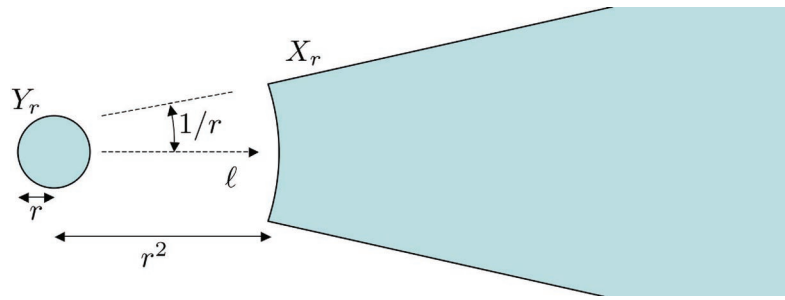


FIG. 1. *Two sets Y_r and X_r which satisfy the directional parabolic separation condition.*

separated point sets, though of large rank, can be applied efficiently by focusing on the far field signature of the potentials. In some sense, this is analogous to the fast Fourier transform. Even if the matrix is dense and of full rank, it can be applied efficiently in almost linear time. The resulting algorithm in the high frequency regime, often named HF-FMM, has $O(N \log N)$ complexity for a given accuracy. Though efficient and accurate, this algorithm is highly technical. It utilizes a large family of tools, such as partial wave expansion, far field signature, exponential expansion, filtering, and interpolation of spherical harmonics (see [37]). Many of these techniques have been developed only recently, and some of them solely for the purpose of the HF-FMM. Good descriptions of the HF-FMM include [18, 27, 41]. Other algorithms using similar techniques are given in [20, 23, 24, 45].

In [40], Michielssen and Boag proposed an interesting development called multilevel matrix decomposition. Their algorithm utilizes the idea of equivalent density, which also serves as the basis of the approaches in [2, 47], and it plays an important role in this paper. Even if their algorithm may not be as efficient as the HF-FMM, its three stage multiplication algorithm is ingenious and reminds one of the FFT.

1.2. Contribution and significance. In this paper, we propose a new directional multilevel algorithm that has $O(N \log N)$ complexity for a given accuracy. The starting point of our approach is a geometric consideration. Suppose Y_r is a ball of radius r centered at a point c, and X_r is the set of all points which are at a distance r^2 or greater from the origin and belong to a cone centered at c with spanning angle $1/r$ (see Figure 1 for an illustration). Whenever two sets Y_r and X_r obey this geometric configuration, we say that Y_r and X_r satisfy the *directional parabolic separation condition*.

At the heart of our algorithm is a *directional low rank property* which states that the interaction between Y_r and X_r via the Helmholtz kernel $G(x, y)$ is approximately of low rank for any fixed accuracy, and more importantly, the rank has an upper bound that is independent of r. To be more precise, the directional low rank property guarantees the following *directional separated representation* which is valid for any $x \in X_r$ and $y \in Y_r$:

$$\left| G(x,y) - \sum_{i=1}^{T(\varepsilon)} \alpha_i(x)\beta_i(y) \right| < \varepsilon$$

where ε is the prescribed accuracy, $T(\varepsilon)$ is a constant which depends only on ε (not on r), and $\{\alpha_i(x)\}$ and $\{\beta_i(y)\}$ are functions of x and y respectively.

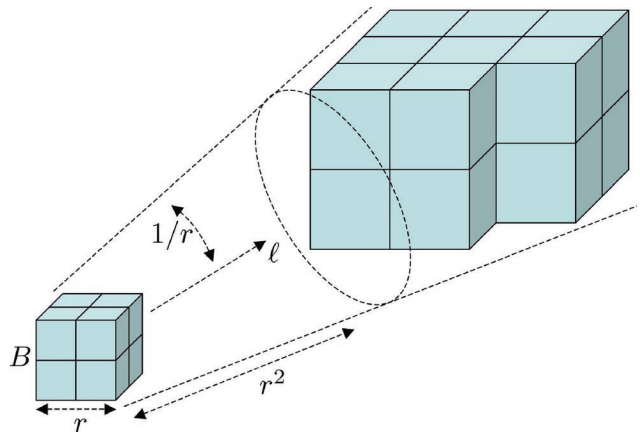


FIG. 2. *For each box B, its far field is partitioned into multiple wedges. Our algorithm constructs one low rank separated representation for each wedge. Each resulting representation can be used to accelerate the interaction computation between B and all the boxes in that wedge.*

The second contribution of this paper is to introduce a randomized procedure for generating the directional separated representation $\{\alpha_i(x)\}$ and $\{\beta_i(x)\}$ for the interaction between X_r and Y_r. This procedure only uses kernel evaluations and stable numerical routines such as the singular value decomposition (SVD) and the QR factorization. The resulting representation allows highly efficient computation of the interaction between sets like X_r and Y_r, and moreover, it can be stored in an economic way.

Our algorithm starts by partitioning the domain recursively using an octree, which is similar to the standard FMM. The top part of the octree that contains the boxes with widths greater than or equal to 1 is called the high frequency regime, while the bottom part that contains the boxes with widths less than 1 is called the low frequency regime. In the low frequency regime, the interactions are computed using the kernel independent FMM [47], which is a variant of the standard FMM. In the high frequency region, the computation is organized in a multidirectional way. For a given box B, all the boxes well separated from it are partitioned into a group of wedges, where each wedge and the box itself follow the directional parabolic separation condition (see Figure 2). The randomized procedure mentioned above is then used repetitively to construct multiple low rank representations about B, one for each wedge. As a result, the calculation of the interactions between B and all the boxes in a specific wedge can be accelerated using the low rank representation associated with this wedge. This framework is repeated recursively at all levels in the high frequency regime to achieve maximum efficiency.

Our directional multilevel algorithm has the following properties. First, it is highly efficient and accurate. A careful complexity analysis shows that, for a finite point set on a two dimensional manifold, our algorithm uses $O(N \log N)$ operations, which is the same as the complexity of the HF-FMM. Second, our algorithm is less technical than the HF-FMM as it uses no analytic expansions and translations. The computational steps of our algorithm only involve kernel evaluation and stable linear algebra routines. Finally, our algorithm works for more general oscillatory kernels as well (see the numerical results in section 5).

This paper is organized as follows. In section 2, we prove the directional low rank property. Following that, we describe the randomized procedure for generating

a directional low rank separated representations in section 3. Our main algorithm is presented in detail in section 4 where also its computational complexity is analyzed. After reporting the numerical results in section 5, we give conclusions and suggestions for future work in section 6.

2. Directional low rank property. In this section, we prove the main theoretical result of this paper: the directional low rank property. Suppose $r \geq \sqrt{3}$ and let

$$Y_r = B(0,r) \quad \text{and} \quad X_r = \{x : \theta(x,\ell) \leq 1/r, |x| \geq r^2\} \tag{2}$$

where ℓ is a given unit vector and $\theta(a,b)$ is the spanning angle between vectors a and b. The geometric relationship between Y_r and X_r is illustrated in Figure 1.

DEFINITION 2.1. *Let $f(x,y)$ be a function for $x \in X$ and $y \in Y$. We say $f(x,y)$ has a T-term ε-expansion for X and Y if there exist functions $\{\alpha_i(x), 1 \leq i \leq T\}$ and $\{\beta_i(y), 1 \leq i \leq T\}$ such that*

$$\left| f(x,y) - \sum_{i=1}^{T} \alpha_i(x)\beta_i(y) \right| \leq \varepsilon$$

for all $x \in X$ and $y \in Y$.

The importance of this definition is that $\{\alpha_i(x)\}$ and $\{\beta_i(y)\}$ depend only on x and y respectively. Expansions of this type are called *separated*. The following theorem is a precise statement of the directional low rank property.

THEOREM 2.2. *For any $\varepsilon > 0$, there exists a number $T(\varepsilon)$ which is independent of r such that $e^{2\pi\imath|x-y|}/|x-y|$ has a $T(\varepsilon)$-term ε-expansion for any X_r and Y_r satisfying* (2).

Results similar to Theorem 2.2 have been discovered before by researchers from applied physics and electric engineering [13, 29, 40]. Their proofs involve special functions such as spherical harmonics and Bessel functions, and give much sharper estimates on the number of terms in the expansion. We believe that our proof in the rest of this section is still of theoretical interest since one can easily adapt it to show similar results for other oscillatory kernels such as $e^{2\pi\imath|x-y|}$. The numerical results in section 5 validate this assertion.

Our proof of Theorem 2.2 is based on the following lemmas.

LEMMA 2.3. *For any $\varepsilon > 0$, there exists a number $T(\varepsilon)$ which is independent of r such that $|x-y|$ has a $T(\varepsilon)$-term ε-expansion for X_r and Y_r.*

Proof. Let us write

$$|x-y| = |x| \cdot f(y/|x|)$$

where $f(p) = |\hat{x}-p|$ and $\hat{x}$ is the unit direction $x/|x|$. Here we suppress the dependence of f on $\hat{x}$. Taylor expansion of f at $p=0$ gives

$$f(p) = \sum_{d=0}^{\infty} \left[\sum_{|\alpha|=d} \frac{f^{(\alpha)}(0)}{\alpha!} p^{\alpha} \right]$$

where α is a multiindex. For any $d \geq 0$, we use $f_d(p)$ to denote the quantity in the square bracket. The series $\sum_{d=0}^{\infty} f_d(p)$ is a convergent series with radius 1. This implies that for any η with $\eta \geq 1$ there exists an integer $D_1(\eta)$ such that $f_d(p) \leq (\eta|p|)^d$ for every $d \geq D_1(\eta)$. We assume, without loss of generality, that $\eta \leq 1.1$.

We argue that there exists a number $D(\varepsilon)$ such that

$$|x| \cdot f_{D(\varepsilon)}(y/|x|) \le \varepsilon.$$

To show this, it is sufficient to find an integer d such that $|x| \cdot (\eta \cdot |y|/|x|)^d \le \varepsilon$. Notice that for $d \ge 2$, $|x| \cdot (\eta \cdot |y|/|x|)^d$ is a decreasing function of $|x|$. Therefore, the function reaches its maximum value when $|x| = r^2$. Since $|y| \le r$,

$$|x| \cdot \left(\eta \cdot \frac{|y|}{|x|}\right)^d \le r^2 \cdot \left(\frac{\eta}{r}\right)^d = \frac{\eta^d}{r^{d-2}} \le \frac{\eta^d}{\sqrt{3}^{d-2}}.$$

Since we have assumed that $\eta \le 1.1$, it is obvious that there exists a $D^2(\eta, \varepsilon)$ such that for any $d \ge D^2(\eta, \varepsilon)$ we have $\eta^d/\sqrt{3}^{d-2} \le \varepsilon$. Therefore, setting $D(\varepsilon) = \max(2, D^1(\eta), D^2(\eta, \varepsilon))$ guarantees that

$$|x| \cdot \left(\eta \cdot \frac{|y|}{|x|}\right)^{D(\varepsilon)} \le \varepsilon.$$

Since the series $\{|x| \cdot (\eta \cdot |y|/|x|)^d : d \ge 0\}$ is geometric and $f_d(y/|x|) \le (\eta \cdot |y|/|x|)^d$, we have the following estimate

$$\sum_{d > D(\varepsilon)} |x| \cdot f_d(y/|x|) \le 4\varepsilon,$$

which is equivalent to

$$\left| |x - y| - \sum_{d=0}^{D(\varepsilon)} |x| \cdot f_d(y/|x|) \right| \le 4\varepsilon. \tag{3}$$

Notice that the number of terms $D(\varepsilon)$ is independent of $|x|$. In fact, it is independent of $\hat{x}$ as well since the kernel is rotation invariant.

Now, each term $f_d(y/|x|)$ can be written as

$$f_d(y/|x|) = \sum_{|\alpha|=d} \frac{f^{(\alpha)}(0)}{\alpha!} \cdot \frac{1}{|x|^{|d|}} \cdot y^\alpha \tag{4}$$

where $f^{(\alpha)}(0)$ depends on $\hat{x}$ implicitly. Since (4) is a separated representation,

$$\sum_{d > D(\varepsilon)} |x| \cdot f_d(y/|x|) \tag{5}$$

has a separated representation as well. Now suppose $T(\varepsilon)$ to be the total number of terms after the expansion of (5). Then (3) states that $|x - y|$ has a $T(\varepsilon)$-term 4ε-expansion. ∎

Similarly, we have the following lemma for the function $1/|x - y|$. In fact, the proof is simpler than the one of Lemma 2.3 because $1/|x - y|$ is bounded for $x \in X_r$ and $y \in Y_r$ for $r \ge \sqrt{3}$.

LEMMA 2.4. *For any $\varepsilon > 0$, there exists a number $T(\varepsilon)$ which is independent of r such that $1/|x - y|$ has a $T(\varepsilon)$-term ε-expansion for X_r and Y_r.*

COROLLARY 2.5. *For any $\varepsilon > 0$, there exists a number $T(\varepsilon)$ which is independent of r such that $|x-y| - \hat{x}\cdot(x-y)$ has a $T(\varepsilon)$-term ε-expansion for X_r and Y_r.*

Proof. Since $\hat{x}\cdot(x-y)$ has a 2-term 0-expansion

$$\hat{x}\cdot(x-y) = |x| - \hat{x}\cdot y,$$

the corollary follows from Lemma 2.3. ∎

The following lemmas show that certain functions are bounded by constants.

LEMMA 2.6. *There exists a constant G such that*

$$|(\hat{x}-\ell)\cdot y| \le G$$

and

$$||x-y| - \hat{x}\cdot(x-y)| \le G$$

for $x \in X_r$ and $y \in Y_r$.

Proof. For the first part, we observe

$$|(\hat{x}-\ell)\cdot y| \le |(\hat{x}-\ell)|\cdot|y| \le \theta(x,\ell)\cdot|y|.$$

From the definitions of X_r and Y_r, we have $|y| \le r$ and $\theta(x,\ell) \le 1/r$. Therefore,

$$|(\hat{x}-\ell)\cdot y| \le r\cdot\frac{1}{r} = 1.$$

For the second part, suppose that θ is the spanning angle between x and $x-y$. From the definitions of X_r and Y_r we have $\theta \le c_1\cdot r/|x-y|$, $|x-y| \ge c_2\cdot r^2$, $1-\cos(\theta) \le c_3\cdot\theta^2$ for constants c_1, c_2 and c_3. Therefore,

$$|x-y| - \hat{x}\cdot(x-y) = |x-y|\cdot(1-\cos(\theta)) \le c_3\cdot|x-y|\cdot\theta^2 \le c_3c_1^2\cdot\frac{r^2}{|x-y|} \le \frac{c_3c_1^2}{c_2}.$$

Setting $G = \max(1, c_3c_1^2/c_2)$ completes the proof. ∎

The next lemma, which is quite elementary, is concerned with the accuracy of the Taylor expansion of the exponential function. We reproduce the proof for completeness (see [14]).

LEMMA 2.7. *For any $Z > 0$ and $\varepsilon > 0$, let $N = \lceil \max(4e\pi Z, \log_2(1/\varepsilon))\rceil$. Then*

$$\left| e^{2\pi\imath x} - \sum_{k=0}^{N-1} \frac{(2\pi\imath x)^k}{k!} \right| \le \varepsilon$$

for any x with $|x| \le Z$.

Proof. Truncated Taylor expansion of $e^{2\pi\imath x}$ at the origin gives

$$e^{2\pi\imath x} = \sum_{k=0}^{N-1} \frac{(2\pi\imath x)^k}{k!} + \frac{(2\pi\imath\tilde{x})^N}{N!}$$

where $\tilde{x}$ lies between 0 and x. Since $|\tilde{x}| \le |x| \le Z$,

$$\left| e^{2\pi\imath x} - \sum_{k=0}^{N-1} \frac{(2\pi\imath x)^k}{k!} \right| \le \frac{(2\pi Z)^N}{N!}.$$

We use the Stirling formula to estimate the last quantity. Since $N! \geq (N/e)^N$,

$$\frac{(2\pi Z)^N}{N!} \leq \left(\frac{2\pi e Z}{N}\right)^N.$$

Now, as $N \geq \max(4e\pi Z, \log_2(1/\varepsilon))$,

$$\left(\frac{2\pi e Z}{N}\right)^N \leq \left(\frac{1}{2}\right)^{\log_2(1/\varepsilon)} = \varepsilon. \qquad \square$$

LEMMA 2.8. *Let $\varepsilon > 0$ and $\eta > 0$. Suppose that a function $f(x,y)$ has a $T^f(\varepsilon)$-term ε-expansion on X_r and Y_r, and $|f(x,y)| \leq B^f$. Suppose also that a function $g(x,y)$ has a $T^g(\eta)$-term η-expansion on X_r and Y_r, and $|g(x,y)| \leq B^g$. Then, $f(x,y) \cdot g(x,y)$ has a $T^f(\varepsilon) \cdot T^g(\eta)$-term $(B^g\varepsilon + B^f\eta + \varepsilon\eta)$-expansion.*

Proof. By assumption, there exist functions $\{\alpha_i^f(x)\}$ and $\{\beta_i^f(x)\}$ for $f(x,y)$ such that

$$\left| f(x,y) - \sum_{i=1}^{T^f(\varepsilon)} \alpha_i^f(x)\beta_i^f(y) \right| \leq \varepsilon.$$

Similarly, there exist functions $\{\alpha_i^g(x)\}$ and $\{\beta_i^g(x)\}$ for $g(x,y)$ such that

$$\left| g(x,y) - \sum_{i=1}^{T^g(\eta)} \alpha_i^g(x)\beta_i^g(y) \right| \leq \eta.$$

Consider two sets of functions $\{\alpha_i^f(x)\alpha_j^g(x) : 1 \leq i \leq T^f(\varepsilon), 1 \leq j \leq T^g(\eta)\}$ and $\{\beta_i^f(y)\beta_j^g(y) : 1 \leq i \leq T^f(\varepsilon), 1 \leq j \leq T^g(\eta)\}$, each containing $T^f(\varepsilon) \cdot T^g(\eta)$ terms.

$$\begin{aligned}
&\left| f(x,y)g(x,y) - \sum_{i,j} (\alpha_i^f(x)\alpha_j^g(x))(\beta_i^f(y)\beta_j^g(y)) \right| \\
&= \left| f(x,y)g(x,y) - \left(\sum_i \alpha_i^f(x)\beta_i^f(y)\right)\left(\sum_j \alpha_j^g(x)\beta_j^g(y)\right) \right| \\
&\leq \left| \left(f(x,y) - \left(\sum_i \alpha_i^f(x)\beta_i^f(y)\right)\right) g(x,y) \right| + \\
&\quad \left| \left(\sum_i \alpha_i^f(x)\beta_i^f(y)\right)\left(g(x,y) - \left(\sum_j \alpha_j^g(x)\beta_j^g(y)\right)\right) \right| \\
&\leq \varepsilon \cdot B^g + (B^f + \varepsilon) \cdot \eta \\
&= (B^g\varepsilon + B^f\eta + \varepsilon\eta). \qquad \square
\end{aligned}$$

We are now ready to prove Theorem 2.2.

Proof of Theorem 2.2. Without loss of generality, we make the assumption that $\varepsilon < 1$.

First, let us construct an expansion for $e^{2\pi\imath(|x-y| - \hat{x}\cdot(x-y))}$. From Corollary 2.5, $|x-y| - \hat{x} \cdot (x-y)$ has a $T^1(\varepsilon)$-term ε-expansion, i.e., there exist functions $\{\alpha_i^1(x)\}$

and $\{\beta_i^1(y)\}$ such that

$$\left|(|x-y| - \hat{x}\cdot(x-y)) - \sum_i \alpha_i^1(x)\beta_i^1(y)\right| \le C^1\varepsilon$$

Lemma 2.6 says that there exists a constant G such that

$$||x-y| - \hat{x}\cdot(x-y)| \le G,$$

which implies

$$\left|\sum_i \alpha_i^1(x)\beta_i^1(y)\right| \le G+\varepsilon.$$

Applying Lemma 2.7 with $Z = G+\varepsilon$ and $N^1(\varepsilon) = \lceil \max(4e\pi(G+\varepsilon), \log_2(1/\varepsilon))\rceil$ gives

$$\left|e^{2\pi\imath\sum_i \alpha_i^1(x)\beta_i^1(y)} - \sum_{k=0}^{N^1(\varepsilon)-1} \frac{(2\pi\imath)^k}{k!}\left(\sum_i \alpha_i^1(x)\beta_i^1(y)\right)^k\right| \le \varepsilon.$$

Expanding the $\left(\sum_i \alpha_i^1(x)\beta_i^1(y)\right)^k$ terms and absorbing the coefficients $\frac{(2\pi\imath)^k}{k!}$, we find that there exists a number $T^2(\varepsilon)$ and two sets of functions $\{\alpha_i^2(x)\}$ and $\{\beta_i^2(y)\}$ such that

$$\left|e^{2\pi\imath\sum_i \alpha_i^1(x)\beta_i^1(y)} - \sum_{i=0}^{T^2(\varepsilon)} \alpha_i^2(x)\beta_i^2(y)\right| \le \varepsilon,$$

or equivalently, that $e^{2\pi\imath\sum_i \alpha_i^1(x)\beta_i^1(y)}$ has a $T^2(\varepsilon)$-term ε-expansion. Since

$$\begin{aligned}\left|e^{2\pi\imath(|x-y|-\hat{x}\cdot(x-y))} - e^{2\pi\imath\sum_i \alpha_i^1(x)\beta_i^1(y)}\right| &\le 2\pi\left|(|x-y| - \hat{x}\cdot(x-y)) - \sum_i \alpha_i^1(x)\beta_i^1(y)\right| \\ &\le 2\pi\varepsilon,\end{aligned}$$

it is obvious that $e^{2\pi\imath(|x-y|-\hat{x}\cdot(x-y))}$ has a $T^2(\varepsilon)$-term $(2\pi+1)\varepsilon$-expansion

$$\left|e^{2\pi\imath(|x-y|-\hat{x}\cdot(x-y))} - \sum_{i=0}^{T^2(\varepsilon)} \alpha_i^2(x)\beta_i^2(y)\right| \le (2\pi+1)\varepsilon. \tag{6}$$

Second, we consider the term $e^{2\pi\imath\hat{x}\cdot(x-y)}$. From Lemma 2.6, we know that there exists a constant G such that $|(\hat{x}-\ell)\cdot y| \le G$. Now applying Lemma 2.7 again with $Z = G$ and $N^3(\varepsilon) = \lceil \max(4e\pi G, \log_2(1/\varepsilon))\rceil$ gives

$$\left|e^{2\pi\imath(\hat{x}-\ell)\cdot y} - \sum_{k=0}^{N^3(\varepsilon)-1} \frac{(2\pi\imath)^k}{k!}((\hat{x}-\ell)\cdot y)^k\right| \le \varepsilon.$$

Expanding the $((\hat{x}-\ell)\cdot y)^k$ terms for all k and absorbing the coefficients $\frac{(2\pi\imath)^k}{k!}$, we have the expansion

$$\left|e^{2\pi\imath(\hat{x}-\ell)\cdot y} - \sum_{i=0}^{T^3(\varepsilon)} \alpha_i^3(x)\beta_i^3(y)\right| \le \varepsilon,$$

where $T^3(\varepsilon)$ is an ε-dependent integer and $\{\alpha_i^3(x)\}$ and $\{\beta_i^3(y)\}$ are two sets of functions. That is to say, $e^{2\pi\imath(\hat{x}-\ell)\cdot y}$ has a $T^3(\varepsilon)$-term ε-expansion. Now exponentiating the identity

$$\hat{x}\cdot(x-y) = |x| - (\hat{x}-\ell)\cdot y - \ell\cdot y$$

gives

$$e^{2\pi\imath\hat{x}\cdot(x-y)} = e^{2\pi\imath|x|}\cdot e^{-2\pi\imath(\hat{x}-\ell)\cdot y}\cdot e^{-2\pi\imath\ell\cdot y}.$$

Then it is obvious that $e^{2\pi\imath\hat{x}\cdot(x-y)}$ has a $T^4(\varepsilon)$-term ε-expansion

$$\left| e^{2\pi\imath\hat{x}\cdot(x-y)} - \sum_{i=0}^{T^4(\varepsilon)} \alpha_i^4(x)\beta_i^4(y) \right| \le \varepsilon, \tag{7}$$

with $T^4(\varepsilon) = T^3(\varepsilon)$ and the functions $\{\alpha_i^4(x)\}$ and $\{\beta_i^4(y)\}$ given by

$$\alpha_i^4(x) = e^{2\pi\imath|x|}\cdot\overline{\alpha_i^3(x)} \quad \text{and} \quad \beta_i^4(y) = e^{-2\pi\imath\ell\cdot y}\cdot\overline{\beta_i^3(y)}.$$

Next, Lemma 2.4 claims that $1/|x-y|$ has $T^5(\varepsilon)$-term ε expansion

$$\left| \frac{1}{|x-y|} - \sum_{i=0}^{T^5(\varepsilon)} \alpha_i^5(x)\beta_i^5(y) \right| \le \varepsilon \tag{8}$$

with functions $\{\alpha_i^5(x)\}$ and $\{\beta_i^5(y)\}$.

Finally, since

$$\frac{e^{2\pi\imath|x-y|}}{|x-y|} = e^{2\pi\imath(|x-y|-\hat{x}\cdot(x-y))}\cdot e^{2\pi\imath\hat{x}\cdot(x-y)}\cdot\frac{1}{|x-y|},$$

applying Lemma 2.8 twice to the product of (6), (7) and (8) shows that $\frac{e^{2\pi\imath|x-y|}}{|x-y|}$ has a $(T^2(\varepsilon)\cdot T^4(\varepsilon)\cdot T^5(\varepsilon))$-term $(12\pi+10)\varepsilon$-expansion. Here we use the facts that

$$\left|e^{2\pi\imath(|x-y|-\hat{x}\cdot(x-y))}\right| < 1, \quad \left|e^{2\pi\imath\hat{x}\cdot(x-y)}\right| < 1 \quad \text{and} \quad \left|\frac{1}{|x-y|}\right| < 2$$

for $x \in X_r$ and $y \in Y_r$ along with the assumption that $\varepsilon < 1$. □

The message of Theorem 2.2 is that, for a fixed ε, the number of terms in an ε-expansion is independent of r, as long as X_r and Y_r satisfy the directional parabolic separation condition, i.e., X_r belongs to a cone with spanning angle $1/r$ and it is an order $O(r^2)$ distance away from Y_r.

The proof given above is not optimal in the sense that $(T^2(\varepsilon)\cdot T^4(\varepsilon)\cdot T^5(\varepsilon))$, the bound of the number of terms in the expansion, can be fairly large. In practice, numerical results show that the actual number of terms grows linearly with $\log(1/\varepsilon)$, see section 5.1.

So far in Theorem 2.2, we have assumed that Y_r is centered at the origin. However, as we have noticed in the proof, what is really important is the relative positions of the sets X_r and Y_r. Since the kernel $e^{2\pi\imath|x-y|}/|x-y|$ is translation invariant, shifting X_r and Y_r together by a constant vector would not change the result of Theorem 2.2. We have also assumed ℓ to be a fixed direction and suppressed the dependence of

X_r on ℓ in our notations. However, all the estimates are independent of the specific choice of ℓ as the kernel $e^{2\pi\imath|x-y|}/|x-y|$ is rotation invariant. Suppose we define $F_r = \{x : |x| \geq r^2\}$. There exists a set of unit vectors $\{\ell\}$ of size $O(r^2)$ such that the union of the cones centered at $\{\ell \in L\}$ with spanning angle $1/r$ cover the whole space. Now let us define

$$X_r^\ell = \{x : \theta(x,\ell) \leq 1/r, |x| \geq r^2\}$$

for each ℓ. Clearly their union is equal to F_r. Applying Theorem 2.2 to each X_r^ℓ results in a different $T(\varepsilon)$-term ε-expansion for Y_r and X_r^ℓ. Even though the expansions are different from one X_r^ℓ to another, the bound $T(\varepsilon)$ on the number of terms is independent of ℓ.

3. Randomized construction of separated representation. The expansion for two sets X_r^ℓ and Y_r introduced in section 2 is a directional separated representation. It is directional since for a given direction ℓ the expansion is valid only for X_r^ℓ that belongs to a directional cone. It is separated since the two sets of functions $\{\alpha_i(x)\}$ and $\{\beta_i(y)\}$ depend only on x and y respectively. The proof of Theorem 2.2 is constructive in the sense that it provides a way to compute the functions $\{\alpha_i(x)\}$ and $\{\beta_i(y)\}$. However, the number of terms in the resulting expansion can be fairly large. In this section, we introduce a procedure which in practice gives expansions with very few terms. This procedure is accurate, efficient and based on random sampling.

In the rest of this section, r and ℓ are not explicitly included in the notation. Our randomized procedure consists of the following steps:

1. Sample Y randomly and densely to obtain a set of samples $\{y_i\}$. By densely, we mean a couple of samples per wavelength. Similarly, sample X to obtain a set of samples $\{x_i\}$. Suppose the numbers of samples in each set are respectively N_y and N_x. Since our problem is confined to a ball of radius $K/2$ where K is the number of wavelengths, there is no reason to sample the part of X that is outside of this ball. As we allow a constant number of samples per wavelength, $N_y = O(vol(Y)) = O(K^{1.5})$. Similarly, $N_x = O(vol(X \cap B(0,K))) = O(K^2)$. Let A be the N_x by N_y matrix defined by

$$A_{ij} = e^{2\pi\imath|x_i-y_j|}/|x_i-y_j|$$

for $1 \leq i \leq N_x$ and $1 \leq j \leq N_y$. In the language of linear algebra, Theorem 2.2 states that A can be factorized, within error $O(\varepsilon)$, into the product of two matrices, the first of size N_x by $T(\varepsilon)$ and the second of size $T(\varepsilon)$ by N_y. In the next few steps, we construct such a factorization.

2. Let A_1 be the submatrix of A containing a set of N_1 randomly selected columns. Here N_1 is set to be a constant multiple of $T(\varepsilon)$. After obtaining the pivoted QR factorization of A_1, we have the decomposition

$$A_1 P_1 = Q_1 R_1$$

where P_1 is a permutation matrix, Q_1 is orthonormal and R_1 is upper triangular. Now, identify the diagonal elements of R_1 which are less than ε and truncate the associated columns of Q_1 and rows of R_1. Denote the resulting matrices by Q_c and R_c. Since A_1, as a submatrix of A, has an approximate factorization of $T(\varepsilon)$, in practice the matrix Q_c has only $O(T(\varepsilon))$ columns after truncation. It is clear from the algorithm of the pivoted QR factorization

that the new matrices Q_c and R_c satisfy the relationship

$$Q_c R_c = A_c$$

where A_c is the submatrix containing the columns of A from which the matrix Q_c is generated. The samples (of Y) associated with these columns are denoted $\{y_i^c\}$. In practice, the columns of Q_c approximately span the column space of A (not just A_1), producing an $O(\varepsilon)$ error with overwhelming probability.

3. Let A_2 be a submatrix of A containing a set of N_2 randomly selected rows. Here N_2 is again set to be a constant multiple of $T(\varepsilon)$. Repeat the previous step on A_2^*. As a result, we have two matrices Q_r and R_r. Q_r is orthonormal and has $O(T(\varepsilon))$ columns again, while R_r is upper triangular. They satisfy the relationship

$$R_r^* Q_r^* = A_r$$

where A_r is a submatrix containing appropriate rows of A. We denote the samples (of X) associated with these rows by $\{x_i^c\}$. Similar to the previous step, the rows of Q_r^* approximately span the row space of A (not just A_2), producing an $O(\varepsilon)$ error with overwhelming probability.

4. Since Q_c and Q_r^* span, respectively, the column and the row spaces of A with error $O(\varepsilon)$, the matrix $M = Q_c^* A Q_r$ satisfies the relationship

$$|A - Q_c M Q_r^*| = O(\varepsilon). \tag{9}$$

Notice that $Q_c M$ and Q_r^* already provide us with a factorization of the matrix A with error $O(\varepsilon)$. However, since A is huge, computing $M = Q_c^* A Q_r$ can be very costly. To this end, we propose the following alternative. We randomly pick a set S of N_s rows and a set T of N_t columns. Set A_3 to be the minor containing the elements from rows in S and columns in T, $Q_{c,S}$ to be the submatrix of Q_c containing the rows in S, and $Q_{r,T}^*$ to be the submatrix of ${Q_r}^*$ containing the columns in T. Now the constraint (9) restricted to S and T becomes

$$\left|A_3 - Q_{c,S} M Q_{r,T}^*\right| = O(\varepsilon).$$

To satisfy this constraint, we choose

$$M = (Q_{c,S})^+ A (Q_{r,T}^*)^+ \tag{10}$$

where $(\ )^+$ stands for pseudo-inverse. In practice, we pick N_s and N_t to be about ten times $T(\varepsilon)$.

5. We claim that

$$|A - Q_c M Q_r^*| = O(\varepsilon).$$

Moreover, since $Q_c R_c = A_c$ and $R_r^* Q_r^* = A_r$, we have

$$\left|A - A_c \cdot (R_c)^+ \cdot M \cdot (R_r^*)^+ \cdot A_r\right| = O(\varepsilon).$$

Let us define D to be the matrix $(R_c)^+ \cdot M \cdot (R_r^*)^+$ sandwiched between A_c and A_r. We then have the approximation

$$\left|A - A_c \cdot D \cdot (R_r^*)^+ \cdot A_r\right| = O(\varepsilon). \tag{11}$$

(11) is often called a pseudoskeleton approximation of A in the literature (see [30, 31]). Notice that the matrix D has only $O(T(\varepsilon))$ rows and columns. Denoting the entries of D by d_{pq}, we can rewrite the previous statement in the form

$$\left| \frac{e^{2\pi\imath|x_i-y_j|}}{|x_i-y_j|} - \sum_{p,q} \frac{e^{2\pi\imath|x_i-y_p^c|}}{|x_i-y_p^c|} \cdot d_{pq} \cdot \frac{e^{2\pi\imath|x_q^c-y_j|}}{|x_q^c-y_j|} \right| = O(\varepsilon)$$

for all x_i and y_j.

6. Finally, since $\{x_i\}$ and $\{y_j\}$ sample the sets X and Y with a constant number of points per wavelength, it is reasonable to expect

$$\left| \frac{e^{2\pi\imath|x-y|}}{|x-y|} - \sum_{p,q} \frac{e^{2\pi\imath|x-y_p^c|}}{|x-y_p^c|} \cdot d_{pq} \cdot \frac{e^{2\pi\imath|x_q^c-y|}}{|x_q^c-y|} \right| = O(\varepsilon) \tag{12}$$

for any $x \in X \cap B(0,K)$ and $y \in Y$.

In (12), both $e^{2\pi\imath|x-y_p^c|}/|x-y_p^c|$ and $e^{2\pi\imath|x_q^c-y|}/|x_q^c-y|$ can be computed directly from the kernel formulae. Therefore, the only quantities that must be stored for (12) are the locations $\{x_q^c\}$ and $\{y_p^c\}$ and the matrix D. This costs only $O(1)$ in storage for a fixed error threshold ε. In fact, for the Helmholtz kernel, one only needs to sample the boundary of the domains X and Y densely in the first step. This improvement, which is due to the uniqueness and existence properties of the Dirichlet boundary value problem of the Helmholtz equation, significantly reduces the computational complexity of our randomized procedure.

The randomized procedure presented here benefits from the existing approaches for constructing low rank separated approximations. Kapur and Long [38] were among the first to use the idea of random sampling to construct low rank representation for electrostatic interaction. The adaptive cross approximation (ACA) by Bebendorf and Rjasanow [4, 5] constructs the approximation in an incremental way. Recently, it has been combined with the H-matrix framework [9] to address several problems in BIE formulations, including the scattering problems in the low frequency regime [22]. The idea of random sampling has been carefully studied in a series of papers by Drineas, Kannan, and Mahoney [25, 26] for general matrices, and our approach shares some similarities with the LinearTimeCUR algorithm [26] proposed by these authors. In [14], a similar algorithm is used to speed up the numerical computation of the Fourier integral operator. Finally, we would like to mention a recent algorithm proposed by Martinsson, Rokhlin, and Tygert [39] which exploits the spectral properties of the Gaussian random matrices and exhibits excellent numerical properties for matrices equipped with fast multiplication algorithms.

Our randomized procedure works well numerically. In section 5, we see that this procedure constructs low rank factorizations in a stable way for ε as small as 10^{-8}. The rank of the resulting factorization, i.e., the minimum of the cardinalities of the sets $\{y_p^c\}$ and $\{x_q^c\}$, is very close to the optimal rank obtained by factorizing A using an SVD.

In most of the cases, we do know the separation rank of A. Since the value of $T(\varepsilon)$ from Theorem 2.2 can be significantly larger than the true separation rank. Setting N_1 and N_2 to be a constant multiple of $T(\varepsilon)$ could be quite wasteful. In our implementation, we adopt the method proposed in [14]. The idea is to start with a relatively small value for N_1. If the columns of Q_c constructed from the second step fail to span the column space of A, we then double N_1 and repeat the second step

until the columns of Q_c span the column space of A within the prescribed error ε. The same strategy is applied to N_2 in the construction of Q_r^*. The time spent on the unsuccessful attempts is bounded by that of the final successful attempt due to the geometric growth rate.

Though we are not able to provide a rigorous proof for the accuracy of this randomized procedure, we argue that its success is based on the following observations:

- For fixed y, the kernel $e^{2\pi\imath|x-y|}/|x-y|$ is a wave-like function of x. The same is true if we fix x and regard $e^{2\pi\imath|x-y|}/|x-y|$ as a function of y. In fact, as verified by numerical computation, the row and column bases of the matrix A contain oscillatory vectors, each of which consists of sufficiently dense samples of an oscillatory function in R^3 with wavelength close to 1. In this regard, one may assume these oscillatory vectors to be the Fourier modes around the frequency 2π. Now, if the row basis contains the Fourier modes, the column vectors of A are incoherent mixtures of the vectors of the column basis. This incoherence property comes from the uncertainty principle between the Fourier and the (canonical) delta bases [15, 44]. As a result, sampling the columns of A with a moderate oversampling factor ensures that the selected vectors contain significant contributions from all of the vectors of the column basis. This is why we are able to construct the column basis of A in a stable way by observing only $N_1 = O(T(\varepsilon))$ of its vectors.
- As the vectors of Q_r are similar to the Fourier basis, the same incoherence property [15, 44] suggests that, as long as the number of rows in S is significantly larger than the number of columns of Q_r, the matrix $Q_{r,S}$ is not too far from an orthogonal matrix and thus has a small condition number. This allows us to invert $Q_{r,S}$ robustly and the same is true for $Q_{r,T}^*$. This allows us to recover M from (10) in a stable way.

In order to prepare for the discussion of our main algorithm in section 4, it is useful to introduce the following definitions. Suppose we have a set of densities $\{f_i\}$ located at points $\{b_i\}$ in Y. Using the representation obtained from the randomized procedure, we have

$$\left| \sum_i \frac{e^{2\pi\imath|x-b_i|}}{|x-b_i|} f_i - \sum_p \frac{e^{2\pi\imath|x-y_p^c|}}{|x-y_p^c|} \sum_q d_{pq} \sum_i \frac{e^{2\pi\imath|x_q^c-b_i|}}{|x_q^c-b_i|} f_i \right| = O(\varepsilon).$$

This states that we can place a set of densities

$$\left\{ \sum_q d_{pq} \sum_i \frac{e^{2\pi\imath|x_q^c-b_i|}}{|x_q^c-b_i|} f_i \right\} \tag{13}$$

at points $\{y_p^c\}$ in order to reproduce the potential generated by the densities $\{f_i\}$ located at points $\{b_i\}$. To this end, the densities in (13) are called the *directional outgoing equivalent densities* of Y in direction ℓ, and the points $\{y_p^c\}$ are called the *directional outgoing equivalent points* of Y in direction ℓ. Here the word outgoing refers to the role of Y in the computation. In addition, we refer to the quantities

$$\left\{ \sum_i \frac{e^{2\pi\imath|x_q^c-b_i|}}{|x_q^c-b_i|} f_i \right\} \tag{14}$$

as the *directional outgoing check potentials* of Y in direction ℓ and the points $\{x_q^c\}$ as the *directional outgoing check points* of Y in direction ℓ.

Notice that in (14) the directional outgoing check potentials can be computed using only kernel evaluation. In addition, as long as one has precomputed the matrix $D = (d_{pq})$, the directional outgoing equivalent densities can be evaluated using a single matrix vector multiplication (i.e., (13)).

Now, let us reverse the situation. Suppose we have a set of densities $\{f_i\}$ located at points $\{a_i\}$ in X so that

$$\left| \sum_i \frac{e^{2\pi\imath|y-a_i|}}{|y-a_i|} f_i - \sum_q \frac{e^{2\pi\imath|y-x_q^c|}}{|y-x_q^c|} \sum_p d_{pq} \sum_i \frac{e^{2\pi\imath|y_p^c-a_i|}}{|y_p^c-a_i|} f_i \right| = O(\varepsilon).$$

This states that we can put a set of densities

$$\left\{ \sum_p d_{pq} \sum_i \frac{e^{2\pi\imath|y_p^c-a_i|}}{|y_p^c-a_i|} f_i \right\} \tag{15}$$

at points $\{x_q^c\}$ and they reproduce the potential generated by the densities $\{f_i\}$ located at points $\{a_i\}$. Therefore, we call the densities in (15) the *directional incoming equivalent densities* of Y in direction ℓ and the locations $\{x_q^c\}$ the *directional incoming equivalent points* of Y in direction ℓ. In analogy to the previous terminology,

$$\left\{ \sum_i \frac{e^{2\pi\imath|y_p^c-a_i|}}{|y_p^c-a_i|} f_i \right\} \tag{16}$$

are called the *directional incoming check potentials* of Y in direction ℓ and the location $\{y_p^c\}$ are called the *directional incoming check points* of Y in direction ℓ.

A couple of remarks are in order here. Let us fix the direction ℓ. Since the kernel $e^{2\pi\imath|x-y|}/|x-y|$ is translation invariant, the equivalent points and check points for a set centered at an arbitrary point can be obtained from those of the set centered at the origin by translation. However, the D matrix remains the same.

The kernel is also rotation invariant. For a fixed radius r, we only need to generate the equivalent points and check points for a fixed direction ℓ. These quantities for any other direction can be obtained by rotation.

4. Algorithm description. Based on the results in the previous section, we are ready to describe our new algorithm for the N-body problem with oscillatory kernels, i.e., the computation of

$$u_i = \sum_{j=1}^{N} G(p_i, p_j) \cdot f_j$$

for all $1 \le i \le N$. Without loss of generality, we assume that $K = 2^{2L}$ for a positive integer L.

4.1. Data structure. Similar to the HF-FMM, our main data structure is an octree. The top level box of width K contains all the points $\{p_i\}$. In the rest of this paper, B denotes a box in the octree and w for its width. We say a box B is in the low frequency regime if $w < 1$ and B is in the high frequency regime if $w \ge 1$.

In the high frequency regime of the octree, no adaptivity is used, i.e., every nonempty box is further partitioned until the width of the box is less than 1. In the low frequency regime, a box B is partitioned as long as the number of points in B is

greater than a fixed constant N_p. In practice, the value of N_p is chosen to optimize the computational complexity.

As we have mentioned already, an FMM algorithm for the Laplace kernel can be easily modified to handle the low frequency case. For a box B in the low frequency regime, its data structure follows the description of the kernel independent FMM in [47]. We modify the notation slightly to accommodate the discussion of the current algorithm.

- The near field N^B is the union of the boxes A that satisfies $dist(A, B) = 0$, where $dist(A, B) = \inf_{x\in A, y\in B} |x - y|$.
- The far field F^B is the complement of N^B.
- The interaction list I^B contains all the boxes in $N^P \backslash N^B$ on B's level, where P is the parent box of B.
- $\{y_k^{B,o}\}$, $\{f_k^{B,o}\}$, $\{x_k^{B,o}\}$ and $\{u_k^{B,o}\}$ are, respectively, the *outgoing* equivalent points, equivalent densities, check points, and check potentials.
- $\{y_k^{B,i}\}$, $\{f_k^{B,i}\}$, $\{x_k^{B,i}\}$ and $\{u_k^{B,i}\}$ are, respectively, the *incoming* equivalent points, equivalent densities, check points, and check potentials.

To simplify the notation, we have omitted the range of the running index k.

Now let us consider a box B in the high frequency regime. The following definitions vary slightly from their low frequency counterparts:

- The near field N^B is the union of all the boxes $\{A\}$ that satisfy $dist(A, B) \leq w^2$.
- The far field F^B is the complement of N^B.
- The interaction list I^B contains all the boxes in $N^P \backslash N^B$ on B's level, where P is B's parent box.

In light of the preceding sections, we need to partition F^B into a group of directional regions, each belonging to a cone with spanning angle $O(1/w)$. To achieve this, we first cut F^B into six pyramids: V_{1+}, V_{1-}, V_{2+}, V_{2-}, V_{3+} and V_{3-}. For example, V_{1+} contains all the points whose first coordinate is positive and greater than the absolute values of the second and third coordinates. The other pyramids are similarly defined. Let us define $C = 4w$. Each part is further partitioned into C^2 wedges. As an example, for each point $p = (p^1, p^2, p^3)$ in V_{1+}, we define

$$\theta(p) = \arctan(p^2/p^1) \quad \text{and} \quad \phi(p) = \arctan(p^3/p^1).$$

Notice $|\theta(p)| \leq \pi/4$ and $|\phi(p)| \leq \pi/4$ for $p \in V_{1+}$. The C^2 wedges of V_{1+} are

$$\left\{p : -\frac{\pi}{4} + \frac{\pi}{2C} i \leq \theta(p) \leq -\frac{\pi}{4} + \frac{\pi}{2C}(i+1), -\frac{\pi}{4} + \frac{\pi}{2C} j \leq \phi(p) \leq -\frac{\pi}{4} + \frac{\pi}{2C}(j+1)\right\}$$

for $0 \leq i < C$ and $0 \leq j < C$. Clearly, the spanning angle for each of these wedges is $O(1/w)$. The wedges for other pyramids are generated in the same way. In total, for a box B of width w, its far field F^B is partitioned into $96 \cdot w^2$ wedges. We index these wedges using their center directions $\{\ell\}$. In Figure 3, we illustrate how these wedges cut the unit sphere into $96 \cdot w^2$ pieces for $w = 1, 2, 4$.

This construction has the advantage of ensuring a hierarchical structure of the wedges across adjacent levels. Suppose $w \geq 2$. For any directional index ℓ of B, one can always find an index ℓ' of the box with width $w/2$ such that the ℓth wedge of B is contained in the ℓ'th wedge of each of B's children. On the other hand, these wedges are not isometric anymore, i.e., one cannot obtain any other wedges by applying a rotation to an existing wedge. This implies that the point sets $\{x_c^q\}$ and $\{y_c^p\}$ and the matrix D can be very different for different wedges, even though the algorithm

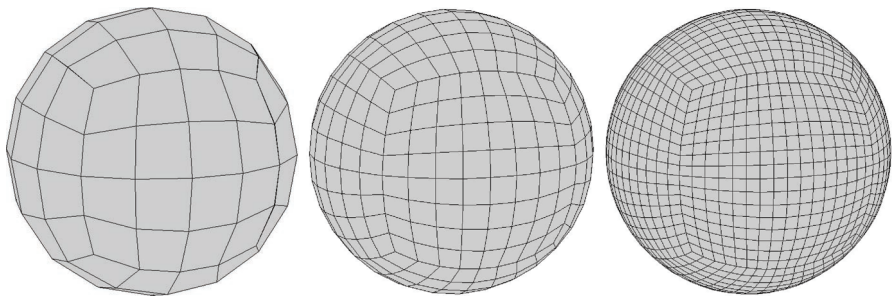


FIG. 3. *The wedges cut the unit sphere into* $96 \cdot w^2$ *pieces. From left to right,* $w = 1, 2, 4$.

used to construct them remains exactly the same. Fortunately, one does not need to construct the point sets $\{x_c^q\}$ and $\{y_c^p\}$ and the matrix D for each of these $96 \cdot w^2$ wedges due to a discrete isometric group acting on them. To be more precise, suppose one has a wedge with center direction $\ell = (\ell^1, \ell^2, \ell^3)$, then it is not difficult to see that we can obtain wedges centered along the following 48 directions through rotations:

$$(\pm\ell^1, \pm\ell^2, \pm\ell^3)(\pm\ell^2, \pm\ell^3, \pm\ell^1)(\pm\ell^3, \pm\ell^1, \pm\ell^2)$$
$$(\pm\ell^3, \pm\ell^2, \pm\ell^1)(\pm\ell^1, \pm\ell^3, \pm\ell^2)(\pm\ell^2, \pm\ell^1, \pm\ell^3).$$

Therefore, one is only required to compute the point sets $\{x_c^q\}$ and $\{y_c^p\}$ and the matrix D for about $2w^2$ wedges.

For each box B and each direction ℓ, we summarize the relevant quantities as follows:

- $\{y_k^{B,o,\ell}\}$, $\{f_k^{B,o,\ell}\}$, $\{x_k^{B,o,\ell}\}$, and $\{u_k^{B,o,\ell}\}$ are the *outgoing directional* equivalent points, equivalent densities, check points and check potentials respectively.
- $\{y_k^{B,i,\ell}\}$, $\{f_k^{B,i,\ell}\}$, $\{x_k^{B,i,\ell}\}$, and $\{u_k^{B,i,\ell}\}$ are the *incoming directional* equivalent points, equivalent densities, check points and check potentials respectively.

4.2. Translation operators. Similar to a standard FMM algorithm, our new algorithm utilizes several translation operators. The translation operators for boxes in the low frequency regime differs from those for boxes in the high frequency regime. Following tradition, we name these operators M2M translation, L2L translation, and L2L translation, though no multipole or local expansions are involved in our algorithm.

Low frequency regime. In the low frequency regime, the M2M, L2L, and M2L translations are exactly the same as the ones introduced in the kernel independent FMM. These operators are non-directional.

M2M operator. For a box B, the M2M operator constructs $\{f_k^{B,o}\}$, the box's outgoing non-directional equivalent densities. We start from a set of source densities, which are either the original densities in B if B is a leaf box or the union of the outgoing non-directional equivalent densities of B's children if B is not. In this procedure, we first calculate B's outgoing non-directional check potentials $\{u_k^{B,o}\}$ using kernel evaluation, and perform a matrix-vector multiplication to obtain $\{f_k^{B,o}\}$.

L2L operator. For a box B, the L2L operator starts with $\{u_k^{B,i}\}$, the box's incoming non-directional check potentials. It first constructs the incoming non-directional equivalent densities $\{f_k^{B,i}\}$ via a matrix-vector multiplication. The second step depends on whether B is a leaf box or not. If B is not a leaf, we evaluate the incoming

non-directional check potentials of B's children. If B is a leaf, then the potentials at the original points inside B are calculated.

M2L operator. The M2L operator works between two boxes A and B on the same level. A and B need to be on each other's interaction lists. The M2L operator transforms the outgoing non-directional equivalent densities of A to the incoming non-directional check potentials of B through kernel evaluation. This operator can be accelerated by the FFT as the kernel is translation invariant.

High frequency regime. The operators in the high frequency regime are more complicated. The main reason is that the computations are now directional.

M2M operator. For a box B in high frequency regime, the M2M operator constructs the outgoing directional equivalent densities of B from the outgoing equivalent densities of B's children. There are two cases to consider. In the first case, $w = 1$. The children boxes only have non-directional equivalent densities. The M2M operator iterates over all the directional indices $\{\ell\}$ of B and the steps for a fixed direction ℓ are:

1. Use $\bigcup_C \{y_k^{C,o}\}$ as source points in B and $\bigcup_C \{f_k^{C,o}\}$ as source densities. Here the union is taken over all the children boxes of B.
2. Compute $\{u_k^{B,o,\ell}\}$ at points $\{x_k^{B,o,\ell}\}$ with kernel evaluation and then obtain $\{f_k^{B,o,\ell}\}$ by multiplying $\{u_k^{B,o,\ell}\}$ with the matrix D associated with B and direction ℓ.

In the second case, $w > 1$. Now the children boxes have directional equivalent densities as well. The M2M operator again iterates over all the directional indices $\{\ell\}$ of B. The steps for a fixed direction ℓ are:

1. Pick ℓ', which is a direction associated with the boxes of width $w/2$, such that the wedge of B indexed by ℓ is contained in the wedge indexed by ℓ' of each of B's children. The existence of ℓ' is ensured by the way we partition F^r.
2. Use $\bigcup_C \{y_k^{C,o,\ell'}\}$ as source points in B and $\bigcup_C \{f_k^{C,o,\ell'}\}$ as source densities. Here the union is taken over all the children boxes of B.
3. Compute $\{u_k^{B,o,\ell}\}$ at $\{x_k^{B,o,\ell}\}$ with kernel evaluation and then obtain $\{f_k^{B,o,\ell}\}$ by multiplying $\{u_k^{B,o,\ell}\}$ with the matrix D associated with B and direction ℓ.

L2L operator. The L2L operator constructs the incoming check potentials of B's children from the incoming directional check potentials of B. Again there are two cases to consider. In the first case $w = 1$. The children boxes only have non-directional check potentials. The L2L operator iterates over all the directional indices $\{\ell\}$ of B and the steps for a fixed direction ℓ are:

1. Compute $\{f_k^{B,i,\ell}\}$ from $\{u_k^{B,i,\ell}\}$ by multiplying it with the appropriate D matrix.
2. For each child C of the box B, add to $\{u_k^{C,i}\}$ the potentials evaluated at $\{x_k^{C,i}\}$ using $\{f_k^{B,i,\ell}\}$ as the source densities at $\{y_k^{B,i,\ell}\}$.

In the second case, $w > 1$. Now the children boxes have directional equivalent densities. The L2L operator iterates over all the directional indices $\{\ell\}$ of B. The steps for a fixed direction ℓ are:

1. Pick ℓ', which is a direction associated with the boxes of width $w/2$, such that the wedge of B indexed by ℓ is contained in the wedge indexed by ℓ' of each of B's children. The existence of ℓ' is ensured by the way we partition F^r.
2. Compute $\{f_k^{B,i,\ell}\}$ from $\{u_k^{B,i,\ell}\}$ by multiplying it with the appropriate D matrix.
3. For each children C of the box B, add to $\{u_k^{C,i,\ell'}\}$ the potentials evaluated at $\{x_k^{C,i,\ell'}\}$ using $\{f_k^{B,i,\ell}\}$ as the source densities at $\{y_k^{B,i,\ell}\}$.

M2L operator. Finally, the M2L operator is applied to pairs of boxes A and B on the same level of the octree. They need to be on each other's interaction lists. Consider ℓ and ℓ' such that B falls into the wedge of A indexed by ℓ while A falls into the wedge of B indexed by ℓ'. The implementation of the M2L operator contains only one step:

1. Add to $\{u_k^{B,i,\ell'}\}$ the potentials evaluated at $\{x_k^{B,i,\ell'}\}$ using the densities $\{f_k^{A,o,\ell}\}$ at points $\{y_k^{A,o,\ell'}\}$.

To summarize the discussion on the transition operators, we would like to emphasize that all these operators involve only kernel evaluation and matrix-vector multiplication with precomputed matrices. Therefore, they are simple to implement and highly efficient.

4.3. Algorithm. Now we are ready to give the overall structure of our new algorithm. It contains the following steps.

1. Construct the octree. In the high frequency regime the boxes are partitioned uniformly. In the low frequency regime, a leaf box contains at most N_p points.
2. Travel up in the octree and visit the boxes in the low frequency regime. These boxes have width less than 1. For each box B, compute its outgoing non-directional equivalent densities $\{f_k^{B,o}\}$. This is done using the low frequency non-directional M2M operator.
3. Travel up in the octree and visit the boxes in the high frequency regime. For every such box B, use the high-frequency directional M2M operator to compute the outgoing directional equivalent densities $\{f_k^{B,o,\ell}\}$ for each outgoing direction ℓ. We skip the boxes with width greater than $\sqrt{K}$ since their interaction lists are empty.
4. Travel down in the octree and visit the boxes in the high frequency regime. For every such box B and for each direction ℓ, perform the following two steps:
   (a) Transform the outgoing directional equivalent densities $\{f_k^{A,o,\ell}\}$ of all the boxes $\{A\}$ in B's interaction list and in direction ℓ via the high-frequency directional M2L operator. Next, add the result to the incoming directional check potentials $\{u_k^{B,i,\ell}\}$.
   (b) Perform the high-frequency directional L2L operator to transform $\{u_k^{B,i,\ell}\}$ to the incoming check potentials for B's children.

   Again, we skip the boxes with width greater than $\sqrt{K}$.
5. Travel down in the octree. For every box B in the low frequency regime, we perform the following two steps:
   (a) Transform the outgoing non-directional equivalent densities $\{f_k^{A,o}\}$ of all the boxes $\{A\}$ in B's interaction list via the low frequency non-directional M2L operator. Next, add the result to the incoming non-directional check potentials $\{u_k^{B,i}\}$.
   (b) Perform the low frequency directional L2L operator. Depending on whether B is a leaf box or not, add the result to the incoming check potentials of B's children or to the potentials at the original points inside B.

An illustration of the various components of the algorithm is given in Figure 4.

In the description of the algorithm, we have assumed that the octree is full. This ensures that the M2L operator itself is sufficient to transform outgoing data to incoming data. When the octree is constructed adaptively, the situation is much more

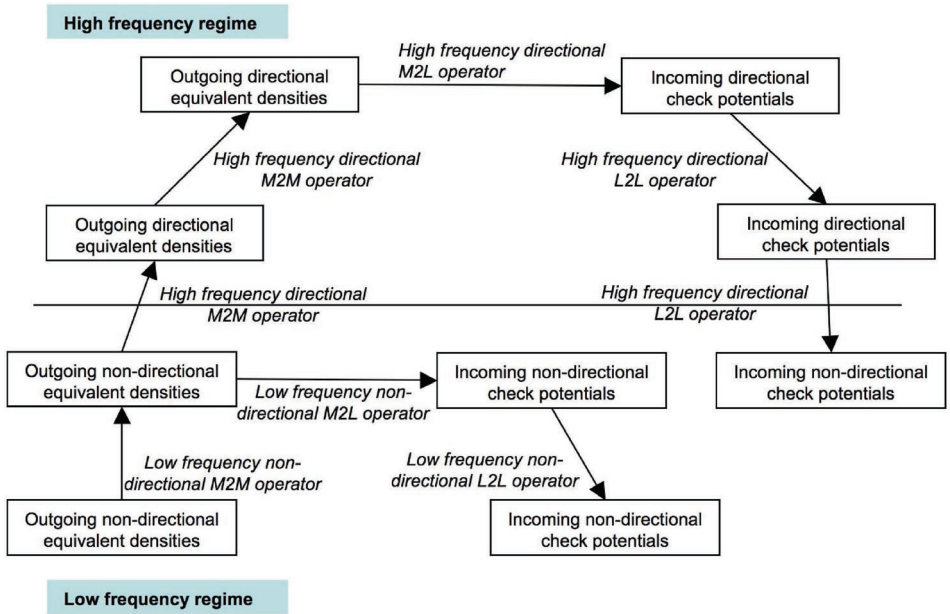


FIG. 4. *A small part of the octree used in the computation. Each rectangular region stands for a box of the octree. The diagram shows how the outgoing non-directional equivalent densities from a leaf box have been transformed into incoming non-directional check potentials at other leaf boxes. Far field interaction involves directional computation in the high frequency regime. The text in each box denotes the quantities involved, while the translation operators are shown in italics.*

complicated as one needs to keep the so-called U, V, W, and X lists for each box B in the low frequency regime. The necessary modifications to include this can be found in [19, 34, 47].

4.4. Complexity. We will now discuss the computational complexity of our algorithm assuming that the N points are sampled from a two dimensinoal surface. This assumption implies that $N = O(K^2)$ and it is generally satisfied for almost all applications which involve boundary integral formulations. In the following analysis, we use K as the main parameter because all the points $\{p_i\}$ satisfy $|p_i| \leq K/2$ by assumption.

THEOREM 4.1. *Let $\mathbb{S}$ be a surface in $B(0, 1/2)$. Suppose that for a fixed K, the points $\{p_i, 1 \leq i \leq N\}$ are samples of $K\mathbb{S}$, where $N = O(K^2)$ and $K\mathbb{S} = \{K \cdot p, p \in \mathbb{S}\}$ (the surface obtained by magnifying $\mathbb{S}$ by a factor of K). Then, for any prescribed accuracy, the proposed algorithm has a computational complexity $O(K^2 \log K) = O(N \log N)$.*

Outline of the proof. We analyze the number of operations for each major step of the algorithm.

- The first step of the algorithm has computational complexity $O(N) = O(K^2)$, due to the efficiency of the octree construction.
- The second step takes $O(N) = O(K^2)$ operations, as there are at most $O(N)$ boxes in the low frequency regime and each one involves a constant number of operations.
- We claim that the third step takes $O(N \log N)$ operations. To verify this, we first look at a single box of a fixed width w $(1 \leq w \leq \sqrt{K})$. Since there are

$O(w^2)$ directions $\{\ell\}$, the M2M operator for this box takes $O(w^2)$ operations. Next count the number of boxes of size w. Noticing that our points are discrete samples from a two dimensional manifold inside $B(0, K/2)$, it is not difficult to see that there are only $O(K^2/w^2)$ of them. This means that the overall complexity of the M2M operator for all the boxes of size w is $O(w^2 \cdot K^2/w^2) = O(K^2)$. Since we have totally $O(\log K)$ levels with $w = 1, 2, 4, \cdots, \sqrt{K}$, the complexity of this step is $O(K^2 \log K) = O(N \log N)$.

- We argue that the fourth step takes $O(N \log N)$ operations. The computation here contains two parts: the M2L operators and the L2L operators. The analysis for the L2L operators is exactly the same as the analysis of the third step, resulting an $O(N \log N)$ operation count. For the M2L operators, we know that, for a box B of width w, the boxes in its high frequency interaction list are approximately located in a ball centered at B with radius $(2w)^2$. Again, the fact that our points are samples from a two dimensional manifold implies that there are at most $O(w^4/w^2) = O(w^2)$ boxes in B's interaction list. Noticing that each M2L operator takes $O(1)$ operations and there are at most $O(K^2/w^2)$ boxes with width w, we conclude that overall complexity of the M2L operators for all the boxes of size w is $O(w^2 \cdot 1 \cdot K^2/w^2) = O(K^2)$. This further implies that the total number of the operations in the M2L operators is $O(K^2 \log K) = O(N \log N)$.
- Similar to the second step, the fifth step takes $O(N) = O(K^2)$ operations by the same argument.

Summing these contributions, we reach the conclusion that the total complexity is $O(N \log N)$, which is the same as the HF-FMM developed by Rokhlin and his collaborators (see, for example, [18]). □

5. Numerical results. In this section, we provide some numerical results to illustrate the properties of our new algorithm. Our implementation is written in C++ and all the computational results below are obtained on a desktop computer with a 3.0 GHz CPU. Due to the memory constraint, we restrict ourselves to problems which are at most 256 wavelengths (i.e., $K \leq 256$). The equivalent points, check points, and the related D matrices are precomputed for boxes with width $w = 1, 2, 4, 8$.

5.1. Separation rank. Let us first study the performance of the randomized procedure presented in section 3. In Table 1, we list the number of terms in the separated representation for two sets X_r and Y_r for different choices of accuracy ε and box width w. Here r is set to be $\sqrt{3}w$ so that the box of width w is contained in Y_r.

We can see from Table 1 that the separation rank is bounded by a constant which is independent of the values of w. This is consistent with our theoretical estimate in Theorem 2.2. In fact, as w grows, it seems that the separation rank decays slightly. The results also show that the separation rank seems to increase linearly with respect to $\log(1/\varepsilon)$. This is compatible with the results mentioned in [40].

TABLE 1

The separation rank of the directional separated representation for different choices of requested accuracy ε and box size w.

	$w = 1$	$w = 2$	$w = 4$	$w = 8$
ε=1e-4	45	45	45	45
ε=1e-6	85	82	81	81
ε=1e-8	121	114	113	111

The actual running time of the high frequency part of our algorithm depends on ε through the separation rank. In fact, the smaller ε, the larger the separation rank, and the longer the running time. As we pointed out already, the low frequency part of our algorithm is almost the same as the one described in [47]. For a prescribed value of ε, we need to choose a value for p, which is the size of the Cartesian grid used for the equivalent and check points. In our implementation, we pick $p = 4$ for ε =1e-4, $p = 6$ for ε=1e-6 and $p = 8$ for ε =1e-8.

5.2. Applications to different geometries. We applied our algorithm, with G being the Helmholtz kernel, to several objects. In our experiments, the surface of each object is represented by a triangular mesh. The point set $\{p_i\}$ is generated by sampling the triangular mesh randomly with 20 points per wavelength on average. This implies that the number of points is roughly 400 times the total surface area. Though these samples are far from optimal for a scattering application, they have the right distribution and are perfectly suitable for studying the performance of our algorithm.

In real applications, the actual sampling density often depends on the geometry of the surface. For example, one often places many more points in regions with large curvature. The result is to shift more computation towards the low frequency regime. Since the computation in the low frequency regime has lower complexity than the one for the high-frequency regime, our algorithm would generally perform better in these cases. Before reporting the results, let us summarize some relevant notations: N is the number of points, K is the size of the problem in terms of the wavelength, and ε is the prescribed error threshold such that the final error of is to be bounded by a constant multiple of ε.

In all experiments, the original densities $\{f_i\}$ are generated from a random distribution with mean 0. Use $\{u_i\}$ to denote the true potentials and $\{u_i^a\}$ to denote the approximations obtained through our algorithm. We estimate the relative error by picking a set S of 200 points from $\{p_i\}$. The true potentials $\{u_i, i \in S\}$ are computed by using direct evaluation. The error is then estimated to be

$$\sqrt{\frac{\sum_{i\in S} |u_i - u_i^a|^2}{\sum_{i\in S} |u_i|^2}}.$$

When reporting the numerical results, we use the following notations:

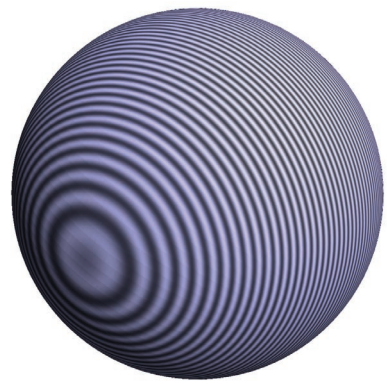
- T_a: the running time of our algorithm in seconds.
- T_d: the running time of the direct evaluation in seconds. This quantity is estimated by first calculating the time used for evaluating the true potentials at 200 points in S and then multiplying it by $N/200$.
- T_d/T_a: the speedup factor.
- ε_a: the error of our algorithm estimated using the method described above.

The first example is a sphere. The results are summarized in Table 2. The second example is a simplified model of an F16. The results are shown in Table 3. The final example is a submarine model. We summarize the computational results in Table 4.

From these tests, we can make the following observations about the performance of our algorithm:

- The running time scales roughly as $O(N \log N)$ in terms of the number of particles, matching well with the complexity analysis. In our implementation, most of the computation has been spent on computing $\{f_k^{B,o,\ell}\}$ and $\{u_k^{B,i,\ell}\}$ in steps 3 and 4(a) of the proposed algorithm (see section 4.3). The construction of one set of densities $\{f_k^{B,o,\ell}\}$ (or equivalently one set of potentials

TABLE 2
Results of the sphere with the Helmholtz kernel.

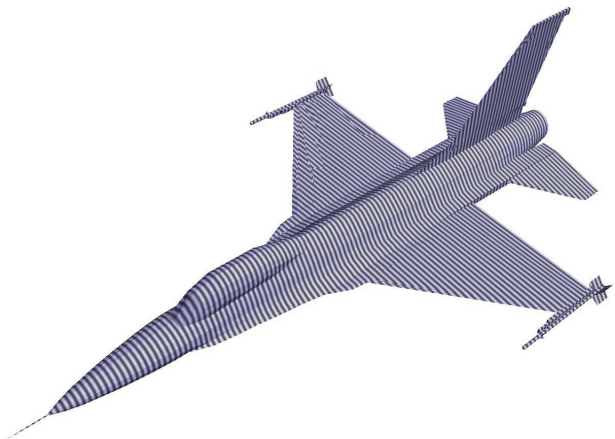


(K, ε)	N	T_a(sec)	T_d(sec)	T_d/T_a	ε_a
(16,1e-4)	3.22e+5	9.50e+1	1.21e+4	1.28e+2	5.08e-4
(32,1e-4)	1.29e+6	4.28e+2	1.95e+5	4.55e+2	5.91e-4
(64,1e-4)	5.15e+6	1.97e+3	3.04e+6	1.54e+3	6.30e-4
(16,1e-6)	3.22e+5	2.42e+2	1.18e+4	4.86e+1	2.92e-6
(32,1e-6)	1.29e+6	1.21e+3	1.87e+5	1.54e+2	2.12e-6
(64,1e-6)	5.15e+6	5.95e+3	3.13e+6	5.27e+2	3.70e-6
(16,1e-8)	3.22e+5	5.11e+2	1.22e+4	2.39e+1	7.16e-8
(32,1e-8)	1.29e+6	2.62e+3	1.96e+5	7.51e+1	9.19e-8
(64,1e-8)	5.15e+6	1.25e+4	3.15e+6	2.52e+2	9.14e-8

$\{u_k^{B,i,\ell}\}$) requires a couple of matrix multiplications, where the dimensions of the matrices involved are approximately equal to the separation rank. Since the complexity of each matrix multiplication is quadratic in the dimension of the matrix, the overall complexity of our algorithm scales quadratically in terms of the separation rank (see Table 1).

- For a fixed threshold ε, the final error of our directional multilevel algorithm seems to depend linearly on $\log_2 \sqrt{K}$, i.e., the number of levels in the octree. This is easy to understand: every time we compute the equivalent densities in a higher level, we introduce some extra error proportional to ε. Since $\log_2 K$ is a quite small number, for example $K = 256$ gives $\log_2 K = 8$, we still have very good control of the overall error.
- For objects on a fixed scale, our algorithm runs faster if the shape of an object is elongated. For example, the results on the submarine are better than the ones for the plane, which are better than the ones for the sphere. In other words, the algorithm prefers slender objects. The reason is quite simple: for an elongated object, the number of directions $\{\ell\}$ a box needs to address is much smaller than the number of directions for an isotropic object. In fact, the sphere seems to be the most difficult surface to work with.
- A close look at the breakdown of the computation time shows that the algorithm spends most of its time in the high frequency regime. This implies that for real problems where certain parts need sub-wavelength sampling, our algorithm would result a larger speedup factor compared with direct evaluation.

TABLE 3
Results of the F16 model with the Helmholtz kernel.



(K, ε)	N	T_a(sec)	T_d(sec)	T_d/T_a	ε_a
(32, 1e-4)	1.87e+5	5.00e+1	4.17e+3	8.34e+1	6.13e-4
(64, 1e-4)	7.46e+5	2.27e+2	6.58e+4	2.90e+2	6.69e-4
(128,1e-4)	2.98e+6	1.04e+3	1.03e+6	9.87e+2	6.89e-4
(256,1e-4)	1.19e+7	5.04e+3	1.64e+7	3.25e+3	7.63e-4
(32, 1e-6)	1.87e+5	1.18e+2	4.06e+3	3.44e+1	2.72e-6
(64, 1e-6)	7.46e+5	6.12e+2	6.56e+4	1.07e+2	3.30e-6
(128,1e-6)	2.98e+6	3.07e+3	1.06e+6	3.45e+2	4.16e-6
(32, 1e-8)	1.87e+5	2.38e+2	4.07e+3	1.71e+1	6.34e-8
(64, 1e-8)	7.46e+5	1.29e+3	6.64e+4	5.14e+1	8.10e-8
(128,1e-8)	2.98e+6	6.42e+3	1.06e+6	1.64e+2	6.55e-8

5.3. Generalization. As we have alluded, our algorithm works well for other oscillatory kernels as well. Here we provide some numerical results for the kernel,

$$e^{2\pi\imath|x-y|}.$$

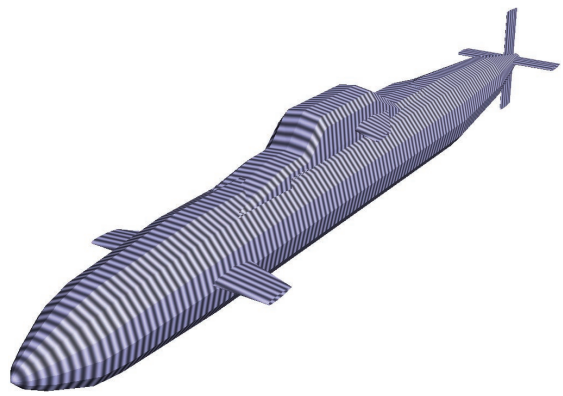
For this kernel, the proof in section 2 remains essentially the same, and the only change is to remove the estimate on the $1/|x-y|$ term. The randomized procedure in section 3 requires no modification as it only relies on the fact that the kernel has a low rank separated representation. Hence, our algorithm remains exactly the same in the high frequency regime. The only modification concerns the low frequency boxes. Our implementation follows the discussion in [46] and we refer to that paper for the details.

In Table 5, we report the results for the sphere with this new kernel. In Table 6, the results of the F16 model are presented.

These results suggest that the overall error of our algorithm for this new kernel with stronger far field oscillations is larger than the error for the Helmholtz kernel. However, the running time still follows the $O(N \log N)$ complexity very closely.

6. Conclusions and future work. In this paper, we introduced a new directional multilevel algorithm for computing the N-body problem with highly oscillatory kernels, and applied it to a boundary integral formulation of the Helmholtz equation. Our algorithm is based on the following three main components:

TABLE 4
Results of the submarine model with the Helmholtz kernel.



(K, ε)	N	T_a(sec)	T_d(sec)	T_d/T_a	ε_a
(32, 1e-4)	1.47e+5	3.90e+1	2.52e+3	6.46e+1	5.35e-4
(64, 1e-4)	5.85e+5	1.83e+2	4.00e+4	2.19e+2	6.37e-4
(128,1e-4)	2.34e+6	8.26e+2	6.57e+5	7.95e+2	6.32e-4
(256,1e-4)	9.36e+6	3.87e+3	1.01e+7	2.61e+3	8.32e-4
(32, 1e-6)	1.47e+5	9.10e+1	2.49e+3	2.74e+1	2.64e-6
(64, 1e-6)	5.85e+5	5.22e+2	4.07e+4	7.80e+1	2.17e-6
(128,1e-6)	2.34e+6	2.55e+3	6.39e+5	2.51e+2	4.64e-6
(32, 1e-8)	1.47e+5	1.81e+2	2.50e+3	1.38e+1	6.37e-8
(64, 1e-8)	5.85e+5	1.08e+3	3.98e+4	3.68e+1	8.05e-8
(128,1e-8)	2.34e+6	5.41e+3	6.63e+5	1.23e+2	8.51e-8

TABLE 5
Results of the sphere with the kernel $e^{2\pi\imath|x-y|}$.

(K, ε)	N	T_a(sec)	T_d(sec)	T_d/T_a	ε_a
(16,1e-4)	3.22e+5	1.12e+2	1.14e+4	1.02e+2	5.09e-3
(32,1e-4)	1.29e+6	5.10e+2	1.83e+5	3.58e+2	5.40e-3
(64,1e-4)	5.15e+6	2.36e+3	2.85e+6	1.21e+3	5.74e-3
(16,1e-6)	3.22e+5	3.82e+2	1.11e+4	2.90e+1	7.83e-6
(32,1e-6)	1.29e+6	1.92e+3	1.77e+5	9.21e+1	7.23e-6
(64,1e-6)	5.15e+6	9.36e+3	2.93e+6	3.13e+2	5.48e-6

TABLE 6
Results of the F16 model with the kernel $e^{2\pi\imath|x-y|}$.

(K, ε)	N	T_a(sec)	T_d(sec)	T_d/T_a	ε_a
(32, 1e-4)	1.87e+5	5.70e+1	3.79e+3	6.64e+1	6.31e-3
(64, 1e-4)	7.46e+5	2.64e+2	6.25e+4	2.37e+2	6.45e-3
(128,1e-4)	2.98e+6	1.24e+3	9.81e+5	7.94e+2	6.75e-3
(32, 1e-6)	1.87e+5	1.81e+2	3.82e+3	2.11e+1	8.79e-6
(64, 1e-6)	7.46e+5	9.52e+2	6.12e+4	6.43e+1	9.00e-6
(128,1e-6)	2.98e+6	4.76e+3	9.45e+5	1.99e+2	7.79e-6

- The directional low rank property, i.e., the interaction between two sets that follow the directional parabolic separation condition is approximately of low rank.
- A randomized procedure which efficiently and accurately constructs low rank, separated and stable representations.

- A multilevel and multidirectional strategy for organizing the overall computation. The fact that the high frequency part of our algorithm is not only multiscale but also multidirectional differs from the approach in the HF-FMM.

Our algorithm has been proved to have $O(N \log N)$ complexity and the numerical results show that it is also highly accurate. Furthermore, it can be adapted to handle kernels other than the Helmholtz kernel quite easily, which is not true for most other existing algorithms.

In future work, we plan to consider the following:

- It would be valuable to have a rigorous proof for the randomized procedure presented in section 3. The discussion presented here provides a possible outline of a proof.
- Currently, our implementation is capable of handling problems of up to about 256 wavelengths, however, many interesting scattering problems involve thousands of wavelengths. One solution is to design a parallel version of our algorithm. It is well-known that parallelizing standard FMMs is difficult since the top part of the octree is a bottleneck (see [48]). Fortunately, this bottleneck is alleviated by our algorithm, as we never visit the boxes of size greater than $\sqrt{K}$ wavelengths. A trivial implementation of partitioning the work at the level with boxes of size $\sqrt{K}$ would provide an appealing and practical solution.
- An FMM-type algorithm has been developed by Michielssen and his collaborators for time-domain scattering problems (see [20, 28]). It utilizes some of the techniques from the HF-FMM. It would be interesting to see whether our algorithm can be extended to handle time-domain problems.

Acknowledgments. The work presented in this paper is partially supported by startup funds of University of Texas at Austin. The authors thank R. Tsai and M. Tygert for stimulating discussions, L. Greengard, J. Qin, and R. Sharp for helping improve the manuscript, and the reviewers for their comments and suggestions.

REFERENCES

[1] B. Alpert, G. Beylkin, R. Coifman, and V. Rokhlin, *Wavelet-like bases for the fast solution of second-kind integral equations*, SIAM J. Sci. Comput., 14 (1993), pp. 159–184.

[2] C. R. Anderson, *An implementation of the fast multipole method without multipoles*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 923–947.

[3] A. Averbuch, E. Braverman, R. Coifman, M. Israeli, and A. Sidi, *Efficient computation of oscillatory integrals via adaptive multiscale local Fourier bases*, Appl. Comput. Harmon. Anal., 9 (2000), pp. 19–53.

[4] M. Bebendorf, *Approximation of boundary element matrices*, Numer. Math., 86 (2000), pp. 565–589.

[5] M. Bebendorf and S. Rjasanow, *Adaptive low-rank approximation of collocation matrices*, Computing, 70 (2003), pp. 1–24.

[6] G. Beylkin, R. Coifman, and V. Rokhlin, *Fast wavelet transforms and numerical algorithms.* I, Comm. Pure Appl. Math., 44 (1991), pp. 141–183.

[7] E. Bleszynski, M. Bleszynski, and T. Jaroszewicz, *AIM: Adaptive integral method for solving large-scale electromagnetic scattering and radiation problems*, Radio Science, 31 (1996), pp. 1225–1252.

[8] N. Bojarski, *K-space formulation of the electromagnetic scattering problems*, Technical report, Air Force Avionic Lab. Technical Report AFAL-TR-71-75, 1971.

[9] S. Börm, L. Grasedyck, and W. Hackbusch, *Hierarchical matrices*, Technical Report 21, Max-Planck-Institut für Mathematik in den Naturwissenschaften, Leipzig, 2003.

[10] B. Bradie, R. Coifman, and A. Grossmann, *Fast numerical computations of oscillatory integrals related to acoustic scattering.* I, Appl. Comput. Harmon. Anal., 1 (1993), pp. 94–99.

[11] A. Brandt and A. A. Lubrecht, *Multilevel matrix multiplication and fast solution of integral equations*, J. Comput. Phys., 90 (1990), pp. 348–370.

[12] O. P. Bruno and L. A. Kunyansky, *A fast, high-order algorithm for the solution of surface scattering problems: basic implementation, tests, and applications*, J. Comput. Phys., 169 (2001), pp. 80–110.

[13] O. M. Bucci and G. Franceschetti, *On the degrees of freedom of scattered fields*, IEEE Trans. Antennas Propagat., 37 (1989), pp. 918–926.

[14] E. J. Candès, L. Demanet, and L. Ying, *Fast computation of Fourier integral operators*, Technical report, California Institute of Technology, 2006.

[15] E. J. Candès, J. Romberg, and T. Tao, *Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information*, IEEE Trans. Inform. Theory, 52 (2006), pp. 489–509.

[16] F. X. Canning, *Sparse approximation for solving integral equations with oscillatory kernels*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 71–87.

[17] J. Carrier, L. Greengard, and V. Rokhlin, *A fast adaptive multipole algorithm for particle simulations*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 669–686.

[18] H. Cheng, W. Y. Crutchfield, Z. Gimbutas, L F. Greengard, J. F. Ethridge, J. Huang, V. Rokhlin, N. Yarvin, and J. Zhao, *A wideband fast multipole method for the Helmholtz equation in three dimensions*, J. Comput. Phys., 216 (2006), pp. 300–325.

[19] H. Cheng, L. Greengard, and V. Rokhlin, *A fast adaptive multipole algorithm in three dimensions*, J. Comput. Phys., 155 (1999), pp. 468–498.

[20] W. C. Chew, E. Michielssen, J. M. Song, and J. M. Jin, eds., *Fast and efficient algorithms in computational electromagnetics*, Artech House, Inc., Norwood, MA, USA, 2001.

[21] D. L. Colton and R. Kress, *Integral equation methods in scattering theory*, Pure and Appl. Math. (New York), John Wiley & Sons Inc., New York, 1983. A Wiley-Interscience Publication.

[22] B. Cranganu-Cretu, J. Ostrowski, and Z. Andjelic, *Fast integral equation solution for electric field shielding*, COMPEL: Internat J. Comput. and Math., in Electrical and Electronic Eng., 24 (2005), pp. 639–647.

[23] E. Darve, *The fast multipole method: Numerical implementation*, J. Comput. Phys., 160 (2000), pp. 195–240.

[24] E. Darve and P. Havé, *A fast multipole method for Maxwell equations stable at all frequencies*, Philos. Trans. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci., 362 (2004), pp. 603–628.

[25] P. Drineas, R. Kannan, and M. W. Mahoney, *Fast Monte Carlo algorithms for matrices.* II. *Computing a low-rank approximation to a matrix*, SIAM J. Comput., 36 (2006), pp. 158–183 (electronic).

[26] P. Drineas, R. Kannan, and M. W. Mahoney, *Fast Monte Carlo algorithms for matrices.* III. *Computing a compressed approximate matrix decomposition*, SIAM J. Comput., 36 (2006), pp. 184–206 (electronic).

[27] M. A. Epton and B. Dembart, *Multipole translation theory for the three-dimensional Laplace and Helmholtz equations*, SIAM J. Sci. Comput., 16 (1995), pp. 865–897.

[28] A. A. Ergin, B. Shankar, and E. Michielssen, *The plane-wave time-domain algorithm for the fast analysis of transient wave phenomena*, IEEE Antennas and Propagation Magazine, 41 (1999), pp. 39–52.

[29] G. T. D. Francia, *Degrees of freedom of an image*, J. Opt. Soc. Am., 59 (1969), pp. 779–804.

[30] S. A. Goreinov, E. E. Tyrtyshnikov, and N. L. Zamarashkin, *A theory of pseudoskeleton approximations*, Linear Algebra Appl., 261 (1997), pp. 1–21.

[31] S. A. Goreinov, N. L. Zamarashkin, and E. E. Tyrtyshnikov, *Pseudoskeleton approximations by submatrices of greatest size*, Mat. Zametki, 62 (1997), pp. 619–623.

[32] L. Greengard, *The rapid evaluation of potential fields in particle systems*, ACM Distinguished Dissertations, MIT Press, Cambridge, MA, 1988.

[33] L. Greengard, J. Huang, V. Rokhlin, and S. Wandzura, *Accelerating fast multipole methods for the Helmholtz equation at low frequencies*, IEEE Comput. Sci. Eng., 5 (1998), pp. 32–38.

[34] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comput. Phys., 73 (1987), pp. 325–348.

[35] W. Hackbusch and Z. P. Nowak, *On the fast matrix multiplication in the boundary element method by panel clustering*, Numer. Math., 54 (1989), pp. 463–491.

[36] J. G. Harris, *Linear elastic waves*, Cambridge Texts in Applied Mathematics, Cambridge University Press, Cambridge, 2001.

[37] R. Jakob-Chien and B. K. Alpert, *A fast spherical filter with uniform resolution*, J. Comput. Phy., 136 (1997), pp. 580–584.

[38] S. Kapur and D. E. Long, *IES*3: *A fast integral equation solver for efficient* 3*-dimensional extraction*, in ICCAD '97: Proceedings of the 1997 IEEE/ACM international conference on Computer-aided design, Washington, DC, USA, 1997, IEEE Computer Society, pp. 448–455.

[39] P.-G. Martinsson, V. Rokhlin, and M. Tygert, *A randomized algorithm for the approximation of matrices*, Technical report, Yale University, 2006.

[40] E. Michielssen and A. Boag, *A multilevel matrix decomposition algorithm for analyzing scattering from large structures*, IEEE Transactions on Antennas and Propagation, 44 (1996), pp. 1086–1093.

[41] N. Nishimura, *Fast multipole accelerated boundary integral equation methods*, Applied Mechanics Reviews, 55 (2002), pp. 299–324.

[42] V. Rokhlin, *Rapid solution of integral equations of scattering theory in two dimensions*, J. Comput. Phys., 86 (1990), pp. 414–439.

[43] V. Rokhlin, *Diagonal forms of translation operators for the Helmholtz equation in three dimensions*, Appl. Comput. Harmon. Anal., 1 (1993), pp. 82–93.

[44] M. Rudelson and R. Vershynin, *Sparse reconstruction by convex relaxation: Fourier and gaussian measurements*, 2006.

[45] J. M. Song and W. C. Chew, *Multilevel fast-multipole algorithm for solving combined field integral equations of electromagnetic scattering*, Microwave Opt. Tech. Lett., 10 (1995), pp. 15–19.

[46] L. Ying, *A kernel independent fast multipole algorithm for radial basis functions*, J. Comput. Phys., 213 (2006), pp. 451–457.

[47] L. Ying, G. Biros, and D. Zorin, *A kernel-independent adaptive fast multipole algorithm in two and three dimensions*, J. Comput. Phys., 196 (2004), pp. 591–626.

[48] L. Ying, G. Biros, D. Zorin, and H. Langston, *A new parallel kernel-independent fast multipole method*, in SC '03: Proceedings of the 2003 ACM/IEEE conference on Supercomputing, Washington, DC, USA, 2003, IEEE Computer Society, p. 14.

SIAM J. SCI. COMPUT.
Vol. 29, No. 4, pp. 1738–1758
© 2007 Society for Industrial and Applied Mathematics

A FAST SHADOWING ALGORITHM FOR HIGH-DIMENSIONAL ODE SYSTEMS*

WAYNE B. HAYES† AND KENNETH R. JACKSON‡

Abstract. Numerical solutions to chaotic dynamical systems are suspect because the exponential divergence of nearby trajectories causes numerical errors to be exponentially magnified with time. To bolster confidence in the reliability of such numerical solutions, we turn to the study of *shadowing:* a *shadow* is an exact trajectory of a chaotic dynamical system that stays close to a numerical trajectory for a long time. Finding shadows of numerical trajectories is very computationally intensive, and until recently it has been infeasible to study shadows of higher-dimensional systems. This paper introduces several optimizations to previous algorithms, which collectively achieve an average speedup of almost two orders of magnitude with no measurable loss in effectiveness. We test the algorithm on systems with up to 180 dimensions. Its application to large gravitational N-body integrations has already led to a deeper understanding of the reliability of galaxy and cosmological simulations. The source code (available from the first author) provides to the experienced user a generic interface capable of searching for shadows of any set of ordinary differential equations.

Key words. error analysis, ordinary differential equations, dynamical systems, chaos, shadowing, computational techniques

AMS subject classifications. 37M05, 65C20, 68U20, 70K99, 81T80

DOI. 10.1137/060654840

1. Introduction.

1.1. Motivation: Ordinary differential equations, numerical error, and chaos. The subject of [illegible] (IVPs) for autonomous ODEs concerns the solution of

$$\mathbf{y}'(t) = \mathbf{f}(\mathbf{y}(t)), \tag{1}$$

$$\mathbf{y}(t_0) = \mathbf{y}_0, \tag{2}$$

where (1) is called the [illegible] and (2) is the [illegible]. Let $\mathbf{y}(t; t_0, \mathbf{y}_0)$ be the solution of (1), (2), and define the time-h solution operator φ_h to be $\varphi_h(\mathbf{x}) \equiv \mathbf{y}(h; 0, \mathbf{x})$. A [illegible] for solving (1), (2) consists of building an approximation $\tilde{\varphi}_h$ to φ_h for small h, and then computing a sequence of discrete points $\mathbf{y}_{i+1} = \tilde{\varphi}_{h_i}(\mathbf{y}_i)$ representing approximations to $\mathbf{y}(t_{i+1}; t_0, \mathbf{y}_0)$, where $t_{i+1} = t_i + h_i$. We call such a discrete sequence of points a [illegible]. If the pseudotrajectory satisfies a [illegible] of δ such that $\|\mathbf{y}_{i+1} - \varphi_{h_i}(\mathbf{y}_i)\| \le \delta$, then we call it a δ [illegible].

The natural first question to ask about pseudotrajectories is how well they represent exact trajectories. If δ is $O(h^{p+1})$, then $\tilde{\varphi}_h$ is called an [illegible] p method, and the [illegible] $\|\mathbf{y}_i - \mathbf{y}(t_i; 0, \mathbf{y}_0)\|$ is bounded by $O(e^{Lt_i} h^p / |L|)$, where L is a bound on

*Received by the editors January 26, 2007; accepted for publication (in revised form) March 9, 2007; published electronically August 22, 2007. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada, and by the Information Technology Research Centre of Ontario.

http://www.siam.org/journals/sisc/29-4/65484.html

†Computer Science Department, University of California, Irvine, Irvine, CA 92697-3435 (wayne@ics.uci.edu).

‡Computer Science Department, University of Toronto, Toronto, ON, Canada M5S 3G4 (krj@cs.toronto.edu).

the [illegible] of the Jacobian of $\mathbf{f}$ [20, subsections 8.1.2, 8.3.6]. If L is negative, then the error is uniformly bounded; otherwise, the forward error is unbounded, and more sophisticated methods of error analysis must be used to gain insight into the value of the numerical solution.

[illegible] relates the pseudotrajectory to the exact solution of a nearby problem. If $\mathbf{x}(t)$ is a continuous approximate solution [24], then the [illegible] is defined as $\delta(t) = \mathbf{x}'(t) - \mathbf{f}(\mathbf{x}(t))$. If for some input tolerance ε we can show that $\|\delta(t)\| \leq \varepsilon$ wherever $\mathbf{x}'(t)$ is defined over the whole of the interpolated solution, then the interpolated solution $\mathbf{x}(t)$ is the exact solution to the defining equation $\mathbf{x}'(t) = \mathbf{f}(\mathbf{x}(t)) + \delta(t)$ [19, 18]. One can sometimes even compute the precise form of $\delta(t)$ analytically [1]. Although defect-based analysis is sometimes appropriate, modifying the defining equation may be undesirable. The present authors are inspired by a desire to understand the meaning of numerical simulations of physical systems in which the physics is well understood and can be well modeled. Since real-world physical systems undergo perturbations, it can be argued that an analysis that allows perturbations of the defining equation is sufficient (see, for example, [18]). However, numerical errors may be biased in qualitatively different ways than other perturbations and may introduce biases into the numerical solution that cause it to behave in an unphysical manner. For example, many physical perturbations would not appreciably change the energy of the system under study, whereas spurious energy dissipation is a major problem in long numerical integrations of systems which should be conservative [7, 41]. Although symplectic integrators [7, 41] and other types of conservative integrators [46] may conserve certain quantities, it is not clear that they do not introduce new biases, such as unphysical energy transport. Physical systems often satisfy properties such as symplecticness, conservation of energy, conservation of phase-space volume, and conservation of various types of momentum. Many of these are well conserved in real systems that experience perturbations, but are not well conserved by many otherwise well-behaved numerical methods. This has been confirmed with several symplectic maps using a fixed-timestep fourth-order Runge–Kutta integrator [7], and by the current authors using the n-body problem and comparing a 7/8-order Runge–Kutta pair [23], two Adams's methods [34, 35], and a Bulirsch–Stöer method [39] to the leapfrog method, which is a second-order symplectic method. In general, for a defect analysis to be fully trustworthy, we want to ensure that changing the field represented by (1) does not affect any quantities of interest [47, 48], but this is difficult to demonstrate.

In contrast, shadowing is a method of backward error analysis in which the defining equation (1) remains fixed, while the initial condition (2) is allowed to change. In other words, a [illegible] is an exact solution to the [illegible] defining equation, which starts close to the pseudotrajectory and remains close to it for a nontrivial duration of time. Shadowing is thus best applied to systems in which the governing equations are well known, and virtually all error is introduced by imprecise knowledge of initial conditions and/or by numerical error in the computation of the solution. It is less applicable to systems in which the mathematics only approximately model the truth, because having an exact solution to an approximate model is of dubious merit.

In addition, "chaotic" systems display [illegible] nearby solutions tend to diverge exponentially away from each other with time. Since numerical methods introduce small errors, it is virtually guaranteed that a pseudotrajectory of a chaotic system will diverge exponentially away from the exact solution with the same initial conditions. Some practitioners (e.g., [25]) have labeled this the "exponential magnification of small errors," leaving open the possibility that numerical trajectories of such simulations are the result of nothing but magnified noise.

Although the presence of exponentially magnified errors is widely recognized, its impact on the qualitative properties of solutions is not well understood.

Nevertheless, numerical solutions can mimic with astounding accuracy the phenomena they purport to simulate. Simulations of galaxies often closely resemble real galaxies [13, 37]. Even galaxy collisions can be modeled in a convincing manner [49, 22]. More generally, exponential divergence of nearby trajectories implies that an initially dense ensemble of points will disperse into a uniform distribution in a relatively short time [47, 48]. This effect is also seen in numerical simulations of chaotic systems [38, 36]. The natural question to ask is whether these simulations are behaving in a fashion similar to real systems, or whether they only superficially mimic real systems but are in fact behaving incorrectly at a more fundamental level. If this were the case, then we could be lulled into a false sense of security while our understanding of these systems becomes compromised.

Since shadowing disallows changes in the defining equation (1), some of these kinds of insidious errors can be ruled out. Furthermore, if the problem (1) arises from a purely mathematical context and not a physical one, we may be earnestly interested in the properties of exact solutions of (1), in which case a recourse to shadowing may be the only option. The only remaining question would then be whether shadows are typical of exact solutions chosen at random.

From the standpoint of physics, the existence of a shadow of a numerical simulation is strong property. In effect, it says that if a shadow exists, then a numerical simulation can be viewed as an [illegible] of an exact trajectory. The distance between the numerical trajectory and the shadow is analogous to experimental error, which is a familiar concept to the physicist. Thus, the study of shadowing is complementary to the study of perturbations of the defining equation.

On the other hand, shadowing algorithms as presented here and elsewhere are extremely expensive. Defect controlled methods are of roughly equal expense compared to more traditional methods of error control, and thus can be used to dynamically control local error at runtime [23]. In contrast, shadowing requires an integration of the variational equation and the inversion of a large matrix, and so costs at least $O(N^3)$, where N is the number of equations. Thus, shadowing is not practical as a method of runtime error control in a numerical integration. Nor is the intent to increase the accuracy of inaccurate trajectories; for this one can simply use a tighter local error tolerance in the original numerical integration. Rather, the intent of shadowing is to perform an a posteriori test of the global accuracy of locally inaccurate integrations, in an effort to discover under what conditions such integrations tend to produce physically meaningful trajectories. These results can then be used to guide error criteria for more efficient (and typically less accurate) integrations. This is precisely the use to which the current algorithm has fruitfully been put elsewhere [31, 30]. More systems can and should be tested for shadowing properties, which is why an efficient and robust algorithm, easily adaptable to new problems, is needed.

To our knowledge, the shadowing algorithm described herein is the first implementation which is demonstrably efficient and effective at shadowing systems with a large number of dimensions. Although shadowing algorithms have previously been described [8, 9, 10, 12, 42, 44, 50], they have been applied exclusively to small test problems with at most a few dimensions. Although all shadowing algorithms, including the one presented here, formally require at least $O(N^3)$ computations, the aim of the current work is to demonstrate that, by the careful analysis of shadowing accuracy requirements and the use of various heuristics, at least two orders of magnitude in runtime can be gained, at no observable cost in efficacy or robustness. We should also

mention that the algorithm here is not rigorous: [illegible] does not prove the existence of a nearby exact trajectory, but only demonstrates the existence of accurate trajectories (i.e., ones with local error close to the machine precision) that remain close to inaccurate ones. Rigorous algorithms do exist [15, 16, 17, 32], but since they require validated integration methods, they are several orders of magnitude more expensive than the ones described here. Our own experiments (as yet unpublished) have also shown that rigorous algorithms tend to be less robust, since the theorems describing the sufficient conditions for shadow existence tend to be more restrictive than is necessary in practice. In other words, the nonrigorous algorithms are able to suggest the existence of shadows under much less restrictive conditions than the rigorous algorithms can cope with.

In this paper we introduce several improvements to the nonrigorous refinement algorithm of [40], resulting in an average speedup of about two orders of magnitude over their implementation. This optimized algorithm has been tested on a smooth, almost hyperbolic Hamiltonian system with 180 dimensions, whose physical size was scaled to be of order unity. To give an idea of the efficacy of the algorithm, recall that nearby solutions of chaotic systems diverge exponentially from each other with time t. If the average distance between the two increases as $e^{\lambda t}$, then λ is called the [illegible] [illegible]. The [illegible] is $1/\lambda$ and defines the average time it takes for the two solutions to diverge by a factor of $\exp(1) \approx 2.718$. Given a numerical trajectory having local error of about 10^{-4}, the algorithm described herein found shadows staying within a distance of about 10^{-3}, for a duration of more than 50 Lyapunov times. Considering that a numerical error of size 10^{-4} is magnified to be comparable to unity after only 9 Lyapunov times (since $e^9 \approx 10^4$), the shadow stays close to the numerical solution for a duration much longer than two solutions would typically remain close together. Thus, the algorithm is capable of finding long, high-dimensional shadows. The algorithm has also been successfully applied to the more realistic problem of gravitational N-body systems with over 200 phase-space dimensions, leading to a deeper understanding of the reliability of galaxy and cosmological simulations [31, 30]. The purpose of this paper is to describe the as-yet unpublished algorithm that was used to produce the above results.

1.2. Background and definitions. We use the terms [illegible], [illegible], and [illegible] interchangeably. For numerical discussions, we assume a machine precision of approximately 16 decimal digits and a well-scaled problem where all macroscopic quantities of interest are of order unity. Like others in the shadowing literature, we abuse the term "noise" to mean "numerical error"; no stochasticity is implied. Throughout this paper, $|\cdot|$ denotes the Euclidean norm, and $\|\cdot\|$ denotes the max norm, although we do not expect the choice of norm to qualitatively affect our results.

An [illegible] $\{\mathbf{x}_i\}_{i=a}^{b}$ of φ satisfies $\mathbf{x}_{i+1} = \varphi(\mathbf{x}_i)$ for $a \le i < b$ (a and b are finite integers). For $a \le i < b$, the 1 [illegible] or [illegible] made between step i and step $i+1$ of the pseudotrajectory $\{\mathbf{y}_i\}_{i=a}^{b}$ is the vector difference $\mathbf{e}_{i+1} = \mathbf{y}_{i+1} - \varphi(\mathbf{y}_i)$. Thus, an exact trajectory is one whose 1-step errors are identically zero, and a δ-pseudotrajectory is one whose 1-step errors satisfy $\|\mathbf{e}_i\| < \delta$ for $a < i \le b$.

An exact trajectory $\{\mathbf{x}_i\}_{i=a}^{b}$ ε-[illegible] the pseudotrajectory $\{\mathbf{y}_i\}_{i=a}^{b}$ if $\|\mathbf{x}_i - \mathbf{y}_i\| < \varepsilon$ for $a \le i \le b$. The pseudotrajectory $\{\mathbf{y}_i\}_{i=a}^{b}$ has a [illegible] at point $i = G_0 < b$ if for some relevant ε there exists an exact trajectory that ε-shadows $\{\mathbf{y}_i\}_{i=a}^{G_0}$, but no exact trajectory that ε-shadows $\{\mathbf{y}_i\}_{i=a}^{G}$, for $G > G_0$.

For the purpose of generating shadows, we must choose an appropriate time interval across which to compute 1-step errors. The obvious choice is to use the same

timesteps as were used by the original numerical integration. However, as we will see later, it is often more efficient to concatenate many timesteps together into a single larger step, which we call a [illegible].

[illegible] is a process similar to Newton's method that iteratively attempts to reduce the noise across a numerical orbit. A refinement iteration is [illegible] if before the iteration the trajectory has noise δ^0, after the iteration it has noise δ^1, and

$$\delta^1 < \mu\delta^0 \text{ for some reasonable } \mu \in (0,1). \tag{3}$$

Otherwise the refinement iteration is [illegible].

Shadowing was first discussed in relation to [illegible] systems by Anosov [2] and Bowen [6]. In a 2-dimensional hyperbolic system, there are two special directions called the [illegible] (or [illegible]) and the [illegible] (or [illegible]) directions, which may vary with time and generally are not orthogonal. Small perturbations along the stable direction decay exponentially in forward time, while small perturbations in the unstable direction grow exponentially in forward time. The two directions reverse roles in backward time. In addition, in a hyperbolic system, the angle between the stable and unstable directions is uniformly bounded away from 0. In such a system, Anosov and Bowen proved that $\forall\varepsilon > 0$, $\exists\delta > 0$ such that every δ-pseudotrajectory is ε-shadowed.

For nonhyperbolic systems, it seems that we must be satisfied with finite-length shadows. The first studies of shadows for nonhyperbolic systems appear to be by [5] and [28]. In the 1990s, there was a flurry of activity in the study of numerical methods for finding shadows [26, 40, 10, 11, 50, 15, 14], as well as studies of when shadowing tends to break down [21, 43, 45]. In this paper, we concern ourselves with the refinement procedure of [40], which is an extension of the 2-dimensional procedure of [26] to arbitrary-dimensional Hamiltonian systems. These two papers will be referred to as QT and GHYS, respectively.

2. The refinement procedure of GHYS and QT.

2.1. Outline. The refinement procedure of GHYS [26] and QT [40] is similar to Newton's method for finding a zero of a function. Here we provide a heuristic explanation. Let $\mathbf{Y} = \{\mathbf{y}_i\}_{i=0}^{S}$ be a trajectory with S steps that has noise $\delta > \eta\varepsilon_{mach}$, where ε_{mach} is the machine precision and η is some constant significantly greater than 1 that allows room for improvement towards the machine precision. Let $\mathbf{e}_{i+1} = \mathbf{y}_{i+1} - \varphi(\mathbf{y}_i)$ be an estimate of the 1-step error at step $i+1$, where $\|\mathbf{e}_i\| < \delta$ for all i. The set of all 1-step errors is represented by $\mathbf{E} = \{\mathbf{e}_i\}_{i=1}^{S}$ and is estimated by a numerical integration technique that has higher accuracy than that used to compute $\mathbf{Y}$. Let g be the function that computes the set of 1-step errors $\mathbf{E}$, given the entire numerical trajectory $\mathbf{Y}$ as input, i.e., $g(\mathbf{Y}) = \mathbf{E}$. Since the 1-step errors are assumed to be small, $\|\mathbf{E}\|$ is small. That is, $\mathbf{Y}$ is close to a zero of g if one exists. A zero of g would represent an orbit in which the 1-step errors are identically zero, i.e., an exact orbit. This is an ideal situation in which to apply Newton's method. Just as with Newton's method, refinement is [illegible], in that we repeatedly compute and apply a correction $\mathbf{C} = \{\mathbf{c}_i\}_{i=0}^{S}$ to $\mathbf{Y}$, ideally giving a less noisy orbit $\mathbf{Y}' = \mathbf{Y} + \mathbf{C}$. Refinement adds a few technical complications that disallow us from directly using Newton's method. For example, the number of constraints (1-step errors) is less than the number of unknowns (correction points in the orbit), and the computation is ill-conditioned. Both problems are solved by imposing boundary conditions on the corrections at both endpoints of the orbit (described below). In practice, we continue iteration until refinement either stops working or produces 1-step errors near the machine precision.

In the latter case we call the refined orbit a [illegible]. The existence of a numerical shadow supports the existence of an exact shadow of comparable duration [40, 44], although rigorously proving the existence of an exact shadow [44, 15, 32] is much more expensive than refinement. We do not pursue rigorous proofs here.

2.2. The GHYS refinement algorithm. Assume that we have a noisy orbit $\mathbf{Y} = \{\mathbf{y}_i\}_{i=0}^{S}$ and that it has a shadow $\{\mathbf{x}_i\}_{i=0}^{S}$. Then

$$\mathbf{x}_{i+1} = \varphi(\mathbf{x}_i) \quad \text{and} \quad \mathbf{y}_{i+1} = \tilde{\varphi}(\mathbf{y}_i),$$

where $\tilde{\varphi}$ is an approximation to φ with a typical accuracy or [illegible] of δ. Using a more accurate integration, we can estimate the 1-step errors $\mathbf{e}_{i+1} = \mathbf{y}_{i+1} - \varphi(\mathbf{y}_i)$. Then, $\hat{\mathbf{c}}_i \equiv \mathbf{x}_i - \mathbf{y}_i$ represents a correction term that perturbs point $\mathbf{y}_i$ towards the shadow $\mathbf{x}_i$, and the next correction term along the orbit is

$$\hat{\mathbf{c}}_{i+1} \equiv \mathbf{x}_{i+1} - \mathbf{y}_{i+1} = \varphi(\mathbf{x}_i) - \varphi(\mathbf{y}_i) - \mathbf{e}_{i+1} = \frac{\partial \varphi(\mathbf{y}_i)}{\partial \mathbf{y}_i}\hat{\mathbf{c}}_i - \mathbf{e}_{i+1} + O(|\hat{\mathbf{c}}_i|^2). \tag{4}$$

In the spirit of Newton's method, we ignore the $O(|\hat{\mathbf{c}}_i|^2)$ in (4), and so one refinement iteration defines the corrections along the entire orbit

$$\mathbf{c}_{i+1} = L_i \mathbf{c}_i - \mathbf{e}_{i+1}, \tag{5}$$

where $L_i \equiv \frac{\partial \varphi(\mathbf{y}_i)}{\partial \mathbf{y}_i}$ is the linearized perturbation map (we will describe how to initialize the iteration below). For a discrete map, L_i is just the Jacobian of the map at step i. For a system of ODEs, L_i is the Jacobian of the integral of the ODE from step i to step $i+1$, computed by solving the linear variational equation [27]. If we let d be the dimension of the problem, then the resolvent has $O(d^2)$ elements in it, and it takes $O(d^3)$ time to compute, assuming that each element takes time $O(d)$ to compute, as is the case in the N-body problem [29].

For simplicity of explanation, we assume $d = 2$ for the remainder of this subsection, deferring discussion of the higher-dimensional case to the next subsection.

If the problem were not chaotic, the correction terms $\mathbf{c}_i$ could be computed directly from (5). But since L_i will amplify any errors in $\mathbf{c}_i$ not lying in the stable direction, computing the $\mathbf{c}_i$'s by iterating (5) forward will amplify errors and typically produce nothing but noise. Therefore, we split the error and correction terms into components in the stable ($\mathbf{s}_i$) and unstable ($\mathbf{u}_i$) directions at each timestep:

$$\mathbf{e}_i = e_{u_i}\mathbf{u}_i + e_{s_i}\mathbf{s}_i, \qquad \mathbf{c}_i = c_{u_i}\mathbf{u}_i + c_{s_i}\mathbf{s}_i. \tag{6}$$

Since it is not known a priori which direction is unstable at each timestep, the unstable vector $\mathbf{u}_0$ at time t_0 is initialized to an arbitrary unit vector. The linearized map is then iterated forward with

$$\bar{\mathbf{u}}_{i+1} = L_i \mathbf{u}_i, \qquad \mathbf{u}_{i+1} = \bar{\mathbf{u}}_{i+1}/|\bar{\mathbf{u}}_{i+1}|. \tag{7}$$

Since L_i magnifies any component that lies in the unstable direction, $\mathbf{u}_i$ converges to the true unstable direction as i increases. Similarly, the stable unit direction vectors $\mathbf{s}_i$ are computed by initializing $\mathbf{s}_S$ to an arbitrary unit vector and iterating backward,

$$\bar{\mathbf{s}}_i = L_i^{-1}\mathbf{s}_{i+1}, \qquad \mathbf{s}_i = \bar{\mathbf{s}}_i/|\bar{\mathbf{s}}_i|. \tag{8}$$

Substituting (6) into (5) yields

$$c_{u_{i+1}}\mathbf{u}_{i+1} + c_{s_{i+1}}\mathbf{s}_{i+1} = L_i(c_{u_i}\mathbf{u}_i + c_{s_i}\mathbf{s}_i) - (e_{u_{i+1}}\mathbf{u}_{i+1} + e_{s_{i+1}}\mathbf{s}_{i+1}). \tag{9}$$

While L_i magnifies errors in the unstable direction, it damps them in the stable direction. Likewise, L_i^{-1} damps errors in the unstable direction and magnifies errors in the stable direction. Thus the c_u terms should be computed backward, and the c_s terms forward. Taking components of (9) in the unstable direction at step $i+1$, we iterate backward on

$$c_{u_i} = (c_{u_{i+1}} + e_{u_{i+1}})/|\bar{\mathbf{u}}_{i+1}|, \tag{10}$$

and taking components in the stable direction, we iterate forward on

$$c_{s_{i+1}} = |L_i \mathbf{s}_i| c_{s_i} - e_{s_{i+1}}. \tag{11}$$

The initial choices for c_{s_0} and c_{u_S} are arbitrary as long as they are small—smaller than the maximum shadowing distance—because (11) damps initial conditions and (10) damps final conditions. GHYS and QT choose them both as 0, as do we for simplicity. This choice is probably as good as any, but it can be seen here that if one shadow exists, there are infinitely many of them. Another way of looking at these initial choices for c_{s_0} and c_{u_S} is that they "pinch" the growing components at the endpoint, and the backward-growing components at the initial point, to be small. That is, [illegible] are being forced on the problem so that the exponential divergence is forcibly masked, if possible, making the solution of (5) numerically stable.

Note that these boundary conditions demand that the initial condition for the shadow and noisy orbits differ along the unstable direction. In fact, this [illegible] occur if the change in initial conditions is to have any effect. That is, when looking for a shadow, if perturbations were allowed only in the stable direction, those perturbations would die out, leading the perturbed exact orbit to follow the original exact orbit that passes through the initial conditions—the one that is already known to diverge exponentially from the noisy orbit.

Algorithm 1 gives pseudocode for the basic 2-dimensional refinement algorithm.

ALGORITHM 1. Basic two-dimensional refinement algorithm.

Input: noisy orbit $\mathbf{Y} = \{\mathbf{y}_i\}_{i=0}^{S}$.
loop
  estimate 1-step errors $\mathbf{e}_i = \mathbf{y}_i - \varphi(\mathbf{y}_{i-1})$ for each step $i = 1, \ldots, S$.
  exit when 1-step errors either ''blow up'' or stop changing between outer loops.
  compute all resolvent matrices L_i by solving the variational equation for each timestep i.
  compute all unstable and stable vectors $\mathbf{u}_i$, $\mathbf{s}_i$ by iterating (7) and (8), respectively.
  split $\mathbf{e}_i$ into stable and unstable components via (6).
  compute split corrections using (10) and (11).
  recombine correction terms $\mathbf{c}_i$ via equation (6).
  apply corrections to $\mathbf{Y}$: $\mathbf{y}_i := \mathbf{y}_i + \mathbf{c}_i$ for each timestep $i = 0, \ldots, S$.
end loop

2.3. QT's generalization to arbitrary Hamiltonian systems. If the configuration space is d-dimensional, then there are $2d$ dimensions in the phase space. It can be shown that in a Hamiltonian system, the number l of stable and unstable directions is equal, although $l < d$ is possible. At time t_i, let $\{\mathbf{u}_i^j\}_{j=1}^{d}$ represent d unstable unit vectors, and let $\{\mathbf{s}_i^j\}_{j=1}^{d}$ represent d stable unit vectors. For any particular timestep, it will be convenient if the unstable vectors are orthonormal to each other and the stable vectors are orthonormal to each other. However, the stable and unstable vectors together will not in general form an orthogonal system.

The vectors are evolved as in the 2-dimensional case, except using modified Gram–Schmidt orthonormalization to produce two sets of d-orthonormal vectors at each timestep. Arbitrary orthonormal bases are chosen, $\mathbf{u}_0^j$ at time t_0, and $\mathbf{s}_S^j$ at time t_S, and then evolved according to

$$\bar{\mathbf{u}}_{i+1}^j = L_i \mathbf{u}_i^j, \qquad \bar{\mathbf{s}}_i^j = L_i^{-1} \mathbf{s}_{i+1}^j.$$

At each step i, two modified Gram–Schmidt orthonormalizations are done: one on $\{\bar{\mathbf{u}}_{i+1}^j\}_{j=1}^d$ to produce $\{\mathbf{u}_{i+1}^j\}_{j=1}^d$, and another on $\{\bar{\mathbf{s}}_i^j\}_{j=1}^d$ to produce $\{\mathbf{s}_i^j\}_{j=1}^d$. After a few Lyapunov times, $\mathbf{u}_i^1$ lies close to the most unstable direction at step i, $\mathbf{u}_i^2$ lies close to the second most unstable direction, etc. Likewise, $\mathbf{s}_i^1$ lies close to the most stable direction at time i, $\mathbf{s}_i^2$ lies close to the second most stable direction, etc.

The multidimensional generalization of (6) is the obvious

$$\mathbf{e}_i = \sum_{j=1}^{d} (e_{u_i^j} \mathbf{u}_i^j + e_{s_i^j} \mathbf{s}_i^j), \qquad \mathbf{c}_i = \sum_{j=1}^{d} (c_{u_i^j} \mathbf{u}_i^j + c_{s_i^j} \mathbf{s}_i^j). \tag{12}$$

To convert the 1-step error at step i from phase-space coordinates $\mathbf{e}_i' = \{e_i^k\}_{k=1}^{2d}$ to the stable and unstable basis $\mathbf{e}_i = \{\{e_{u_i^j}\}_{j=1}^d, \{e_{s_i^j}\}_{j=1}^d\}$, one constructs the matrix V_i whose columns are the unstable and stable unit vectors, $V_i = (\mathbf{u}_i^1 \mathbf{u}_i^2 \cdots \mathbf{u}_i^d \mathbf{s}_i^1 \mathbf{s}_i^2 \cdots \mathbf{s}_i^d)$, and solves the system $V_i \mathbf{e}_i = \mathbf{e}_i'$.

From (5) and (12) the equation for the correction coefficients in the unstable subspace at step $i+1$ is

$$\sum_{j=1}^{d} (c_{u_{i+1}^j} + e_{u_{i+1}^j}) \mathbf{u}_{i+1}^j = \sum_{j=1}^{d} c_{u_i^j} L_i \mathbf{u}_i^j,$$

which is projected out along $\mathbf{u}_{i+1}^k$ producing

$$c_{u_{i+1}^k} + e_{u_{i+1}^k} = \sum_{j=k}^{d} c_{u_i^j} U_i^{kj},$$

where the scalar $U_i^{kj} = \mathbf{u}_{i+1}^k \cdot L_i \mathbf{u}_i^j = \mathbf{u}_{i+1}^k \cdot \bar{\mathbf{u}}_{i+1}^j$ and the modified Gram–Schmidt process ensures that $U_i^{kj} = 0$ if $j < k$. As stated previously, the the boundary condition requires that the unstable component of the corrections be small at timestep S and their stable components be small at timestep 0. For simplicity we take $\{c_{u_S^j} = 0\}_{j=1}^d$ as did QT, and the coefficients are computed backward using

$$c_{u_i^k} = \frac{1}{U_i^{kk}} \left(c_{u_{i+1}^k} + e_{u_{i+1}^k} - \sum_{j=k+1}^{d} c_{u_i^j} U_i^{kj} \right).$$

We first solve for $\{c_{u_i^d}\}_{i=0}^S$, which does not require knowledge of the other c_u coefficients, and then solve for $\{c_{u_i^{d-1}}\}_{i=0}^S$, etc.

Again from (5) and (12), the equation for the correction coefficients in the stable subspace at timestep i is

$$\sum_{j=1}^{d} (c_{s_{i+1}^j} + e_{s_{i+1}^j}) L_i^{-1} \mathbf{s}_{i+1}^j = \sum_{j=1}^{d} c_{s_i^j} \mathbf{s}_i^j,$$

which is projected out along $\mathbf{s}_i^k$ producing

$$c_{s_i^k} = \sum_{j=k}^{d} (c_{s_{i+1}^j} + e_{s_{i+1}^j}) S_i^{kj},$$

where $S_i^{kj} = \mathbf{s}_i^k \cdot L_i^{-1}\mathbf{s}_{i+1}^j = \mathbf{s}_i^k \cdot \bar{\mathbf{s}}_i^j$ and the modified Gram–Schmidt process ensures that $S_i^{kj} = 0$ if $j < k$. The boundary condition at step 0 is $\{c_{s_0^j} = 0\}_{j=1}^d$, and the co-efficients are computed forward using

$$c_{s_{i+1}^k} = \frac{1}{S_i^{kk}} \left(c_{s_i^k} - \sum_{j=k+1}^{d} (c_{s_{i+1}^j} + e_{s_{i+1}^j}) S_i^{kj} \right) - e_{s_{i+1}^k}.$$

As with the unstable corrections, we first compute $\{c_{s_i^d}\}_{i=0}^S$, which does not require knowledge of the other c_s coefficients, and then we compute $\{c_{s_i^{d-1}}\}$, etc.

Applying the above computed corrections to the noisy orbit, producing a nearby orbit with (ideally) less noise, concludes one refinement iteration.

3. Optimizations to the GHYS/QT refinement algorithm.

3.1. Asymptotic runtimes. Most of the expense of refinement of an N-body orbit is in computing the resolvents L_i. We tested our algorithm on an artificial system of N particles, only M of which move. Manually counting floating-point operations in the N-body Jacobian gives an operation count of roughly $60MN + 50M^2$. The total operation count to compute the right-hand side of the variational equation, including the matrix-matrix multiply and the computation of the Jacobian, is about $110M^3 + 70M^2 + 85MN$. The M^2 term is probably negligible, and the MN term is only relevant if $N \gg M$. Thus, if there are S shadow steps, and the function computing the right-hand side is invoked an average of k times per shadow step, the total cost of computing all the resolvents is about $k \times S \times (110M^3 + 85MN)$. We found empirically that, using an Adams's integrator [34], k seems to be independent of M and roughly equal to 500 for the shadow steps used in our N-body simulations.

A straightforward, unoptimized implementation of the high-dimensional QT algorithm can take a prohibitively long time to find shadows of nontrivial duration. Furthermore, if the dimension of the problem is d, then the time scales as $O(d^3)$, making it virtually impossible to study shadows of high-dimensional systems. However, a vital scientific question is how shadow lengths scale with the dimensionality of the system.

3.2. Brief overview of the optimizations. This paper introduces eight major optimizations to the GHYS/QT refinement procedure. Each provides a constant speedup of roughly between 1.5- and 8-fold, with a cumulative average speedup of about a factor of 100. We briefly introduce all eight optimizations here. The first three are trivial and are not discussed further, while the remaining five are discussed in more detail in the following subsections. Most of the effort is focused on reducing both the number of resolvents to recompute and the time spent recomputing each. This necessitates handling the ensuing complications of when a resolvent [illegible] be recomputed and when it would be worthless to do so. Furthermore, each of the optimizations is independent of all the others. It is conceivable that some optimizations may significantly decrease the reliability of refinement for some systems, although this was not observed for the systems we tested.

Each optimization (except the first) has a name and a single-letter identification for later discussion. We use the acronym [illegible] to mean the set of all [illegible]esolvents, [illegible]nstable, and [illegible]table vectors for the entire trajectory. The RUS at a particular timestep i is referred to as RUS_i.

[illegible] $3M$ [illegible] QT recomputed a new resolvent for each stable and unstable vector; i.e., at each timestep they computed $6M$ resolvents (this is obvious only from

viewing their code). However, a resolvent $R(t_1, t_0)$ is a matrix operator that applies to small perturbations of the orbit from time t_0 to t_1. Thus only two resolvents per timestep need to be computed: one for evolving perturbations forward in time, and another to evolve them backward. Thus QT would have been required to perform $O(M^4)$ work for M moving particles. Since most of the time is spent computing the resolvents, fixing this oversight provides an immediate speedup of a factor of about $3M$. The effect of fixing this oversight is included in the speedup values of Table 1, but it partly explains why QT thought that shadowing is wholly impractical for large N systems.

The above optimization can be extended further by noting that the resolvent matrix L_i that maps perturbations from time t_i to t_{i+1} is precisely the inverse matrix of the resolvent that maps small perturbations from t_{i+1} to t_i. Thus, rather than doing an integration from t_{i+1} backward to t_i to compute L_i^{-1}, L_i can be inverted using matrix operations, or the system $L_i\mathbf{x} = \mathbf{b}$ can be solved, whenever necessary, using standard elimination schemes. This gives a speedup of about 2.

Our experiments show that for smooth problems like the N-body problem, the Jacobian need not be extremely accurate, even in late refinements where the 1-step errors approach the machine precision. In these N-body experiments, when this optimization was used, resolvents were always computed to a fixed tolerance of 10^{-6}. We found that the movement of the orbit during refinement dictates recomputation of a resolvent, not the numerical error in its construction.

When computing the resolvent and 1-step errors during early refinements, it is not necessary to compute them to extremely high accuracy. Since the initial 1-step errors may have magnitudes of, say, 10^{-4}, the resolvents and 1-step errors need only be computed to a tolerance of, say, 10^{-7}, i.e., three extra digits. QT always computed the 1-step errors and resolvents to a tolerance of 10^{-13}. Of course, if the optimization is used, this only affects the 1-step error tolerances.

If a shadow exists, then by definition it cannot be too far away from the original noisy orbit. It is reasonable to assume that sometimes the RUS will change little as the orbit is perturbed towards the shadow. Thus, it may not be necessary to recompute the RUS for every refinement iteration. This is probably the biggest savings, because in combination with the cheaper accurate integrator, it means that the RUS often needs only to be computed once to a lax tolerance. This is analogous to a Newton's method that does not recompute the Jacobian at every iteration.

To find the longest possible shadow of a noisy orbit, we attempt to shadow for longer and longer times until shadowing fails. Assume that shadowing for S shadow steps produces a shadow A. When attempting to shadow $S + k$ timesteps for some integer k, assume that a successful shadow B will be found. Since A and the first S steps of B both shadow the same noisy orbit, they must be close to one another. By the same argument as in the previous paragraph, the RUS that was computed for A can probably be reused for the first S steps of B. Thus only k new RUS_i's need to be computed.

If it can be shown that a particular RUS_i, or a small set of them, is causing Constant RUS refinements to fail, then perhaps it is sufficient only to recompute the "bad" RUS_i's, rather than all of them along the entire orbit. It is not trivial to decide which ones need recomputing, though.

QT used every internal timestep of their integrator's noisy orbit as a shadow step, but this is not necessary. It is reasonable to skip timesteps in the original orbit to build larger shadow steps. This means that fewer resolvents and stable/unstable directions need to be computed and stored. This optimization alone produced speedups of more than an order of magnitude in some cases.

3.3. Constant RUS directions ($\boldsymbol{R}$). For a smooth, continuous, well-behaved function, the Jacobian is also smooth and continuous. Most Hamiltonian systems (that we have seen) are at least piecewise well-behaved in this sense. The N-body problem, in particular, is of this class, assuming that no 0-distance collisions occur, or if the forces are artificially bounded through "softening." This implies that the resolvent along a particular path will not change much if the path is perturbed slightly. Since the unstable and stable unit vectors are computed solely from the resolvents, they will also change only slightly when the path is perturbed. Thus, the RUS computed on one refinement iteration may be reusable on some subsequent iterations. Computing the new 1-step errors is then the only significant work to be performed on each refinement iteration after the first.

We have observed that computing the RUS to a tolerance of between 10^{-6} and 10^{-9} usually suffices to refine the orbit down to 1-step errors near 10^{-15}. If refinement fails using Constant RUS, then perhaps there is a spot in the orbit where some RUS_i's change quickly with small perturbations in the orbit. In that case, the RUS can be recomputed for a refinement or two, after which Constant RUS may be tried again. However, when Constant RUS fails it often suggests that there is a glitch somewhere in the portion of the orbit which is currently being tested for a shadow.

3.3.1. Switching between constant and recomputed RUS. To switch between Constant RUS and Recomputed RUS, we use a running-average "memory" scheme to track the progress of the norm of the maximum 1-step error over the previous few refinements. We originally used a simple algorithm [40] that signals failure after K successive failed refinements, for $\mu = 0.9$ (see (3)) and $K = 3$. However, we found that refinement would sometimes get into a "loop" in which a few successful refinements would be followed by an unsuccessful one that would "undo" the previous improvements. Thus, refinement would be at a standstill even though there would never be more than one consecutive "bad" iteration, but setting $K = 1$ was too stringent, because it aggressively stops refinements that would eventually succeed.

The [illegible] is defined by

$$A_{j+1} = \alpha A_j + (1-\alpha)\nu, \qquad \alpha \in (0,1), \tag{13}$$

where ν is the newest element added to the set and α is the [illegible] constant. The higher the memory constant, the longer the memory—i.e., the less the effect of each new element; A is roughly an average of the most recent $2/(1-\alpha)$ elements. However, (13) is not suited for measuring errors such as those in refinement that can change by many orders of magnitude, and is especially unsuited when smaller means better. A [illegible] equivalent of (13) allows values differing by orders of magnitude to average meaningfully. We define the [illegible] G as

$$G_{j+1} = \sqrt[n+1]{(G_j)^n \nu}, \tag{14}$$

where ν is the new element and $\frac{n}{n+1}$ is analogous to α in (13), so higher N implies longer memory. Both (13) and (14) require initialization to some reasonable starting value A_0 and G_0, respectively. N should not be too large; we found 2 or 3 worked best.

Finally, we want to measure the [illegible] that is being made by successive refinements, rather than the absolute error, because it is the [illegible] in 1-step errors that indicates whether current refinements are succeeding, not their absolute size. Note that the 1-step errors may stop decreasing for two reasons: (a) 1-step errors have gotten as close to machine precision as is possible, or (b) refinement is failing to find a shadow. Using the improvement rather than the absolute value allows us to use the same algorithm to halt refinement in either case. We use the geometric running average (14) of the improvement to measure the progress of refinement, where the improvement I is

$$I = \left| \frac{\text{the maximum 1-step error of this refinement}}{\text{maximum 1-step error of previous refinement}} \right|.$$

If I is close to or greater than 1, then the current refinement did not improve on the previous one; if $I < \mu$ (μ from (3)), then the current refinement is successful.

We use the improvement I to switch between methods of various expense as follows. In general, we continue to use cheap refinement until improvement stops, at which point we either give up or switch to a more expensive refinement method. We currently have only two refinement methods. In order of decreasing cost, they are Recomputed RUS and Constant RUS. Below is the heuristic we have created using much trial and error. Let μ_R be the success value (from (3)) when the RUS is being recomputed, and μ_C the success value when using Constant RUS. We used $\mu_R = 0.1$, $\mu_C = 0.5$, because we expect a Constant RUS iteration not to be as successful as one that recomputes the RUS, but they are much cheaper, so we can afford a lower expected improvement per Constant RUS refinement.

- When using Recomputed RUS, it is safe to switch to Constant RUS when the geometric running average improvement becomes $< \mu_R$. At this time the current orbit and all its statistics must be saved in case Constant RUS refinement fails.
- When using Constant RUS, it is necessary to switch back to Recomputed RUS when the geometric running average improvement stays $> \mu_C$ for three successive refinements. We then discard all progress made by Constant RUS and revert to the previous orbit that used Recomputed RUS. We believe it is advantageous to discard all progress made by Constant RUS, even if significant progress was made, for the following reasons:
  - A refinement that computes the RUS is far more expensive than one that does not, so discarding all progress made by Constant RUS refinements usually makes little difference, percentagewise, in the final runtime.
  - Future Constant RUS refinements, which will be performed after the RUS is recomputed, will converge faster than the current ones that just failed, because the RUS will be more accurate.
  - Recomputing the RUS with small 1-step errors would mean, using the simple "three extra digits of accuracy" heuristic, recomputing it at a much higher accuracy than is necessary. (Unless the [illegible] optimization is used.)
- Finally, after having switched back to Recomputed RUS, it is time to admit failure if the geometric running average improvement stays $> \mu_R$ for three successive refinements; and it is safe to switch again to Constant RUS when the geometric running average improvement becomes less than μ_R.

We found that there are usually no half-measures when using recomputed RUS—refinement either succeeds geometrically or fails miserably. Finally, there may be

a place for other noise-reduction procedures in the universe of expensive and cheap refinement algorithms.

3.4. Reuse RUS from a previous successful shadow (P). This idea is based on the same observation as Constant RUS, and is useful only if Constant RUS is employed. It takes advantage of the proximity of an RUS for an orbit B to the RUS for a nearby orbit A. So, when trying to find a shadow for a segment B which is an extension to segment A, the RUS of A can be reused on the segment of B that overlaps A.

One interesting question is which of A's RUS's to use if A's RUS was ever recomputed. An argument in favor of using the first is that the noisy orbit is exactly the same, since B is just an extension of A. However, if A had to recompute the RUS, then probably so will B if A's first RUS is used. Recall that the RUS needed for the early refinements need not be as accurate as the ones used later. Furthermore, we assume that the segment of B's shadow that overlaps A's shadow will be [illegible] to A's shadow than the first few iterations of B is to the first few iterations of A. Thus, if we use the last RUS that A used, we are less likely to recompute the RUS for B. If B is a [illegible] of A rather than an extension, then obviously no new RUS_i's need to be computed.

3.5. Recompute specific RUS_i's (i). When Constant RUS refinements fail, it may be because some of the RUS_i's are invalid due to the current trajectory drifting too far from the trajectory for which the RUS_i was computed. If the set of RUS_i's that are invalid is small, then it would be advantageous to recompute only those RUS_i's that are causing the failure.

A difficulty arises, however, when trying to pinpoint which RUS_i's are causing the problem. Note that refinement using the GHYS/QT algorithm is a [illegible] method: the correction $\mathbf{c}_j$ is a function of the resolvents and 1-step errors of [illegible] the timesteps, not just L_j and $\mathbf{e}_j$. Thus, it is not trivial to decide which RUS_i's are causing a particular $\mathbf{c}_j$ to be invalid.

What we would like is some measure M_i of how bad an RUS_i is. Then we can insert RUS_i's into a priority queue based on M_i, extract the RUS_i's that are worse than some criteria, and recompute them. Some obvious measures for M_i are the following:

- the 1-step error at step i,
- the distance $\mathbf{y}_i$ has moved since the last time its RUS_i was computed,
- the norm $\|L_i\|$, which is a measure of how sensitive that area of the orbit is to small changes.

Once the M_i's are ordered by some measure, a criterion is needed to decide where to put the dividing line between those RUS_i's that need recomputing and those that don't. Some obvious criteria are the following:

- recompute all RUS_i's for which $M_i > M'$ for some constant M',
- recompute some constant number of the RUS_i's,
- recompute some constant fraction of the RUS_i's,
- recompute all whose $M_i > w(\bar{M})$ for some function w of the geometric mean measure $\bar{M}$.

None of these measures taken alone with any of the above criteria taken alone seem sufficient, since there were cases in which recomputing the entire RUS resulted in a successful refinement, but a single-measure–single-criterion method for recomputing particular RUS_i's resulted in a long sequence of unsuccessful refinements.

After much trial and error, a measure was found which gives a speedup no better than a factor of 2, which is much less than we expected. Let us focus on a particular

timestep i. Let k be the current refinement iteration, and k' be the refinement at which the RUS_i was last computed. Let $L_i^{k'}$ be the resolvent for step i computed during refinement k'. Let $\mathbf{y}_i^{k'}$ be the orbit point at step i from refinement k', and let $\mathbf{y}_i^k$ be the orbit point at step i for the current refinement k. When trying to decide if we need to compute L_i^k, the measure we devised is

$$M_i = \|L_i^{k'}(\mathbf{y}_i^{k'} - \mathbf{y}_i^k)\|.$$

The intent is to combine the effects of the norm of L_i and how much the orbit has moved at that point since L_i was last computed. L_i^k is recomputed if it satisfies [illegible] of the following criteria:

- always recompute the worst K RUS_i's, for some integer K;
- always recompute the worst F RUS_i's, for some fraction F of all of them;
- always recompute RUS_i if $M_i > M'$ for some constant M';
- for each RUS_i satisfying one of the above, also recompute RUS_{i-1}, since L_i starts at $\mathbf{y}_i$, while L_{i-1} ends at $\mathbf{y}_i$.

We used $K = 1$, $F = \frac{1}{8}$, and $M' = 10^{-3}$.

Unfortunately, this set of measures and criteria eventually recomputes most of the RUS_i's except in the final few refinements of a failing search. In other words, this set of criteria gives the best speedups only during shadow searches that will eventually fail to find a shadow. This is a useful speedup, because a large fraction of the time spent trying to find the longest shadow is spent in the failing search for a shadow longer than the longest eventually found. However, it would be better if some criterion could be found that also aids in searches that are eventually successful. It seems plausible that some such measure and criterion should exist. Since the benefits are potentially enormous for orbits with many timesteps, more study seems appropriate.

3.6. Large shadow steps (L). A numerical solution of an ODE is represented by an ordered, discrete sequence of points. Assume that we construct a continuous solution $\mathbf{y}(t)$ from these points, for example by using a suitably smooth and accurate interpolant. Then we could extend the definition of ε-shadowing to continuous systems as follows: if $\mathbf{x}(t)$ is an exact solution, then $\mathbf{x}(t)$ ε-shadows $\mathbf{y}(t)$ on $t_0 \le t \le t_1$ if $\forall t \in [t_0, t_1],\ \|\mathbf{x}(t) - \mathbf{y}(t)\| < \varepsilon$.

Now, it should be clear that we can choose [illegible] representative sequence of points along $\mathbf{y}(t)$ to use in the refinement algorithm; we need not choose the points from which $\mathbf{y}(t)$ was originally constructed. In particular, we can choose a subset of the original set of points, as long as enough points are chosen to be "representative" of $\mathbf{y}(t)$. The steps that are finally chosen are called [illegible].

There are at least two reasons to desire as few shadow steps as possible. First, if Constant RUS is being used, then the RUS needs to be stored. Each RUS_i requires a matrix (the resolvent), and two sets of basis vectors.[1] Second, if the "accurate" integrator has a large startup cost, then we wish to minimize the number of startups. For example, the Adams's method that we used takes extremely small internal timesteps at the beginning of each shadow step.

For the N-body simulations reported in this paper, a set of "standardized" units [33] was used such that the scale of the system in all macroscopic units of interest was of order 1—i.e., the system had a diameter of order 1, the crossing time and total

[1]For example, with $M = 25$ moving particles, an RUS of length $S = 128$ requires about 100 megabytes of memory, and this space scales as $O(SM^2)$. ((25 particles × 6 dimensions per particle)^2× 8 bytes per double precision number × 4 (2 resolvents, 2 sets of basis vectors each covering half the space) × 128 steps).

energy were of order 1, and the N particles each had mass $1/N$. In such a system, the timesteps of QT's integrator averaged about 0.01, and they used each timestep as a shadow step. We found empirically that, in this system, using shadow steps about ten times longer, of size 0.1, works well. Smaller shadow steps use more time and memory but could not find shadows any better; shadow steps of 0.2 were slightly less reliable, and steps of size 0.5 were unreliable. A significant area of further work would be to dynamically determine in an algorithmic fashion what size shadow step is appropriate in the general case.

It might be reasonable to assume that as refinement proceeds to decrease the 1-step errors, it would be advantageous to construct larger-and-larger shadow steps. However, we found that this does not give much of a speedup, because (1) if refinement is working well, then Constant RUS will be invoked, in which case there is no need to recompute the RUS at all; (2) conversely, if refinement is failing, then there is no justification for increasing the size of shadow steps—in fact, it may be advantageous to decrease them. Note that, if refinement is succeeding, it may be advantageous to use larger-and-larger shadow steps in the computation of the 1-step errors, even though the RUS is not being recomputed. We did not do this.

Finally, implementing dynamic shadow step sizes may help in another area. As described above, we currently discard all progress made by Constant RUS when it begins to fail, reverting to the orbit and RUS of the most recent refinement that computed the RUS. However, if Constant RUS works for several refinements, but then starts to fail, perhaps it would be advantageous to use the orbit of the most recent successful refinement to build new shadow steps using the above dynamic algorithm, regardless of whether the most recent successful refinement computed the RUS or not.

3.7. Cheaper accurate integrator (C). When computing the 1-step errors of a noisy orbit, an integrator must be used that has smaller 1-step errors than the integrator used to compute the noisy orbit. The question is, how much more accurate does it need to be? During early refinements when the 1-step errors are large, the errors in the correction factors may be dominated by the first-order approximation of the resolvents, so the 1-step errors do not need to be computed to machine precision. In practice, we found that it is sufficient to estimate the 1-step errors of the noisy trajectory to an accuracy of 10^{-3} times that of the size of the maximum 1-step error of the previous refinement; computing them any more accurately does not speed refinement, although computing them only 10^{-2} times more accurately sometimes results in more iterations of the refinement procedure.[2] In addition, a factor of only 10^{-2} is not enough because some integrators have sufficiently gross control of their errors that requesting only two extra digits will sometimes result in the integrator returning the same result. This is because it did too much work in the less accurate case, so that the solution it computed had at least two more accurate digits than were requested. This is less likely to happen with a difference of 10^{-3}. Note that it is necessary to loosen this criterion when the 1-step errors are within 10^{-3} of the machine precision, in order not to request more accuracy than the machine precision allows.

We have tested several different integrators as "accurate", including a 7/8 order Runge–Kutta pair [23], two Adams's methods [34, 35], and a Bulirsch–Stöer method [39]. Although all integrators gave roughly the same reliability results (that is, the shadows produced did not change significantly based upon which accurate integrator was chosen), we found that LSODE [34] was most efficient. However, it would not be

[2]More iterations, but each iteration takes less time. There is clearly a trade-off here that is probably problem-dependent.

surprising if efficiency were a function of the problem being integrated. In particular, if the shadow steps are small, it may pay to use a less sophisticated routine, such as a high-order explicit Runge–Kutta method. If the shadow steps are large, an Adams or a Bulirsch–Stöer method is probably apt, because even though they both have high startup cost, they can eventually move quickly along the solution if they are not restarted too often. It may also be interesting to attempt using extended precision integration to test the accuracy of the standard (double precision) routines.

Finally, an important factor in problems, like the N-body problem, that have a large variance of scales all occurring at once (i.e., some pairs of stars are several orders of magnitude closer to each other than other pairs) is a concept called [illegible]. The modern hand-coded N-body integrator gives each star a personalized integration timestep; stars that have high accelerations are integrated with smaller timesteps than those with lower accelerations. Perhaps it would be fruitful to attempt shadowing with an integration routine that uses individual timesteps for each star, because currently the timestep of the entire system is restricted to the smallest required timestep.

4. Results on a test case: The gravitational *N*-body problem. To quantify the speedups of the optimizations, we did timing tests while applying the algorithm to trajectories of a simplified gravitational n-body system. The system consists of 100 particles, 99 of which are fixed in 3-dimensional space [40, 31, 30]. Units are scaled so that the gravitational constant is 1, the physical extent of the system and the total mass are both unity, while one particle moves "pinball" fashion among the fixed particles with a velocity of order unity. For our simplified system, the moving particle feels an acceleration $\mathbf{a}$ arising from Newtonian gravitational forces from all the other particles, according to

$$\mathbf{a} = \sum_{j=1}^{99} m_j \hat{\mathbf{r}}_j / r_j^2,$$

where m_j is the mass of particle j, $\hat{\mathbf{r}}_j$ is a unit vector lying along the line between the particle j and the moving particle, and r_j is the scalar distance between the fixed particle j and the moving particle. Thirty-two initial conditions were chosen at random. Each was shadowed using several different combinations of the optimizations. The 32 orbits were chosen by generating random 3-dimensional positions for all particles from the uniform distribution on $(0, 1)$; a 3-dimensional random initial velocity for the moving particle was also chosen uniformly on $(0, 1)$. The standardized units of Heggie and Mathieu [33] were used, in which each particle has mass $1/N$. The pseudorandom number generator was the popular Unix 48-bit `drand48()`, with seeds 1 through 32.

Once the initial conditions were set, each noisy orbit was generated by integrating for 1.28 standard time units (about 1 crossing time) with local errors of about 10^{-6}. This figure agreed well with the magnitude of the initial 1-step errors computed during the first refinement. Although one crossing time sounds short, it is long enough that five of the orbits contained trouble spots.[3] The number of unoptimized QT refinements required to find a shadow that has no trouble spots seems independent of the length of the orbit—each refinement takes much longer, but the convergence is still geometric per refinement. Thus, the timing results below should be independent

[3]If the orbit has a trouble spot, then Constant resolvent and Reuse RUS from previous successful shadow will have little effect, because the RUS will be recomputed in an attempt to find a shadow. The other optimizations—C, L, F, and I—still offer significant performance improvement.

of the length of the orbit. This is what has been observed with the longer orbits we have shadowed, although we do not document them here.

For short shadow steps, a constant shadow step size of 0.01 was used, which approximates the average-sized shadow step in QT. This results in 128 shadow steps. For large shadow steps, 16 shadow steps of size 0.08 were used. As in a usual shadowing attempt, longer and longer segments are shadowed in an attempt to find the longest shadow. Here, each successive segment had twice as many shadow steps as the previous one, up to 16 and 128, for long and short shadow steps, respectively. No attempt was made to isolate glitches more accurately than this factor of two.

The most stringent tolerance requested of the accurate integrator was 10^{-15}. A successful numerical shadow was defined to be one whose 1-step errors were all less than 2×10^{-14}. This number was chosen simply because it was the smallest number that geometrically converging refinement sequences could consistently achieve; 10^{-14} was too small, because many refinement sequences would proceed geometrically until their maximum 1-step error was about 1.5×10^{-14}, and then they would "bounce around" for many refinements until, apparently by chance, the maximum 1-step error would fall below 10^{-14}. The maximum allowed shadow distance was 0.1, although none over 0.0066 were observed with successful shadows.

Table 1 displays the speedups of the various optimizations. All ratios are speedup factors with respect to the time column, which was originally timed on a Sun Sparcstation. However, similar total speedups were recorded (under less rigorous conditions in which only the fully optimized versus unoptimized versions were tested) for many platforms, including Sun Sparcstations using both Sun's C compiler and GCC, Intel machines running Microsoft Windows or Linux, Silicon Graphics machines, a Cray Y-MP, and a Fujitsu VPX vector supercomputer. There are several interesting observations to make about this table. First, note that the run times of the original QT algorithm (after correction of the 3M oversight) are comparable to the run times of our unoptimized version (differing on average by a factor of 1.07), as would be expected. Now looking at each optimization acting alone, we see that using large shadow steps (column L) 8 times longer gives an average speedup of about 5.5. Large shadow steps are trivial to implement, so we would recommend that, even if no other optimizations are adopted, large shadow steps be used in any new shadowing implementation. Third, inverting the forward resolvent to produce the inverse resolvent (column I) produces the expected average speedup of 2.0. Fourth, the cheap accurate integrator (column C) gives an average speedup of 2.36. This is about what is expected, because both the QT and C algorithms on average require just over three refinements to converge, and all C refinements are cheaper than a QT refinement except the last one, which is about equal in expense. Finally, using Constant RUS gives a speedup of almost 3. Again this is about what is expected, because QT requires about three refinements to converge, while R has one RUS computation followed by several cheap Constant RUS refinements. The P, F, and i optimizations were not tested alone.

Next we look at combinations of optimizations. First, it is interesting to note that combining the cheap accurate integrator C with Constant RUS R results in an average speedup that is greater than the product of the two individual speedups ($2.36 \times 2.68 = 6.32 < 7.55$). This is because, when using CR, only $'$ $'$ RUS is computed, and it is computed cheaply. Using R alone requires computing the one RUS to high accuracy; using C alone requires computing at least one RUS (the final one) to high accuracy. Second, reusing the RUS of a previous successful shadow (P) makes sense only when R is also used. This gives an average speedup of 43% over CR. The next three columns show other combinations. An interesting point to note

TABLE 1

Speedups of the optimizations. "Time" represents unoptimized version. For other abbreviations, see text.

Seed	Time(s)	QT	L	I	C	R	CR	CPR	CLR	KR	KRI	KRIF	KRIFi
1	1442	1.06	4.54	2.01	1.96	3.00	8.87	14.2	30.8	37.2	47.4	53.53	57.37
2	9379	0.70	6.08	1.82	3.09	3.48	11.2	15.7	52.3	68.3	75.7	79.83	79.67
3	7847	0.72	6.30	1.88	2.36	2.56	9.29	13.0	44.0	59.7	65.5	87.61	86.78
4	7011	0.86	11.1	1.98	2.74	2.84	7.88	12.3	40.5	55.8	74.3	79.19	78.52
5	12204	0.93	12.2	2.13	2.42	1.08	1.66	1.54	5.31	4.96	8.0	22.14	37.55
6	9443	1.14	7.24	2.01	3.18	3.98	10.8	13.5	5.36	56.8	29.8	80.13	137.64
7	6277	1.13	5.51	1.85	2.63	2.71	8.50	11.6	46.9	63.7	82.9	91.60	90.58
8	6582	1.13	4.79	2.04	1.91	2.79	7.09	10.6	33.8	45.4	56.9	72.76	67.95
9	7154	1.14	4.33	1.92	1.96	2.70	7.74	11.5	30.5	39.6	51.9	54.12	53.78
10	5832	1.13	4.32	1.87	1.54	2.72	6.46	10.7	29.1	45.5	57.2	63.89	63.59
11	6659	1.13	5.10	1.93	2.34	2.65	7.40	10.7	37.1	50.7	62.9	67.15	62.76
12	8623	0.86	5.00	2.44	2.57	3.37	9.64	13.7	44.8	52.4	75.4	79.77	79.19
13	14566	0.45	4.34	1.00	4.52	0.78	0.93	0.93	5.67	5.15	7.6	23.63	35.86
14	6976	0.74	4.54	2.00	2.17	2.85	7.23	11.1	32.9	45.2	49.4	58.08	57.66
15	5664	0.72	4.79	2.02	1.89	2.71	6.83	10.2	33.5	45.4	62.9	67.03	66.43
16	6634	1.15	4.89	1.97	1.82	2.64	7.35	10.6	33.8	44.8	58.7	62.65	62.08
17	6854	0.94	4.96	2.06	2.16	2.88	8.97	12.7	47.2	60.8	83.2	89.45	88.57
18	4932	0.62	2.88	1.74	1.69	2.17	5.38	7.84	25.7	34.1	45.7	49.70	49.27
19	8470	1.06	5.16	2.98	2.96	1.32	4.37	4.02	31.1	27.5	53.5	64.21	151.73
20	4566	1.12	5.59	1.91	2.04	2.78	6.21	9.50	33.7	48.6	64.3	62.95	66.32
21	7409	1.07	6.01	2.00	2.92	2.86	9.59	13.4	46.6	61.3	82.2	84.44	86.69
22	6988	1.22	4.89	2.17	2.00	3.03	7.34	11.4	36.1	44.0	57.2	61.00	60.52
23	7699	0.75	4.66	1.93	1.91	3.20	8.27	10.4	35.2	46.3	51.1	62.82	62.32
24	8851	1.18	2.53	1.79	2.78	2.89	10.2	15.4	52.9	71.1	93.1	95.29	100.39
25	7287	1.13	3.58	1.96	2.67	2.53	7.58	11.5	34.1	47.4	59.3	62.81	62.34
26	10736	0.85	7.74	2.84	2.79	4.33	12.4	16.9	58.2	76.5	81.1	63.96	63.50
27	7001	1.13	5.07	2.01	2.24	2.79	8.35	12.1	38.9	54.5	70.1	76.20	75.58
28	7324	1.13	4.49	2.06	1.93	2.72	8.01	11.2	33.2	49.3	55.0	58.98	55.21
29	13779	0.80	4.01	1.89	2.01	0.89	1.16	1.15	4.03	3.99	6.5	17.00	29.02
30	7932	1.14	10.9	2.01	2.17	2.81	8.82	13.0	43.3	67.5	86.7	91.38	90.70
31	6989	1.20	5.57	2.26	2.37	2.81	8.44	12.3	47.4	57.2	77.6	84.06	83.29
32	6720	1.08	4.80	1.89	1.90	2.78	7.61	11.5	34.0	50.5	65.5	70.33	69.67
Avg	7682	1.07	5.56	2.01	2.36	2.68	7.55	10.8	34.6	47.5	59.3	66.8	72.3

is that, except for CR, the combinations give a speedup less than the product of appropriate previous columns. Perhaps this is at least partially owing to an effect similar to Ahmdal's law: there still exist parts of the algorithm that have not been sped up, and as the remainder of the program that ▪ ▪ being optimized speeds up, these unoptimized sections take a greater proportion of the time.

Finally, it can be seen that the effect of the last two optimizations, ▪ and ▪, observed cumulatively with other optimizations ▪ ▪ in the last two columns of Table 1, contribute little to the average. This is because most of the systems in this table were shadowable. In fact, F slightly slows down refinement in one case. However, orbits 5, 6, 13, 19, and 29 obtained speedups significantly less than the average in the KRI column, but gain was made in the final two columns of up to a factor of 5. These were precisely the orbits that were most difficult to shadow (as the time column shows), and in fact some of them were not shadowable for their entire lengths. For these orbits, significant time was spent in the failed search to find a shadow lasting 1.28 time units. This indicates that the F and i optimizations are most effective in speeding up failing searches.

The final version of our algorithm corresponds to the combination of all optimizations, as illustrated in the last column of Table 1. However, in our source code, each of the optimizations can be turned on or off at compile time if the user so wishes.

5. Discussion. QT argues that if the refinement algorithm fails, there is good reason to believe that no shadow exists. This seems reasonable for several reasons. First, from the more rigorous study of simpler systems, glitches are known to exist and are not just a failure of any particular refinement algorithm. Second, QT's results are consistent with a conjecture by GHYS on the frequency of glitches [40]. Finally, [44] finds that 1-step errors close to the machine precision are often enough to rigorously imply the existence of an exact shadow (assuming sufficient hyperbolicity). However, in a nonnegligible number of cases, a shadow is not found for shadow steps $0..S$, but a shadow is found for the set $0..2S$. In other words, the algorithm failed to find a shadow of length S even though one exists. This also occurs when all optimizations are disabled, so it is not merely that the optimizations reduce reliability. Thus, in the search for the longest shadow, if our algorithm finds a shadow of length S but none of length $2S$, we also try $4S$ before giving up.

The GHYS/QT refinement algorithm is trivially parallelizable, since the computation of each RUS_i is independent of all the others; for the same reason, it also has excellent locality of reference in a serial implementation, so virtual memory paging is minimized.

There may be further optimizations possible. For example, in the N-body problem, the phase-space dimensions are split into sets of 6, one set for each particle in the system. Instead of computing the $6M \times 6M$ resolvent in $O((6M)^3)$ time, perhaps it may be possible to refine the trajectories of each particle separately, requiring M resolvents each of dimension 6×6, and $O(6^3 M)$ time to compute. This approach was explored with partial success in [31] and [30], but more work is needed to solidify the technique.

That a Hamiltonian system is shadowable for a nontrivial duration of time seems to surprise many people, since it can be shown that Hamiltonian systems are not hyperbolic. However, in practice, a system can hyperbolic for a nontrivial duration of time, before the global nonhyperbolicity inevitably asserts itself. In practice, it is the duration of this interval of "pseudohyperbolicity" [32] that shadowing measures, and in many cases the duration is nontrivial. We intend to apply our shadowing algorithm to a wider variety of problems in the near future. It has already been applied to a simple molecular dynamics test case with interesting results.

The source code can in principle be adapted to any ODE. However, the current major stumbling block is that the Jacobian must be hand-coded by the user, an extremely tedious and error-prone process. We intend in the near future to incorporate automatic differentiation [3, 4], but until then we are reluctant to publicly release the source code. However, individual requests for the source code can be addressed to the first author.

Acknowledgments. We thank Gerry Quinlan and Scott Tremaine for helpful discussions throughout this research, and for providing their original source code. The University of Toronto undergraduate Computing Disciplines Facility generously provided workstation clusters on which we consumed several CPU years during this research.

REFERENCES

[1] M. O. Ahmed and R. M. Corless, *The method of modified equations in Maple*, in Electronic Proceedings of the 3rd International IMACS conference on Applications of Computer Algebra, Maui, 1997, M. Wester and S. Steiberg eds., Marcel Dekker, New York, pp. 1–11.

[2] D. V. Anosov, *Geodesic flows and closed Riemannian manifolds with negative curvature*, Proc. Steklov Inst. Math., 90 (1967).

[3] C. Bendtsen and O. Stauling, *FADBAD, A Flexible C++ Package for Automatic Differentiation, Using the Forward and Backward Methods*, Technical Report IMM-REP-1996-17, Department of Mathematical Modelling, Technical University of Denmark, Kgs. Lyngby, Denmark, 1996.

[4] C. Bendtsen and O. Stauling, *TADIFF, A Flexible C++ Package for Automatic Differentiation, Using Taylor Series Expansion*, Technical Report IMM-REP-1997-07, Department of Mathematical Modelling, Technical University of Denmark, Kgs. Lyngby, Denmark, 1997.

[5] W.-J. Beyn, *On invariant closed curves for one-step methods*, Numer. Math., 51 (1987), pp. 103–122.

[6] R. Bowen, *ω-limit sets for axiom a diffeomorphisms*, J. Differential Equations, 18 (1975), p. 333–339.

[7] P. J. Channell and C. Scovel, *Symplectic integration of Hamiltonian systems*, Nonlinearity, 3 (1990), pp. 231–259.

[8] S.-N. Chow and K. J. Palmer, *On the numerical computation of orbits of dynamical systems: The one-dimensional case*, Dynam. Differential Equations, 3 (1991), pp. 361–380.

[9] S.-N. Chow and K. J. Palmer, *On the numerical computation of orbits of dynamical systems: The higher dimensional case*, J. Complexity, 8 (1992), pp. 398–423.

[10] S. N. Chow and E. S. Van Vleck, *Shadowing of lattice maps*, in Chaotic Numerics, P. E. Kloeden and K. J. Palmer, eds., AMS, Providence, RI, 1993, pp. 97–113.

[11] S.-N. Chow and E. S. Van Vleck, *A shadowing lemma approach to global error analysis for initial value ODEs*, SIAM J. Sci. Comput., 15 (1994), pp. 959–976.

[12] S. N. Chow and E. S. Van Vleck, *Shadowing of lattice maps*, Contemp. Math., 172 (1994), pp. 97–113.

[13] D. A. Clarke and M. J. West, eds., *The 12th "Kingston Meeting": Computational Astrophysics*, ASP Conference Series 123, Astronomical Society of the Pacific, San Francisco, 1997.

[14] B. A. Coomes, *Shadowing orbits of ordinary differential equations on invariant submanifolds*, Trans. Amer. Math. Soc., 349 (1997), pp. 203–216.

[15] B. A. Coomes, H. Koçak, and K. J. Palmer, *Shadowing orbits of ordinary differential equations*, J. Comput. Appl. Math., 52 (1994), pp. 35–43.

[16] B. A. Coomes, H. Koçak, and K. J. Palmer, *Rigorous computational shadowing of orbits of ordinary differential equations*, Numer. Math., 69 (1995), pp. 401–421.

[17] B. A. Coomes, H. Koçak, and K. J. Palmer, *A shadowing theorem for ordinary differential equations*, Z. Angew. Math. Phys., 46 (1995), pp. 85–106.

[18] R. M. Corless, *Error backward*, Contemp. Math., 172 (1994), pp. 31–62.

[19] R. M. Corless and G. F. Corliss, *Rationale for guaranteed ODE defect control*, in Computer Arithmetic and Enclosure Methods (1992 IMACS), L. Atanassova and J. Herzberger, eds., Elsevier Science, North-Holland, Amsterdam, 1992, pp. 1–9.

[20] G. Dahlquist and Å. Björck, *Numerical Methods*, Prentice-Hall Series in Automatic Computation, Prentice-Hall, Englewood Cliffs, NJ, 1974.

[21] S. Dawson, C. Grebogi, T. Sauer, and J. A. Yorke, *Obstructions to shadowing when a Lyapunov exponent fluctuates about zero*, Phys. Rev. Lett., 73 (1994), pp. 1927–1930.

[22] J. Dubinski, *The dynamical evolution of galaxies in clusters*, in Galaxy Dynamics: A Rutgers Symposium, D. Merritt, J. A. Sellwood, and M. Valluri, eds., ASP Conference Series 182, Astronomical Society of the Pacific, San Francisco, 1999, pp. 491–494.

[23] W. H. Enright, *The relative efficiency of alternative defect control schemes for high-order continuous Runge–Kutta formulas*, SIAM J. Numer. Anal., 30 (1993), pp. 1419–1445.

[24] W. H. Enright, K. R. Jackson, S. P. Nørsett, and P. G. Thomsen, *Interpolants for Runge–Kutta formulas*, ACM TOMS, 12 (1986), pp. 193–218.

[25] J. Goodman, D. C. Heggie, and P. Hut, *On the exponential instability of n-body systems*, Astrophys. J., 415 (1993), pp. 715–733.

[26] C. Grebogi, S. M. Hammel, J. A. Yorke, and T. Sauer, *Shadowing of physical trajectories in chaotic dynamics: Containment and refinement*, Phys. Rev. Lett., 65 (1990), pp. 1527–1530.

[27] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations*, 2nd ed., Springer-Verlag, New York, 1993.

[28] S. M. Hammel, J. A. Yorke, and C. Grebogi, *Do numerical orbits of chaotic dynamical processes represent true orbits?*, J. Complexity, 3 (1987), pp. 136–145.

[29] W. B. Hayes, *Efficient Shadowing of High Dimensional Chaotic Systems with the Large Astrophysical n-Body Problem as an Example*, Master's thesis, Department of Computer Science, University of Toronto, Toronto, ON, 1995.

[30] W. B. Hayes, *Shadowing-based reliability decay in softened n-body simulations*, Astrophys. J. Lett., 587 (2003), pp. L59–L62.

[31] W. B. Hayes, *Shadowing high-dimensional Hamiltonian systems: The gravitational n-body problem*, Phys. Rev. Lett., 90 (2003), pp. 1–4.
[32] W. B. Hayes and K. R. Jackson, *Rigorous shadowing of numerical solutions of ordinary differential equations by containment*, SIAM J. Numer. Anal., 41 (2003), pp. 1948–1973.
[33] D. C. Heggie and R. D. Mathieu, *Standardized units and time scales*, in The Use of Supercomputers in Stellar Dynamics, P. Hut and S. L. W. McMillan, eds., Springer-Verlag, New York, 1986, pp. 233–235.
[34] A. C. Hindmarsh, *LSODE and LSODI, Two new initial value ordinary differential equation solvers*, ACM-SIGNUM Newsletter, 15 (1980), pp. 10–11.
[35] D. Kahaner, C. Moler, and S. Nash, *Numerical Methods and Software*, Prentice-Hall Series in Computational Mathematics, Prentice-Hall, Englewood Cliffs, NJ, 1989.
[36] D. Merritt, *Elliptical galaxy dynamics*, Pub. Astronom. Soc. Pacific, 111 (1999), pp. 129–168.
[37] D. Merritt, J. A. Sellwood, and M. Valluri, eds., *Galaxy Dynamics: A Rutgers Symposium*, ASP Conference Series 182, Astronomical Society of the Pacific, San Francisco, 1999.
[38] D. Merritt and M. Valluri, *Chaos and mixing in triaxial stellar systems*, Astrophys. J., 471 (1996), pp. 82–105.
[39] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*, 2nd ed., Cambridge University Press, Cambridge, UK, 1992.
[40] G. D. Quinlan and S. Tremaine, *On the reliability of gravitational n-body integrations*, Monthly Notices of the Royal Astronomical Society, 259 (1992), pp. 505–518.
[41] J. M. Sanz-Serna, *Symplectic integrators for Hamiltonian problems: An overview*, in Acta Numerica 1992, A. Iserles, ed., Cambridge University Press, Cambridge, UK, 1992, pp. 243–286.
[42] J. M. Sanz-Serna and S. Larsson, *Shadows, chaos, and saddles*, Appl. Numer. Math., 13 (1993), pp. 181–190.
[43] T. Sauer, C. Grebogi, and J. A. Yorke, *How long do numerical chaotic solutions remain valid?*, Phys. Rev. Lett., 79 (1997), 59–62.
[44] T. Sauer and J. A. Yorke, *Rigorous verification of trajectories for the computer simulation of dynamical systems*, Nonlinearity, 4 (1991), pp. 961–979.
[45] T. D. Sauer, *Shadowing breakdown and large errors in dynamical simulations of physical systems*, Phys. Rev. E, 65 (2002), paper 036220.
[46] B. A. Shadwick, J. C. Bowman, and P. J. Morrison, *Exactly conservative integrators*, SIAM J. Appl. Math., 59 (1999), pp. 1112–1133.
[47] R. Skeel, *The Meaning of Molecular Dynamics*, manuscript, Department of Computer Science, University of Illinois at Urbana, Champaign, Urbana, IL, 1996.
[48] R. D. Skeel, *Integration schemes for molecular dynamics and related applications*, in The Graduate Student's Guide to Numerical Analysis, M. Ainsworth, J. Levesley, and M. Marletta, eds., Special Radon Semester in Computational Mathematics (SSCM), Springer-Verlag, New York, 1999, pp. 119–176.
[49] C. Struck, *Galaxy splashes: The effects of collisions between gas-rich galaxy disks*, in The 12th "Kingston Meeting": Computational Astrophysics, D. A. Clarke and M. J. West, eds., ASP Conference Series 123, Astronomical Society of the Pacific, San Francisco, 1997, pp. 225–230.
[50] E. S. Van Vleck, *Numerical shadowing near hyperbolic trajectories*, SIAM J. Sci. Comput., 16 (1995), pp. 1177–1189.

SIAM J. SCI. COMPUT.
Vol. 29, No. 4, pp. 1759–1780
© 2007 Society for Industrial and Applied Mathematics

THREE-LEVEL BDDC IN THREE DIMENSIONS*

XUEMIN TU†

Abstract. Balancing domain decomposition by constraints (BDDC) methods are nonoverlapping iterative substructuring domain decomposition methods for the solution of large sparse linear algebraic systems arising from the discretization of elliptic boundary value problems. Their coarse problems are given in terms of a small number of continuity constraints for each subdomain, which are enforced across the interface. The coarse problem matrix is generated and factored by a direct solver at the beginning of the computation and it can ultimately become a bottleneck if the number of subdomains is very large. In this paper, two three-level BDDC methods are introduced for solving the coarse problem approximately for problems in three dimensions. This is an extension of previous work for the two-dimensional case. Edge constraints are considered in this work since vertex constraints alone, which work well in two dimensions, result in a noncompetitive algorithm in three dimensions. Some new technical tools are then needed in the analysis and this makes the three-dimensional case more complicated. Estimates of the condition numbers are provided for two three-level BDDC methods, and numerical experiments are also discussed.

Key words. BDDC, three-level, three dimensions, domain decomposition, coarse problem, condition number, Chebyshev iteration

AMS subject classifications. 65N30, 65N55

DOI. 10.1137/050629902

1. Introduction.

Balancing domain decomposition by constraints (BDDC) methods, which were introduced and analyzed in [4, 11, 12], are similar to the balancing Neumann–Neumann algorithms. The coarse problem in a BDDC algorithm is given in terms of a set of primal constraints chosen for each subdomain, and the matrix of the coarse problem is generated and factored by using a direct solver at the beginning of the computation. We note that there are now computer systems with more than 100,000 powerful processors, which allow very large and detailed simulations. If there is a one to one or one to several relationship between processors and subdomains, then we can have a large number of subdomains. The coarse component of a two-level preconditioner can therefore become a bottleneck if the number of subdomains is very large. One way to remove this difficulty is to introduce one or more additional levels. In our recent paper [17], two three-level BDDC methods were introduced for two-dimensional problems with vertex constraints. We solve the coarse problem approximately by using the BDDC idea recursively and show that a good rate of convergence still can be maintained. However, in three dimensions, vertex constraints alone are not enough to obtain good polylogarithmic condition number bounds due to much weaker interpolation estimates, and constraints on the averages over edges or faces are needed. The new constraints lead to a considerably more complicated coarse problem and the need for new technical tools in the analysis. In this paper, we extend the two three-level BDDC methods in [17] to the three-dimensional

*Received by the editors April 24, 2005; accepted for publication (in revised form) February 16, 2007; published electronically September 19, 2007. This work was supported in part by the U.S. Department of Energy under contracts DE-FG02-92ER25127 and DE-FC02-01ER25482 while the author was a Ph.D. student at Courant Institute of Mathematical Sciences, New York University. This work was also supported in part by the Director, Office of Science, Advanced Scientific Computing Research, U.S. Department of Energy under contract DE-AC02-05CH11231.

http://www.siam.org/journals/sisc/29-4/62990.html

†Department of Mathematics, University of California, and Lawrence Berkeley National Laboratory, Berkeley, CA 94720-3840 (xuemin@math.berkeley.edu).

case using primal edge average constraints. With the help of new technical tools, we are able to provide estimates of the condition numbers of the system with these two new preconditioners.

We note that, since this paper was submitted, several papers have appeared on inexact solvers for BDDC (see [5, 10, 16, 15]) and dual-primal finite element tearing and interconnecting (FETI-DP), which are iterative substructuring algorithms closely related to BDDC (see [6]). For the study of the convergence rates of the BDDC algorithms and their connection with the FETI-DP algorithms, see [11, 12, 9, 2].

The rest of the paper is organized as follows. We first review the two-level BDDC methods briefly in section 2. We introduce our first three-level BDDC method and the corresponding preconditioner $\widetilde{M}^{-1}$ in section 3. We give some auxiliary results in section 4. In section 5, we provide an estimate of the condition number for the system with the preconditioner $\widetilde{M}^{-1}$, which is of the form $C(1+\log\frac{\hat{H}}{H})^2(1+\log\frac{H}{h})^2$, where $\hat{H}$, H, and h are typical diameters of the subregions, subdomains, and elements, respectively. (We decompose the whole domain into subregions and each subregion is then partitioned into several subdomains; see section 3 for details.) In section 6, we introduce a second three-level BDDC method which uses Chebyshev iterations. We denote the corresponding preconditioner by $\widehat{M}^{-1}$. We show that the condition number bound of the system with the preconditioner $\widehat{M}^{-1}$ is of the form $CC(k)\left(1+\log\frac{H}{h}\right)^2$, where $C(k)$ is a function of k, the number of Chebyshev iterations, and also depends on the eigenvalues of the preconditioned coarse problem and on the two parameters chosen for the Chebyshev iteration. $C(k)$ goes to 1 as k goes to ∞; i.e., the condition number approaches that of the two-level case. Finally, some computational results are presented in section 7.

2. The two-level BDDC method. The two-level BDDC methods have been studied extensively; see [4, 11, 12, 9]. In this section, we will briefly review this work and introduce notation which will be used in the rest of the paper.

We will consider a second order scalar elliptic problem in a three-dimensional region Ω as follows: Find $u \in H_0^1(\Omega)$ such that

$$\int_\Omega \rho\nabla u\cdot\nabla v = \int_\Omega fv \quad \forall v \in H_0^1(\Omega), \tag{2.1}$$

where $\rho(x) > 0$ for all $x \in \Omega$. We decompose Ω into N nonoverlapping subdomains Ω_i with diameters H_i, $i = 1,\dots,N$, and set $H = \max_i H_i$. We then introduce a triangulation of all the subdomains. Let Γ be the interface between the subdomains and let the set of interface nodes Γ_h be defined by $\Gamma_h = (\cup_i \partial\Omega_{i,h}) \setminus \partial\Omega_h$, where $\partial\Omega_{i,h}$ is the set of nodes on $\partial\Omega_i$ and $\partial\Omega_h$ is the set of nodes on $\partial\Omega$. The nodes of the different triangulations match across Γ.

Let $\mathbf{W}^{(i)}$ be the standard finite element space of continuous, piecewise trilinear functions on Ω_i; the algorithms and theory developed in this paper work for other lower order finite elements as well. We assume that these functions vanish on $\partial\Omega$. Each $\mathbf{W}^{(i)}$ can be decomposed into a subdomain interior part $\mathbf{W}_I^{(i)}$ and a subdomain interface part $\mathbf{W}_\Gamma^{(i)}$, i.e., $\mathbf{W}^{(i)} = \mathbf{W}_I^{(i)} \bigoplus \mathbf{W}_\Gamma^{(i)}$, where the subdomain interface part $\mathbf{W}_\Gamma^{(i)}$ will be further decomposed into a primal subspace $\mathbf{W}_\Pi^{(i)}$ and a dual subspace $\mathbf{W}_\Delta^{(i)}$, i.e., $\mathbf{W}_\Gamma^{(i)} = \mathbf{W}_\Pi^{(i)} \bigoplus \mathbf{W}_\Delta^{(i)}$. (They are called primal and dual in earlier works on FETI algorithms, where the dual variables are controlled by Lagrange multipliers.) Here, we will consider only edge average constraints over all the edges of all subdomains as primal variables. We change the variables to make the edge average degrees

of freedom explicit; see [7, section 4.2.1] and [9, section 2.3]. From now on, we assume that all the matrices are written in terms of the new variables.

We denote the associated product spaces by $\mathbf{W} := \prod_{i=1}^{N} \mathbf{W}^{(i)}$, $\mathbf{W}_\Gamma := \prod_{i=1}^{N} \mathbf{W}_\Gamma^{(i)}$, $\mathbf{W}_\Delta := \prod_{i=1}^{N} \mathbf{W}_\Delta^{(i)}$, $\mathbf{W}_\Pi := \prod_{i=1}^{N} \mathbf{W}_\Pi^{(i)}$, and $\mathbf{W}_I := \prod_{i=1}^{N} \mathbf{W}_I^{(i)}$. Correspondingly, we have $\mathbf{W} = \mathbf{W}_I \bigoplus \mathbf{W}_\Gamma$, and $\mathbf{W}_\Gamma = \mathbf{W}_\Pi \bigoplus \mathbf{W}_\Delta$.

The elements of $\mathbf{W}$ can be discontinuous across the interface. However, the finite element approximation of the elliptic problem is continuous across Γ. We denote the corresponding subspace of $\mathbf{W}$ by $\widehat{\mathbf{W}}$. Similarly, we denote the corresponding subspaces of $\mathbf{W}_\Gamma$, $\mathbf{W}_\Delta$, and $\mathbf{W}_\Pi$ by $\widehat{\mathbf{W}}_\Gamma$, $\widehat{\mathbf{W}}_\Delta$, and $\widehat{\mathbf{W}}_\Pi$, respectively.

In order to define the BDDC preconditioner, we further introduce an interface subspace $\widetilde{\mathbf{W}}_\Gamma \subset \mathbf{W}_\Gamma$, for which all the edge average primal constraints are enforced. The space $\widetilde{\mathbf{W}}_\Gamma$ can be decomposed into $\widetilde{\mathbf{W}}_\Gamma = \widehat{\mathbf{W}}_\Pi \bigoplus \mathbf{W}_\Delta$. We also have $\widehat{\mathbf{W}}_\Gamma \subset \widetilde{\mathbf{W}}_\Gamma$.

The global problem has the following form: Find $(\mathbf{u}_I, \mathbf{u}_\Delta, \mathbf{u}_\Pi) \in (\mathbf{W}_I, \widehat{\mathbf{W}}_\Delta, \widehat{\mathbf{W}}_\Pi)$ such that

$$\begin{pmatrix} A_{II} & A_{\Delta I}^T & A_{\Pi I}^T \\ A_{\Delta I} & A_{\Delta\Delta} & A_{\Pi\Delta}^T \\ A_{\Pi I} & A_{\Pi\Delta} & A_{\Pi\Pi} \end{pmatrix} \begin{pmatrix} \mathbf{u}_I \\ \mathbf{u}_\Delta \\ \mathbf{u}_\Pi \end{pmatrix} = \begin{pmatrix} \mathbf{f}_I \\ \mathbf{f}_\Delta \\ \mathbf{f}_\Pi \end{pmatrix}. \tag{2.2}$$

This problem is assembled from the subdomain problems

$$\begin{pmatrix} A_{II}^{(i)} & A_{\Delta I}^{(i)^T} & A_{\Pi I}^{(i)^T} \\ A_{\Delta I}^{(i)} & A_{\Delta\Delta}^{(i)} & A_{\Pi\Delta}^{(i)^T} \\ A_{\Pi I}^{(i)} & A_{\Pi\Delta}^{(i)} & A_{\Pi\Pi}^{(i)} \end{pmatrix} \begin{pmatrix} \mathbf{u}_I^{(i)} \\ \mathbf{u}_\Delta^{(i)} \\ \mathbf{u}_\Pi^{(i)} \end{pmatrix} = \begin{pmatrix} \mathbf{f}_I^{(i)} \\ \mathbf{f}_\Delta^{(i)} \\ \mathbf{f}_\Pi^{(i)} \end{pmatrix}. \tag{2.3}$$

We also denote by $\mathbf{F}_\Gamma$, $\widehat{\mathbf{F}}_\Gamma$, and $\widetilde{\mathbf{F}}_\Gamma$ the dual spaces, that is, the spaces of the right-hand sides corresponding to $\mathbf{W}_\Gamma$, $\widehat{\mathbf{W}}_\Gamma$, and $\widetilde{\mathbf{W}}_\Gamma$, respectively.

In order to describe the BDDC algorithms, we need to introduce several restriction, extension, and scaling operators between different spaces.

The restriction operators from the product spaces to the subdomain local spaces are

$$\widehat{\mathbf{W}}_\Gamma \xrightarrow{R_\Gamma^{(i)}} \mathbf{W}_\Gamma^{(i)}, \quad \widehat{\mathbf{W}}_\Gamma \xrightarrow{\widehat{R}_\Delta^{(i)}} \mathbf{W}_\Delta^{(i)}, \quad \widetilde{\mathbf{W}}_\Gamma \xrightarrow{\overline{R}_\Gamma^{(i)}} \mathbf{W}_\Gamma^{(i)}, \quad \widehat{\mathbf{W}}_\Pi \xrightarrow{R_\Pi^{(i)}} \mathbf{W}_\Pi^{(i)}, \quad \text{and} \quad \mathbf{W}_\Delta \xrightarrow{R_\Delta^{(i)}} \mathbf{W}_\Delta^{(i)}.$$

Additionally, there are three restriction operators:

$$\widehat{\mathbf{W}}_\Gamma \xrightarrow{\widehat{R}_\Pi} \widehat{\mathbf{W}}_\Pi, \quad \widetilde{\mathbf{W}}_\Gamma \xrightarrow{R_{\Gamma\Delta}} \mathbf{W}_\Delta, \quad \text{and} \quad \widetilde{\mathbf{W}}_\Gamma \xrightarrow{R_{\Gamma\Pi}} \widehat{\mathbf{W}}_\Pi.$$

We also introduce two extension operators:

$$\widehat{\mathbf{W}}_\Gamma \xrightarrow{\widetilde{R}_\Gamma} \widetilde{\mathbf{W}}_\Gamma \xrightarrow{\overline{R}_\Gamma} \mathbf{W}_\Gamma,$$

where $\widetilde{R}_\Gamma$ is the direct sum of the operators $\widehat{R}_\Delta^{(i)}$ and $\widehat{R}_\Pi$, and $\overline{R}_\Gamma$ is the direct sum of the operators $\overline{R}_\Gamma^{(i)}$.

Multiplying each element of the matrix $\widehat{R}_\Delta^{(i)}$, which corresponds to a node $x \in \partial\Omega_i$, with $\delta_i^\dagger(x)$ gives us $\widehat{R}_{D,\Delta}^{(i)}$. Here, we define the scale factor $\delta_i^\dagger(x)$ as follows: For $\gamma \in [1/2, \infty)$,

$$\delta_i^\dagger(x) = \frac{\rho_i^\gamma(x)}{\sum_{j \in \mathcal{N}_x} \rho_j^\gamma(x)}, \quad x \in \partial\Omega_{i,h} \cap \Gamma_h, \tag{2.4}$$

where $\mathcal{N}_x$ is the set of indices j of the subdomains such that $x \in \partial\Omega_j$, and $\rho_j(x)$ is the coefficient of (2.1) at x in the subdomain Ω_j.

The scaled extension operator $\widetilde{R}_{D,\Gamma}$ is the direct sum of the operators $\widehat{R}^{(i)}_{D,\Delta}$ and $\widehat{R}_{\Pi}$. Equivalently, we can write $\widetilde{R}_{D,\Gamma} = D\widetilde{R}_{\Gamma}$, where D is a diagonal scaling matrix. The diagonal elements of D, corresponding to the primal variables, are 1, and all others are given by $\delta_i^{\dagger}(x)$.

We also use the same restriction, extension, and scaled extension operators for $\mathbf{F}_\Gamma$, $\widehat{\mathbf{F}}_\Gamma$, and $\widetilde{\mathbf{F}}_\Gamma$.

We now reduce the global problem (2.2) to an interface problem. We first introduce the subdomain Schur complement $S_\Gamma^{(i)}$ by eliminating the subdomain interior variables $u_I^{(i)}$ in (2.3) as follows:

$$S_\Gamma^{(i)} = \begin{pmatrix} A^{(i)}_{\Delta\Delta} & A^{(i)^T}_{\Pi\Delta} \\ A^{(i)}_{\Pi\Delta} & A^{(i)}_{\Pi\Pi} \end{pmatrix} - \begin{pmatrix} A^{(i)}_{\Delta I} \\ A^{(i)}_{\Pi I} \end{pmatrix} A^{(i)^{-1}}_{II} \begin{pmatrix} A^{(i)^T}_{\Delta I} & A^{(i)^T}_{\Pi I} \end{pmatrix}$$

and let

$$S_\Gamma = \begin{pmatrix} S_\Gamma^{(1)} & & \\ & \ddots & \\ & & S_\Gamma^{(N)} \end{pmatrix}.$$

The partially assembled Schur complement $\widetilde{S}_\Gamma$ is obtained from S_Γ by assembling the primal variables on the subdomain interface, i.e.,

$$\widetilde{S}_\Gamma = \overline{R}_\Gamma^T S_\Gamma \overline{R}_\Gamma.$$

$\widetilde{S}_\Gamma$ can be further assembled with respect to the variables of the $\mathbf{W}_\Delta^{(i)}$ and the reduced interface problem of (2.2) can be written as follows: Find $\mathbf{u}_\Gamma \in \widehat{\mathbf{W}}_\Gamma$ such that

$$\widetilde{R}_\Gamma^T \widetilde{S}_\Gamma \widetilde{R}_\Gamma \mathbf{u}_\Gamma = \mathbf{g}_\Gamma,$$

where

$$\mathbf{g}_\Gamma = \sum_{i=1}^N R_\Gamma^{(i)^T} \left\{ \begin{pmatrix} \mathbf{f}_\Delta^{(i)} \\ \mathbf{f}_\Pi^{(i)} \end{pmatrix} - \begin{pmatrix} A^{(i)}_{\Delta I} \\ A^{(i)}_{\Pi I} \end{pmatrix} A^{(i)^{-1}}_{II} \mathbf{f}_I^{(i)} \right\}.$$

The preconditioned two-level BDDC equation is of the form

$$M^{-1} \widetilde{R}_\Gamma^T \widetilde{S}_\Gamma \widetilde{R}_\Gamma \mathbf{u}_\Gamma = M^{-1} \mathbf{g}_\Gamma,$$

where the preconditioner $M^{-1} = \widetilde{R}_{D,\Gamma}^T \widetilde{S}_\Gamma^{-1} \widetilde{R}_{D,\Gamma}$ has the following form:

(2.5)
$$\widetilde{R}_{D,\Gamma}^T \left\{ R_{\Gamma\Delta}^T \left(\sum_{i=1}^N \begin{pmatrix} \mathbf{0} & R_\Delta^{(i)^T} \end{pmatrix} \begin{pmatrix} A^{(i)}_{II} & A^{(i)}_{I\Delta} \\ A^{(i)}_{\Delta I} & A^{(i)}_{\Delta\Delta} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{0} \\ R_\Delta^{(i)} \end{pmatrix} \right) R_{\Gamma\Delta} + \Phi S_\Pi^{-1} \Phi^T \right\} \widetilde{R}_{D,\Gamma}.$$

Here Φ is the matrix given by the coarse level basis functions of minimal energy defined by

$$\Phi = R_{\Gamma\Pi}^T - R_{\Gamma\Delta}^T \sum_{i=1}^N \begin{pmatrix} \mathbf{0} & R_\Delta^{(i)^T} \end{pmatrix} \begin{pmatrix} A^{(i)}_{II} & A^{(i)}_{I\Delta} \\ A^{(i)}_{\Delta I} & A^{(i)}_{\Delta\Delta} \end{pmatrix}^{-1} \begin{pmatrix} A^{(i)^T}_{\Pi I} \\ A^{(i)^T}_{\Pi\Delta} \end{pmatrix} R_\Pi^{(i)}.$$

The coarse level problem matrix S_Π is determined by

(2.6)
$$S_\Pi = \sum_{i=1}^{N} R_\Pi^{(i)^T} \left\{ A_{\Pi\Pi}^{(i)} - \begin{pmatrix} A_{\Pi I}^{(i)} & A_{\Pi\Delta}^{(i)} \end{pmatrix} \begin{pmatrix} A_{II}^{(i)} & A_{I\Delta}^{(i)} \\ A_{\Delta I}^{(i)} & A_{\Delta\Delta}^{(i)} \end{pmatrix}^{-1} \begin{pmatrix} A_{\Pi I}^{(i)^T} \\ A_{\Pi\Delta}^{(i)^T} \end{pmatrix} \right\} R_\Pi^{(i)},$$

which is obtained by assembling subdomain matrices; for additional details, cf. [4, 11, 9].

We know that, under certain assumptions, and for any $\mathbf{u}_\Gamma \in \widehat{\mathbf{W}}_\Gamma$,

$$\mathbf{u}_\Gamma^T M \mathbf{u}_\Gamma \leq \mathbf{u}_\Gamma^T \widetilde{R}_\Gamma^T \widetilde{S}_\Gamma \widetilde{R}_\Gamma \mathbf{u}_\Gamma \leq C \left(1 + \log(H/h)\right)^2 \mathbf{u}_\Gamma^T M \mathbf{u}_\Gamma. \tag{2.7}$$

These estimates can be established directly by using methods very similar to those of certain studies of the FETI-DP algorithms. Denote by E_D and P_D, respectively, the average and jump operators (see [14, Formulas (6.4) and (6.38)]) on the space $\widetilde{\mathbf{W}}_\Gamma$. Central to obtaining the condition number estimate for the preconditioned two-level BDDC operator is a bound for the E_D operators (see [12, Theorem 25]). Since $E_D + P_D = I$ (see [14, Lemma 6.10]), we need only find a bound for the P_D operator. We obtain a bound for the P_D operator by using [14, Lemma 6.36] under [14, Assumption 4.3.1] for the triangulation and using [14, Assumption 6.27.2] for the coefficient $\rho(x)$ of (2.1).

3. A three-level BDDC method. For the three-level cases, as in [17], the coarse problem matrix S_Π defined in (2.6) will not be factored by a direct solver. Instead, a new level is introduced and the coarse problem is solved approximately. Call the new level the subregion level. To distinguish the spaces and operators for the subregion level from those for the subdomain level, we use the subscript c for the former.

We decompose Ω into N_c subregions Ω^j with diameters $\hat{H}^j$, $j = 1, \ldots, N_c$. Each subregion Ω^j is the union of N_j subdomains Ω_i^j with diameters H_i^j. Let $\hat{H} = \max_j \hat{H}^j$ and $H = \max_{i,j} H_i^j$, for $j = 1, \ldots, N_c$, and $i = 1, \ldots, N_j$. Then N, the total number of subdomains, can be written as $N = N_1 + \cdots + N_{N_c}$.

We introduce the subregional Schur complement as

$$S_\Pi^{(j)} = \sum_{i=1}^{N_j} R_\Pi^{(i)^T} \left\{ A_{\Pi\Pi}^{(i)} - \begin{pmatrix} A_{\Pi I}^{(i)} & A_{\Pi\Delta}^{(i)} \end{pmatrix} \begin{pmatrix} A_{II}^{(i)} & A_{I\Delta}^{(i)} \\ A_{\Delta I}^{(i)} & A_{\Delta\Delta}^{(i)} \end{pmatrix}^{-1} \begin{pmatrix} A_{\Pi I}^{(i)^T} \\ A_{\Pi\Delta}^{(i)^T} \end{pmatrix} \right\} R_\Pi^{(i)} \tag{3.1}$$

and note that the coarse problem matrix S_Π can be assembled from the $S_\Pi^{(j)}$.

In the two-level case, S_Π is factored by a direct solver at the beginning of the computation; cf. (2.5). Here, we build $\widetilde{S}_\Pi^{-1}$ to approximate S_Π^{-1}. Replacing S_Π^{-1} in (2.5) with $\widetilde{S}_\Pi^{-1}$ gives us the three-level preconditioner $\widetilde{M}^{-1}$:

$$\widetilde{R}_{D,\Gamma}^T \left\{ R_{\Gamma\Delta}^T \left(\sum_{i=1}^{N} \begin{pmatrix} \mathbf{0} & R_\Delta^{(i)^T} \end{pmatrix} \begin{pmatrix} A_{II}^{(i)} & A_{I\Delta}^{(i)} \\ A_{\Delta I}^{(i)} & A_{\Delta\Delta}^{(i)} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{0} \\ R_\Delta^{(i)} \end{pmatrix} \right) R_{\Gamma\Delta} + \Phi \widetilde{S}_\Pi^{-1} \Phi^T \right\} \widetilde{R}_{D,\Gamma}.$$

To define $\widetilde{S}_\Pi^{-1}$ in detail, we need to introduce several spaces and operators.

Let Γ_c be the interface between the subregions; note that $\Gamma_c \subset \Gamma$. For each subregion Ω^i, we denote by $\mathbf{W}_c^{(i)}$ the space corresponding to the subdomain edge

average variables in this subregion. Let $\mathbf{W}_c = \prod_{i=1}^{N_c} \mathbf{W}_c^{(i)}$ and let $\widehat{\mathbf{W}}_c$ be the subspace of $\mathbf{W}_c$ of elements that are continuous across Γ_c. $\mathbf{W}_c^{(i)}$ can be decomposed into a subregion interior part $\mathbf{W}_{c,I_c}^{(i)}$ and a subregion interface part $\mathbf{W}_{c,\Gamma_c}^{(i)}$, i.e., $\mathbf{W}_c^{(i)} = \mathbf{W}_{c,I_c}^{(i)} \bigoplus \mathbf{W}_{c,\Gamma_c}^{(i)}$. We further decompose the subregion interface part $\mathbf{W}_{c,\Gamma_c}^{(i)}$ into a primal subspace $\mathbf{W}_{c,\Pi_c}^{(i)}$ and a dual subspace $\mathbf{W}_{c,\Delta_c}^{(i)}$, i.e., $\mathbf{W}_{c,\Gamma_c}^{(i)} = \mathbf{W}_{c,\Pi_c}^{(i)} \bigoplus \mathbf{W}_{c,\Delta_c}^{(i)}$. Here, we will consider only the use of edge average constraints over subregion edges. Again, we should change the variables for all local coarse matrices corresponding to these edge average constraints. We will assume that all matrices are written in the new variables.

We denote the associated subregion interface product space by $\mathbf{W}_{c,\Gamma_c} := \prod_{i=1}^{N_c} \mathbf{W}_{c,\Gamma_c}^{(i)}$. We note that the elements in $\mathbf{W}_{c,\Gamma_c}$ can be discontinuous across the subregion interface Γ_c. Let $\widehat{\mathbf{W}}_{c,\Gamma_c}$ and $\widetilde{\mathbf{W}}_{c,\Gamma_c}$ be two subsets of $\mathbf{W}_{c,\Gamma_c}$. The elements are continuous across Γ_c in $\widehat{\mathbf{W}}_{c,\Gamma_c}$, whereas only the primal variables are continuous across Γ_c in $\widetilde{\mathbf{W}}_{c,\Gamma_c}$. We have $\widehat{\mathbf{W}}_{c,\Gamma_c} \subset \widetilde{\mathbf{W}}_{c,\Gamma_c} \subset \mathbf{W}_{c,\Gamma_c}$. We also need two extension operators $\widetilde{R}_{\Gamma_c}$ and $\overline{R}_{\Gamma_c}$,

$$\widehat{\mathbf{W}}_{c,\Gamma_c} \xrightarrow{\widetilde{R}_{\Gamma_c}} \widetilde{\mathbf{W}}_{c,\Gamma_c} \xrightarrow{\overline{R}_{\Gamma_c}} \mathbf{W}_{c,\Gamma_c},$$

which are similar to $\widetilde{R}_{\Gamma}$ and $\overline{R}_{\Gamma}$, respectively.

We denote by $\widehat{\mathbf{F}}_c$ and $\widehat{\mathbf{F}}_{\Gamma_c}$ the dual spaces of $\widehat{\mathbf{W}}_c$ and $\widehat{\mathbf{W}}_{c,\Gamma_c}$, respectively. We use the same operators for $\widehat{\mathbf{F}}_c$ and $\widehat{\mathbf{F}}_{\Gamma_c}$.

We are now ready to explain how $\widetilde{S}_{\Pi}^{-1}$ works on a vector in $\widehat{\mathbf{F}}_c$. Given a vector $\Psi \in \widehat{\mathbf{F}}_c$, let $\mathbf{y} = S_{\Pi}^{-1}\Psi$ and $\widetilde{\mathbf{y}} = \widetilde{S}_{\Pi}^{-1}\Psi$. We write Ψ, $\mathbf{y}$, and $\widetilde{\mathbf{y}}$ in terms of interior and interface parts, i.e., $\Psi = (\Psi_{I_c}^{(1)}, \ldots, \Psi_{I_c}^{(N_c)}, \Psi_{\Gamma_c})^T$, $\mathbf{y} = (\mathbf{y}_{I_c}^{(1)}, \ldots, \mathbf{y}_{I_c}^{(N_c)}, \mathbf{y}_{\Gamma_c})^T$, and $\widetilde{\mathbf{y}} = (\widetilde{\mathbf{y}}_{I_c}^{(1)}, \ldots, \widetilde{\mathbf{y}}_{I_c}^{(N_c)}, \widetilde{\mathbf{y}}_{\Gamma_c})^T$.

To obtain $\mathbf{y}$, we can solve $S_{\Pi}\mathbf{y} = \Psi$ by block factorization. This vector satisfies

$$\begin{pmatrix} S_{\Pi_{I_c I_c}}^{(1)} & \mathbf{0} & \mathbf{0} & S_{\Pi_{\Gamma_c I_c}}^{(1)^T} R_{\Gamma_c}^{(1)} \\ \mathbf{0} & \ddots & \mathbf{0} & \vdots \\ \mathbf{0} & \mathbf{0} & S_{\Pi_{I_c I_c}}^{(N_c)} & S_{\Pi_{\Gamma_c I_c}}^{(N_c)^T} R_{\Gamma_c}^{(N_c)} \\ R_{\Gamma_c}^{(1)^T} S_{\Pi_{\Gamma_c I_c}}^{(1)} & \cdots & R_{\Gamma_c}^{(N_c)^T} S_{\Pi_{\Gamma_c I_c}}^{(N_c)} & \sum_{i=1}^{N_c} R_{\Gamma_c}^{(i)^T} S_{\Pi_{\Gamma_c \Gamma_c}}^{(i)} R_{\Gamma_c}^{(i)} \end{pmatrix} \begin{pmatrix} \mathbf{y}_{I_c}^{(1)} \\ \vdots \\ \mathbf{y}_{I_c}^{(N_c)} \\ \mathbf{y}_{\Gamma_c} \end{pmatrix} = \begin{pmatrix} \Psi_{I_c}^{(1)} \\ \vdots \\ \Psi_{I_c}^{(N_c)} \\ \Psi_{\Gamma_c} \end{pmatrix},$$

where $R_{\Gamma_c}^{(i)} : \widehat{\mathbf{W}}_{c,\Gamma_c} \to \mathbf{W}_{c,\Gamma_c}^{(i)}$ is a restriction operator.

We solve $\mathbf{y}_{I_c}^{(i)}$ in terms of $\mathbf{y}_{\Gamma_c}$ and have

$$\mathbf{y}_{I_c}^{(i)} = S_{\Pi_{I_c I_c}}^{(i)^{-1}} \left(\Psi_{I_c}^{(i)} - S_{\Pi_{I_c \Gamma_c}}^{(i)} R_{\Gamma_c}^{(i)} \mathbf{y}_{\Gamma_c} \right). \tag{3.2}$$

We then obtain the subregion interface problem

(3.3)
$$\left(\sum_{i=1}^{N_c} R_{\Gamma_c}^{(i)^T}(S_{\Pi_{\Gamma_c\Gamma_c}}^{(i)} - S_{\Pi_{\Gamma_c I_c}}^{(i)} S_{\Pi_{I_c I_c}}^{(i)^{-1}} S_{\Pi_{\Gamma_c I_c}}^{(i)^T}) R_{\Gamma_c}^{(i)}\right) \mathbf{y}_{\Gamma_c} = \Psi_{\Gamma_c} - \sum_{i=1}^{N_c} R_{\Gamma_c}^{(i)^T} S_{\Pi_{\Gamma_c I_c}}^{(i)} S_{\Pi_{I_c I_c}}^{(i)^{-1}} \Psi_{I_c}^{(i)}.$$

Let

$$T^{(i)} = S_{\Pi_{\Gamma_c\Gamma_c}}^{(i)} - S_{\Pi_{\Gamma_c I_c}}^{(i)} S_{\Pi_{I_c I_c}}^{(i)^{-1}} S_{\Pi_{\Gamma_c I_c}}^{(i)^T}$$

be the subregion Schur complement in (3.3).

Denote their direct sum by T:

$$T = \begin{pmatrix} T^{(1)} & & \\ & \ddots & \\ & & T^{(N_c)} \end{pmatrix}.$$

As on the subdomain level case, we introduce a partially assembled Schur complement of S_Π, and denote it by $\widetilde{T}$. $\widetilde{T}$ can be written as

$$\widetilde{T} = \overline{R}_{\Gamma_c}^T T \overline{R}_{\Gamma_c}. \tag{3.4}$$

We define $\mathbf{h}_{\Gamma_c} \in \widehat{\mathbf{F}}_{\Gamma_c}$ by

$$\mathbf{h}_{\Gamma_c} = \Psi_{\Gamma_c} - \sum_{i=1}^{N_c} R_{\Gamma_c}^{(i)^T} S_{\Pi_{\Gamma_c I_c}}^{(i)} S_{\Pi_{I_c I_c}}^{(i)^{-1}} \Psi_{I_c}^{(i)}. \tag{3.5}$$

The reduced subregion interface problem (3.3) can be written as follows: Find $\mathbf{y}_{\Gamma_c} \in \widehat{\mathbf{W}}_{c,\Gamma_c}$ such that

$$\widetilde{R}_{\Gamma_c}^T \widetilde{T} \widetilde{R}_{\Gamma_c} \mathbf{y}_{\Gamma_c} = \mathbf{h}_{\Gamma_c}. \tag{3.6}$$

To obtain the approximation $\widetilde{\mathbf{y}} = \widetilde{S}_\Pi^{-1}\Psi$, we do not solve (3.6) exactly. Instead, we compute $\widetilde{\mathbf{y}}_{\Gamma_c}$ as

$$\widetilde{\mathbf{y}}_{\Gamma_c} = \widetilde{R}_{D_c,\Gamma_c}^T \widetilde{T}^{-1} \widetilde{R}_{D_c,\Gamma_c} \mathbf{h}_{\Gamma_c}. \tag{3.7}$$

Here $\widetilde{R}_{D_c,\Gamma_c}$ is a scaled operator which is similar to $\widetilde{R}_{D,\Gamma}$; we can write $\widetilde{R}_{D_c,\Gamma_c} = D_c \widetilde{R}_{\Gamma_c}$, where D_c is a diagonal scaling matrix. The diagonal elements of D_c, corresponding to the primal variables, are 1, and all others are given by $\delta_{c,i}^\dagger(x)$. Here $\delta_{c,i}^\dagger(x)$ is similar to $\delta_i^\dagger(x)$, which is defined in (2.4), except that $\delta_{c,i}^\dagger(x)$ is defined for the subregion interface instead of the subdomain interface nodes. For an x on the subregion interface, $\delta_{c,i}^\dagger(x)$ is defined as follows: For $\gamma \in [\frac{1}{2}, \infty)$, $\delta_{c,i}^\dagger(x) = \frac{\rho_i^\gamma(x)}{\sum_{j\in\mathcal{N}_x} \rho_j^\gamma(x)}$, where $\mathcal{N}_x$ is the set of indices j of the subregions such that $x \in \partial\Omega^j$ and $\rho_j(x)$ is the coefficient of (2.1) at $x \in \partial\Omega^j$. (In our theory, we assume the ρ_i are constant in the subregions.)

We will maintain the same relation between $\widetilde{\mathbf{y}}_{I_c}^{(i)}$ and $\widetilde{\mathbf{y}}_{\Gamma_c}$as for $\mathbf{y}_{I_c}^{(i)}$ and $\mathbf{y}_{\Gamma_c}$ in (3.2), i.e.,

$$\widetilde{\mathbf{y}}_{I_c}^{(i)} = S_{\Pi_{I_c I_c}}^{(i)^{-1}} \left(\Psi_{I_c}^{(i)} - S_{\Pi_{I_c\Gamma_c}}^{(i)} R_{\Gamma_c}^{(i)} \widetilde{\mathbf{y}}_{\Gamma_c}\right). \tag{3.8}$$

4. Some auxiliary results. In this section, we will collect a number of results which are needed in our theory. In order to avoid a proliferation of constants, we will use the notation $A \approx B$. This means that there are two constants c and C,

independent of any mesh parameter and the coefficients of (2.1), such that $cA \le B \le CA$, where $C < \infty$ and $c > 0$. For the definition of discrete harmonic functions, see [14, section 4.4].

LEMMA 4.1. [illegible] $\mathcal{D}$ [illegible] $A_1 = (0,0,0)$ $B_1 = (H,0,0)$ $C_1 = (H,H,0)$ $D_1 = (0,H,0)$ $A_2 = (0,0,H)$ $B_2 = (H,0,H)$ $C_2 = (H,H,H)$ [illegible] $D_2 = (0,H,H)$ [illegible] h [illegible] v [illegible] $\mathcal{D}$ [illegible] $\bar{v}_{A_1B_1} \approx 1 + \log \frac{H}{h}$ [illegible] $\bar{v}_{A_1B_1}$ [illegible] v [illegible] A_1B_1 $|v|^2_{H^1(\mathcal{D})} \approx H\left(1 + \log \frac{H}{h}\right)$ [illegible] v [illegible]. This lemma follows from a result by Brenner and He [1, Lemma 4.2]. Let N be an integer and G_N the function defined on $(0,1)$ by

$$G_N(x) = \sum_{n=1}^{N} \left(\frac{1}{4n-3} \sin\left((4n-3)\pi x\right) \right).$$

$G_N(x)$ is even with respect to the midpoint of $(0,1)$, where it attains its maximum in absolute value. Moreover, we have

$$|G_N|^2_{H_{00}^{1/2}(0,1)} \approx 1 + \log N \text{ and } \|G_N\|_{L^2(0,1)} \approx 1;$$

see [1, Lemma 3.7].

Let $[-H, 0]$ and $[0, H]$ have a mesh inherited from the quasi-uniform meshes on D_1A_1 and A_1B_1, respectively, and let $g_h(x)$ be the nodal interpolation of $G_N(\frac{x+H}{2H})$. Then, we have

$$\|g_h\|_{L^\infty(-H,H)} \approx 1 + \log \frac{H}{h}, \quad |g_h|^2_{H_{00}^{1/2}(-H,H)} \approx 1 + \log \frac{H}{h}, \tag{4.1}$$
$$\text{and} \quad \|g_h\|_{L^2(-H,H)} \approx H;$$

see [1, Lemma 3.7] or [17, Lemma 1].

Let $\tau_h(x)$ be a function on $[0, H]$ defined as follows:

$$\tau_h(x) = \begin{cases} \frac{x}{h_1}, & 0 \le x \le h_1, \\ 1, & h_1 \le x \le H - h_2, \\ \frac{H-x}{h_2}, & H - h_2 \le x \le H, \end{cases}$$

where h_1 and h_2 are the lengths of the two end mesh intervals of $[0, H]$. Then the following estimates hold:

$$\|\tau_h\|^2_{L^2(0,H)} \approx H \quad \text{and} \quad |\tau_h|^2_{H_{00}^{1/2}(0,H)} \approx 1 + \log \frac{H}{h}; \tag{4.2}$$

see [1, Lemma 3.6].

Define the discrete harmonic function v as 0 everywhere on the boundary of $\mathcal{D}$ except on the two open faces $A_1B_1C_1D_1$ and $A_1B_1B_2A_2$. On these two faces it is defined by

$$v(x_1, x_2, 0) = g_h(x_2)\tau_h(x_1) \quad \text{for } (x_1, x_2) \in A_1B_1C_1D_1,$$

$$v(x_1, 0, x_3) = g_h(-x_3)\tau_h(x_1) \quad \text{for } (x_1, x_3) \in A_1B_1B_2A_2.$$

It is clear that $\bar{v}_{A_1B_1} \approx 1 + \log \frac{H}{h}$ and that v has a zero average over the other edges. Since v is discrete harmonic in $\mathcal{D}$, we have

$$\begin{aligned} |v|^2_{H^1(\mathcal{D})} &\approx |v|^2_{H^{1/2}(\partial\mathcal{D})} \\ &\approx |g_h|^2_{H^{1/2}_{00}(-H,H)} \|\tau_h\|^2_{L^2(0,H)} + |\tau_h|^2_{H^{1/2}_{00}(0,H)} \|g_h\|^2_{L^2(-H,H)} \\ &\approx H\left(1 + \log \frac{H}{h}\right), \end{aligned}$$

where we have used (4.1), (4.2), and [1, Corollary 3.5]. ∎

In Lemma 4.1, we have constructed the function v for a cube $\mathcal{D}$. By using similar ideas, we can construct functions v for other shape-regular polyhedra which will satisfy similar properties and bounds.

LEMMA 4.2. Ω^i_j Ω^i $j = 1, \ldots, N_i$ $V^h_{i,j}$ Ω^i_j h $\mathcal{E}_k$ $k = 1, \ldots, K_j$ Ω^i_j u $\bar{u}_{\mathcal{E}_k}$ $u \in V^h_{i,j}$ Ω^i_j Ω^i_j $j = 1, \ldots, N_i$

$$\left(1 + \log \frac{H}{h}\right) \left(\sum_{j=1}^{N_i} |u|^2_{H^1(\Omega^i_j)}\right) \approx \sum_{j=1}^{N_i} \sum_{k_1,k_2=1}^{K_j} H |\bar{u}_{\mathcal{E}_{k_1}} - \bar{u}_{\mathcal{E}_{k_2}}|^2.$$

Without loss of generality, we assume that the subdomains are hexahedral. Denote the edges of the subdomain Ω^i_j by $\mathcal{E}_k$, $k = 1, \ldots, 12$, and denote the average values of u over these 12 edges by $\bar{u}_{\mathcal{E}_k}$, $k = 1, \ldots, 12$, respectively.

According to Lemma 4.1, we can construct 11 discrete harmonic functions ϕ_m, $m = 2, \ldots, 12$, on Ω^i_j such that

$$(\overline{\phi}_m)_{\mathcal{E}_k} = \begin{cases} (\bar{u}_{\mathcal{E}_m} - \bar{u}_{\mathcal{E}_1})\,(1 + \log \frac{H}{h}), & m = k, \\ 0, & m \neq k, \end{cases}$$

and with

$$|\phi_m|^2_{H^1(\Omega^i_j)} \approx (\bar{u}_{\mathcal{E}_m} - \bar{u}_{\mathcal{E}_1})^2 \, H\left(1 + \log \frac{H}{h}\right), \quad m = 2, \ldots, 12. \tag{4.3}$$

Let $v_j = \frac{1}{1+\log \frac{H}{h}} (\sum_{m=2}^{12} \phi_m) + \bar{u}_{\mathcal{E}_1}$; we then have $(\bar{v}_j)_{\mathcal{E}_k} = \bar{u}_{\mathcal{E}_k}$, for $k = 1, \ldots, 12$, and

$$\begin{aligned} |v_j|^2_{H^1(\Omega^i_j)} &= \Big|\frac{1}{1 + \log \frac{H}{h}} \left(\sum_{m=2}^{12} \phi_m\right) + \bar{u}_{\mathcal{E}_1}\Big|^2_{H^1(\Omega^i_j)} \\ &= \left(\frac{1}{1 + \log \frac{H}{h}}\right)^2 \left|\sum_{m=2}^{12} \phi_m\right|^2_{H^1(\Omega^i_j)} \leq 11 \left(\frac{1}{1 + \log \frac{H}{h}}\right)^2 \sum_{m=2}^{12} |\phi_m|^2_{H^1(\Omega^i_j)} \\ &\leq \left(\frac{1}{c^{1/2}(1 + \log \frac{H}{h})}\right)^2 H\left(1 + \log \frac{H}{h}\right) \sum_{m=2}^{12} (\bar{u}_{\mathcal{E}_m} - \bar{u}_{\mathcal{E}_1})^2 \\ &\leq \frac{1}{c(1 + \log \frac{H}{h})} \sum_{k=1}^{12} H (\bar{u}_{\mathcal{E}_k} - \bar{u}_{\mathcal{E}_1})^2. \end{aligned}$$

Here, we have used (4.3) for the penultimate inequality.

By the definition of u, we have

$$|u|^2_{H^1(\Omega^i_j)} \leq |v_j|^2_{H^1(\Omega^i_j)} \leq \frac{1}{c(1+\log\frac{H}{h})} \sum_{k=1}^{12} H(\bar{u}_{\mathcal{E}_k} - \bar{u}_{\mathcal{E}_1})^2.$$

Summing over all the subdomains in the subregion Ω^i, we have

$$c\left(1+\log\frac{H}{h}\right)\left(\sum_{j=1}^{N_i} |u|^2_{H^1(\Omega^i_j)}\right) \leq \sum_{j=1}^{N_i}\sum_{k=1}^{12} H(\bar{u}_{\mathcal{E}_k} - \bar{u}_{\mathcal{E}_1})^2.$$

This proves one side of the equivalence.

We prove the other side as follows:

$$\begin{aligned}
&\sum_{j=1}^{N_i}\sum_{k=1}^{12} H(\bar{u}_{\mathcal{E}_k} - \bar{u}_{\mathcal{E}_1})^2 = \sum_{j=1}^{N_i}\sum_{k=1}^{12} H|\overline{(u-\bar{u}_{\mathcal{E}_1})}_{\mathcal{E}_k}|^2 \\
&\leq C\left(\sum_{j=1}^{N_i}\sum_{k=1}^{12} H\frac{1}{H}\|u-\bar{u}_{\mathcal{E}_1}\|^2_{L^2(\mathcal{E}_k)}\right) \leq C\left(\sum_{j=1}^{N_i}\left(1+\log\frac{H}{h}\right)|u|^2_{H^1(\Omega^i_j)}\right) \\
&\leq C\left(1+\log\frac{H}{h}\right)\left(\sum_{j=1}^{N_i}|u|^2_{H^1(\Omega^i_j)}\right).
\end{aligned}$$

Here, we have used a standard finite element Sobolev inequality; see [14, Lemma 4.30] for the second inequality and [14, Lemma 4.16] for the penultimate inequality.

We complete the proof of the other side of the equivalence by using the triangle inequality. ∎

We now introduce a new mesh on each subregion; we follow [3, 13]. The purpose of introducing this mesh is to relate the quadratic form of Lemma 4.2 to one for a more conventional finite element space.

Given a subregion Ω^i and subdomains Ω^i_j, $j = 1, \ldots, N_i$, let $\mathcal{T}$ be a quasi-uniform subtriangulation of Ω^i such that its set of vertices includes the vertices and the midpoints of the edges of Ω^i_j. For the hexahedral case, we decomposed each hexahedron into eight hexahedra by connecting the midpoints of the edges. We then partition the vertices of the new mesh $\mathcal{T}$ into two sets. The midpoints of edges are called primary and the other vertices of the new mesh $\mathcal{T}$ are called secondary. We call two vertices in the triangulation $\mathcal{T}$ adjacent if there is an edge of $\mathcal{T}$ between them, as in the standard finite element context; see Figures 1 and 2.

Let $U_H(\Omega)$ be the continuous piecewise trilinear finite element function space with respect to the new triangulation $\mathcal{T}$. For a subregion Ω^i, $U_H(\Omega^i)$ and $U_H(\partial\Omega^i)$ are defined as restrictions:

$$U_H(\Omega^i) = \{u|_{\Omega^i} : \ u \in U_H(\Omega)\}, \qquad U_H(\partial\Omega^i) = \{u|_{\partial\Omega^i} : \ u \in U_H(\Omega)\}.$$

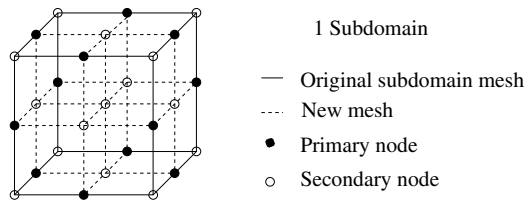


FIG. 1. *The new mesh and primary and secondary nodes in one subdomain of a subregion. (Note that all the lines of the original subdomain mesh are drawn in the same way.)*

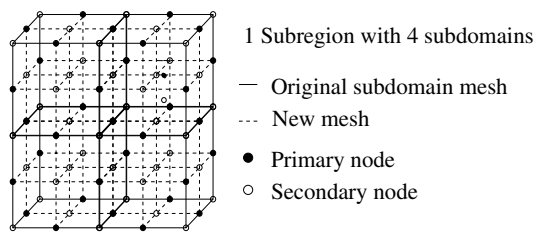


FIG. 2. *The new mesh and primary and secondary nodes in a subregion with four subdomains. (Note that all the lines of the original subdomain mesh are drawn in the same way.)*

We define a mapping $I_H^{\Omega^i}$ of any function ϕ, defined at the primary vertices in Ω^i, to $U_H(\Omega^i)$ by

(4.4)

$$I_H^{\Omega^i}\phi(x) = \begin{cases} \phi(x) \text{ if } x \text{ is a primary node;} \\ \text{the average of the values at all adjacent primary nodes} \\ \text{on the edges of } \Omega^i \text{ if } x \text{ is a vertex of } \Omega^i; \\ \text{the average of the values at two adjacent primary nodes} \\ \text{on the same edge of } \Omega^i \text{ if } x \text{ is an edge secondary node of } \Omega^i; \\ \text{the average of the values at all adjacent primary nodes on the} \\ \text{boundary of } \Omega^i \text{ if } x \text{ is a face secondary boundary node of } \Omega^i; \\ \text{the average of the values at all adjacent primary nodes} \\ \text{if } x \text{ is an interior secondary node of } \Omega^i \text{ with some adjacent} \\ \text{primary nodes;} \\ \text{the average of the values at all adjacent nodes} \\ \text{if } x \text{ is an interior secondary node of } \Omega^i \text{ without any adjacent} \\ \text{primary nodes;} \\ \text{the result of trilinear interpolation using the vertex values} \\ \text{if } x \text{ is not a vertex of } \mathcal{T}. \end{cases}$$

We recall that $\mathbf{W}_c^{(i)}$ is the discrete space of the values at the primary nodes given by the subdomain edge average values. $I_H^{\Omega^i}$ can be considered as a map from $\mathbf{W}_c^{(i)}$ to $U_H(\Omega^i)$ or as a map from $U_H(\Omega^i)$ to $U_H(\Omega^i)$.

Let $I_H^{\partial\Omega^i}$ be the mapping of a function ϕ defined at the primary vertices on the boundary of Ω^i to $U_H(\partial\Omega^i)$ and defined by $I_H^{\partial\Omega^i}\phi = (I_H^{\Omega^i}\phi_e)|_{\partial\Omega^i}$, where ϕ_e is any function in $\mathbf{W}_c^{(i)}$ such that $\phi_e|_{\partial\Omega^i} = \phi$. The map is well defined since the boundary values of $I_H^{\Omega^i}\phi_e$ depend only on the boundary values of ϕ_e.

Finally, let

$$\widetilde{U}_H(\Omega^i) = \{\psi = I_H^{\Omega^i}\phi,\ \phi \in U_H(\Omega^i)\}, \qquad \widetilde{U}_H(\partial\Omega^i) = \{\psi|_{\partial\Omega^i},\ \psi \in \widetilde{U}_H(\Omega^i)\}.$$

$I_H^{\partial\Omega^i}$ also can be considered as a map from $\mathbf{W}_{c,\Gamma_c}^{(i)}$ to $\widetilde{U}_H(\partial\Omega^i)$.

We carefully define the operators $I_H^{\Omega^i}$ and $I_H^{\partial\Omega^i}$ so that, if the edge averages of $w_i \in \mathbf{W}_{c,\Gamma_c}^{(i)}$ and $w_j \in \mathbf{W}_{c,\Gamma_c}^{(j)}$ over an edge $\mathcal{E}$ are the same, we have $(\overline{I_H^{\partial\Omega^i} w_i})_{\mathcal{E}} = (\overline{I_H^{\partial\Omega^j} w_j})_{\mathcal{E}}$. Here we need to use a weighted average which has a larger weight at the two end points since we consider an edge as an open set and the two end primary points have only one neighboring secondary node on the edge. But this will not affect our analysis. We could also define a weighted edge average of w_i and w_j and obtain $(\overline{I_H^{\partial\Omega^i} w_i})_{\mathcal{E}} = (\overline{I_H^{\partial\Omega^j} w_j})_{\mathcal{E}}$ for the usual average.

We list some useful lemmas from [3]. For proofs of Lemmas 4.3 and 4.4, see [3, Lemmas 6.1 and 6.2], respectively.

LEMMA 4.3. There exists a constant $C > 0$ such that for any H and $|\Omega^i|$ the

$$|I_H^{\Omega^i}\phi|_{H^1(\Omega^i)} \le C|\phi|_{H^1(\Omega^i)} \quad \text{and} \quad \|I_H^{\Omega^i}\phi\|_{L^2(\Omega^i)} \le C\|\phi\|_{L^2(\Omega^i)} \quad \forall \phi \in U_H(\Omega^i).$$

LEMMA 4.4. For $\hat{\phi} \in \widetilde{U}_H(\partial\Omega^i)$

$$\inf_{\phi \in \widetilde{U}_H(\Omega^i),\ \phi|_{\partial\Omega^i} = \hat{\phi}} \|\phi\|_{H^1(\Omega^i)} \approx \|\hat{\phi}\|_{H^{1/2}(\partial\Omega^i)},$$

$$\inf_{\phi \in \widetilde{U}_H(\Omega^i),\ \phi|_{\partial\Omega^i} = \hat{\phi}} |\phi|_{H^1(\Omega^i)} \approx |\hat{\phi}|_{H^{1/2}(\partial\Omega^i)}.$$

LEMMA 4.5. For $w_i \in \mathbf{W}_{c,\Gamma_c}^{(i)}$

$$\rho_i C_1 |I_H^{\partial\Omega^i} w_i|^2_{H^{1/2}(\partial\Omega^i)} \approx \left(1 + \log\frac{H}{h}\right)(T^{(i)} w_i, w_i),$$

where $(T^{(i)} w_i, w_i) = w_i^T T^{(i)} w_i = |w_i|^2_{T^{(i)}}$ and $T^{(i)} = S^{(i)}_{\Pi_{\Gamma_c\Gamma_c}} - S^{(i)}_{\Pi_{\Gamma_c I_c}} S^{(i)^{-1}}_{\Pi_{I_c I_c}} S^{(i)^T}_{\Pi_{\Gamma_c I_c}}$

By the definition of $T^{(i)}$, we have

$$\begin{aligned}
\left(1+\log\frac{H}{h}\right)(T^{(i)}w_i,w_i) &= \left(1+\log\frac{H}{h}\right)\inf_{v\in\mathbf{W}_c^{(i)},\ v|_{\partial\Omega^i}=w_i}|v|^2_{S_\Pi^{(i)}}\\
&= \inf_{v\in\mathbf{W}_c^{(i)},\ v|_{\partial\Omega^i}=w_i}\rho_i\left(1+\log\frac{H}{h}\right)\left(\sum_{j=1}^{N_i}\inf_{u\in V_{i,j}^h,\ \bar{u}_{\mathcal{E}_l}=v_l,\ \mathcal{E}_l\subset\partial\Omega_j^i}|u|^2_{H^1(\Omega_j^i)}\right)\\
&\approx \inf_{v\in\mathbf{W}_c^{(i)},\ v|_{\partial\Omega^i}=w_i}\rho_i\sum_{j=1}^{N_i}\sum_{k_1,k_2=1}^{K_j}H|\bar{u}_{\mathcal{E}_{k_1}}-\bar{u}_{\mathcal{E}_{k_2}}|^2\\
&\approx \inf_{v\in\mathbf{W}_c^{(i)},\ v|_{\partial\Omega^i}=w_i}\rho_i\sum_{j=1}^{N_i}\sum_{k_1,k_2=1}^{K_j}H|v_{k_1}-v_{k_2}|^2\\
&\approx \inf_{v\in\mathbf{W}_c^{(i)},\ v|_{\partial\Omega^i}=w_i}\rho_i|I_H^{\Omega^i}v|^2_{H^1(\Omega^i)} \approx \inf_{\phi\in\widetilde{U}_H(\Omega^i),\ \phi|_{\partial\Omega^i}=I_H^{\partial\Omega^i}w_i}\rho_i|\phi|^2_{H^1(\Omega^i)}\\
&\approx \rho_i|I_H^{\partial\Omega^i}w_i|_{H^{1/2}(\partial\Omega^i)}.
\end{aligned}$$

We use Lemma 4.2 for the third bound, the definitions of $I_H^{\Omega^i}$ and $I_H^{\partial\Omega^i}$ for the fourth and fifth bounds, and Lemma 4.4 for the final bound. ∎

To be fully rigorous, we assume that there is a quasi-uniform coarse triangulation of each subregion. We can then obtain uniform upper and lower bounds for each subregion as is required in Lemma 4.5.

We define the interface average operator E_{D_c} on $\widetilde{\mathbf{W}}_{c,\Gamma_c}$ as $E_{D_c}=\widetilde{R}_{\Gamma_c}\widetilde{R}^T_{D_c,\Gamma_c}$, which computes the averages across the subregion interface Γ_c and then adopts these averages at the boundary points of the subregions.

The interface average operator E_{D_c} has the following property.

LEMMA 4.6.

$$|E_{D_c}\mathbf{w}_{\Gamma_c}|^2_{\tilde{T}} \le C\left(1+\log\frac{\hat{H}}{H}\right)^2|\mathbf{w}_{\Gamma_c}|^2_{\tilde{T}}$$

$\mathbf{w}_{\Gamma_c}\in\widetilde{\mathbf{W}}_{c,\Gamma_c}$, C $\hat{H}$ H h (2.1) $\widetilde{T}$ (3.4).

Let $w_i=\overline{R}^{(i)}_{\Gamma_c}\mathbf{w}_{\Gamma_c}\in\mathbf{W}^{(i)}_{c,\Gamma_c}$, where $\overline{R}^{(i)}_{\Gamma_c}$ is the restriction operator from $\widetilde{\mathbf{W}}_{c,\Gamma_c}$ to $\mathbf{W}^{(i)}_{c,\Gamma_c}$. We rewrite the formula for $v:=w_{\Gamma_c}-E_{D_c}w_{\Gamma_c}$ for an arbitrary element $\mathbf{w}_{\Gamma_c}\in\widetilde{\mathbf{W}}_{c,\Gamma_c}$, and find that for $i=1,\dots,N_c$,

$$v_i(x):=(\mathbf{w}_{\Gamma_c}(x)-E_{D_c}\mathbf{w}_{\Gamma_c}(x))_i=\sum_{j\in\mathcal{N}_x}\delta^\dagger_{c,j}(w_i(x)-w_j(x)),\quad x\in\partial\Omega^i\cap\Gamma_c. \tag{4.5}$$

Here $\mathcal{N}_x$ is the set of indices of the subregions that have x on their boundaries.

We have

$$|E_{D_c}\mathbf{w}_{\Gamma_c}|^2_{\tilde{T}}=\sum_{i=1}^{N_c}|w_i-v_i|^2_{T^{(i)}}\le 2\sum_{i=1}^{N_c}|w_i|^2_{T^{(i)}}+2\sum_{i=1}^{N_c}|v_i|^2_{T^{(i)}}\text{ and }|\mathbf{w}_{\Gamma_c}|^2_{\tilde{T}}=\sum_{i=1}^{N_c}|w_i|^2_{T^{(i)}}.$$

We can therefore focus on the estimate of the contribution from a single subregion Ω^i and proceed as in the proof of [14, Lemma 6.36].

We will also use the simple inequality

$$\rho_i \delta_{c,j}^{\dagger^2} \leq \min(\rho_i, \rho_j) \text{ for } \gamma \in [1/2, \infty). \tag{4.6}$$

By Lemma 4.5,

$$(T^{(i)} v_i, v_i) \leq C \frac{1}{(1 + \log \frac{H}{h})} \rho_i |I_H^{\partial\Omega^i}(v_i)|^2_{H^{1/2}(\partial\Omega^i)}. \tag{4.7}$$

Let $l_i = I_H^{\partial\Omega^i}(v_i)$. By using a partition of unity as in [14, Lemma 6.36], we have

$$l_i = \sum_{\mathcal{F} \subset \partial\Omega^i} I^H(\theta_{\mathcal{F}} l_i) + \sum_{\mathcal{E} \subset \partial\Omega^i} I^H(\theta_{\mathcal{E}} l_i) + \sum_{\mathcal{V} \in \partial\Omega^i} \theta_{\mathcal{V}} l_i(\mathcal{V}),$$

where I^H is the nodal piecewise linear interpolant on the coarse mesh $\mathcal{T}$. We note that the analysis of face and edge terms is almost identical to that in [14, Lemma 6.36]. But the vertex terms are different because of $I_H^{\partial\Omega^i}$. We need only consider the vertex term when two subregions share at least an edge. This make the analysis simpler than in the proof of [14, Lemma 6.36].

Face terms. First, consider

$$I^H(\theta_{\mathcal{F}} l_i) = I^H(\theta_{\mathcal{F}} I_H^{\partial\Omega^i}(\delta_{c,j}^{\dagger}(w_i - w_j))).$$

Similar to [14, Lemma 6.36], we obtain, by using (4.6),

$$\begin{aligned}
&\rho_i |I^H(\theta_{\mathcal{F}} I_H^{\partial\Omega^i}(\delta_{c,j}^{\dagger}(w_i - w_j)))|^2_{H^{1/2}(\partial\Omega^i)} \\
&\quad = \rho_i \delta_{c,j}^{\dagger^2} |I^H(\theta_{\mathcal{F}} I_H^{\partial\Omega^i}(w_i - w_j))|^2_{H^{1/2}(\partial\Omega^i)} \\
&\quad \leq \min(\rho_i, \rho_j) |I^H(\theta_{\mathcal{F}}((I_H^{\partial\Omega^i} w_i - (\overline{I_H^{\partial\Omega^i} w_i})_{\mathcal{F}}) - (I_H^{\partial\Omega^i} w_j - (\overline{I_H^{\partial\Omega^i} w_j})_{\mathcal{F}}) \\
&\qquad + ((\overline{I_H^{\partial\Omega^i} w_i})_{\mathcal{F}} - (\overline{I_H^{\partial\Omega^i} w_j})_{\mathcal{F}})))|^2_{H^{1/2}(\partial\Omega^i)} \\
&\quad \leq 3 \min(\rho_i, \rho_j) \Big(|I^H(\theta_{\mathcal{F}}(I_H^{\partial\Omega^i} w_i - (\overline{I_H^{\partial\Omega^i} w_i})_{\mathcal{F}}))|^2_{H^{1/2}(\partial\Omega^i)} \\
&\qquad + |I^H(\theta_{\mathcal{F}}(I_H^{\partial\Omega^i} w_j - (\overline{I_H^{\partial\Omega^i} w_j})_{\mathcal{F}}))|^2_{H^{1/2}(\partial\Omega^i)} \\
&\qquad + |\theta_{\mathcal{F}}((\overline{I_H^{\partial\Omega^i} w_i})_{\mathcal{F}} - (\overline{I_H^{\partial\Omega^i} w_j})_{\mathcal{F}})|^2_{H^{1/2}(\partial\Omega^i)} \Big).
\end{aligned} \tag{4.8}$$

By the definition of $I_H^{\partial\Omega^i}$,

$$I^H(\theta_{\mathcal{F}}(I_H^{\partial\Omega^i} w_j)) = I^H(\theta_{\mathcal{F}}(I_H^{\partial\Omega^j} w_j)) \quad \text{and} \quad (\overline{I_H^{\partial\Omega^i} w_j})_{\mathcal{F}} = (\overline{I_H^{\partial\Omega^j} w_j})_{\mathcal{F}}.$$

By [14, Lemma 4.26], the first and second terms in (4.8) can be estimated as follows:

$$\begin{aligned}
&\min(\rho_i, \rho_j)(|I^H(\theta_{\mathcal{F}}(I_H^{\partial\Omega^i} w_i - (\overline{I_H^{\partial\Omega^i} w_i})_{\mathcal{F}}))|^2_{H^{1/2}(\partial\Omega^i)} \\
&+ |I^H(\theta_{\mathcal{F}}(I_H^{\partial\Omega^i} w_j - (\overline{I_H^{\partial\Omega^i} w_j})_{\mathcal{F}}))|^2_{H^{1/2}(\partial\Omega^i)}) \\
&\quad = \min(\rho_i, \rho_j)(|I^H(\theta_{\mathcal{F}}(I_H^{\partial\Omega^i} w_i - (\overline{I_H^{\partial\Omega^i} w_i})_{\mathcal{F}}))|^2_{H^{1/2}(\partial\Omega^i)} \\
&+ |I^H(\theta_{\mathcal{F}}(I_H^{\partial\Omega^j} w_j - (\overline{I_H^{\partial\Omega^j} w_j})_{\mathcal{F}}))|^2_{H^{1/2}(\partial\Omega^i)}) \\
&\quad \leq C \left(1 + \log \frac{\hat{H}}{H}\right)^2 \left(\rho_i |I_H^{\partial\Omega^i} w_i|^2_{H^{1/2}(\partial\Omega^i)} + \rho_j |I_H^{\partial\Omega^j} w_j|^2_{H^{1/2}(\partial\Omega^j)} \right).
\end{aligned}$$

Let $\mathcal{E} \subset \partial\mathcal{F}$. Since the edge averages of w_i and w_j are the same, we have, by the definition of $I_H^{\partial\Omega^i}$ and $I_H^{\partial\Omega^j}$, that $(\overline{I_H^{\partial\Omega^i} w_i})_{\mathcal{E}} = (\overline{I_H^{\partial\Omega^j} w_j})_{\mathcal{E}}$. As we have pointed out before, we use a weighted average which has a larger weight at the two end points.

We then have

$$\begin{aligned}
&|(\overline{I_H^{\partial\Omega^i} w_i})_{\mathcal{F}} - (\overline{I_H^{\partial\Omega^j} w_j})_{\mathcal{F}})|^2 \\
&\quad \leq 2\left(|(\overline{I_H^{\partial\Omega^i} w_i})_{\mathcal{E}} - (\overline{I_H^{\partial\Omega^i} w_i})_{\mathcal{F}}|^2 + |(\overline{I_H^{\partial\Omega^j} w_j})_{\mathcal{E}} - (\overline{I_H^{\partial\Omega^j} w_j})_{\mathcal{F}}|^2\right).
\end{aligned} \tag{4.9}$$

It is sufficient to consider the first term on the right-hand side. Using [14, Lemma 4.30], we find

$$\begin{aligned}
&|(\overline{I_H^{\partial\Omega^i} w_i})_{\mathcal{E}} - (\overline{I_H^{\partial\Omega^i} w_i})_{\mathcal{F}}|^2 \\
&\quad = |\overline{(I_H^{\partial\Omega^i} w_i - (\overline{I_H^{\partial\Omega^i} w_i})_{\mathcal{F}})_{\mathcal{E}}}|^2 \leq C/\hat{H}_i \|I_H^{\partial\Omega^i} w_i - (\overline{I_H^{\partial\Omega^i} w_i})_{\mathcal{F}}\|^2_{L^2(\mathcal{E})},
\end{aligned}$$

and, by using [14, Lemma 4.17] and the Poincaré inequality given as [14, Lemma A.17], we have

$$|(\overline{I_H^{\partial\Omega^i} w_i})_{\mathcal{E}} - (\overline{I_H^{\partial\Omega^i} w_i})_{\mathcal{F}}|^2 \leq \frac{C}{\hat{H}_i}\left(1 + \log\frac{\hat{H}}{H}\right)|I_H^{\partial\Omega^i} w_i - (\overline{I_H^{\partial\Omega^i} w_i})_{\mathcal{F}}|^2_{H^{1/2}(\mathcal{F})}.$$

Combining this with the bound for $\theta_{\mathcal{F}}$ in [14, Lemma 4.26], we have

$$\begin{aligned}
&\min(\rho_i, \rho_j)|\theta_{\mathcal{F}}((\overline{I_H^{\partial\Omega^i} w_i})_{\mathcal{F}} - (\overline{I_H^{\partial\Omega^i} w_j})_{\mathcal{F}})|^2_{H^{1/2}(\partial\Omega^i)} \\
&\quad \leq C\left(1 + \log\frac{\hat{H}}{H}\right)^2 \left(\rho_i |I_H^{\partial\Omega^i} w_i|^2_{H^{1/2}(\partial\Omega^i)} + \rho_j |I_H^{\partial\Omega^j} w_j|^2_{H^{1/2}(\partial\Omega^j)}\right).
\end{aligned}$$

Edge terms. We can develop the same estimate as in [14, Lemma 6.34]. For simplicity, we consider only an edge $\mathcal{E}$ common to four subregions Ω^i, Ω^j, Ω^k, and Ω^l. Then,

$$\begin{aligned}
&\rho_i |I^H(\theta_{\mathcal{E}} l_i)|^2_{H^{1/2}(\partial\Omega^i)} \\
&\quad \leq \rho_i \left(|I^H(\theta_{\mathcal{E}} I_H^{\partial\Omega^i}(\delta^\dagger_{c,j}(w_i - w_j)))|^2_{H^{1/2}(\partial\Omega^i)}\right. \\
&\quad + |I^H(\theta_{\mathcal{E}} I_H^{\partial\Omega^i}(\delta^\dagger_{c,k}(w_i - w_k)))|^2_{H^{1/2}(\partial\Omega^i)} \\
&\quad \left. + |I^H(\theta_{\mathcal{E}} I_H^{\partial\Omega^i}(\delta^\dagger_{c,l}(w_i - w_l)))|^2_{H^{1/2}(\partial\Omega^i)}\right).
\end{aligned} \tag{4.10}$$

We recall that $\delta^\dagger_{c,j}$, $\delta^\dagger_{c,k}$, and $\delta^\dagger_{c,l}$ are constants.

By the definition of $I_H^{\partial\Omega^i}$, $I_H^{\partial\Omega^j}$, $I_H^{\partial\Omega^k}$, and $I_H^{\partial\Omega^l}$, we have

$$\theta_{\mathcal{E}}(I_H^{\partial\Omega^i} w_j) = \theta_{\mathcal{E}}(I_H^{\partial\Omega^j} w_j), \quad \theta_{\mathcal{E}}(I_H^{\partial\Omega^i} w_k) = \theta_{\mathcal{E}}(I_H^{\partial\Omega^k} w_k), \quad \theta_{\mathcal{E}}(I_H^{\partial\Omega^i} w_l) = \theta_{\mathcal{E}}(I_H^{\partial\Omega^l} w_l),$$

and

$$(\overline{I_H^{\partial\Omega^i} w_i})_{\mathcal{E}} = (\overline{I_H^{\partial\Omega^j} w_j})_{\mathcal{E}} = (\overline{I_H^{\partial\Omega^k} w_k})_{\mathcal{E}} = (\overline{I_H^{\partial\Omega^l} w_l})_{\mathcal{E}}.$$

We assume that Ω^i shares a face with Ω^j as well as Ω^l and shares an edge only with Ω^k.

First, we consider the second term in (4.10). By [14, Lemmas 4.19 and 4.17] and (4.6), we have

$$
\begin{aligned}
&\rho_i |I^H(\theta_{\mathcal{E}} I_H^{\partial\Omega^i}(\delta_{c,k}^{\dagger}(w_i - w_k)))|^2_{H^{1/2}(\partial\Omega^i)} \\
&\quad \le C\rho_i \delta_{c,k}^{\dagger^2} \|I^H(\theta_{\mathcal{E}}(I_H^{\partial\Omega^i} w_i - (\overline{I_H^{\partial\Omega^i} w_i})_{\mathcal{E}}) - \theta_{\mathcal{E}}(I_H^{\partial\Omega^k} w_k - (\overline{I_H^{\partial\Omega^k} w_k})_{\mathcal{E}}))\|^2_{L^2(\mathcal{E})} \\
&\quad \le 2C \Big(\rho_i \|I^H(\theta_{\mathcal{E}}(I_H^{\partial\Omega^i} w_i - (\overline{I_H^{\partial\Omega^i} w_i})_{\mathcal{E}}))\|^2_{L^2(\mathcal{E})} \\
&\qquad + \rho_k \|I^H(\theta_{\mathcal{E}}(I_H^{\partial\Omega^k} w_k - (\overline{I_H^{\partial\Omega^k} w_k})_{\mathcal{E}}))\|^2_{L^2(\mathcal{E})}\Big) \\
&\quad \le 2C \Big(\rho_i \|I_H^{\partial\Omega^i} w_i - (\overline{I_H^{\partial\Omega^i} w_i})_{\mathcal{E}}\|^2_{L^2(\mathcal{E})} + \rho_k \|I_H^{\partial\Omega^k} w_k - (\overline{I_H^{\partial\Omega^k} w_k})_{\mathcal{E}}\|^2_{L^2(\mathcal{E})}\Big) \\
&\quad \le 2C \left(1 + \log \frac{\hat{H}}{H}\right) \Big(\rho_i |I_H^{\partial\Omega^i} w_i|^2_{H^{1/2}(\mathcal{F}^i)} + \rho_k |I_H^{\partial\Omega^k} w_k|^2_{H^{1/2}(\mathcal{F}^k)}\Big) \\
&\quad \le 2C \left(1 + \log \frac{\hat{H}}{H}\right) \Big(\rho_i |I_H^{\partial\Omega^i} w_i|^2_{H^{1/2}(\partial\Omega^i)} + \rho_k |I_H^{\partial\Omega^k} w_k|^2_{H^{1/2}(\partial\Omega^k)}\Big),
\end{aligned}
$$

where $\mathcal{F}^i$ is a face of Ω^i, $\mathcal{F}^k$ is a face of Ω^k, and $\mathcal{F}^i$ and $\mathcal{F}^k$ share the edge $\mathcal{E}$.

The first and third terms can be estimated similarly.

Vertex terms. We can apply techniques similar to those of the proof in [14, Lemma 6.36]. We have

$$
\rho_i |\theta_{\mathcal{V}} l_i(\mathcal{V})|^2_{H^{1/2}(\partial\Omega^i)} = \rho_i |\theta_{\mathcal{V}}(I_H^{\partial\Omega^i} v_i)(\mathcal{V})|^2_{H^{1/2}(\partial\Omega^i)}. \tag{4.11}
$$

By (4.5) and the definition of $I_H^{\partial\Omega^i}$, we see that $(I_H^{\partial\Omega^i} v_i)(\mathcal{V})$ is nonzero only when two subregions share one or several edges with a common vertex $\mathcal{V}$.

In the definition of $I_H^{\Omega^i}$, we denote by $\mathcal{E}_{i,m}$, $m = 1, 2, 3, \ldots$, the edges in $\partial\Omega^i$ which share $\mathcal{V}$. Denote by $p_{i,m}$ the primary nodes on the edges $\mathcal{E}_{i,m}$ which are adjacent to $\mathcal{V}$.

By the definition of $I_H^{\Omega^i}$, (4.11), and $|\theta_{\mathcal{V}}|^2_{H^{1/2}(\partial\Omega^i)} \le CH_i$, we have

$$
\begin{aligned}
\rho_i |\theta_{\mathcal{V}}(I_H^{\partial\Omega^i} v_i)(\mathcal{V})|^2_{H^{1/2}(\partial\Omega^i)} &\le C\rho_i |\sum_m v_i(p_{i,m})|^2 |\theta_{\mathcal{V}}|^2_{H^{1/2}(\partial\Omega^i)} \\
&\le C\rho_i H_i \sum_m |v_i(p_{i,m})|^2. \qquad (4.12)
\end{aligned}
$$

Let us look at the first term in (4.12); the other terms can be estimated in the

same way. We find that

$$
\begin{aligned}
&\rho_i H_i |v_i(p_{i,1})|^2 \\
&\quad = \rho_i H_i | \sum_{j,\ \mathcal{E}_{i,1}\subset\partial\Omega^j} \delta^\dagger_{c,j}(w_i(p_{i,1}) - w_j(p_{i,1}))|^2 \\
&\quad \le C \sum_{j,\ \mathcal{E}_{i,1}\subset\partial\Omega^j} \min(\rho_i,\rho_j) H_i |w_i(p_{i,1}) - w_j(p_{i,1})|^2 \\
&\quad = C \sum_{j,\ \mathcal{E}_{i,1}\subset\partial\Omega^j} \min(\rho_i,\rho_j) H_i |I_H^{\partial\Omega^i} w_i(p_{i,1}) - I_H^{\partial\Omega^j} w_j(p_{i,1})|^2 \\
&\quad \le C \sum_{j,\ \mathcal{E}_{i,1}\subset\partial\Omega^j} \min(\rho_i,\rho_j) H_i \left(|I_H^{\partial\Omega^i} w_i(p_{i,1}) - (\overline{I_H^{\partial\Omega^i} w_i})_{\mathcal{E}_{i,1}}|^2 \right. \\
&\qquad \left. + |I_H^{\partial\Omega^j} w_j(p_{i,1}) - (\overline{I_H^{\partial\Omega^j} w_j})_{\mathcal{E}_{i,1}}|^2 \right) \\
&\quad \le C \sum_{j,\ \mathcal{E}_{i,1}\subset\partial\Omega^j} \min(\rho_i,\rho_j) \left(H_i | \left(I_H^{\partial\Omega^i} w_i - (\overline{I_H^{\partial\Omega^i} w_i})_{\mathcal{E}_{i,1}} \right)(p_{i,1})|^2 \right. \\
&\qquad \left. + H_i | \left(I_H^{\partial\Omega^j} w_j - (\overline{I_H^{\partial\Omega^j} w_j})_{\mathcal{E}i,1} \right)(p_{i,1})|^2 \right) \\
&\quad \le C \sum_{j,\ \mathcal{E}i,1\subset\partial\Omega^j} \min(\rho_i,\rho_j) \left(\| I_H^{\partial\Omega^i} w_i - (\overline{I_H^{\partial\Omega^i} w_i})_{\mathcal{E}_{i,1}} \|^2_{L^2(\mathcal{E}_{i,1})} \right. \\
&\qquad \left. + \| I_H^{\partial\Omega^j} w_j - (\overline{I_H^{\partial\Omega^j} w_j})_{\mathcal{E}_{i,1}} \|^2_{L^2(\mathcal{E}_{i,1})} \right) \\
&\quad \le C \sum_{j,\ \mathcal{E}i,1\subset\partial\Omega^j} \left(1 + \log \frac{\hat{H}}{H} \right) \left(\rho_i |I_H^{\partial\Omega^i} w_i|^2_{H^{1/2}(\partial\Omega^i)} + \rho_j |I_H^{\partial\Omega^j} w_j|^2_{H^{1/2}(\partial\Omega^i)} \right).
\end{aligned}
$$

For the third equality, we use that $p_{i,1}$ is a primary node. For the fourth inequality, we use that $(\overline{I_H^{\partial\Omega^i} w_i})_{\mathcal{E}_{i,1}} = (\overline{I_H^{\partial\Omega^j} w_j})_{\mathcal{E}_{i,1}}$. We use [14, Lemmas B.5] for the sixth inequality and [14, Lemma 4.17] for the last inequality.

Combining all face, edge, and vertex estimates, we obtain

$$
\rho_i |I_H^{\partial\Omega^i}(v_i)|^2_{H^{1/2}(\partial\Omega^i)} \le C \left(1 + \log \frac{\hat{H}}{H} \right)^2 \sum_{j:\ \partial\Omega^j\cap\partial\Omega^i\neq\emptyset} \rho_j |I_H^{\partial\Omega^i}(w_j)|^2_{H^{1/2}(\partial\Omega^j)}. \tag{4.13}
$$

Using (4.13), Lemma 4.5, and (4.7), we obtain

$$
\begin{aligned}
(T^{(i)} v_i, v_i) = |v_i|^2_{T^{(i)}} &\le C \frac{1}{(1+\log\frac{H}{h})} \rho_i |I_H^{\partial\Omega^i}(v_i)|^2_{H^{1/2}(\partial\Omega^i)} \\
&\le C \frac{\left(1+\log\frac{\hat{H}}{H}\right)^2}{(1+\log\frac{H}{h})} \sum_{j:\ \partial\Omega^j\cap\partial\Omega^i\neq\emptyset} \rho_j |I_H^{\partial\Omega^i}(w_j)|^2_{H^{1/2}(\partial\Omega^j)} \\
&\le \frac{C}{c} \left(1 + \log \frac{\hat{H}}{H} \right)^2 \sum_{j:\ \partial\Omega^j\cap\partial\Omega^i\neq\emptyset} (T^{(j)} w_j, w_j) \\
&= \frac{C}{c} \left(1 + \log \frac{\hat{H}}{H} \right)^2 \sum_{j:\ \partial\Omega^j\cap\partial\Omega^i\neq\emptyset} |w_j|^2_{T^{(j)}}. \qquad \square
\end{aligned}
$$

LEMMA 4.7. [illegible] $\mathbf{u}_\Gamma \in \widehat{\mathbf{W}}_\Gamma$ [illegible] $\Psi = \Phi^T \widetilde{R}_{D,\Gamma} \mathbf{u}_\Gamma$ [illegible]

$$\Psi^T S_\Pi^{-1} \Psi \le \Psi^T \widetilde{S}_\Pi^{-1} \Psi \le C \left(1 + \log \frac{\hat{H}}{H}\right)^2 \Psi^T S_\Pi^{-1} \Psi.$$

[illegible] Using (3.2), (3.5), and (3.6), we have

$$\begin{aligned}
\Psi^T S_\Pi^{-1} \Psi &= \sum_{i=1}^{N_c} {\Psi_{I_c}^{(i)}}^T \mathbf{y}_{I_c}^{(i)} + \Psi_{\Gamma_c}^T \mathbf{y}_{\Gamma_c} \\
&= \sum_{i=1}^{N_c} {\Psi_{I_c}^{(i)}}^T \left({S_{\Pi_{I_c} I_c}^{(i)}}^{-1} (\Psi_{I_c}^{(i)} - S_{\Pi_{I_c}\Gamma_c}^{(i)} R_{\Gamma_c}^{(i)} \mathbf{y}_{\Gamma_c}) \right) + \left(\mathbf{h}_{\Gamma_c} + \sum_{i=1}^{N_c} {R_{\Gamma_c}^{(i)}}^T {S_{\Pi_{\Gamma_c} I_c}^{(i)}} {S_{\Pi_{I_c} I_c}^{(i)}}^{-1} \Psi_{I_c}^{(i)} \right)^T \mathbf{y}_{\Gamma_c} \\
&= \sum_{i=1}^{N_c} {\Psi_{I_c}^{(i)}}^T {S_{\Pi_{I_c} I_c}^{(i)}}^{-1} \Psi_{I_c}^{(i)} + \mathbf{h}_{\Gamma_c}^T \mathbf{y}_{\Gamma_c} = \sum_{i=1}^{N_c} {\Psi_{I_c}^{(i)}}^T {S_{\Pi_{I_c} I_c}^{(i)}}^{-1} \Psi_{I_c}^{(i)} + \mathbf{h}_{\Gamma_c}^T \left(\widetilde{R}_{\Gamma_c}^T \widetilde{T} \widetilde{R}_{\Gamma_c} \right)^{-1} \mathbf{h}_{\Gamma_c}.
\end{aligned}$$

Using (3.8), (3.5), and (3.7), we also have

$$\begin{aligned}
\Psi^T \widetilde{S}_\Pi^{-1} \Psi &= \sum_{i=1}^{N_c} {\Psi_{I_c}^{(i)}}^T \widetilde{\mathbf{y}}_{I_c{}^{(i)}} + \Psi_{\Gamma_c}^T \widetilde{\mathbf{y}}_{\Gamma_c} \\
&= \sum_{i=1}^{N_c} {\Psi_{I_c}^{(i)}}^T \left({S_{\Pi_{I_c} I_c}^{(i)}}^{-1} (\Psi_{I_c}^{(i)} - S_{\Pi_{I_c}\Gamma_c}^{(i)} R_{\Gamma_c}^{(i)} \widetilde{\mathbf{y}}_{\Gamma_c}) \right) + \left(\mathbf{h}_{\Gamma_c} + \sum_{i=1}^{N_c} {R_{\Gamma_c}^{(i)}}^T {S_{\Pi_{\Gamma_c} I_c}^{(i)}} {S_{\Pi_{I_c} I_c}^{(i)}}^{-1} \Psi_{I_c}^{(i)} \right)^T \widetilde{\mathbf{y}}_{\Gamma_c} \\
&= \sum_{i=1}^{N_c} {\Psi_{I_c}^{(i)}}^T {S_{\Pi_{I_c} I_c}^{(i)}}^{-1} \Psi_{I_c}^{(i)} + \mathbf{h}_{\Gamma_c}^T \widetilde{\mathbf{y}}_{\Gamma_c} = \sum_{i=1}^{N_c} {\Psi_{I_c}^{(i)}}^T {S_{\Pi_{I_c} I_c}^{(i)}}^{-1} \Psi_{I_c}^{(i)} + \mathbf{h}_{\Gamma_c}^T \left(\widetilde{R}_{D_c,\Gamma_c}^T \widetilde{T}^{-1} \widetilde{R}_{D_c,\Gamma_c} \right) \mathbf{h}_{\Gamma_c}.
\end{aligned}$$

We need only compare $\mathbf{h}_{\Gamma_c}^T (\widetilde{R}_{\Gamma_c}^T \widetilde{T} \widetilde{R}_{\Gamma_c})^{-1} \mathbf{h}_{\Gamma_c}$ and $\mathbf{h}_{\Gamma_c}^T (\widetilde{R}_{D_c,\Gamma_c}^T \widetilde{T}^{-1} \widetilde{R}_{D_c,\Gamma_c}) \mathbf{h}_{\Gamma_c}$ for any $\mathbf{h}_{\Gamma_c} \in \widehat{\mathbf{F}}_{\Gamma_c}$. We follow the proofs of [8, Theorem 1].

Let

$$\mathbf{w}_{\Gamma_c} = \left(\widetilde{R}_{\Gamma_c}^T \widetilde{T} \widetilde{R}_{\Gamma_c} \right)^{-1} \mathbf{h}_{\Gamma_c} \in \widehat{\mathbf{W}}_{c,\Gamma_c} \text{ and } \mathbf{v}_{\Gamma_c} = \widetilde{T}^{-1} \widetilde{R}_{D_c,\Gamma_c} \mathbf{h}_{\Gamma_c} \in \widetilde{\mathbf{W}}_{c,\Gamma_c}. \tag{4.14}$$

Noting the fact that $\widetilde{R}_{\Gamma_c}^T \widetilde{R}_{D_c,\Gamma_c} = \widetilde{R}_{D_c,\Gamma_c}^T \widetilde{R}_{\Gamma_c} = I$ and using (4.14), we have

$$\begin{aligned}
\mathbf{h}_{\Gamma_c}^T \left(\widetilde{R}_{\Gamma_c}^T \widetilde{T} \widetilde{R}_{\Gamma_c} \right)^{-1} \mathbf{h}_{\Gamma_c} &= \mathbf{h}_{\Gamma_c}^T \mathbf{w}_{\Gamma_c} = \mathbf{h}_{\Gamma_c}^T \widetilde{R}_{D_c,\Gamma_c}^T \widetilde{R}_{\Gamma_c} \mathbf{w}_{\Gamma_c} \\
&= \mathbf{h}_{\Gamma_c}^T \widetilde{R}_{D_c,\Gamma_c}^T \widetilde{T}^{-1} \widetilde{T} \widetilde{R}_{\Gamma_c} \mathbf{w}_{\Gamma_c} = \left(\widetilde{T}^{-1} \widetilde{R}_{D_c,\Gamma_c} \mathbf{h}_{\Gamma_c} \right)^T \widetilde{T} \widetilde{R}_{\Gamma_c} \mathbf{w}_{\Gamma_c} \\
&= \mathbf{v}_{\Gamma_c}^T \widetilde{T} \widetilde{R}_{\Gamma_c} \mathbf{w}_{\Gamma_c} = \langle \mathbf{v}_{\Gamma_c}, \widetilde{R}_{\Gamma_c} \mathbf{w}_{\Gamma_c} \rangle_{\widetilde{T}} \\
&\le \langle \mathbf{v}_{\Gamma_c}, \mathbf{v}_{\Gamma_c} \rangle_{\widetilde{T}}^{1/2} \langle \widetilde{R}_{\Gamma_c} \mathbf{w}_{\Gamma_c}, \widetilde{R}_{\Gamma_c} \mathbf{w}_{\Gamma_c} \rangle_{\widetilde{T}}^{1/2} \\
&= \left(\mathbf{h}_{\Gamma_c}^T \left(\widetilde{R}_{D_c,\Gamma_c}^T \widetilde{T}^{-1} \widetilde{R}_{D_c,\Gamma_c} \right) \mathbf{h}_{\Gamma_c} \right)^{1/2} \left(\mathbf{h}_{\Gamma_c}^T \left(\widetilde{R}_{\Gamma_c}^T \widetilde{T} \widetilde{R}_{\Gamma_c} \right)^{-1} \mathbf{h}_{\Gamma_c} \right)^{1/2}.
\end{aligned}$$

We obtain

$$\mathbf{h}_{\Gamma_c}^T \left(\widetilde{R}_{\Gamma_c}^T \widetilde{T} \widetilde{R}_{\Gamma_c} \right)^{-1} \mathbf{h}_{\Gamma_c} \le \mathbf{h}_{\Gamma_c}^T \left(\widetilde{R}_{D_c,\Gamma_c}^T \widetilde{T}^{-1} \widetilde{R}_{D_c,\Gamma_c} \right) \mathbf{h}_{\Gamma_c}.$$

On the other hand,

$$\begin{aligned}
&\mathbf{h}_{\Gamma_c}^T \left(\widetilde{R}_{D_c,\Gamma_c}^T \widetilde{T}^{-1} \widetilde{R}_{D_c,\Gamma_c}\right) \mathbf{h}_{\Gamma_c} \;=\; \mathbf{w}_{\Gamma_c}^T \left(\widetilde{R}_{\Gamma_c}^T \widetilde{T} \widetilde{R}_{\Gamma_c}\right) \left(\widetilde{R}_{D_c,\Gamma_c}^T \widetilde{T}^{-1} \widetilde{R}_{D_c,\Gamma_c}\right) \mathbf{h}_{\Gamma_c} \\
&= \left\langle \mathbf{w}_{\Gamma_c}, \widetilde{R}_{D_c,\Gamma_c}^T \left(\widetilde{T}^{-1} \widetilde{R}_{D_c,\Gamma_c} \mathbf{h}_{\Gamma_c}\right)\right\rangle_{\left(\widetilde{R}_{\Gamma_c}^T \widetilde{T} \widetilde{R}_{\Gamma_c}\right)} \;=\; \left\langle \mathbf{w}_{\Gamma_c}, \widetilde{R}_{D_c,\Gamma_c}^T \mathbf{v}_{\Gamma_c}\right\rangle_{\left(\widetilde{R}_{\Gamma_c}^T \widetilde{T} \widetilde{R}_{\Gamma_c}\right)} \\
&\le \langle \mathbf{w}_{\Gamma_c}, \mathbf{w}_{\Gamma_c}\rangle_{\left(\widetilde{R}_{\Gamma_c}^T \widetilde{T} \widetilde{R}_{\Gamma_c}\right)}^{1/2} \left\langle \widetilde{R}_{D_c,\Gamma_c}^T \mathbf{v}_{\Gamma_c}, \widetilde{R}_{D_c,\Gamma_c}^T \mathbf{v}_{\Gamma_c}\right\rangle_{\left(\widetilde{R}_{\Gamma_c}^T \widetilde{T} \widetilde{R}_{\Gamma_c}\right)}^{1/2} \\
&= \left(\mathbf{h}_{\Gamma_c}^T \left(\widetilde{R}_{\Gamma_c}^T \widetilde{T} \widetilde{R}_{\Gamma_c}\right)^{-1} \mathbf{h}_{\Gamma_c}\right)^{1/2} \left\langle \widetilde{R}_{\Gamma_c} \widetilde{R}_{D_c,\Gamma_c}^T \mathbf{v}_{\Gamma_c}, \widetilde{R}_{\Gamma_c} \widetilde{R}_{D_c,\Gamma_c}^T \mathbf{v}_{\Gamma_c}\right\rangle_{\widetilde{T}}^{1/2} \\
&= \left(\mathbf{h}_{\Gamma_c}^T \left(\widetilde{R}_{\Gamma_c}^T \widetilde{T} \widetilde{R}_{\Gamma_c}\right)^{-1} \mathbf{h}_{\Gamma_c}\right)^{1/2} |E_{D_c} \mathbf{v}_{\Gamma_c}|_{\widetilde{T}} \\
&\le C \left(1 + \log \frac{\hat{H}}{H}\right) \left(\mathbf{h}_{\Gamma_c}^T \left(\widetilde{R}_{\Gamma_c}^T \widetilde{T} \widetilde{R}_{\Gamma_c}\right)^{-1} \mathbf{h}_{\Gamma_c}\right)^{1/2} |\mathbf{v}_{\Gamma_c}|_{\widetilde{T}} \\
&= C \left(1 + \log \frac{\hat{H}}{H}\right) \left(\mathbf{h}_{\Gamma_c}^T \left(\widetilde{R}_{\Gamma_c}^T \widetilde{T} \widetilde{R}_{\Gamma_c}\right)^{-1} \mathbf{h}_{\Gamma_c}\right)^{1/2} \left(\mathbf{h}_{\Gamma_c}^T \left(\widetilde{R}_{D_c,\Gamma_c}^T \widetilde{T}^{-1} \widetilde{R}_{D_c,\Gamma_c}\right) \mathbf{h}_{\Gamma_c}\right)^{1/2},
\end{aligned}$$

where we use Lemma 4.6 for the penultimate inequality.

We finally obtain

$$\mathbf{h}_{\Gamma_c}^T \left(\widetilde{R}_{D_c,\Gamma_c}^T \widetilde{T}^{-1} \widetilde{R}_{D_c,\Gamma_c}\right) \mathbf{h}_{\Gamma_c} \le C \left(1 + \log \frac{\hat{H}}{H}\right)^2 \left(\mathbf{h}_{\Gamma_c}^T \left(\widetilde{R}_{\Gamma_c}^T \widetilde{T} \widetilde{R}_{\Gamma_c}\right)^{-1} \mathbf{h}_{\Gamma_c}\right). \qquad \square$$

5. Condition number estimate for the new preconditioner. In order to estimate the condition number for the system with the new preconditioner $\widetilde{M}^{-1}$, we compare it to the system with the preconditioner M^{-1}.

LEMMA 5.1. [illegible] $\mathbf{u}_\Gamma \in \widehat{\mathbf{W}}_\Gamma$

$$\mathbf{u}_\Gamma^T M^{-1} \mathbf{u}_\Gamma \le \mathbf{u}_\Gamma^T \widetilde{M}^{-1} \mathbf{u}_\Gamma \le C \left(1 + \log \frac{\hat{H}}{H}\right)^2 \mathbf{u}_\Gamma^T M^{-1} \mathbf{u}_\Gamma. \tag{5.1}$$

[illegible] We have, for any $\mathbf{u}_\Gamma \in \widehat{\mathbf{W}}_\Gamma$,

$$\begin{aligned}
&\mathbf{u}_\Gamma^T M^{-1} \mathbf{u}_\Gamma \\
&= \mathbf{u}_\Gamma^T \widetilde{R}_{D,\Gamma}^T \left\{ R_{\Gamma\Delta}^T \sum_{i=1}^N \begin{pmatrix} \mathbf{0} & R_\Delta^{(i)^T} \end{pmatrix} \begin{pmatrix} A_{II}^{(i)} & A_{I\Delta}^{(i)} \\ A_{\Delta I}^{(i)} & A_{\Delta\Delta}^{(i)} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{0} \\ R_\Delta^{(i)} \end{pmatrix} R_{\Gamma\Delta} \right\} \widetilde{R}_{D,\Gamma} \mathbf{u}_\Gamma \\
&+ \mathbf{u}_\Gamma^T \widetilde{R}_{D,\Gamma}^T \Phi S_\Pi^{-1} \Phi^T \widetilde{R}_{D,\Gamma} \mathbf{u}_\Gamma
\end{aligned}$$

and

$$\begin{aligned}
&\mathbf{u}_\Gamma^T \widetilde{M}^{-1} \mathbf{u}_\Gamma \\
&= \mathbf{u}_\Gamma^T \widetilde{R}_{D,\Gamma}^T \left\{ R_{\Gamma\Delta}^T \sum_{i=1}^N \begin{pmatrix} \mathbf{0} & R_\Delta^{(i)^T} \end{pmatrix} \begin{pmatrix} A_{II}^{(i)} & A_{I\Delta}^{(i)} \\ A_{\Delta I}^{(i)} & A_{\Delta\Delta}^{(i)} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{0} \\ R_\Delta^{(i)} \end{pmatrix} R_{\Gamma\Delta} \right\} \widetilde{R}_{D,\Gamma} \mathbf{u}_\Gamma \\
&+ \mathbf{u}_\Gamma^T \widetilde{R}_{D,\Gamma}^T \Phi \widetilde{S}_\Pi^{-1} \Phi^T \widetilde{R}_{D,\Gamma} \mathbf{u}_\Gamma.
\end{aligned}$$

We obtain our result by using Lemma 4.7. ∎

THEOREM 5.2. $\widetilde{M}^{-1}$ $C(1+\log\frac{\hat{H}}{H})^2(1+\log\frac{H}{h})^2$

Combining the condition number bound, given in (2.7) for the two-level BDDC method, and Lemma 5.1, we find that the condition number for the three-level method is bounded by $C(1+\log\frac{\hat{H}}{H})^2(1+\log\frac{H}{h})^2$. ∎

6. Using Chebyshev iterations. Another approach to the three-level BDDC methods is to use an iterative method with a preconditioner to solve (3.6). Here, we consider a Chebyshev method with a fixed number of iterations and use $\widetilde{R}^T_{D_c,\Gamma_c}\widetilde{T}^{-1}\widetilde{R}_{D_c,\Gamma_c}$ as a preconditioner. Denoting the eigenvalues of $(\widetilde{R}^T_{D_c,\Gamma_c}\widetilde{T}^{-1}\widetilde{R}_{D_c,\Gamma_c})(\widetilde{R}^T_{\Gamma_c}\widetilde{T}\widetilde{R}_{\Gamma_c})$ by λ_j, we need two input parameters l and u, which are estimates for the minimum and maximum values of λ_j, for the Chebyshev iterations. From our analysis above, we know that $l=1$ and $\max_j\lambda_j \le C(1+\log\frac{\hat{H}}{H})^2(1+\log\frac{H}{h})^2$. We can use the conjugate gradient method to obtain an estimate for the largest eigenvalue at the beginning of the computation to choose a proper u.

Let $\alpha=\frac{2}{l+u}$, $\mu=\frac{u+l}{u-l}$, and $\sigma_j=1-\alpha\lambda_j$. As for the two-dimensional case in [17, section 6], we have the following theorem. No new ideas are required.

THEOREM 6.1. $\widehat{M}^{-1}$ k $C\frac{C_2(k)}{C_1(k)}(1+\log\frac{H}{h})^2$

$$C_1(k)=\min_j\left(1-\frac{\cosh(k\cosh^{-1}(\mu\sigma_j))}{\cosh(k\cosh^{-1}(\mu))}\right),$$

$$C_2(k)=\max_j\left(1-\frac{\cosh(k\cosh^{-1}(\mu\sigma_j))}{\cosh(k\cosh^{-1}(\mu))}\right),$$

$\frac{C_2(k)}{C_1(k)}\to 1$ $k\to\infty$

7. Numerical experiments. We have applied our two three-level BDDC algorithms to the model problem (2.1), where $\Omega=[0,1]^3$. We decompose the unit cube into $\widehat{N}\times\widehat{N}\times\widehat{N}$ subregions with the side-length $\hat{H}=1/\widehat{N}$ and each subregion into $N\times N\times N$ subdomains with the side-length $H=\hat{H}/N$. Equation (2.1) is discretized, in each subdomain, by conforming piecewise trilinear elements with an element diameter h. The preconditioned conjugate gradient iteration is stopped when the norm of the residual has been reduced by a factor of 10^{-6}.

We have carried out two different sets of experiments to obtain iteration counts and condition number estimates. All the experimental results are fully consistent with our theory.

In the first set of experiments, we use the first preconditioner $\widetilde{M}^{-1}$. We take the coefficient $\rho\equiv 1$ in case 1. In case 2, ρ is constant in one direction with a checkerboard pattern in the cross sections, where we take $\rho=1$ or $\rho=100$. The coefficients in both cases satisfy [14, Assumption 6.27.2]; i.e., for all pairs of subdomains which have a vertex but not an edge in common, there exists an acceptable edge path (see [14, Definition 6.26]) between these two subdomains. Table 1 gives the iteration counts and condition number estimates with a change of the number of subregions. We find that the condition numbers are independent of the number of subregions. Table 2 gives results with a change of the number of subdomains and the size of the subdomain problems.

TABLE 1
Eigenvalue bounds and iteration counts with the preconditioner $\widetilde{M}^{-1}$ with a change of the number of subregions, $\frac{\hat{H}}{H} = 3$ and $\frac{H}{h} = 3$.

	Case 1		Case 2	
Num. of subregions	Iter.	Cond. #	Iter.	Cond. #
$3 \times 3 \times 3$	9	2.6603	9	2.2559
$4 \times 4 \times 4$	10	2.8701	10	2.5245
$5 \times 5 \times 5$	11	2.9668	11	2.8074
$6 \times 6 \times 6$	11	3.0190	11	2.8477

TABLE 2
Eigenvalue bounds and iteration counts with the preconditioner $\widetilde{M}^{-1}$ with a change of the number of subdomains and the size of subdomain problems with $3 \times 3 \times 3$ subregions.

	Case 1		Case 2			Case 1		Case 2	
$\frac{\hat{H}}{H}$	Iter.	Cond. #	Iter.	Cond. #	$\frac{H}{h}$	Iter.	Cond. #	Iter.	Cond. #
3	9	2.6603	9	2.2559	3	9	2.6603	9	2.2559
4	9	3.0446	10	2.5183	4	9	2.7261	10	2.3299
5	10	3.3570	11	2.7782	5	10	2.8381	10	2.4353
6	10	3.6402	11	3.0078	6	10	2.9601	11	2.5488

TABLE 3
Eigenvalue bounds and iteration counts with the preconditioner $\widehat{M}^{-1}$, $u = 2.3$, $3 \times 3 \times 3$ subregions, $\frac{\hat{H}}{H} = 6$, and $\frac{H}{h} = 3$.

k	Iter.	$C_1(k)$	$\lambda_{\min}$	$\lambda_{\max}$	Cond. #
1	13	0.6061	0.6167	2.3309	3.7797
2	9	0.9159	0.9255	1.8968	2.0496
3	8	0.9827	1.0000	1.8835	1.8836
4	8	0.9964	1.0016	1.8854	1.8825
5	8	0.9993	1.0009	1.8797	1.8780

TABLE 4
Eigenvalue bounds and iteration counts with the preconditioner $\widehat{M}^{-1}$, $u = 3$, $3\times3\times3$ subregions, $\frac{\hat{H}}{H} = 6$, and $\frac{H}{h} = 3$.

k	Iter.	$C_1(k)$	$\lambda_{\min}$	$\lambda_{\max}$	Cond. #
1	15	0.5000	0.5093	2.0150	3.9562
2	10	0.8571	0.8678	1.9744	2.2753
3	8	0.9615	0.9900	1.8821	1.9012
4	8	0.9897	1.0015	1.8955	1.8927
5	8	0.9972	1.0020	1.8903	1.8866

In the second set of experiments, we use the second preconditioner $\widehat{M}^{-1}$ and take the coefficient $\rho \equiv 1$. We use the preconditioned conjugate gradient (PCG) to estimate the largest eigenvalue of $(\widetilde{R}^T_{D_c,\Gamma_c}\widetilde{T}^{-1}\widetilde{R}_{D_c,\Gamma_c})(\widetilde{R}^T_{\Gamma_c}\widetilde{T}\widetilde{R}_{\Gamma_c})$, which is approximately 2.3249. For $18 \times 18 \times 18$ subdomains and $\frac{H}{h} = 3$, we have a condition number estimate of 1.8767 for the two-level preconditioned BDDC operator. We select different values of u, the upper bound eigenvalue estimate of the preconditioned system, and of k to see how the condition number changes. We take $u = 2.3$ and $u = 3$ in Tables 3 and 4, respectively. We also evaluate $C_1(k)$ for $k = 1, 2, 3, 4, 5$. From these two tables, we find that the smallest eigenvalue is bounded from below by $C_1(k)$ and the condition number estimate becomes closer to 1.8767, the value for the two-level case, as k increases. We also see that if we can get a more precise estimate for the largest

eigenvalue of $(\widetilde{R}^T_{D_c,\Gamma_c}\widetilde{T}^{-1}\widetilde{R}_{D_c,\Gamma_c})(\widetilde{R}^T_{\Gamma_c}\widetilde{T}\widetilde{R}_{\Gamma_c})$, we need fewer Chebyshev iterations to get a condition number close to that of the two-level case. However, the iteration count is not very sensitive to the choice of u.

Acknowledgment. The author is grateful to Professor Olof Widlund for suggesting this problem and all the help and time he has devoted to this work.

REFERENCES

[1] S. C. Brenner and Q. He, *Lower bounds for three-dimensional nonoverlapping domain decomposition algorithms*, Numer. Math., 93 (2003), pp. 445–470.

[2] S. C. Brenner and L.-Y. Sung, *BDDC and FETI-DP without matrices or vectors*, Comput. Methods Appl. Mech. Engrg., 196 (2007), pp. 1429–1435.

[3] L. C. Cowsar, J. Mandel, and M. F. Wheeler, *Balancing domain decomposition for mixed finite elements*, Math. Comp., 64 (1995), pp. 989–1015.

[4] C. R. Dohrmann, *A preconditioner for substructuring based on constrained energy minimization*, SIAM J. Sci. Comput., 25 (2003), pp. 246–258.

[5] C. R. Dohrmann, *An approximate BDDC preconditioner*, Numer. Linear Algebra Appl., 14 (2007), pp. 149–168.

[6] A. Klawonn and O. Rheinbach, *Inexact FETI-DP methods*, Internat. J. Numer. Methods Engrg., 69 (2007), pp. 284–307.

[7] A. Klawonn and O. B. Widlund, *Dual-primal FETI methods for linear elasticity*, Comm. Pure Appl. Math., 59 (2006), pp. 1523–1572.

[8] J. Li and O. Widlund, *BDDC algorithms for incompressible Stokes equations*, SIAM J. Numer. Anal., 44 (2006), pp. 2432–2455.

[9] J. Li and O. B. Widlund, *FETI–DP, BDDC, and block Cholesky methods*, Internat. J. Numer. Methods Engrg., 66 (2006), pp. 250–271.

[10] J. Li and O. B. Widlund, *On the use of inexact subdomain solvers for BDDC algorithms.*, Comput. Methods Appl. Mech. Engrg., 196 (2007), pp. 1415–1428.

[11] J. Mandel and C. R. Dohrmann, *Convergence of a balancing domain decomposition by constraints and energy minimization*, Numer. Linear Algebra Appl., 10 (2003), pp. 639–659.

[12] J. Mandel, C. R. Dohrmann, and R. Tezaur, *An algebraic theory for primal and dual substructuring methods by constraints*, Appl. Numer. Math., 54 (2005), pp. 167–193.

[13] M. V. Sarkis, *Nonstandard coarse spaces and Schwarz methods for elliptic problems with discontinuous coefficients using non-conforming elements*, Numer. Math., 77 (1997), pp. 383–406.

[14] A. Toselli and O. B. Widlund, *Domain Decomposition Methods—Algorithms and Theory*, Springer Ser. Comput. Math. 34, Springer-Verlag, Berlin, 2005.

[15] X. Tu, *BDDC Domain Decomposition Algorithms: Methods with Three Levels and for Flow in Porous Media*, Ph.D. thesis, Courant Institute, New York University, New York, NY, 2006; Tech. report TR2005-879, Department of Computer Science, Courant Institute, New York University, NY, 2005.

[16] X. Tu, *Three-level BDDC*, in Domain Decomposition Methods in Science and Engineering XVI, O. B. Widlund and D. E. Keyes, eds., Lecture Notes in Comput. Sci. Engrg. 55, Springer-Verlag, Berlin, 2006, pp. 439–446.

[17] X. Tu, *Three-level BDDC in two dimensions*, Internat. J. Numer. Methods Engrg., 69 (2007), pp. 33–59.

SIAM J. SCI. COMPUT.
Vol. 29, No. 4, pp. 1781–1824
© 2007 Society for Industrial and Applied Mathematics

A CELL-CENTERED LAGRANGIAN SCHEME FOR TWO-DIMENSIONAL COMPRESSIBLE FLOW PROBLEMS*

PIERRE-HENRI MAIRE†, RÉMI ABGRALL‡, JÉRÔME BREIL†, AND JEAN OVADIA§

Abstract. We present a new Lagrangian cell-centered scheme for two-dimensional compressible flows. The primary variables in this new scheme are cell-centered, i.e., density, momentum, and total energy are defined by their mean values in the cells. The vertex velocities and the numerical fluxes through the cell interfaces are not computed independently, contrary to standard approaches, but are evaluated in a consistent manner due to an original solver located at the nodes. The main new feature of the algorithm is the introduction of four pressures on each edge, two for each node on each side of the edge. This extra degree of freedom allows us to construct a nodal solver which fulfills two properties. First, the conservation of momentum and total energy is ensured. Second, a semidiscrete entropy inequality is provided. In the case of a one-dimensional flow, the solver reduces to the classical Godunov acoustic solver: it can be considered as its two-dimensional generalization. Many numerical tests are presented. They are representative test cases for compressible flows and demonstrate the robustness and the accuracy of this new solver.

Key words. Godunov-type schemes, hyperbolic conservation laws, Lagrangian gas dynamics

AMS subject classifications. 65M06, 76L05, 76N05

DOI. 10.1137/050633019

1. Introduction.

The physical model that is considered throughout this paper is based on the equations of fluid dynamics written in Lagrangian form. This form is well adapted to the simulation of multimaterial compressible fluid flows, such as those encountered in the domain of inertial confinement fusion (ICF); see [19] and [25]. Our aim is to propose a new Lagrangian cell-centered scheme for two-dimensional gas dynamics equations. Before describing our method, let us briefly give a historical overview of the Lagrangian schemes.

Lagrangian schemes are characterized by a mesh that follows the fluid flow. In this way, these methods treat interfaces in a natural manner. The main numerical difficulty lies in the node motion discretization, especially for multidimensional situations. The most natural way to overcome this difficulty is to use a staggered discretization, where the momentum is defined at the nodes and the other variables (density, pressure, and specific internal energy) are cell-centered. This type of scheme was first introduced by von Neumann and Richtmyer in [27] for one-dimensional flows. The bidimensional extension was proposed by Wilkins in [29]. It is based on an internal energy formulation. The entropy production inherent to shock waves is ensured by an artificial viscosity. In its initial version, this scheme was not conservative, and it admitted numerical spurious modes. However, in spite of these drawbacks, this scheme has been widely used in the domain of multimaterial flow simulations during the last forty years. Moreover, in the past decade, many improvements have been made in order to solve the previous problems. In their paper [7], Caramana and Shashkov show

*Received by the editors June 3, 2005; accepted for publication (in revised form) March 8, 2007; published electronically September 19, 2007.
http://www.siam.org/journals/sisc/29-4/63301.html
†UMR CELIA CEA–CNRS–Université Bordeaux I, 33405 Talence Cedex, France (maire@celia.u-bordeaux1.fr, breil@celia.u-bordeaux1.fr).
‡Institut Universitaire de France et Mathématiques Appliquées de Bordeaux, Université Bordeaux I, 33405 Talence Cedex, France (abgrall@math.u.-bordeaux1.fr).
§CEA CESTA BP2, 33114 Le Barp, France (jean.ovadia@cea.fr).

that with an appropriate discretization of the subzonal forces resulting from subzonal pressures, hourglass motion and spurious vorticity can be eliminated. By using the method of support operators proposed in [6], they constructed a staggered scheme which ensures the conservation of total energy. Moreover, the discretization of artificial viscosity has been considerably improved: first, by introducing formulations for multidimensional shock wave computations in [5] and then by using a discretization based on a mimetic finite difference scheme in [4]. With all these improvements, the staggered Lagrangian scheme is an accurate and robust method, which can produce impressive results, even on unstructured polygonal grids; see, for instance, [21].

An alternative to the staggered discretization is to use a conservative cell-centered discretization. This method for Lagrangian gas dynamics in one dimension was introduced by Godunov; see [12] and [24]. The multidimensional extension of this method was performed during the 1980s [2], [11]. This multidimensional scheme is a cell-centered finite volume scheme on moving structured or unstructured meshes. It is constructed by integrating directly the system of conservation laws on each moving cell. The primary variables—density, momentum, and total energy—are defined in the cells. The flux across the boundary of the cell is computed by solving exactly or approximately a one-dimensional Riemann problem in the direction normal to the boundary. The main problem with this type of method lies in the fact that the node velocity needed to move the mesh cannot be directly calculated. In [2], the node velocity is computed via a special least squares procedure. It consists in minimizing the error between the normal velocity coming from the Riemann solver and the normal projection of the vertex velocity. It turns out that it leads to an artificial grid motion, which requires a very expensive treatment [10]. Moreover, with this approach the flux calculation is not consistent with the node motion.

Recently, new cell-centered methods have been proposed in [15] and [1]. These new approaches use a fully Lagrangian form of the gas dynamics equations; that is, the gradient and divergence operators are expressed in the Lagrangian coordinates. This type of discretization needs to compute the Jacobian matrix associated to the map between Lagrangian and Eulerian spaces. However, these methods are purely Lagrangian and cannot be interpreted as moving mesh methods. This drawback has motivated another approach proposed by Després and Mazeran. In [9], they made a theoretical analysis of the Lagrangian gas dynamics equations written in a fully Lagrangian form and they derived a new conservative and entropy consistent two-dimensional Lagrangian scheme of the finite volume type. It is a moving grid scheme based on a nodal solver. The node velocity is computed in a coherent manner with the face fluxes. The numerical results shown in [9] are quite impressive, in particular, those related to the difficult Saltzmann's test case. However, it appears that in the case of one-dimensional flows, this scheme leads to a nodal velocity, which depends on the cell aspect ratio. This drawback has motivated our study to develop a new cell-centered scheme that retains the good feature of the Després–Mazeran scheme but resolves the aspect ratio problem. Here, we propose a new Lagrangian cell-centered scheme in which the vertex velocities and the numerical fluxes through the cell interfaces are not computed independently as usual but in a consistent manner with an original solver located at the nodes. The main new feature of the algorithm is the introduction of four pressures on each edge, two for each node on each side of the edge. This is the main difference from [9]. We show, in the limit of a one-dimensional flow computed by our two-dimensional solver, or for flows in a cylindrical geometry, that the scheme recovers the classical Godunov approximate Riemann solver. Moreover, our scheme locally conserves momentum and total energy and satisfies a local entropy inequality.

The boundary conditions are taken into account in a natural way. This scheme is first order in time and space.

The paper is organized as follows. First we recall the gas dynamics equations written in the Lagrangian form. In section 2 we derive space approximations based on face and node fluxes. Then we build a nodal solver using conservation arguments and a discrete entropy inequality. In section 4 we give the time discretization. Finally, we validate our new scheme with several test cases. They are representative test cases for compressible fluid flows and demonstrate the robustness and the accuracy of this new solver.

2. Derivation of the Euler equations in the Lagrangian formalism. The aim of this section is to recall briefly how the gas dynamics equations are obtained in the Lagrangian formalism. For such a derivation, we follow the approach developed in [8].

Let $\mathcal{D}$ be a region of the two-dimensional space $\mathbb{R}^2$, filled with an inviscid fluid and equipped with an orthonormal frame (O, X, Y). The Lagrangian formalism consists in following the time evolution of fluid particles along their trajectories. Consider a fluid particle which is moving through the point M at time $t > 0$ and whose initial position is point m. The coordinates of point M are denoted by (X, Y) and are named Eulerian coordinates. The coordinates of point m are denoted by (x, y) and are named Lagrangian coordinates. The Eulerian coordinates are obtained from the trajectory equations

$$\begin{aligned} \frac{dX}{dt} &= u(X, Y, t), \quad X(x, y, 0) = x, \\ \frac{dY}{dt} &= v(X, Y, t), \quad Y(x, y, 0) = y, \end{aligned} \tag{2.1}$$

where (u, v) are the coordinates of the fluid velocity $\boldsymbol{V}$. If the velocity field is smooth enough, the system (2.1) admits a unique solution $(X(x, y, t), Y(x, y, t))$. This enables us to define the map

$$\mathcal{M}_t : (x, y) \mapsto (X, Y), \tag{2.2}$$

where (X, Y) is the unique solution of (2.1). With fixed t, this map advances each fluid particle from its position at time $t = 0$ to its position at time t. If ω denotes a region in $\mathcal{D}$, then $\mathcal{M}_t(\omega) = \Omega$ is the volume ω moving with the fluid. We assume that for each $t > 0$, $\mathcal{M}_t$ is invertible. Let us introduce the Jacobian of this map:

$$J(x, y, t) = \begin{vmatrix} \dfrac{\partial X}{\partial x} & \dfrac{\partial X}{\partial y} \\ \dfrac{\partial Y}{\partial x} & \dfrac{\partial Y}{\partial y} \end{vmatrix}. \tag{2.3}$$

We notice that $J(x, y, 0) = 1$ and since $\mathcal{M}_t$ is invertible we have for each $t > 0$, $J(x, y, t) > 0$. Time differentiation of (2.3) gives the classical result

$$\frac{dJ}{dt} - J\nabla \cdot \boldsymbol{V} = 0, \tag{2.4}$$

where $\nabla\cdot$ denotes the divergence operator, namely, $\nabla \cdot \boldsymbol{V} = \frac{\partial u}{\partial X} + \frac{\partial v}{\partial Y}$.

Let us consider a flow variable φ depending on Eulerian coordinates: $\varphi \equiv \varphi(X, Y, t)$. Time differentiation of φ gives

$$\frac{d\varphi}{dt} = \frac{\partial \varphi}{\partial t} + \frac{dX}{dt}\frac{\partial \varphi}{\partial X} + \frac{dY}{dt}\frac{\partial \varphi}{\partial Y},$$

and using (2.1) one obtains

$$\frac{d\varphi}{dt} = \frac{\partial \varphi}{\partial t} + \boldsymbol{\nabla}\varphi \cdot \boldsymbol{V}, \tag{2.5}$$

where $\boldsymbol{\nabla}$ is the gradient operator, namely $\boldsymbol{\nabla}\varphi = (\frac{\partial \varphi}{\partial X}, \frac{\partial \varphi}{\partial Y})^t$. The time derivative of φ depending on Eulerian coordinates is named the material derivative; it represents the variation of φ along a fluid trajectory. Finally, combining (2.4) and (2.5) it is straightforward to write

$$\frac{d}{dt}(\varphi J) = J\left[\frac{\partial \varphi}{\partial t} + \nabla \cdot (\varphi \boldsymbol{V})\right]. \tag{2.6}$$

With this last equation we are able to transform gas dynamics equations written in Eulerian formalism. These equations for an inviscid compressible fluid (see [18]) are

$$\begin{aligned} &\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{V}) = 0, \\ &\frac{\partial}{\partial t}(\rho \boldsymbol{V}) + \boldsymbol{\nabla} \cdot (\rho \boldsymbol{V} \otimes \boldsymbol{V}) + \boldsymbol{\nabla} P = \mathbf{0}, \\ &\frac{\partial}{\partial t}(\rho E) + \nabla \cdot (\rho E \boldsymbol{V}) + \nabla \cdot (P \boldsymbol{V}) = 0. \end{aligned} \tag{2.7}$$

In (2.7), ρ denotes the density, P is the pressure, and E is the specific total energy. We denote by $\varepsilon = E - \frac{1}{2}\|\boldsymbol{V}\|^2$ the specific internal energy. The thermodynamic closure of (2.7) is given by the equation of state, $P \equiv P(\rho, \varepsilon)$.

Now, using (2.6) for the conservative variables $\varphi = 1,\ \rho,\ \rho\boldsymbol{V},\ \rho E$ and after substitution in (2.7), one obtains the gas dynamics system in Lagrangian formalism:

$$\begin{aligned} &\frac{dJ}{dt} - J\nabla \cdot \boldsymbol{V} = 0, \\ &\frac{d}{dt}(\rho J) = 0, \\ &\frac{d}{dt}(\rho J \boldsymbol{V}) + J\boldsymbol{\nabla} P = \mathbf{0}, \\ &\frac{d}{dt}(\rho J E) + J\nabla \cdot (P \boldsymbol{V}) = 0. \end{aligned} \tag{2.8}$$

We notice that these equations are only in a semi-Lagrangian formalism since the gradient and divergence operate on variables which depend on Eulerian coordinates. In order to write these equations in a full Lagrangian way, one has to express the gradient and divergence operators with Lagrangian coordinates using the map $\mathcal{M}_t$. For such an approach, the reader should refer to [15], [20], and [9].

In order to perform the space discretization in the next section, we give the integral form of system (2.8). This form is obtained by integration of (2.8) on the Lagrangian domain ω. Knowing that $\mathcal{M}_t(\omega) = \Omega$ and $J\mathrm{d}\omega = \mathrm{d}\Omega$, using Green's formula one gets

$$\frac{d}{dt}\int_\Omega \mathrm{d}\Omega - \int_{\partial\Omega} \boldsymbol{V}\cdot\boldsymbol{N}\,\mathrm{d}l = 0, \tag{2.9 i}$$

$$\frac{d}{dt}\int_\Omega \rho\,\mathrm{d}\Omega = 0, \tag{2.9 ii}$$

$$\frac{d}{dt}\int_\Omega \rho\boldsymbol{V}\,\mathrm{d}\Omega + \int_{\partial\Omega} P\boldsymbol{N}\,\mathrm{d}l = \boldsymbol{0}, \tag{2.9 iii}$$

$$\frac{d}{dt}\int_\Omega \rho E\,\mathrm{d}\Omega + \int_{\partial\Omega} P\boldsymbol{V}\cdot\boldsymbol{N}\,\mathrm{d}l = 0, \tag{2.9 iv}$$

where $\partial\Omega$ is the boundary of Ω, $\boldsymbol{N}$ is the unit outward normal to $\partial\Omega$, and $\mathrm{d}l$ is the length element on $\partial\Omega$. The equations (2.9 i)–(2.9 iv) are well known and have been used in this form in many papers; see, for example, [14]. Let m_Ω denote the mass of fluid enclosed in Ω. Then (2.9 ii) simply states mass conservation; hence for each time one gets $m_\Omega = m_\omega$. Let us introduce the area of domain Ω:

$$\mathcal{A}_\Omega = \int_\Omega \mathrm{d}\Omega.$$

We define the density $\rho_\Omega = \frac{m_\omega}{\mathcal{A}_\Omega}$ and the specific volume $\tau_\Omega = \frac{1}{\rho_\Omega}$. We also define the mass averaged velocity and total energy

$$\boldsymbol{V_\Omega} = \frac{1}{m_\omega}\int_\Omega \rho\boldsymbol{V}\mathrm{d}\Omega,$$

$$E_\Omega = \frac{1}{m_\omega}\int_\Omega \rho E\mathrm{d}\Omega.$$

With this notation and mass conservation, one gets the final integral form which will be useful for space discretization:

$$m_\omega\frac{d}{dt}\tau_\Omega - \int_{\partial\Omega} \boldsymbol{V}\cdot\boldsymbol{N}\,\mathrm{d}l = 0, \tag{2.10 i}$$

$$m_\omega\frac{d}{dt}\boldsymbol{V_\Omega} + \int_{\partial\Omega} P\boldsymbol{N}\,\mathrm{d}l = \boldsymbol{0}, \tag{2.10 ii}$$

$$m_\omega\frac{d}{dt}E_\Omega + \int_{\partial\Omega} P\boldsymbol{V}\cdot\boldsymbol{N}\,\mathrm{d}l = 0, \tag{2.10 iii}$$

where m_ω is the mass of fluid enclosed initially in ω.

3. Spatial approximation. Using classical finite volume approach, we derive in this section the spatial approximation of the gas dynamics equations written in Lagrangian formalism. Let us introduce some notation.

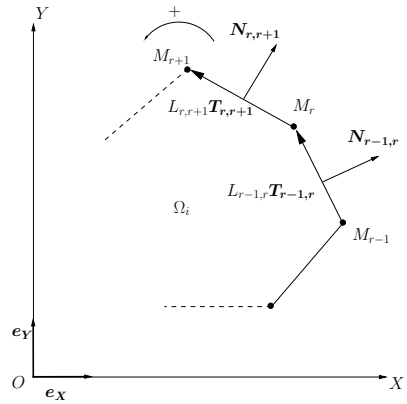


FIG. 3.1. *Notation.*

3.1. Notation, assumptions, and problem statement. Let $\{\omega_i,\ i = 1 \dots I\}$ be a collection of nonoverlapping polygons whose reunion covers ω, the subdomain of $\mathbb{R}^2$ that is initially filled by the fluid. We set $\Omega_i = \mathcal{M}_t(\omega_i)$, where $\mathcal{M}_t$ is the map defined by (2.2). Ω_i is also a polygonal cell[1] whose vertices are denoted by $M_r,\ r = 1, \dots, R(i)$. We denote by $R(i)$ the number of vertices (or faces) of the cell Ω_i. The vertices are indexed counterclockwise; see Figure 3.1. The numbering is also periodic; i.e., $M_{R(i)+1} = M_1$ and $M_0 = M_{R(i)}$. The frame (O, X, Y) is equipped with an orthonormal basis $(\boldsymbol{e_X}, \boldsymbol{e_Y})$ which is completed by the vector $\boldsymbol{e_Z} = \boldsymbol{e_X} \times \boldsymbol{e_Y}$. We denote the Eulerian coordinates of the vertex M_r by (X_r, Y_r). Note that M_r and its coordinates are time dependent. Considering any edge $[M_r, M_{r+1}]$ of Ω_i, $L_{r,r+1}$ denotes its length, $\boldsymbol{T_{r,r+1}}$ represents its unit tangent vector in a counterclockwise orientation, and $\boldsymbol{N_{r,r+1}}$ is its unit outward normal vector. Following these definitions and conventions, we have

$$
\begin{aligned}
&L_{r,r+1}\boldsymbol{T_{r,r+1}} = \boldsymbol{M_r M_{r+1}},\\
&L_{r,r+1}\boldsymbol{N_{r,r+1}} = L_{r,r+1}\boldsymbol{T_{r,r+1}} \times \boldsymbol{e_Z}.
\end{aligned}
\tag{3.1}
$$

The fluid in cell Ω_i is described by the discrete variables $(\tau_i, \boldsymbol{V_i}, E_i)$, respectively, the averaged specific volume, velocity, and specific total energy. A set of discrete equations is written for these discrete unknowns, using the integral formulation (2.10 i) applied for cell Ω_i:

$$
m_i \frac{d}{dt}\tau_i - \int_{\partial\Omega_i} \boldsymbol{V}\cdot\boldsymbol{N}\,\mathrm{d}l = 0,
$$

$$
m_i \frac{d}{dt}\boldsymbol{V_i} + \int_{\partial\Omega_i} P\boldsymbol{N}\,\mathrm{d}l = \boldsymbol{0},
$$

$$
m_i \frac{d}{dt}E_i + \int_{\partial\Omega_i} P\boldsymbol{V}\cdot\boldsymbol{N}\,\mathrm{d}l = 0,
$$

[1]Here, we assume that Ω_i is still a polygon. This means that we implicitly assume that the velocity field variation is linear in space. This is not restrictive here, because the aim of the paper is to develop a scheme that is first order in space.

where m_i denotes the mass of the cell Ω_i. We also introduce the density $\rho_i = \frac{1}{\tau_i}$, the specific internal energy $\varepsilon_i = E_i - \frac{1}{2}\|\boldsymbol{V_i}\|^2$, and the pressure given by the equation of state $P_i = P(\rho_i, \varepsilon_i)$. Since all fluid variables are assumed to be constant in cell Ω_i, we obtain a spatial approximation which is first order accurate.

Finally, we introduce some notation to express the discrete face fluxes. Let us denote by $\boldsymbol{V^\star_{r,r+1}} \cdot \boldsymbol{N_{r,r+1}}$, $P^\star_{r,r+1}\boldsymbol{N_{r,r+1}}$, and $(P\boldsymbol{V})^\star_{r,r+1} \cdot \boldsymbol{N_{r,r+1}}$ the volume, momentum, and total energy fluxes on the face $[M_r, M_{r+1}]$. They are defined by the following equations:

$$
\begin{aligned}
L_{r,r+1}\boldsymbol{V^\star_{r,r+1}} \cdot \boldsymbol{N_{r,r+1}} &= \int_{M_r}^{M_{r+1}} \boldsymbol{V} \cdot \boldsymbol{N}\, \mathrm{d}l,\\
L_{r,r+1}\boldsymbol{P^\star_{r,r+1}}\boldsymbol{N_{r,r+1}} &= \int_{M_r}^{M_{r+1}} P\boldsymbol{N}\, \mathrm{d}l,\\
L_{r,r+1}\,(P\boldsymbol{V})^\star_{r,r+1} \cdot \boldsymbol{N_{r,r+1}} &= \int_{M_r}^{M_{r+1}} P\boldsymbol{V} \cdot \boldsymbol{N}\, \mathrm{d}l.
\end{aligned}
\tag{3.2}
$$

With this notation the previous system is now written in the following manner:

$$
m_i \frac{d}{dt}\tau_i - \sum_{r=1}^{R(i)} L_{r,r+1}\boldsymbol{V^\star_{r,r+1}} \cdot \boldsymbol{N_{r,r+1}} = 0, \tag{3.3 i}
$$

$$
m_i \frac{d}{dt}\boldsymbol{V_i} + \sum_{r=1}^{R(i)} L_{r,r+1}P^\star_{r,r+1}\boldsymbol{N_{r,r+1}} = \boldsymbol{0}, \tag{3.3 ii}
$$

$$
m_i \frac{d}{dt}E_i + \sum_{r=1}^{R(i)} L_{r,r+1}\,(P\boldsymbol{V})^\star_{r,r+1} \cdot \boldsymbol{N_{r,r+1}} = 0. \tag{3.3 iii}
$$

We notice that with this Lagrangian discretization we have to take into account the cell motion. For this purpose we write the node motion equation using the trajectory equation

$$
\begin{aligned}
\frac{d}{dt}X_r &= u^\star_r, \quad X_r(0) = x_r,\\
\frac{d}{dt}Y_r &= v^\star_r, \quad Y_r(0) = y_r,
\end{aligned}
\tag{3.4}
$$

where $(u^\star_r, v^\star_r)$ are the components of the velocity $\boldsymbol{V^\star_r}$ of the vertex M_r.

In order to complete the space approximation, the following important problems arise:

1. How do we compute the face fluxes defined by (3.2)?
2. How do we compute the node velocities $\boldsymbol{V^\star_r}$?
3. These velocities being known, how can we ensure the compatibility between the mesh motion and the volume variation of the cells?

Thus, our task consists in building a numerical solver that can compute the fluxes $\boldsymbol{V^\star_{r,r+1}}$, $P^\star_{r,r+1}$, and $(P\boldsymbol{V})^\star_{r,r+1}$ but also the node velocities $\boldsymbol{V^\star_r}$. Moreover, this must be done coherently. We resolve these questions in the next sections.

3.2. Approximation of the volume equation. Using a geometrical interpretation of (3.3 ii), we can link the volume flux $\boldsymbol{V}^{\star}_{\boldsymbol{r,r+1}} \cdot \boldsymbol{N_{r,r+1}}$ to the node velocity $\boldsymbol{V}^{\star}_{\boldsymbol{r}}$.

First, we notice that $m_i \tau_i = \mathcal{A}_i$, where $\mathcal{A}_i$ is the area of the cell Ω_i. Thus, (3.3 ii) provides the time variation of $\mathcal{A}_i$,

$$\frac{d}{dt}\mathcal{A}_i = \sum_{r=1}^{R(i)} L_{r,r+1} \boldsymbol{V}^{\star}_{\boldsymbol{r,r+1}} \cdot \boldsymbol{N_{r,r+1}}. \tag{3.5}$$

The area of the cell Ω_i can also be evaluated from the location of the vertices M_r using the relation (see [28])

$$\mathcal{A}_i = \frac{1}{2} \sum_{r=1}^{R(i)} \boldsymbol{OM_r} \times \boldsymbol{OM_{r+1}} \cdot \boldsymbol{e_Z}.$$

This amounts to summing the area of the triangles (O, M_r, M_{r+1}) (see Figure 3.1) over all the vertices of the cell. Using the coordinates of the vertices and after time differentiation, we get

$$\frac{d}{dt}\mathcal{A}_i = \frac{1}{2} \sum_{r=1}^{R(i)} \left(Y_{r+1} \frac{d}{dt} X_r - X_{r+1} \frac{d}{dt} Y_r + X_r \frac{d}{dt} Y_{r+1} - Y_r \frac{d}{dt} X_{r+1} \right).$$

We rearrange the last two terms of the sum using the transformation $r \to r-1$ and the periodicity of the indices and get

$$\frac{d}{dt}\mathcal{A}_i = \frac{1}{2} \sum_{r=1}^{R(i)} \left[\left(Y_{r+1} - Y_r + Y_r - Y_{r-1}\right) \frac{d}{dt} X_r - \left(X_{r+1} - X_r + X_r - X_{r-1}\right) \frac{d}{dt} Y_r \right].$$

This formula can be rewritten as

$$\frac{d}{dt}\mathcal{A}_i = \frac{1}{2} \sum_{r=1}^{R(i)} \boldsymbol{V}^{\star}_{\boldsymbol{r}} \cdot \left(L_{r-1,r} \boldsymbol{N_{r-1,r}} + L_{r,r+1} \boldsymbol{N_{r,r+1}} \right), \tag{3.6}$$

and using the change of indices $r-1 \to r$ for the first term of the sum, we have the equivalent formulation

$$\frac{d}{dt}\mathcal{A}_i = \sum_{r=1}^{R(i)} \frac{1}{2} L_{r,r+1} \left(\boldsymbol{V}^{\star}_{\boldsymbol{r}} + \boldsymbol{V}^{\star}_{\boldsymbol{r+1}} \right) \cdot \boldsymbol{N_{r,r+1}}. \tag{3.7}$$

The comparison between (3.5) and (3.7) enables us to give the face velocities in term of the vertex velocities, namely,

$$\boldsymbol{V}^{\star}_{\boldsymbol{r,r+1}} = \frac{1}{2} \left(\boldsymbol{V}^{\star}_{\boldsymbol{r}} + \boldsymbol{V}^{\star}_{\boldsymbol{r+1}} \right). \tag{3.8}$$

This fundamental relation enables us to write two equivalent discretizations of the specific volume variation. It can be given in terms of the flux through the faces, or equivalently, in term of vertex fluxes. Moreover, the two discretizations are [illegible] with the node motion. We get one of the results of [9], but with a much simpler geometrical argument. We note that (3.8) is consistent with the assumption that the velocity field is linear along face $[M_r, M_{r+1}]$.

Hence, one can consider two methods for computing the face velocities:

- The first one relies on the evaluation of the normal velocity $\boldsymbol{V}^{\star}_{\boldsymbol{r,r+1}} \cdot \boldsymbol{N_{r,r+1}}$ using a one-dimensional Riemann solver at faces. The vertex velocities $\boldsymbol{V}^{\star}_{\boldsymbol{r}}$ have to be computed by solving a linear system built from (3.8) written for all the faces. This system is in general singular, which is why we give up this approach and shall adopt a more robust method.
- In the second method, the velocities $\boldsymbol{V}^{\star}_{\boldsymbol{r}}$ are first evaluated using some still-to-be-defined solver. The face velocities are then computed from (3.8). This is the technique we will use in what follows because this method will guarantee the compatibility between vertex motion and cell area variation.

3.3. Approximation of momentum flux. Following the same methodology, we approximate the momentum flux defined at the vertices in such a way that it is compatible with the approximation at faces. Thus, similarly to (3.8) we set

$$P^{\star,i}_{r,r+1} = \frac{1}{2}\left(P^{\star,i}_{r,r+\frac{1}{2}} + P^{\star,i}_{r+\frac{1}{2},r+1}\right), \tag{3.9}$$

where $P^{\star,i}_{r,r+\frac{1}{2}}$ (resp., $P^{\star,i}_{r+\frac{1}{2},r+1}$) represents the pressure on the half-face $[M_r, M_{r+\frac{1}{2}}]$ (resp., $[M_{r+\frac{1}{2}}, M_{r+1}]$) seen from the cell Ω_i; see Figure 3.2. We have added the superscript i to remove any possible ambiguity. Hence, $\frac{1}{2}L_{r,r+1}P^{\star,i}_{r,r+\frac{1}{2}}\boldsymbol{N_{r,r+1}}$ represents the momentum flux for half-face $[M_r, M_{r+\frac{1}{2}}]$ seen from the cell Ω_i. We observe that at node M_r, seen from the cell Ω_i, there exist two pressures: $P^{\star,i}_{r,r+\frac{1}{2}}$ and $P^{\star,i}_{r-\frac{1}{2},r}$. Because of this, our approach is different from that of [9], where only one pressure is defined at the vertex M_r.

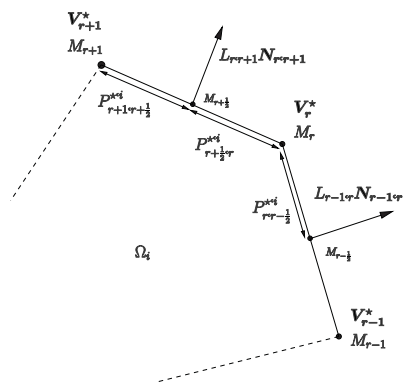


FIG. 3.2. *Pressures at faces.*

Using this definition, the equation for momentum (3.3 iii) is written as

$$m_i \frac{d}{dt}\boldsymbol{V_i} + \sum_{r=1}^{R(i)} \frac{1}{2} L_{r,r+1}\left(P^{\star,i}_{r,r+\frac{1}{2}} + P^{\star,i}_{r+\frac{1}{2},r+1}\right)\boldsymbol{N_{r,r+1}} = \boldsymbol{0}. \tag{3.10}$$

If we shift the index r (i.e., $r \to r-1$) in the second term of the summation, we get

$$m_i \frac{d}{dt}\boldsymbol{V_i} + \sum_{r=1}^{R(i)} \frac{1}{2}\left(L_{r-1,r}P^{\star,i}_{r-\frac{1}{2},r}\boldsymbol{N_{r-1,r}} + L_{r,r+1}P^{\star,i}_{r,r+\frac{1}{2}}\boldsymbol{N_{r,r+1}}\right) = \boldsymbol{0}. \tag{3.11}$$

It amounts to summing the fluxes over all nodes of the cell. Equations (3.10) and (3.11) are equivalent discretizations. They use either face fluxes or vertex fluxes. The evaluation of these fluxes uses two pressures at a face or at a node for cell Ω_i.

1. We notice that in the case of a one-dimensional Riemann solver at faces, as classically done, only one single pressure is computed for $[M_r, M_{r+1}]$. The conservation of the momentum is a simple consequence of this approximation. Here, this is no longer the case. In fact, if the cell Ω_j shares face $[M_r, M_{r+1}]$ with Ω_i, the pressures on this face seen from Ω_i may be in general different from those seen from Ω_j. That is why we have put the superscript denoting the cell in the definition of edge pressures. This is an important fact, which is one of the key points of our construction: in adding degrees of freedom as here, we will see later in the text that this enables us to build a node solver which simultaneously provides us with the node velocity and the local conservation of momentum and energy.

3.4. Approximation of the energy flux. In order to get an equivalent discretization in terms of face and nodal fluxes for the total energy equation, we set

$$(P\boldsymbol{V})^{\star,i}_{r,r+1} = \frac{1}{2}\left(P^{\star,i}_{r,r+\frac{1}{2}}\boldsymbol{V_r^\star} + P^{\star,i}_{r+\frac{1}{2},r+1}\boldsymbol{V_{r+1}^\star}\right). \tag{3.12}$$

Using (3.12), the total energy equation in its face flux formulation is written as

$$m_i\frac{d}{dt}E_i + \sum_{r=1}^{R(i)}\frac{1}{2}L_{r,r+1}\left(P^{\star,i}_{r,r+\frac{1}{2}}\boldsymbol{V_r^\star} + P^{\star,i}_{r+\frac{1}{2},r+1}\boldsymbol{V_{r+1}^\star}\right)\cdot\boldsymbol{N_{r,r+1}} = 0. \tag{3.13}$$

Shifting once more the index r in the second term of the sum, we get the node flux formulation

$$m_i\frac{d}{dt}E_i + \sum_{r=1}^{R(i)}\frac{1}{2}\left(L_{r-1,r}P^{\star,i}_{r-\frac{1}{2},r}\boldsymbol{N_{r-1,r}} + L_{r,r+1}P^{\star,i}_{r,r+\frac{1}{2}}\boldsymbol{N_{r,r+1}}\right)\cdot\boldsymbol{V_r^\star} = 0. \tag{3.14}$$

3.5. Summary. We obtain an approximation of the Euler equations written in Lagrangian coordinates which relies on approximated vertex fluxes:

$$m_i\frac{d}{dt}\tau_i - \sum_{r=1}^{R(i)}\frac{1}{2}\left(L_{r-1,r}\boldsymbol{N_{r-1,r}} + L_{r,r+1}\boldsymbol{N_{r,r+1}}\right)\cdot\boldsymbol{V_r^\star} = 0,$$

$$m_i\frac{d}{dt}\boldsymbol{V_i} + \sum_{r=1}^{R(i)}\frac{1}{2}\left(L_{r-1,r}P^{\star,i}_{r-\frac{1}{2},r}\boldsymbol{N_{r-1,r}} + L_{r,r+1}P^{\star,i}_{r,r+\frac{1}{2}}\boldsymbol{N_{r,r+1}}\right) = \boldsymbol{0}, \tag{3.15}$$

$$m_i\frac{d}{dt}E_i + \sum_{r=1}^{R(i)}\frac{1}{2}\left(L_{r-1,r}P^{\star,i}_{r-\frac{1}{2},r}\boldsymbol{N_{r-1,r}} + L_{r,r+1}P^{\star,i}_{r,r+\frac{1}{2}}\boldsymbol{N_{r,r+1}}\right)\cdot\boldsymbol{V_r^\star} = 0.$$

The system (3.15) is equivalent to (3.3) provided that the face fluxes are defined by

$$\begin{aligned}
\boldsymbol{V^\star_{r,r+1}} &= \tfrac{1}{2}\left(\boldsymbol{V_r^\star} + \boldsymbol{V_{r+1}^\star}\right),\\
P^{\star,i}_{r,r+1} &= \tfrac{1}{2}\left(P^{\star,i}_{r,r+\frac{1}{2}} + P^{\star,i}_{r+\frac{1}{2},r+1}\right),\\
(P\boldsymbol{V})^{\star,i}_{r,r+1} &= \tfrac{1}{2}\left(P^{\star,i}_{r,r+\frac{1}{2}}\boldsymbol{V_r^\star} + P^{\star,i}_{r+\frac{1}{2},r+1}\boldsymbol{V_{r+1}^\star}\right).
\end{aligned} \tag{3.16}$$

Moreover, this definition is coherent with the vertex motion.

Now, we are going to show how to compute, for each vertex M_r , the velocity $\boldsymbol{V_r^\star}$ and the pressures $P^{\star,i}_{r-\frac{1}{2},r}$ and $P^{\star,i}_{r,r+\frac{1}{2}}$.

4. Construction of a solver at the vertices. The evaluation of $\boldsymbol{V_r^\star}$, $P^{\star,i}_{r-\frac{1}{2},r}$, and $P^{\star,i}_{r,r+\frac{1}{2}}$ relies on the following arguments:

- the global conservation of momentum,
- a local entropy inequality.

The first argument is a consequence of Remark 1: the pressure on the face $[M_r, M_{r+1}]$ is not unique, contrary to the standard finite volume approach. Thus, we lose the automatic conservation of momentum and total energy. Of course, we must guarantee the conservation of momentum and total energy, so we have to add additional constraints on $\boldsymbol{V_r^\star}$, $P^{\star,i}_{r-\frac{1}{2},r}$, and $P^{\star,i}_{r,r+\frac{1}{2}}$.

The second argument is about the thermodynamical consistency of the scheme, which is necessary for a correct computation of the discontinuities. We will build a sufficient condition which will supply a local dissipation of entropy.

The two sets of constraints (conservation and dissipation of entropy) will lead, as we shall see in this section, to a linear system whose unique solution will give us $\boldsymbol{V_r^\star}$, $P^{\star,i}_{r-\frac{1}{2},r}$, and $P^{\star,i}_{r,r+\frac{1}{2}}$.

4.1. Notation around a vertex. In order to exhibit conservation equations for momentum and total energy around a node, let us introduce some new notation. We denote by M_q a generic internal vertex of the mesh, for $q = 1, \dots, Q$, where Q is the total number of internal nodes. The case of the nodes located on the boundary $\partial\Omega$ will be treated in section 4.7. There are $K(q)$ cells around M_q which are denoted by Ω_k; see Figure 4.1. For the cell Ω_k we denote by $[M_q, M_k]$ and $[M_q, M_{k+1}]$ the edges coming from node M_q. The outward normal to these edges are $\boldsymbol{N_k^k}$, $\boldsymbol{N_{k+1}^k}$ and their lengths L_k, L_{k+1}. Let $\boldsymbol{V_q^\star}$ be the velocity of node M_q and $P^{\star,k}_{q,k}$ (resp., $P^{\star,k}_{q,k+1}$), the half-pressures on edge $[M_q, M_k]$ (resp., $[M_q, M_{k+1}]$) seen from cell Ω_k.

4.2. Conservation relations. Omitting the boundary conditions and summing (3.11) on each cell of the domain, we make a global balance of momentum

$$\frac{d}{dt}\left(\sum_{i=1}^{I} m_i \boldsymbol{V_i}\right) = -\sum_{i=1}^{I}\sum_{r=1}^{r(i)} \frac{1}{2}\left(L^i_{r-1,r} P^{\star,i}_{r-\frac{1}{2},r} \boldsymbol{N^i_{r-1,r}} + L^i_{r,r+1} P^{\star,i}_{r,r+\frac{1}{2}} \boldsymbol{N^i_{r,r+1}}\right).$$

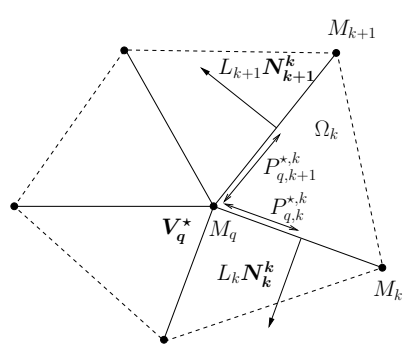


FIG. 4.1. *Notation around the vertex M_q.*

We have added the superscript i for the lengths and normals in order to remove any possible ambiguity. If we replace the global summation over cells by a global summation over nodes, using the notation previously defined, we can rewrite the right-hand side in the following way:

$$\frac{d}{dt}\left(\sum_{i=1}^{I} m_i \boldsymbol{V_i}\right) = -\sum_{q=1}^{Q}\sum_{k=1}^{K(q)} \frac{1}{2}\left(L_k P_{q,k}^{\star,k} \boldsymbol{N_k^k} + L_{k+1} P_{q,k+1}^{\star,k} \boldsymbol{N_{k+1}^k}\right).$$

The (global) conservation of momentum is satisfied provided that the right-hand side vanishes. A sufficient condition is

$$\sum_{k=1}^{K(q)} \frac{1}{2}\left(L_k P_{q,k}^{\star,k} \boldsymbol{N_k^k} + L_{k+1} P_{q,k+1}^{\star,k} \boldsymbol{N_{k+1}^k}\right) = \boldsymbol{0}. \tag{4.1}$$

This summation is done over all the cells k surrounding vertex M_q; see Figure 4.1. The relation (4.1) can be interpreted as the local equilibrium of vertex M_q under pressure forces. This is also a local conservation relation.

It can be checked that condition (4.1) also implies the global conservation of total energy. To do so, we sum (3.14) on each cell, switch the summation for the cells' and the vertices' indices, and get

$$\frac{d}{dt}\left(\sum_{i=1}^{I} m_i E_i\right) = -\sum_{q=1}^{Q}\sum_{k=1}^{K(q)} \frac{1}{2}\left(L_k P_{q,k}^{\star,k} \boldsymbol{N_k^k} + L_{k+1} P_{q,k+1}^{\star,k} \boldsymbol{N_{k+1}^k}\right) \cdot \boldsymbol{V_q^\star} = 0,$$

using (4.1) and because the velocity $\boldsymbol{V_q^\star}$ is single valued at M_q.

The conservation of volume is easily checked. To do so, we sum the first equation of (3.15) for each cell:

$$\frac{d}{dt}\left(\sum_{i=1}^{I} m_i \tau_i\right) = \sum_{q=1}^{Q}\sum_{k=1}^{K(q)} \frac{1}{2}\left(L_k \boldsymbol{N_k^k} + L_{k+1} \boldsymbol{N_{k+1}^k}\right) \cdot \boldsymbol{V_q^\star}.$$

We know that

$$\sum_{k=1}^{K(q)} \frac{1}{2}\left(L_k \boldsymbol{N_k^k} + L_{k+1} \boldsymbol{N_{k+1}^k}\right) = \boldsymbol{0}, \tag{4.2}$$

since the polygon whose vertices are M_k, $k = 1, \ldots, K(q)$, is closed (dotted line in Figure 4.1).

4.3. Entropy inequality. We first compute the time variation of entropy in cell Ω_i. Let us denote it by σ_i. If T_i is the cell average temperature, we have

$$m_i T_i \frac{d}{dt}\sigma_i = m_i\left(\frac{d}{dt}\varepsilon_i + P_i \frac{d}{dt}\tau_i\right). \tag{4.3}$$

This quantity is evaluated in two steps. First, we compute the variation of internal energy $\frac{d}{dt}\varepsilon_i = \frac{d}{dt}E_i - \boldsymbol{V_i} \cdot \frac{d}{dt}\boldsymbol{V_i}$, and then we evaluate the pressure work $P_i \frac{d}{dt}\tau_i$.

- Internal energy: we dot-multiply (3.11) by $\boldsymbol{V_i}$ to get the kinetic energy variation:

$$m_i \boldsymbol{V_i} \cdot \frac{d}{dt}\boldsymbol{V_i} = -\sum_{r=1}^{R(i)} \frac{1}{2}\left(L_{r-1,r} P_{r-\frac{1}{2},r}^{\star,i} \boldsymbol{N_{r-1,r}} + L_{r,r+1} P_{r,r+\frac{1}{2}}^{\star,i} \boldsymbol{N_{r,r+1}}\right) \cdot \boldsymbol{V_i}. \tag{4.4}$$

Then (4.4) is subtracted from the total energy variation equation and we have

(4.5)
$$m_i \frac{d}{dt}\varepsilon_i = -\sum_{r=1}^{R(i)} \frac{1}{2}\left(L_{r-1,r}P^{\star,i}_{r-\frac{1}{2},r}\boldsymbol{N_{r-1,r}} + L_{r,r+1}P^{\star,i}_{r,r+\frac{1}{2}}\boldsymbol{N_{r,r+1}}\right)\cdot\left(\boldsymbol{V_r^\star} - \boldsymbol{V_i}\right).$$

- Pressure work: we multiply the first relation of (3.15) by P_i:

$$m_i P_i \frac{d}{dt}\tau_i = \sum_{r=1}^{R(i)} \frac{1}{2}\left(L_{r-1,r}P_i\boldsymbol{N_{r-1,r}} + L_{r,r+1}P_i\boldsymbol{N_{r,r+1}}\right)\cdot\boldsymbol{V_r^\star}.$$

An equivalent formulation is

$$m_i P_i \frac{d}{dt}\tau_i = \sum_{r=1}^{R(i)} \frac{1}{2}\left(L_{r-1,r}P_i\boldsymbol{N_{r-1,r}} + L_{r,r+1}P_i\boldsymbol{N_{r,r+1}}\right)\cdot\left(\boldsymbol{V_r^\star} - \boldsymbol{V_i}\right). \tag{4.6}$$

Knowing that,

$$\sum_{r=1}^{R(i)} \frac{1}{2}\left(L_{r-1,r}\boldsymbol{N_{r-1,r}} + L_{r,r+1}\boldsymbol{N_{r,r+1}}\right) = \boldsymbol{0},$$

because the boundary of Ω_i is a closed polygon.

Last, the summation of (4.5) and (4.6) gives

$$m_i T_i \frac{d}{dt}\sigma_i = \sum_{r=1}^{R(i)} \frac{1}{2}\left[L_{r-1,r}\left(P_i - P^{\star,i}_{r-\frac{1}{2},r}\right)\boldsymbol{N_{r-1,r}} + L_{r,r+1}\left(P_i - P^{\star,i}_{r,r+\frac{1}{2}}\right)\boldsymbol{N_{r,r+1}}\right] \cdot\left(\boldsymbol{V_r^\star} - \boldsymbol{V_i}\right).$$

A sufficient condition for the right-hand side of this relation to be positive is

$$\begin{cases} P_i - P^{\star,i}_{r-\frac{1}{2},r} = \alpha_i\left(\boldsymbol{V_r^\star} - \boldsymbol{V_i}\right)\cdot\boldsymbol{N_{r-1,r}}, \quad r = 1,\dots,R(i), \\ \\ P_i - P^{\star,i}_{r,r+\frac{1}{2}} = \alpha_i\left(\boldsymbol{V_r^\star} - \boldsymbol{V_i}\right)\cdot\boldsymbol{N_{r,r+1}}, \quad r = 1,\dots,R(i), \end{cases} \tag{4.7}$$

where α_i is a positive coefficient that has the dimension of a mass flux. These two relations can be interpreted as (discretized) Riemann invariants along the directions $\boldsymbol{N_{r-1,r}}$ and $\boldsymbol{N_{r,r+1}}$. Our main motivation is to recover the approximate acoustic solver [24] for one-dimensional flows, which is why the parameter α_i is set to the value

$$\alpha_i = \rho_i c_i, \tag{4.8}$$

where $c_i = \sqrt{\left(\frac{\partial P}{\partial \rho}\right)_\sigma}$ represents the isentropic sound speed. Under condition (4.7), the variation of entropy is

(4.9)
$$m_i T_i \frac{d}{dt}\sigma_i = \sum_{r=1}^{R(i)} \frac{1}{2}\rho_i c_i \left\{L_{r-1,r}\left[\left(\boldsymbol{V_r^\star} - \boldsymbol{V_i}\right)\cdot\boldsymbol{N_{r-1,r}}\right]^2 + L_{r,r+1}\left[\left(\boldsymbol{V_r^\star} - \boldsymbol{V_i}\right)\cdot\boldsymbol{N_{r,r+1}}\right]^2\right\}.$$

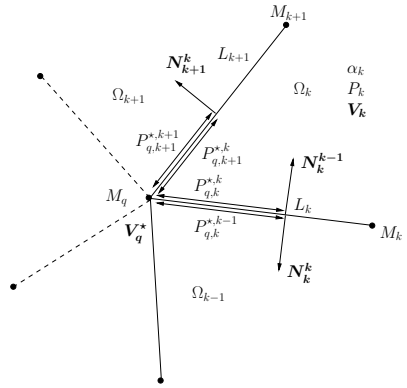


FIG. 4.2. *States around internal node M_q.*

REMARK 2. The relation (4.9) is very similar to the entropy production term resulting from a two-dimensional linear artificial viscosity term; see [5]. We notice that (4.9) implies a positive production of entropy, even in the case of isentropic flows. For such flows our scheme does not conserve entropy. This is typical for Godunov-type schemes. If we rewrite the sufficient condition (4.7) using notation around a generic vertex M_q, we obtain

$$\begin{cases} P_k - P_{q,k}^{\star,k} = \alpha_k \left(\boldsymbol{V_q^\star} - \boldsymbol{V_k} \right) \cdot \boldsymbol{N_k^k}, \quad k = 1, \dots, K(q), \\ \\ P_k - P_{q,k+1}^{\star,k} = \alpha_k \left(\boldsymbol{V_q^\star} - \boldsymbol{V_k} \right) \cdot \boldsymbol{N_{k+1}^k}, \quad k = 1, \dots, K(q), \end{cases} \tag{4.10}$$

where α_k, P_k, and $\boldsymbol{V_k}$ stand for acoustic impedance, pressure, and velocity in cell Ω_k. We point out that for a vertex M_q (not on the boundary of the computational domain), we have to compute $2K(q)$ pressures, that is, together with the velocity $\boldsymbol{V_q^\star}$, to evaluate $2K(q) + 2$ scalar unknowns. Equation (4.1) provides 2 scalar relations, and (4.10) gives $2K(q)$ more. Hence, we can compute the pressures and the velocity at the vertex M_q. This is the topic of the next section.

4.4. Evaluation of the velocity and pressures at the vertices. Using sufficient conditions (4.1) and (4.10), we write the linear system satisfied by the components of the velocity $\boldsymbol{V}_q^\star$ of an internal vertex M_q. This generic vertex is not on the boundary of the domain, so that it is surrounded by the cells Ω_k, $k = 1, \dots, K(q)$; see Figure 4.2. With this new notation, if we shift index $(k \to k-1)$ in the second term of (4.1), we get

$$\sum_{k=1}^{K(q)} L_k \left(P_{q,k}^{\star,k} - P_{q,k}^{\star,k-1} \right) \boldsymbol{N_k^{k-1}} = \boldsymbol{0}, \tag{4.11}$$

where we have used $\boldsymbol{N_k^k} = -\boldsymbol{N_k^{k-1}}$. We proceed in the same way for (4.10):

$$\begin{cases} P_{q,k}^{\star,k} = P_k + \alpha_k \left(\boldsymbol{V_q^\star} - \boldsymbol{V_k} \right) \cdot \boldsymbol{N_k^{k-1}}, \\ \\ P_{q,k}^{\star,k-1} = P_{k-1} - \alpha_{k-1} \left(\boldsymbol{V_q^\star} - \boldsymbol{V_{k-1}} \right) \cdot \boldsymbol{N_k^{k-1}}. \end{cases} \tag{4.12}$$

After substitution of (4.12) into (4.11), we obtain the following equation satisfied by $\boldsymbol{V_q^\star}$:

(4.13)
$$\sum_{k=1}^{K(q)} L_k\left(\alpha_{k-1}+\alpha_k\right) \times \left[\boldsymbol{V_q^\star}\cdot\boldsymbol{N_k^{k-1}} - \frac{P_{k-1}-P_k+\alpha_{k-1}\boldsymbol{V_{k-1}}\cdot\boldsymbol{N_k^{k-1}}+\alpha_k\boldsymbol{V_k}\cdot\boldsymbol{N_k^{k-1}}}{\alpha_{k-1}+\alpha_k}\right]\boldsymbol{N_k^{k-1}} = \mathbf{0}.$$

If we set

$$\mathcal{V}_k^\star = \frac{P_{k-1}-P_k+\alpha_{k-1}\boldsymbol{V_{k-1}}\cdot\boldsymbol{N_k^{k-1}}+\alpha_k\boldsymbol{V_k}\cdot\boldsymbol{N_k^{k-1}}}{\alpha_{k-1}+\alpha_k},$$

$\mathcal{V}_k^\star$ is the normal velocity given by the classical one-dimensional acoustic Riemann solver for face $[M_q, M_k]$. With this notation, we can rewrite (4.13) and give an interesting interpretation of it:

$$\sum_{k=1}^{K(q)} L_k\left(\alpha_{k-1}+\alpha_k\right)\left[\boldsymbol{V_q^\star}\cdot\boldsymbol{N_k^{k-1}} - \mathcal{V}_k^\star\right]\boldsymbol{N_k^{k-1}} = \mathbf{0}.$$

A straightforward calculation shows that the left-hand side of this last equation is the gradient of the following quadratic functional:

$$F(u_q^\star, v_q^\star) = \sum_{k=1}^{K(q)} L_k\left(\alpha_{k-1}+\alpha_k\right)\left[\boldsymbol{V_q^\star}\cdot\boldsymbol{N_k^{k-1}} - \mathcal{V}_k^\star\right]^2, \tag{4.14}$$

where $(u_q^\star, v_q^\star)$ denotes the components of $\boldsymbol{V_q^\star}$. Consequently, the solution of (4.13) reaches the minimum of the functional $F(u_q^\star, v_q^\star)$. Hence, it appears that the nodal velocity $\boldsymbol{V_q^\star}$ is obtained from [illegible]. This least squares procedure corresponds to the overdetermined system equating the projection of the nodal velocity onto the edge normal N_k^{k-1} with the normal velocity $\mathcal{V}_k^\star$ obtained from the acoustic approximate Riemann solver. For each edge impinging on node M_q, the weight is $L_k(\alpha_{k-1}+\alpha_k)$.

Finally, we write the 2×2 linear system satisfied by $(u_q^\star, v_q^\star)$:

$$\begin{cases} Au_q^\star + Cv_q^\star = SM_X, \\ Cu_q^\star + Bv_q^\star = SM_Y, \end{cases} \tag{4.15}$$

where A, B, and C are defined by

$$A = \sum_{k=1}^{K(q)} L_k\left(\alpha_{k-1}+\alpha_k\right)\left(N_{k,X}^{k-1}\right)^2,$$
$$B = \sum_{k=1}^{K(q)} L_k\left(\alpha_{k-1}+\alpha_k\right)\left(N_{k,Y}^{k-1}\right)^2,$$
$$C = \sum_{k=1}^{K(q)} L_k\left(\alpha_{k-1}+\alpha_k\right)N_{k,X}^{k-1}N_{k,Y}^{k-1}.$$

Here, $(N_{k,X}^{k-1}, N_{k,Y}^{k-1})$ are the coordinates of $\boldsymbol{N}_k^k$. The right-hand side (SM_X, SM_Y) of (4.15) are the components of the vector $\boldsymbol{SM}$ defined by

$$\boldsymbol{SM} = \sum_{k=1}^{K(q)} L_k \left(\alpha_{k-1} + \alpha_k\right) \mathcal{V}_k^\star \boldsymbol{N_k^{k-1}}.$$

The determinant Δ of (4.15) is $\Delta = AB - C^2$. We show that it is always positive. In order to simplify notation, let us introduce $\beta_k = \sqrt{L_k \left(\alpha_{k-1} + \alpha_k\right)}$ and $\boldsymbol{N_k} = \boldsymbol{N_k^{k-1}}$ for $k = 1, \ldots, K(q)$. We define

$$\begin{aligned} \boldsymbol{U_X} &= \left(\beta_1 N_{1,X}, \ldots, \beta_K N_{K,X}\right)^t, \\ \boldsymbol{U_Y} &= \left(\beta_1 N_{1,Y}, \ldots, \beta_K N_{K,Y}\right)^t. \end{aligned}$$

We have immediately $A = \|\boldsymbol{U_X}\|^2$, $B = \|\boldsymbol{U_Y}\|^2$, and $C = \langle \boldsymbol{U_X}, \boldsymbol{U_Y} \rangle$, where $\langle , \rangle$ is the inner scalar product of $\mathbb{R}^K$ and $\| \ \|$ its associated norm. From the Cauchy–Schwarz inequality, we know that $\Delta \geq 0$. In fact, $\Delta = 0$ if and only if one of the two vectors is null or $\boldsymbol{U_X}$ and $\boldsymbol{U_Y}$ are colinear. This situation is generally impossible unless the edges around the node merge into a single line.

We have shown that the system (4.15) always has a unique solution which determines the velocity $\boldsymbol{V_q^\star}$. From (4.12), we can get the pressures $P_{q,k}^{\star,k}$ and $P_{q,k}^{\star,k-1}$. Moreover, this nodal solver is invariant under translation, homothety, and rotation of center M_q.

3. If the flow is uniform (i.e., with a uniform velocity $\boldsymbol{V}^0$ and pressure P^0), we can easily check that the solution of (4.15) reduces to $\boldsymbol{V}_q^\star = \boldsymbol{V}^0$

4.5. Summary of the discrete evolution equations. We give in this section the summary of the discrete evolution equations that constitute a closed set of equations for the unknowns $\{\tau_i, \boldsymbol{V_i}, E_i\}$:

$$\begin{aligned} & m_i \frac{d}{dt}\tau_i - \sum_{r=1}^{R(i)} \frac{1}{2} \left(L_{r-1,r} \boldsymbol{N_{r-1,r}} + L_{r,r+1} \boldsymbol{N_{r,r+1}}\right) \cdot \boldsymbol{V_r^\star} = 0, \\ & m_i \frac{d}{dt}\boldsymbol{V_i} + \sum_{r=1}^{R(i)} \frac{1}{2} \left(L_{r-1,r} P_{r-\frac{1}{2},r}^{\star,i} \boldsymbol{N_{r-1,r}} + L_{r,r+1} P_{r,r+\frac{1}{2}}^{\star,i} \boldsymbol{N_{r,r+1}}\right) = \boldsymbol{0}, \\ & m_i \frac{d}{dt}E_i + \sum_{r=1}^{R(i)} \frac{1}{2} \left(L_{r-1,r} P_{r-\frac{1}{2},r}^{\star,i} \boldsymbol{N_{r-1,r}} + L_{r,r+1} P_{r,r+\frac{1}{2}}^{\star,i} \boldsymbol{N_{r,r+1}}\right) \cdot \boldsymbol{V_r^\star} = 0. \end{aligned} \tag{4.16}$$

We note that these equations have been written using a vertex flux formulation (see section 3). Those fluxes are obtained via a nodal solver which is derived in section 4. The edge pressures $P_{r-\frac{1}{2},r}^{\star,i}, P_{r,r+\frac{1}{2}}^{\star,i}$ are computed as follows:

$$\begin{cases} P_i - P_{r-\frac{1}{2},r}^{\star,i} = \rho_i c_i \left(\boldsymbol{V_r^\star} - \boldsymbol{V_i}\right) \cdot \boldsymbol{N_{r-1,r}}, \ \ r = 1, \ldots, R(i), \\ \\ P_i - P_{r,r+\frac{1}{2}}^{\star,i} = \rho_i c_i \left(\boldsymbol{V_r^\star} - \boldsymbol{V_i}\right) \cdot \boldsymbol{N_{r,r+1}}, \ \ r = 1, \ldots, R(i). \end{cases} \tag{4.17}$$

If q denotes the global index corresponding to the local index r, the nodal velocity $\boldsymbol{V_r^\star} \equiv \boldsymbol{V_q^\star} = (u_q^\star, v_q^\star)$ is given by the solution of the 2×2 linear system

$$\begin{cases} A u_q^\star + C v_q^\star = SM_X, \\ C u_q^\star + B v_q^\star = SM_Y, \end{cases} \tag{4.18}$$

where A, B, and C are defined by

$$A = \sum_{k=1}^{K(q)} L_k \left(\rho_{k-1}c_{k-1} + \rho_k c_k\right) \left(N_{k,X}^{k-1}\right)^2,$$
$$B = \sum_{k=1}^{K(q)} L_k \left(\rho_{k-1}c_{k-1} + \rho_k c_k\right) \left(N_{k,Y}^{k-1}\right)^2,$$
$$C = \sum_{k=1}^{K(q)} L_k \left(\rho_{k-1}c_{k-1} + \rho_k c_k\right) N_{k,X}^{k-1} N_{k,Y}^{k-1},$$

where the index k denotes the label of the cells (edges) that surround the node M_q; see Figure 4.2. The right-hand side of (4.18), $\boldsymbol{SM} = (SM_X, SM_Y)^t$, is given by

$$\boldsymbol{SM} = \sum_{k=1}^{K(q)} L_k \left(\rho_{k-1}c_{k-1} + \rho_k c_k\right) \mathcal{V}_k^\star \boldsymbol{N_k^{k-1}},$$

where $\mathcal{V}_k^\star$ corresponds to the normal velocity given by the classical one-dimensional acoustic Riemann solver

$$\mathcal{V}_k^\star = \frac{P_{k-1} - P_k + \rho_{k-1}c_{k-1}\boldsymbol{V_{k-1}} \cdot \boldsymbol{N_k^{k-1}} + \rho_k c_k \boldsymbol{V_k} \cdot \boldsymbol{N_k^{k-1}}}{\rho_{k-1}c_{k-1} + \rho_k c_k}.$$

Before we detail our implementation of the boundary conditions, we provide in the next section an interpretation of our results in the case of a one-dimensional flow with a planar symmetry and in the case of a flow with a cylindrical symmetry.

4.6. Interpretation of the solver in two simple cases.

4.6.1. One-dimensional flow with planar symmetry. Let us consider M_q surrounded by four quadrangular cells. They are numbered from 1 to 4 in the counterclockwise direction; see Figure 4.3. We consider the frame $(M_q, \boldsymbol{e_X}, \boldsymbol{e_Y})$. The vector $\boldsymbol{N}_k^{k-1}$ is the normal to the edge shared by cells $k-1$ and k. The length of this edge is L_k. The indexing system is periodic with period 4, and the flow is one-dimensional in the $\boldsymbol{e}_X$ direction so that the velocity field in the cell k reduces to $\boldsymbol{V}_k = u_k \boldsymbol{e_X}$. Moreover, any scalar quantity φ satisfies $\varphi_1 \equiv \varphi_4$ and $\varphi_2 \equiv \varphi_3$. The cells 1 and 4 contain the state defined by P_1, c_1, $\boldsymbol{V_1} = u_1 \boldsymbol{e_X}$. The cells 2 and 3 contain the state defined by P_2, c_2, $\boldsymbol{V_2} = u_2 \boldsymbol{e_X}$. Using this notation, the evaluation of the matrix

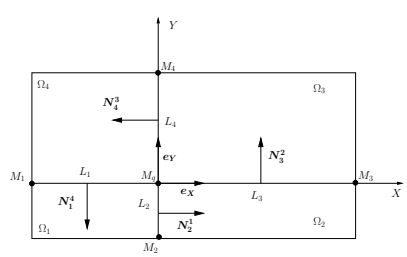


FIG. 4.3. *One-dimensional planar flow.*

coefficients A, B, C is straightforward:

$$A = (L_2 + L_4)(\rho_1 c_1 + \rho_2 c_2),$$

$$B = 2(L_1 \rho_1 c_1 + L_3 \rho_2 c_2),$$

$$C = 0.$$

The calculation of $\boldsymbol{SM}$ gives

$$SM_X = (L_2 + L_4)(P_1 - P_2 + \rho_1 c_1 u_1 + \rho_2 c_2 u_2),$$

$$SM_Y = 0.$$

Therefore $v_q^\star = 0$ and

$$u_q^\star = \frac{P_1 - P_2 + \rho_1 c_1 u_1 + \rho_2 c_2 u_2}{\rho_1 c_1 + \rho_2 c_2}.$$

We recover exactly the acoustic Riemann solver!

4.6.2. One-dimensional flow with cylindrical symmetry. We consider an equiangular cylindrical mesh centered at O. We denote by θ the angle of any sector. The mesh may be nonregular in the radial direction. Any vertex M_q is surrounded by four trapezoidal cells. In order to simplify the algebra, we choose to work in the local orthonormal frame $(M_q, \boldsymbol{e_X}, \boldsymbol{e_Y})$, where $\boldsymbol{e_X}$ is the unit vector colinear to $\boldsymbol{OM_q}$; see Figure 4.4. We use the same conventions as in the previous section. The cells 1 and 4 contain the states defined by P_1, ρ_1, c_1. The cells 2 and 3 contain the states defined by P_2, ρ_2, c_2. Due to cylindrical symmetry, the velocities are defined by $\boldsymbol{V_1} = V_1 \boldsymbol{N_2^1}$, $\boldsymbol{V_2} = V_2 \boldsymbol{N_2^1}$, $\boldsymbol{V_3} = -V_2 \boldsymbol{N_4^3}$, and $\boldsymbol{V_4} = -V_1 \boldsymbol{N_4^3}$. In the frame $(M_q, \boldsymbol{e_X}, \boldsymbol{e_Y})$, the normals are written as

$$\boldsymbol{N_2^1} = \begin{pmatrix} \cos(\theta/2) \\ -\sin(\theta/2) \end{pmatrix}, \quad \boldsymbol{N_2^3} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad \boldsymbol{N_4^3} = \begin{pmatrix} -\cos(\theta/2) \\ -\sin(\theta/2) \end{pmatrix}, \quad \text{and } \boldsymbol{N_1^4} = \begin{pmatrix} 0 \\ -1 \end{pmatrix}.$$

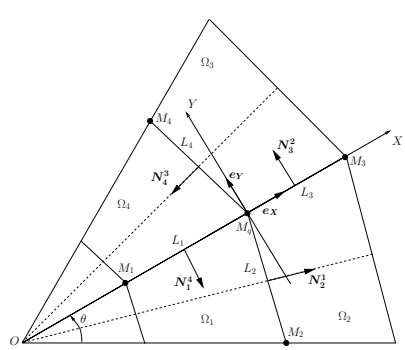


FIG. 4.4. *One-dimensional cylindrical flow.*

After calculations we get

$$A = (L_2 + L_4)\cos^2\left(\frac{\theta}{2}\right)(\rho_1 c_1 + \rho_2 c_2),$$
$$B = 2L_1\rho_1 c_1 + (L_2 + L_4)\sin^2\left(\frac{\theta}{2}\right)(\rho_1 c_1 + \rho_2 c_2) + 2L_3\rho_2 c_2,$$
$$C = (L_4 - L_2)\sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\theta}{2}\right)(\rho_1 c_1 + \rho_2 c_2).$$

We also obtain

$$\boldsymbol{SM} = (P_1 - P_2 + \rho_1 c_1 V_1 + \rho_2 c_2 V_2)\begin{pmatrix}(L_2 + L_4)\cos(\theta/2)\\ (L_4 - L_2)\sin(\theta/2)\end{pmatrix}.$$

Using the symmetry we have $L_2 = L_4$; thus $C = 0$ and $SM_Y = 0$. Therefore $v_q^\star = 0$, and the velocity is of the form $\boldsymbol{V_q^\star} = u_q^\star \boldsymbol{e_X}$. The evaluation of $u_q^\star$ is straightforward:

$$u_q^\star = \frac{P_1 - P_2 + \rho_1 c_1 V_1 + \rho_2 c_2 V_2}{\rho_1 c_1 + \rho_2 c_2}\frac{1}{\cos(\frac{\theta}{2})}.$$

This is once more the acoustic solver modified by a geometrical factor that is a consequence of the projection of $\boldsymbol{V_k}$ on the axis $[O, M_q]$. This term has no consequence since $\cos(\frac{\theta}{2}) \to 1$ when $\theta \to 0$. We also notice that the velocity is radial because the angle between two sectors is uniform. If the mesh does not satisfy this property, therefore $L_2 \neq L_4$ and $C \neq 0$, and consequently, $v_q^\star \neq 0$.

4.7. Boundary conditions. In this section, we explain our implementation of boundary conditions which is consistent with our internal solver. In the Lagrangian formalism, we have to consider two types of boundary conditions on the boundary $\partial\Omega$: either we impose the pressure or we impose the normal component of the velocity. We use the same type of notation as in section 4.1. Let $M_q \in \partial\Omega$. It is surrounded by $K(q) = K$ cells contained in the domain Ω. There are $K + 1$ edges impinging on M_q. They are numbered counterclockwise; see Figure 4.5. The first and last cells Ω_1 and Ω_K have an edge on the boundary $\partial\Omega$. The outward normals to the two boundary edges coming out of M_q are denoted by $-\boldsymbol{N_1^0}$ and $\boldsymbol{N_{K+1}^K}$ coherently with our notation.

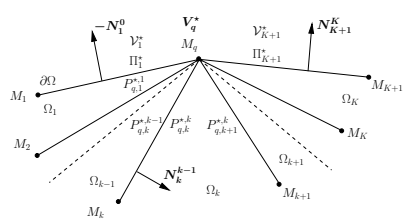


FIG. 4.5. *Notation for the boundary conditions.*

4.7.1. Case of a prescribed pressure. We denote by $\Pi_1^\star$ and $\Pi_{K+1}^\star$ the pressures that are imposed on the boundary edges; see Figure 4.5. We have to compute the $2K$ pressures $P_{q,k}^{\star,k-1}$, $k = 2, \ldots, K+1$, and $P_{q,k}^{\star,k}$, $k = 1, \ldots, K$; adding the velocity $\boldsymbol{V_q^\star}$ of the vertex M_q, we have a total number of $2K + 2$ scalar unknowns. These unknowns satisfy the previous relations (conservation of momentum, entropy

inequality). For the conservation relation, we make a balance around the vertex M_q by counting the pressure forces due to the boundary conditions. We get

$$\begin{aligned} &-L_1 P_{q,1}^{\star,1} \boldsymbol{N_1^0} + \sum_{k=2}^{K} L_k \left(P_{q,k}^{\star,k-1} - P_{q,k}^{\star,k} \right) \boldsymbol{N_k^{k-1}} + L_{K+1} P_{q,K+1}^{\star,K} \boldsymbol{N_{K+1}^K} \\ &\quad = -L_1 \Pi_1^\star \boldsymbol{N_1^0} + L_{K+1} \Pi_{K+1}^\star \boldsymbol{N_{K+1}^K}. \end{aligned} \tag{4.19}$$

The entropy inequality provides

$$\begin{aligned} P_{q,k}^{\star,k} &= P_k + \alpha_k \left(\boldsymbol{V_q^\star} - \boldsymbol{V_k} \right) \cdot \boldsymbol{N_k^{k-1}}, \quad k = 1, \dots, K, \\ P_{q,k}^{\star,k-1} &= P_{k-1} - \alpha_{k-1} \left(\boldsymbol{V_q^\star} - \boldsymbol{V_{k-1}} \right) \cdot \boldsymbol{N_k^{k-1}}, \quad k = 2, \dots, K+1. \end{aligned} \tag{4.20}$$

The use of (4.20) in (4.19) leads to

$$\begin{cases} \tilde{A} u_q^\star + \tilde{C} v_q^\star = \widetilde{SM_X}, \\ \tilde{C} u_q^\star + \tilde{B} v_q^\star = \widetilde{SM_Y}, \end{cases} \tag{4.21}$$

where the coefficients are

$$\begin{aligned} \tilde{A} &= \sum_{k=2}^{K} L_k \left(\alpha_{k-1} + \alpha_k \right) \left(N_{k,X}^{k-1} \right)^2 + L_1 \alpha_1 \left(N_{1,X}^0 \right)^2 + L_{K+1} \alpha_{K+1} \left(N_{K+1,X}^K \right)^2, \\ \tilde{B} &= \sum_{k=2}^{K} L_k \left(\alpha_{k-1} + \alpha_k \right) \left(N_{k,Y}^{k-1} \right)^2 + L_1 \alpha_1 \left(N_{1,Y}^0 \right)^2 + L_{K+1} \alpha_{K+1} \left(N_{K+1,Y}^K \right)^2, \\ \tilde{C} &= \sum_{k=2}^{K} L_k \left(\alpha_{k-1} + \alpha_k \right) N_{k,X}^{k-1} N_{k,Y}^{k-1} + L_1 \alpha_1 N_{1,X}^0 N_{1,Y}^0 + L_{K+1} \alpha_{K+1} N_{K+1,X}^K N_{K+1,Y}^K. \end{aligned}$$

We see that we recover the coefficients A, B and C, where the contribution of the boundary faces are now taken into account. The right-hand side of (4.21) is written as

$$\begin{aligned} \widetilde{\boldsymbol{SM}} = &\sum_{k=2}^{K} L_k \left[P_{k-1} - P_k + (\alpha_{k-1} \boldsymbol{V_{k-1}} + \alpha_k \boldsymbol{V_k}) \cdot \boldsymbol{N_{k-1}^k} \right] \boldsymbol{N_{k-1}^k} \\ &- L_1 \left[P_1 - \Pi_1^\star - \alpha_1 \boldsymbol{V_1} \cdot \boldsymbol{N_1^0} \right] \boldsymbol{N_1^0} \\ &+ L_{K+1} \left[P_K - \Pi_{K+1}^\star + \alpha_K \boldsymbol{V_K} \cdot \boldsymbol{N_K^{K+1}} \right] \boldsymbol{N_K^{K+1}}. \end{aligned}$$

From the Cauchy–Schwartz inequality, we know that $\tilde{C}^2 < \tilde{A}\tilde{B}$. Hence, the linear system (4.21) always has a unique solution $(u_q^\star, v_q^\star)$. The pressures $P_{q,k}^{\star,k-1}, k = 2, \dots, K+1$, and $P_{q,k}^{\star,k}, k = 1, \dots, K$, are obtained from (4.20).

4.7.2. Case of a prescribed normal velocity. Let $\mathcal{V}_1^\star$ and $\mathcal{V}_{K+1}^\star$ be the values of the prescribed normal velocities on the boundary edges coming out of M_q. We distinguish the following two cases:

- $-\boldsymbol{N_1^0}$ and $\boldsymbol{N_K^{K+1}}$ are not colinear: in this case the value of $\boldsymbol{V_q^\star}$ is given by the boundary conditions, and the components of the vertex velocity are the solution of the linear system

$$\begin{cases} -N_{1,X}^0 u_q^\star - N_{1,Y}^0 v_q^\star = \mathcal{V}_1^\star, \\ N_{K,X}^{K+1} u_q^\star + N_{K,Y}^{K+1} v_q^\star = \mathcal{V}_{K+1}^\star. \end{cases}$$

Since the normals are not colinear, this linear system always has a unique solution. The pressures are computed from (4.20).

- $-\boldsymbol{N_1^0}$ and $\boldsymbol{N_K^{K+1}}$ are colinear: in this case, $\boldsymbol{V_q^\star}$ is not given directly, and we need to know the balance of momentum around the vertex M_q which takes the boundary conditions into account:

$$-L_1 P_{q,1}^{\star,1}\boldsymbol{N_1^0} + \sum_{k=2}^{K} L_k\left(P_{q,k}^{\star,k-1} - P_{q,k}^{\star,k}\right)\boldsymbol{N_k^{k-1}} + L_{K+1}P_{q,K+1}^{\star,K}\boldsymbol{N_{K+1}^K} = \left(-L_1\boldsymbol{N_1^0} + L_{K+1}\boldsymbol{N_K^{K+1}}\right)\Pi^\star, \tag{4.22}$$

where $\Pi^\star$ is an average pressure applied on the external side of the edges coming out of M_q. The pressure $\Pi^\star$ is a new unknown, but we have an additional equation corresponding to the boundary condition

$$\left(-L_1\boldsymbol{N_1^0} + L_{K+1}\boldsymbol{N_K^{K+1}}\right)\cdot\boldsymbol{V_q^\star} = L_1\mathcal{V}_1^\star + L_{K+1}\mathcal{V}_{K+1}^\star. \tag{4.23}$$

Using (4.20) in (4.22) combined with (4.23), we get

$$\begin{cases} \tilde{A}u_q^\star + \tilde{C}v_q^\star + D\Pi^\star = \widehat{SM_X}, \\ \tilde{C}u_q^\star + \tilde{B}v_q^\star + E\Pi^\star = \widehat{SM_Y}, \\ Du_q^\star + Ev_q^\star = L_1\mathcal{V}_1^\star + L_{K+1}\mathcal{V}_{K+1}^\star. \end{cases} \tag{4.24}$$

The coefficients $\tilde{A}$, $\tilde{B}$, and $\tilde{C}$ have already been defined in the previous section and we have set $D = -L_1N_{1,X}^0 + L_{K+1}N_{K,X}^{K+1}$, $E = -L_1N_{1,Y}^0 + L_{K+1}N_{K,Y}^{K+1}$. The right-hand side of (4.24) is

$$\widehat{\boldsymbol{SM}} = \sum_{k=2}^{K} L_k\left[P_{k-1} - P_k + (\alpha_{k-1}\boldsymbol{V_{k-1}} + \alpha_k\boldsymbol{V_k})\cdot\boldsymbol{N_{k-1}^k}\right]\boldsymbol{N_{k-1}^k} - L_1\left[P_1 - \alpha_1\boldsymbol{V_1}\cdot\boldsymbol{N_1^0}\right]\boldsymbol{N_1^0} + L_{K+1}\left[P_K + \alpha_K\boldsymbol{V_K}\cdot\boldsymbol{N_K^{K+1}}\right]\boldsymbol{N_K^{K+1}}.$$

The determinant of (4.24) is $\Delta = -\tilde{A}E + 2\tilde{C}DE - \tilde{B}D^2$. Using the fact that $|\tilde{C}| < \sqrt{\tilde{A}\tilde{B}}$, one can show that $\Delta < 0$ provided the mesh is not degenerated: (4.24) always admits a unique solution $(u_q^\star, v_q^\star)$, and then we can always define a unique set of pressures due to (4.20).

4. We have implemented a numerical test in order to determine the colinearity of the unit normal $-\boldsymbol{N_1^0}$ and $\boldsymbol{N_K^{K+1}}$ with a fixed threshold $\varepsilon > 0$. For numerical applications we have set $\varepsilon = 10^{-10}$. Namely, for $|\det(\boldsymbol{N_1^0}, \boldsymbol{N_K^{K+1}})| \leq \varepsilon$, $-\boldsymbol{N_1^0}$ and $\boldsymbol{N_K^{K+1}}$ are almost colinear, and we use the second alternative in order to avoid an ill-conditioned system.

4.8. Some remarks. From conservation arguments and one entropy inequality, we have been able to construct a nodal solver. This scheme shares some characteristics with the method developed in [9] and can be interpreted as a two-dimensional extension of the acoustic solver. It is interesting to realize that we need only the knowledge of the isentropic speed of sound: it is very easy to extend it to more general equation

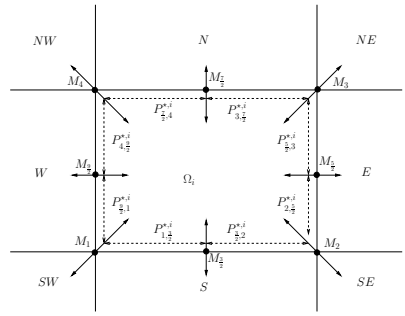


FIG. 4.6. *Stencil for a quadrangular mesh.*

of state. The precise form of the equation of state, analytical or tabulated, does not matter.

We have rigorously shown that the linear system which provides the components of the vertex velocity admits a unique solution, provided that the mesh is not degenerated; see the end of section 4.4. This is also true for vertices located on the boundary, whereas with the method developed in [9] some difficulties have been encountered for vertices located at the corner on the boundary of a rectangular domain.

We have also checked that our solver recovers the one-dimensional acoustic solver in the case of one-dimensional planar and cylindrical flows, contrarily to [9]. More precisely, the calculation of the node velocity with the solver described in [9] gives, for a one-dimensional flow with planar symmetry,

$$u_q^\star = \frac{P_1 - P_2 + \rho_1 c_1 u_1 + \rho_2 c_2 u_2}{\rho_1 c_1 + \rho_2 c_2} \frac{\sqrt{\Delta X^2 + \Delta Y^2}}{\Delta Y},$$

where $L_1 = L_3 = \Delta X$ and $L_2 = L_4 = \Delta Y$ are mesh spacing along directions $\boldsymbol{e_X}$ and $\boldsymbol{e_Y}$. With this nodal solver, one recovers the one-dimensional acoustic solver velocity multiplied by a factor which depends on the aspect ratio of the cells.

Last, our method fully answers the questions raised at the beginning of this paper since it uniquely provides the vertex velocity and the face fluxes. The main new features of our scheme is the introduction of four pressures on each edge, two for each node on each side of the edge, that is, eight pressures at each node for a quadrangular mesh.

We can compute, in a coherent way, the vertex motion as well as the face fluxes. This solver has been constructed delocalizing the fluxes from the faces to the vertices. Consequently, the conservation of momentum and energy is obtained around the node. This implies a fundamental difference with a standard finite volume scheme.

This difference can be illustrated by the consideration of a mesh made of quadrangular cells. We denote by 1, 2, 3, and 4 the vertices of the cell Ω_i and by S, SE, E, NE, N, NW, O, and SW the neighboring cells; see Figure 4.6. In a standard finite volume scheme with an approximate Riemann solver, the cell Ω_i exchanges information only with the neighboring cells having a common face, i.e., the cells S, E, N, and O. This is a 5 point scheme.

With our scheme, the cell Ω_i exchanges momentum and energy with the neighboring cells having a common vertex, i.e., the cells S, SE, E, NE, N, NW, W, and SW. This is a 9 point scheme. The four additional cells reinforce the genuinely multidimensional nature of our scheme.

5. Time discretization. In this section, we time discretize the system that describes the evolution of the physical variables $(\tau_i, \boldsymbol{V_i}, E_i)$ in the cell Ω_i:

$$m_i \frac{d}{dt}\tau_i - \sum_{r=1}^{R(i)} L_{r,r+1} \boldsymbol{V^\star_{r,r+1}} \cdot \boldsymbol{N_{r,r+1}} = 0, \tag{5.1i}$$

$$m_i \frac{d}{dt}\boldsymbol{V_i} + \sum_{r=1}^{R(i)} L_{r,r+1} P^\star_{r,r+1} \boldsymbol{N_{r,r+1}} = \boldsymbol{0}, \tag{5.1ii}$$

$$m_i \frac{d}{dt}E_i + \sum_{r=1}^{R(i)} L_{r,r+1} \left(P\boldsymbol{V}\right)^\star_{r,r+1} \cdot \boldsymbol{N_{r,r+1}} = 0, \tag{5.1iii}$$

where the face fluxes $\boldsymbol{V^\star_{r,r+1}}$, $P^\star_{r,r+1}$ are evaluated by our solver. For the time discretization of (5.1) we use a classical forward Euler scheme. A more sophisticated time discretization is not required here, since the scheme is only first order accurate in space. However, time discretization of (5.1) should be done carefully so that all the properties of the semidiscrete system are kept. They are as follows:

- the variation of volume is coherent with the vertex motion;
- momentum and total energy are conserved;
- the scheme satisfies an entropy inequality.

The first two properties would impose an explicit time discretization; the third would impose an implicit discretization. We choose an explicit time discretization, knowing that the entropy inequality will be a priori satisfied under a CFL-type condition.

We assume to know the physical properties in the cell Ω_i and the geometrical characteristics of the cell at the beginning of the time step t_n, i.e., τ_i^n, $\boldsymbol{V_i^n}$, E_i^n, P_i^n, X_r^n, and Y_r^n. We are going to compute their values at t^{n+1} and we set $\Delta t = t^{n+1} - t^n$.

5.1. Mesh motion and variation of volume. The nodal solver provides the vertex velocities $\boldsymbol{V_r^\star}$ and the face pressures at $P^{\star,i}_{r,r+\frac{1}{2}}$, $P^{\star,i}_{r+\frac{1}{2},r+1}$ from the physical variables and geometry characteristics evaluated at time t^n. The explicit time integration of the trajectory equation provides the location of vertices for any time $t \in [t^n, t^{n+1}]$:

$$\begin{aligned} X_r(t) &= X_r^n + (t - t^n) u_r^\star, \\ Y_r(t) &= Y_r^n + (t - t^n) v_r^\star. \end{aligned} \tag{5.2}$$

In this way, one gets the location of the vertices at time t^{n+1}:

$$\begin{aligned} X_r^{n+1} &= X_r^n + u_r^\star \Delta t, \\ Y_r^{n+1} &= Y_r^n + v_r^\star \Delta t. \end{aligned} \tag{5.3}$$

From this we deduce that $\left(L_{r,r+1}\boldsymbol{N_{r,r+1}}\right)(t) = \left[Y_{r+1}(t) - Y_r(t), -\left(X_{r+1}(t) - X_r(t)\right)\right]^t$ is linear in time, so that

$$\int_{t^n}^{t^{n+1}} \left(L_{r,r+1}\boldsymbol{N_{r,r+1}}\right)(t) \mathrm{d}t = \frac{\Delta t}{2}\left[\left(L_{r,r+1}\boldsymbol{N_{r,r+1}}\right)^n + \left(L_{r,r+1}\boldsymbol{N_{r,r+1}}\right)^{n+1}\right]. \tag{5.4}$$

This last result enables us to write an approximation of the volume equation which is coherent with the mesh motion, namely,

(5.5)
$$m_i\left(\tau_i^{n+1}-\tau_i^n\right)-\frac{\Delta t}{4}\sum_{r=1}^{R(i)}\left[\left(L_{r,r+1}\boldsymbol{N_{r,r+1}}\right)^n+\left(L_{r,r+1}\boldsymbol{N_{r,r+1}}\right)^{n+1}\right]\cdot\left(\boldsymbol{V_r^\star}+\boldsymbol{V_{r+1}^\star}\right)=0.$$

5. We recall that (5.1i) is completely equivalent to the evolution equation of $\mathcal{A}_i$ (area of cell Ω_i), since $m_i\tau_i=\mathcal{A}_i$ and m_i is constant over time; see section 3.2. Knowing that the nodal velocity is computed at time t^n, the right-hand side of (5.1i) is a linear function of time over $[t^n,t^{n+1}]$ thanks to (5.2). Hence, the time integration formula given by (5.4) is exact. Consequently, we can write $\tau_i^{n+1}=\frac{\mathcal{A}_i^{n+1}}{m_i}$. In a nutshell, the second order time integration of the volume flux provides a rigorous compatibility between the specific volume equation and the mesh motion.

5.2. Momentum and total energy. The approximation of the momentum and the total energy equation is fully explicit in order to conserve exactly the momentum and the total energy. The lengths (of edges) and the normals have the same definition as previously. We get

$$m_i\left(\boldsymbol{V_i^{n+1}}-\boldsymbol{V_i^n}\right)+\frac{\Delta t}{2}\sum_{r=1}^{R(i)}L_{r,r+1}^n\left(P^{\star,i}_{r,r+\frac12}+P^{\star,i}_{r+\frac12,r+1}\right)\boldsymbol{N_{r,r+1}^n}=\boldsymbol{0},\tag{5.6}$$

and for the total energy

$$m_i\left(E_i^{n+1}-E_i^n\right)+\frac{\Delta t}{2}\sum_{r=1}^{R(i)}L_{r,r+1}^n\left(P^{\star,i}_{r,r+\frac12}\boldsymbol{V_r^\star}+P^{\star,i}_{r+\frac12,r+1}\boldsymbol{V_{r+1}^\star}\right)\cdot\boldsymbol{N_{r,r+1}^n}=0.\tag{5.7}$$

5.3. Time step limitation. The time step is evaluated following two criteria. The first is a standard CFL criterion which guarantees heuristically the monotone behavior of the entropy. The second is more intuitive, but appears very useful in practice: we limit the variation of the volume of cells over one time step.

5.3.1. CFL criterion. We propose a CFL-like criterion in order to ensure a positive entropy production in cell Ω_i during the time step. At time t_n, for each cell Ω_i we denote by λ_i^n the minimal value of the distance between two nodes of the cell. We define

$$\Delta t_E=C_E\min_{i=1,\dots,I}\frac{\lambda_i^n}{c_i^n},$$

where C_E is a strictly positive coefficient and c_i is the sound speed in the cell. The coefficient C_E is computed heuristically, and we provide no rigorous analysis which allows such formula. However, extensive numerical experiments show that $C_E=0.3$ is a value which provides good numerical results. We have also checked that this value is compatible with a monotone behavior of entropy. The rigorous derivation of this criterion could be obtained by computing the time step which ensures a positive entropy production in cell Ω_i from time t^n to t^{n+1}.

5.3.2. Criterion on the variation of volume. Using (5.2), we can compute the area of the cell i at any time $t \in [t^n, t^{n+1}]$. Indeed, we have

$$\mathcal{A}_i(t) = \frac{1}{2}\sum_{r=1}^{R(i)} \left[X_r^n + (t-t^n)u_r^\star\right]\left[Y_{r+1}^n + (t-t^n)v_{r+1}^\star\right] - \left[Y_r^n + (t-t^n)v_r^\star\right]\left[X_{r+1}^n + (t-t^n)u_{r+1}^\star\right]. \tag{5.8}$$

Simple algebra gives

$$\frac{d}{dt}\mathcal{A}_i(t^n) = \frac{1}{2}\sum_{r=1}^{R(i)} \left(u_r^\star Y_{r+1}^n + v_{r+1}^\star X_r^n - u_{r+1}^\star Y_r^n - v_r^\star X_{r+1}^n\right).$$

After a Taylor expansion we estimate the area at $t = t^{n+1}$ by

$$\mathcal{A}_i^{n+1} = \mathcal{A}_i^n + \frac{d}{dt}\mathcal{A}_i(t^n)\Delta t.$$

Let C_V be a strictly positive coefficient, $C_V \in]0,1[$. We look for Δt such that

$$\frac{|\mathcal{A}_i^{n+1} - \mathcal{A}_i^n|}{\mathcal{A}_i^n} \leq C_V.$$

To do so, we define

$$\Delta t_V = C_V \min_{i=1,\dots,I}\left\{\frac{\mathcal{A}_i^n}{\left|\frac{d}{dt}\mathcal{A}_i(t^n)\right|}\right\}.$$

For numerical applications, we choose $C_V = 0.1$.

Finally, the estimation of the next time step Δt^{n+1} is given by

$$\Delta t^{n+1} = \min\left(\Delta t_E, \Delta t_V, C_M \Delta t^n\right), \tag{5.9}$$

where Δt^n is the current time step and C_M is a multiplicative coefficient, which allows the time step to increase. We generally set $C_M = 1.01$.

5.4. Description of the algorithm.

1. **Initialization**
   At time $t = t^n$ we know in each cell Ω_i, $i = 1, \dots, I$, the fluid variables τ_i^n, $\boldsymbol{V_i^n}$, E_i^n, ρ_i^n, c_i^n, P_i^n and the geometrical characteristics X_r^n,Y_r^n, $L_{r,r+1}^n$, $\boldsymbol{N_{r,r+1}^n}$ for $r = 1, \dots, R(i)$.
2. **Nodal solver**
   - For each internal vertex M_q, $q = 1, \dots, Q$, we first compute the velocity $\boldsymbol{V_q^\star}$ by solving the linear system (4.15). Then we evaluate the edge pressures $P_{q,k}^{\star,k}$ and $P_{q,k}^{\star,k-1}$ with (4.12) for each edge impinging on M_q.
   - For each boundary vertex, we first compute the velocity $\boldsymbol{V_q^\star}$ by solving (4.21) or (4.24). Then we calculate the edge pressures $P_{q,k}^{\star,k}$ and $P_{q,k}^{\star,k-1}$ using (4.20).
3. **Time step limitations**
   We compute Δt_V and Δt_E; then we predict Δt^{n+1} from $\Delta t^{n+1} = \min(\Delta t_E, \Delta t_V, C_M \Delta t^n)$.

4. **Update of the geometrical quantities**
   We compute X_r^{n+1} and Y_r^{n+1} from (5.3); then we deduce $L_{r,r+1}^{n+1}$ and $\boldsymbol{N_{r,r+1}^{n+1}}$.
5. **Update of the physical variables**
   We compute the face fluxes from the vertex velocity and the edge pressures; then we get τ_i^{n+1}, $\boldsymbol{V_i^{n+1}}$, and E_i^{n+1} from (5.5), (5.6), and (5.7).
6. **Equation of state**
   The internal energy is given by $\varepsilon_i^{n+1} = E_i^{n+1} - \frac{1}{2}\|\boldsymbol{V_i^{n+1}}\|^2$; then we get the pressure P_i^{n+1} and the isentropic sound speed c_i from the equation of state.

6. Numerical results. In this section, we present several test cases in order to validate our numerical scheme. For each case we use a perfect gas equation of state, namely $P = (\gamma - 1)\rho\varepsilon$, where γ is the adiabatic index. We begin by several classical one-dimensional test cases characterized by the presence of shock waves, of various strengths, rarefaction waves, and contact discontinuities. We also provide an isentropic compression. For all these test cases an analytical solution is available so that we can make accurate comparisons. Then, we give an original test representative of the domain of hydrodynamic instabilities. The aim of this test is to show the ability of our scheme to reproduce the growth of such instabilities in the linear regime. Last, we propose a two-dimensional test case which consists in imploding a double layer cylindrical shell under an anisotropic pressure load.

6.1. Multimaterial Sod's shock tube problem. We are concerned with a multimaterial variant of Sod's shock tube problem defined in [26]. We consider a shock tube of unity length (see Figure 6.1). At the initial time, the states on the left- and right-hand sides of $x = 0.5$ are constant. The left-hand state is a high pressure fluid characterized by $(\rho_l, P_l, u_l) = (1, 1, 0)$, and the right-hand state is a low pressure fluid defined by $(\rho_r, P_r, u_r) = (0.125, 0.1, 0)$. The equation of state is defined on the left by $\gamma_l = \frac{7}{5}$ and by $\gamma_r = \frac{5}{3}$ on the right. The computational domain is defined by $(x, y) \in [0, 1] \times [0, 0.1]$. The initial mesh is a Cartesian grid, and we denote by n_x and n_y the number of cells in the x and y directions. The boundary conditions are wall boundary conditions: the normal velocity is set to zero. Three simulations are done with $n_y = 10$ and $n_x = 100$, $n_x = 200$, and finally $n_x = 400$ in order to check the mesh convergence of the scheme. The time step is evaluated following (5.9). In Figure 6.2, we compare the profiles of the physical quantities at $t = 0.2$ with the analytical quantities for three different meshes. The profiles are similar to what could be obtained with a one-dimensional first order Lagrangian scheme based on the acoustic solver; see the interpretation of our scheme in section 4.6.1. The shock is smeared on several cells, and the fan is correctly described for a first order scheme. The contact discontinuity is resolved in several cells; this is a consequence of the entropic nature of the solver. We notice an undershoot for the density profile and an overshoot for the internal energy profile which are classical for Lagrangian methods. As expected, the finer the mesh is, the better the results are.

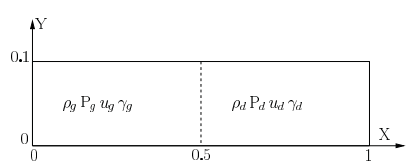


FIG. 6.1. *Initial data for Sod's shock tube test case.*

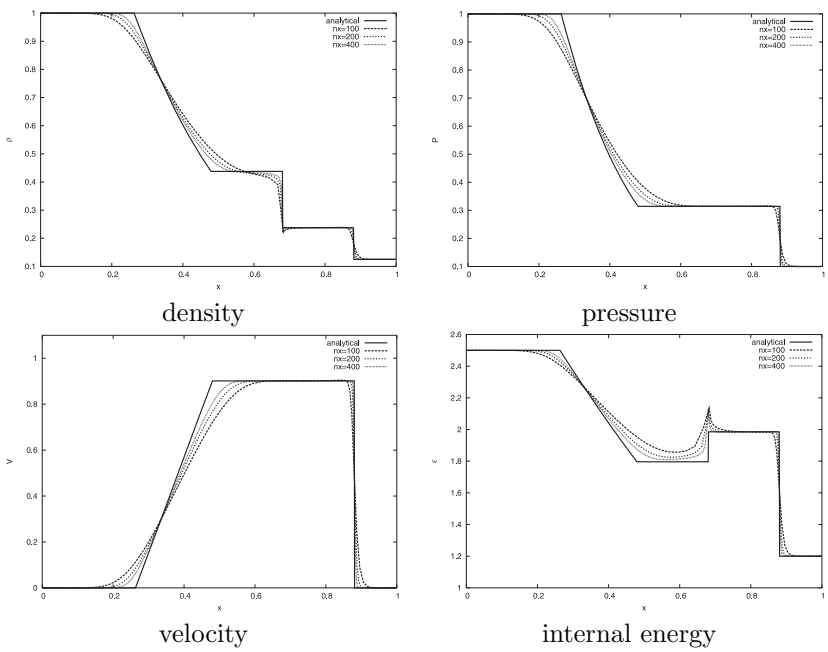


FIG. 6.2. *Cross-section at $t = 0.2$, comparison with the analytical solution.*

6.2. Noh's test case. Noh's test case is the implosion of a cylinder of unit radius. The cylinder is filled with a monoatomic perfect gas ($\gamma = 5/3$). The initial state is $(\rho^0, P^0, \boldsymbol{V^0}) = (1, 0, -\boldsymbol{e_R})$, where $\boldsymbol{e}_R$ represents the radial unit vector. This case, defined by Noh in [23], admits a self-similar solution: a shock wave moves inwards at the constant speed $D = 1/3$. The symmetry of the problem enables us to limit the computational domain to an angular sector of angle Θ; see Figure 6.3. The mesh consists of regular angular sectors, and the domain is meshed with triangles near the center and with quadrangles elsewhere. This enables us to respect the radial symmetry of the solution. We denote by n_x (resp., n_y) the number of layers (resp., angular sectors). In what follows, $\theta = \frac{\Theta}{n_y}$. The test case is initialized with $P^0 = 10^{-6}$. The boundary conditions are wall conditions on the two external edges (hence zero normal velocity), and we impose a constant pressure $P^\star = P^0$ on the external radius $R = 1$. The time step is computed following the condition (5.9). In Figure 6.4 we have represented the results at time $t = 0.6$ for three meshes defined with $\Theta = 12°$ and $(n_x, \theta) = (100, 4°)$, $(200, 2°)$, and $(400, 1°)$. The convergence behavior of the scheme is very satisfactory. The timing, the density levels, and the pressure levels after the shock wave are correct. As for Sod's problem, we notice an overshoot for the internal energy and an undershoot for the density that are characteristic of the wall heating phenomena; see [23]. Figure 6.5 displays the mesh at $t = 0.6$ for $\Theta = 90°$ and $(n_x, \theta) = (100, 2°)$, and one can see that the cylindrical symmetry is perfectly respected.

6.3. Kidder's isentropic compression. In [17], Kidder has analytically computed the solution of the isentropic compression of a cylindrical shell filled with perfect gas. We briefly recall the main features of this solution in order to define the test case. Initially, the shell has the internal (resp., external) radius r_1 (resp., r_2). Let P_1, P_2,

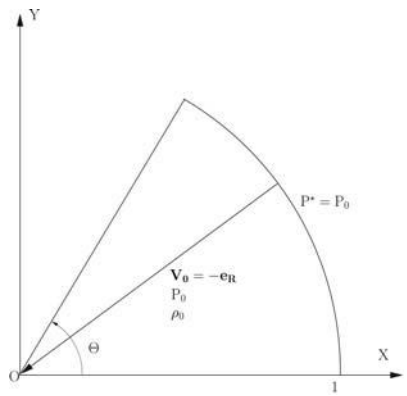


FIG. 6.3. *Noh's case: initial conditions.*

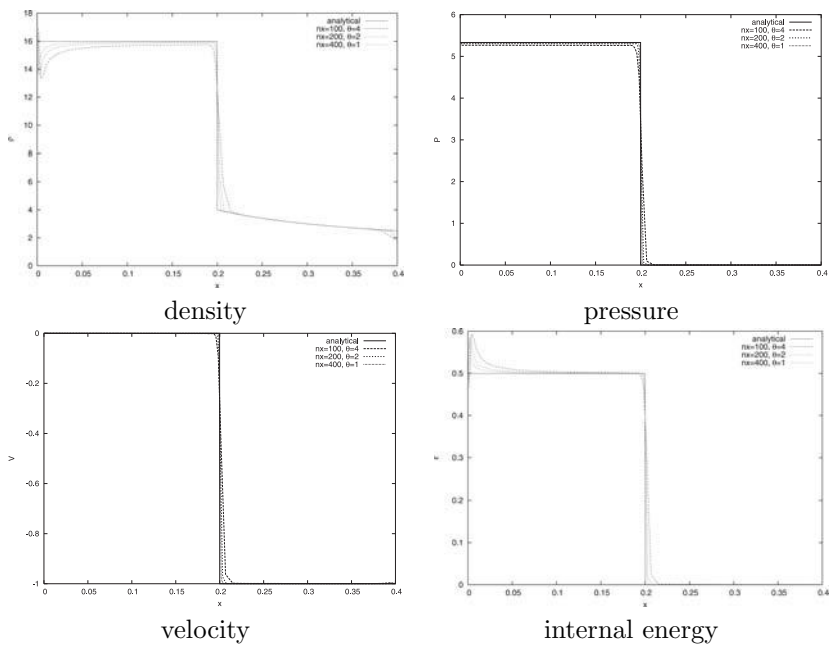


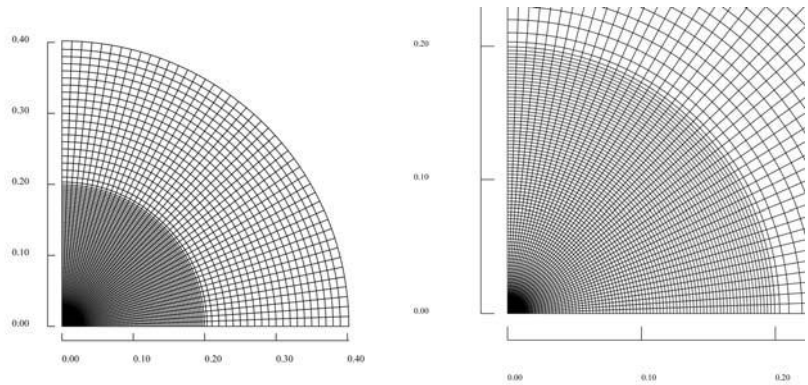
FIG. 6.4. *Cross-section for Noh's case at $t = 0.6$, comparison with the analytical solution.*

ρ_1, and ρ_2 be the pressures and densities located at r_1 and r_2. Since the compression is isentropic, we define $s = \frac{P_2}{\rho_2^\gamma}$ and we have $\rho_1 = \rho_2(\frac{P_1}{P_2})^{\frac{1}{\gamma}}$. The initial density and pressure profiles are given by (for $r \in [r_1, r_2]$)

$$\rho^0(r) = \left(\frac{r_2^2 - r^2}{r_2^2 - r_1^2} \rho_1^{\gamma-1} + \frac{r^2 - r_1^2}{r_2^2 - r_1^2} \rho_2^{\gamma-1} \right)^{\frac{1}{\gamma-1}},$$

$$P^0(r) = s \left(\rho^0(r) \right)^\gamma.$$

The initial velocity is set to zero: $u_0(r) = 0$. The value of γ is set to $\gamma = 1 + \frac{2}{\nu}$ with $\nu = 1, 2, 3$ whether we have a planar, cylindrical, or spherical symmetry. These values are used in order to obtain a fully analytical solution. In our case we have $\nu = 2$, and



Mesh Zoom near the origin

FIG. 6.5. *Noh's case: mesh at* $t = 0.6$.

therefore $\gamma = 2$. The focusing time τ of the shell is defined (for these parameters) by

$$\tau = \sqrt{\frac{\gamma - 1}{2} \frac{r_2^2 - r_1^2}{c_2^2 - c_1^2}},$$

where $c_i = \sqrt{s\gamma\rho_i^{\gamma-1}}$ for $i = 1, 2$ is the isentropic sound speed at $r = r_i$. The isentropic compression is obtained by imposing the following pressure laws at the internal and external faces of the shell:

$$P(R(r_1, t), t) = P_1 h(t)^{-\frac{2\gamma}{\gamma-1}},$$
$$P(R(r_2, t), t) = P_2 h(t)^{-\frac{2\gamma}{\gamma-1}}.$$

Let $R(r, t)$ be the Eulerian radius (i.e., $R = \sqrt{X^2 + Y^2}$) at $t > 0$ of a fluid particle initially located on the circle of radius r. Looking for a solution of the form $R(r, t) = h(t)r$, we get

$$h(t) = \sqrt{1 - \left(\frac{t}{\tau}\right)^2}.$$

The analytical forms of the physical variables at $t \in [0, \tau[$ are

$$\rho(R(r, t), t) = h(t)^{-\frac{2}{\gamma-1}} \rho^0 \left[\frac{R(r, t)}{h(t)}\right],$$
$$u(R(r, t), t) = \frac{d}{dt} h(t) \frac{R(r, t)}{h(t)},$$
$$P(R(r, t), t) = h(t)^{-\frac{2\gamma}{\gamma-1}} P^0 \left[\frac{R(r, t)}{h(t)}\right].$$

In this test case, we have used the parameters $r_1 = 0.9$, $r_2 = 1$, $P_1 = 0.1$, $P_2 = 10$, and $\rho_2 = 10^{-2}$. From this, we get $\rho_1 = 10^{-3}$, $s = 10^5$, and $\tau = 7.265\ 10^{-3}$. The computational domain consists of the quarter of an annulus; see Figure 6.6. Once

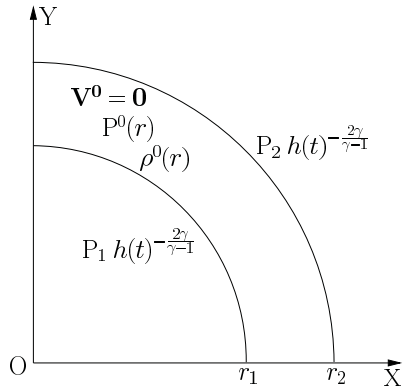


FIG. 6.6. *Initial conditions for Kidder's isentropic compression.*

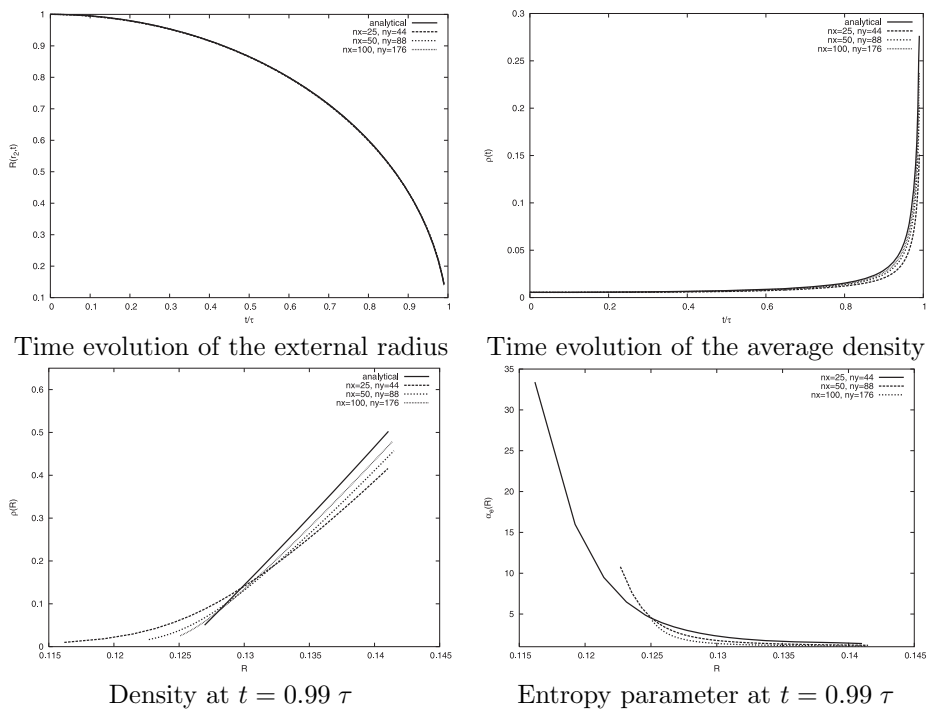


FIG. 6.7. *Kidder's isentropic compression, comparison with the analytical solution.*

more, we impose wall boundary conditions on the axes $X = 0$, $Y = 0$ and the previous pressure laws on the internal and external faces of the domain. The simulations are run on regular polar meshes defined by $(n_x, n_y) = (25, 44),\ (50, 88),\ (100, 176)$. The computations are run up to a final time which is very close to the focusing time, i.e., $t = 0.99\ \tau$. In all that follows, the time is adimensionalized by the focusing time τ.

Figure 6.7, where the variation of the external radius of the shell is plotted, shows the excellent agreement with the analytical solution. The agreement is good for the average density[2] $\bar{\rho}(t)$ near the focusing point; however, the numerical values

[2] $\bar{\rho}(t)$ is evaluated from the analytical formula $\bar{\rho}(t) = 0.5(\rho_1 + \rho_2)h(t)^{-2}$.

are smaller than the exact ones. There exists a gap between numerical and analytical density profiles at the final time. This gap gets smaller and smaller with mesh refinement; thus we believe it is a consequence of the numerical diffusion. This observation is confirmed by the profile of the entropy parameter $\alpha_e = \frac{P}{s\rho^\gamma}$. It indicates the entropy production of the scheme, and we have $\alpha_e = 1$ in the analytical case. The gap between the exact solution and the numerical one is large on the internal face of the shell when the mesh is coarse, and it also gets smaller with mesh refinement. The large entropy production of the scheme degrades the simulated compression, which explains the shape of the density profile on the internal face of the shell.

6.4. Saltzman's shock tube. We consider now the movement of a planar shock wave on a Cartesian grid that has been stretched [10]. This is a well-known difficult test case that enables us to validate the robustness of our scheme when the mesh is not aligned with the fluid flow. The computational domain is the rectangle $(x, y) \in [0, 1] \times [0, 0.1]$. The initial mesh is obtained by transforming a uniform grid of 100×10 cells with the mapping

$$\begin{aligned} x_{str} &= x + (0.1 - y)\sin(x\pi), \\ y_{str} &= y. \end{aligned}$$

For the material we use the equation of state of a monoatomic gas ($\gamma = 5/3$). The initial state is $(\rho^0, P^0, \boldsymbol{V^0}) = (1, 0, 0)$. The boundary condition at $x = 0$ is a normal velocity $V^\star = -1$ (inflow velocity). On all the other boundaries, we set up wall conditions.

The exact solution is a planar shock wave that moves at speed $D = 4/3$ from left to right. The propagation of the shock wave at $t = 0.6$ is displayed in Figure 6.8. The important result is that our scheme preserves the one-dimensional solution very well (except for the first and last layers in the vertical direction, this is an effect of up and down wall boundary conditions). Figure 6.9 shows the mesh at $t = 0.75$ when the shock wave hits the right boundary ($x = 1$). Behind the shock wave, the initial mesh is distorted, and all the horizontal lines stay almost parallel with respect to the others. The density profile at $t = 0.6$ is displayed in Figure 6.10. The shock level is not uniform, but it oscillates around the exact value $\rho = 4$.

The robustness of our scheme is clearly demonstrated by this test case: we are able to reach time $t = 0.93$ which corresponds to two successive rebounds of the shock wave on the vertical boundaries of the domain; see Figure 6.11. After this time, the computation stops because the time step becomes too small (less than 10^{-15}). This too small time step occurs when nodes are too close to each other (see Figure 6.11, top left corner). In order to go beyond this time we have to improve the quality of the mesh by using a rezoning procedure and a remapping phase. This is the arbitrary Lagrangian–Eulerian (ALE) extension of our scheme which is under development; it will be the subject of a forthcoming paper.

6.5. Sedov's test case. We consider the propagation of a high intensity cylindrical shock wave generated by a strong explosion; see, for instance, [30]. All the boundary conditions are wall conditions. The gas of interest is monoatomic ($\gamma = \frac{5}{3}$). It is initially at rest ($(\rho^0, P^0, \boldsymbol{V^0}) = (1, 0, 0)$) and we set up an energy spike at the center. The simulation is done first on a Cartesian grid and then on a polar grid. In order to compare the results, the same amount of energy is added for each computation.

On the Cartesian grids, the computational domain is $(x, y) \in [0, 1] \times [0, 1]$. In the case of the coarsest mesh (i.e., $(n_x, n_y) = (50, 50)$), the internal specific energy of

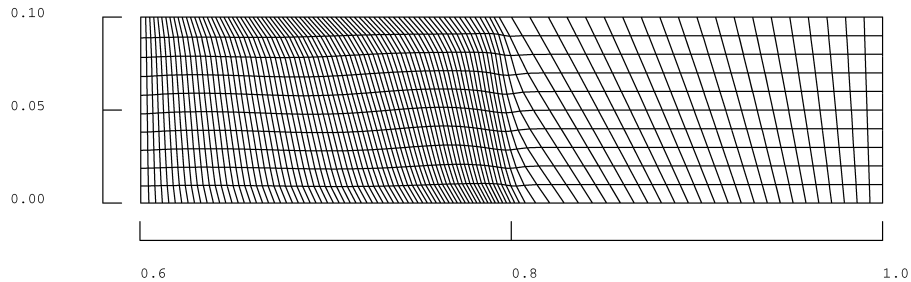


FIG. 6.8. *Saltzman's test: mesh at* $t = 0.6$.

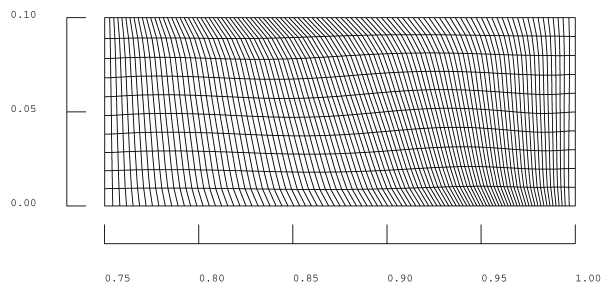


FIG. 6.9. *Saltzman's test: mesh at* $t = 0.75$.

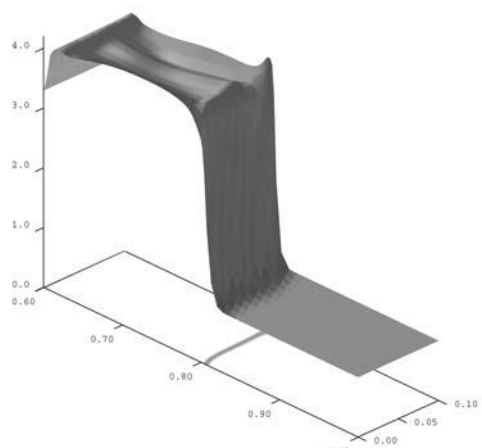


FIG. 6.10. *Saltzman's test: density profile at* $t = 0.6$.

the first cell is set to $\epsilon^0 = 5000$. This corresponds to subdomain $(x, y) \in [0, 1/50] \times [0, 1/50]$. The finer meshes are $(n_x, n_y) = (100, 100)$ and $(n_x, n_y) = (200, 200)$. In order to have the same amount of energy, we also initialize with the specific internal energy $\epsilon^0 = 5000$ in $[0, 1/50] \times [0, 1/50]$.

In the case of the polar grids, the domain is defined by $(r, \theta) \in [0, 1] \times [0, 90°]$. The

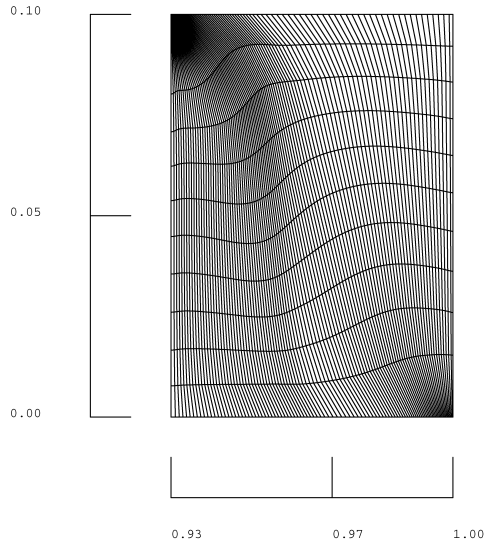


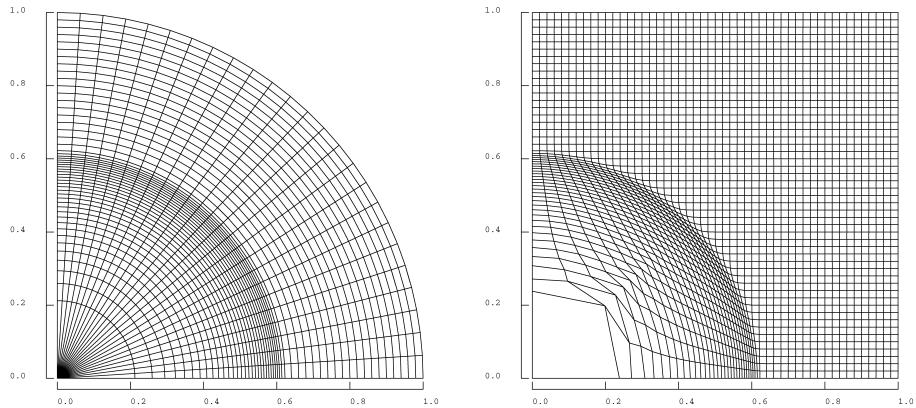
FIG. 6.11. *Saltzman's test: mesh at* $t = 0.93$.

mesh is made of quadrangles with triangles at the center. The parameters (n_x, n_y) correspond to the number of layers in the radial direction and of angular sectors. The coarsest mesh is defined by $(n_x, n_y) = (50, 25)$. We initialize the specific internal energy on triangles and define the subdomain $(r, \theta) \in [0, 1/50] \times [0, 90°]$. As the area to initialize is different in this case, we need to modify the internal energy so that the energy amount is the same as in the case of the Cartesian grid. Thus we choose $\epsilon^0 = 5000\frac{4}{\pi}$. As for the Cartesian mesh, the finest polar grids are $(n_x, n_y) = (100, 50)$ and $(n_x, n_y) = (200, 100)$, and in both cases, the cells to initialize are those in the subdomain $(r, \theta) \in [0, 1/50] \times [0, 90°]$.

Figure 6.12 represents the final mesh at $t = 0.1$ obtained for the two types of meshes. Clearly, our initialization leads to the same results. However, we notice that on the Cartesian grid nonconvex cells appear, but this does not pose any problem to the nodal solver.

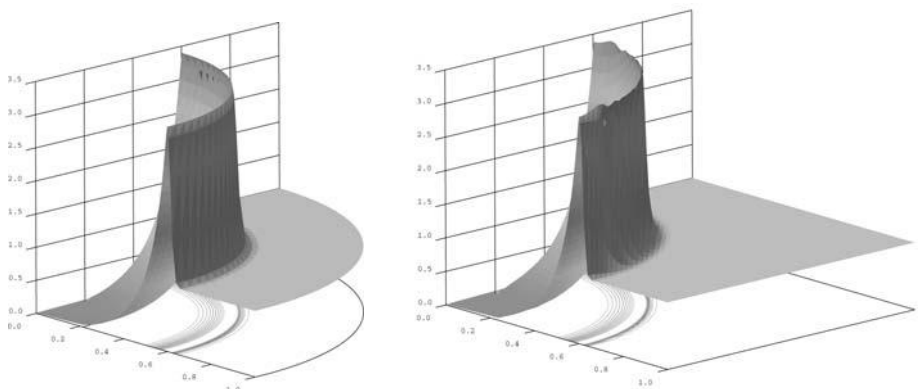
The comparison of the density profiles for the two types of grids is done in Figures 6.13, 6.15, and 6.16. We notice that for the polar grid, the cylindrical symmetry of the flow is perfectly preserved; see Figures 6.12 and 6.13. The symmetry is also well preserved on Cartesian grids, as shown in Figure 6.14 on the left, where the density profiles on the diagonal are slightly different from those on the x- and y- axes (which are identical). In Figure 6.14 on the right are represented the density in all cells as function of the centroid radius of each cell. We also observe that the symmetry is well preserved. Consequently, these results on Cartesian grids show the ability of our new scheme to provide good results even when the mesh is not aligned with the flow. Moreover, Figures 6.15 and 6.16 demonstrate that a mesh convergence study for both grids leads to similar results.

6.6. Perturbation of the isentropic Kidder's compression. In this section, we give an original test case representative of the field of hydrodynamic instabilities.



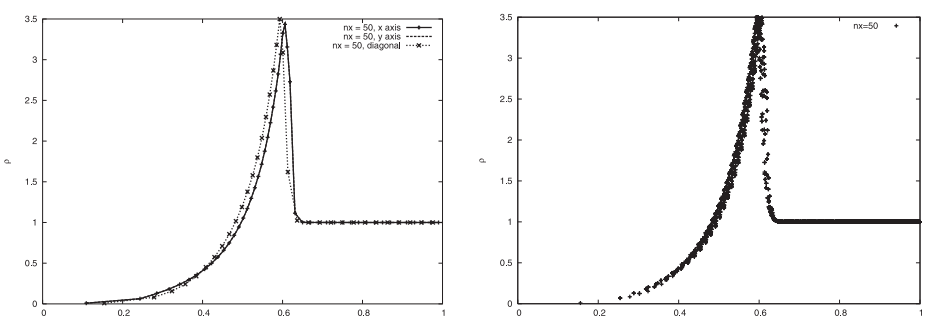
Polar grid Cartesian grid

FIG. 6.12. *Sedov's test case: mesh at* $t = 0.1$ *for the polar and Cartesian meshes.*



Polar grid Cartesian grid

FIG. 6.13. *Sedov's test case: density level at* $t = 0.1$.



Density profiles along different axes Density in all the cells

FIG. 6.14. *Sedov's test case on Cartesian grid at* $t = 0.1$.

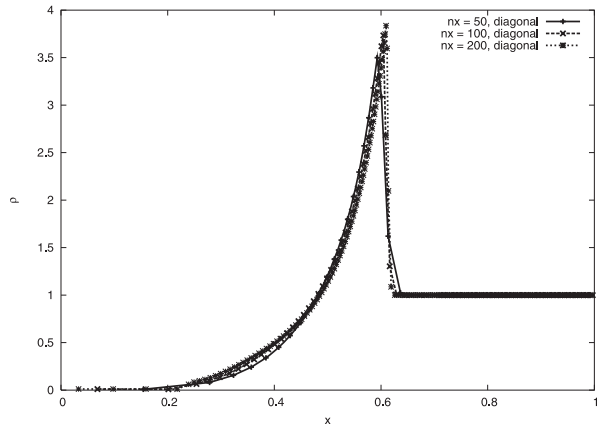


FIG. 6.15. *Sedov's test case: Cartesian grid convergence for density profiles at* $t = 0.1$

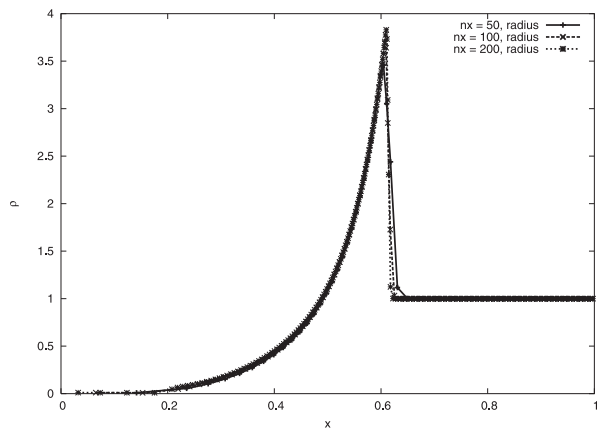


FIG. 6.16. *Sedov's test case: polar grid convergence for density profiles at* $t = 0.1$.

The aim of this test is to show the ability of our new scheme to reproduce the growth of such instabilities. In order to define the test, we consider the isentropic implosion of a cylindrical shell as in section 6.3. We initially perturb the internal and external faces of the shell; see Figure 6.6. Then, we study the time evolution of the perturbations. An approximate analytical solution is available for the perturbation amplification in the linear regime. This solution is obtained with the technique developed in [13]. We briefly recall the main results about the analytical solution in order to set up the test case. More details can be found in [22] and [3].

First we define the mesh perturbation. We are given a Cartesian frame $(0, x, y)$ and polar coordinates (r, θ). The polar coordinates of the vertices are given by

$$\begin{aligned} r^p &= r + a_0 \xi_r(r, \theta), \\ \theta^p &= \theta + a_0 \xi_\theta(r, \theta), \end{aligned}$$

where a_0 is the amplitude of the initial perturbation. The vector $\boldsymbol{\xi} = \binom{\xi_r}{\xi_\theta}$ is defined by

$$\begin{aligned} \xi_r(r, \theta) &= n \left(A_1 r^{-n-1} + A_2 r^{n-1} \right) \cos(n\theta), \\ \xi_\theta(r, \theta) &= -n \left(A_1 r^{-n-1} + A_2 r^{n-1} \right) \sin(n\theta). \end{aligned} \tag{6.1}$$

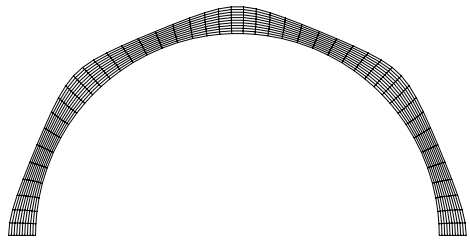


FIG. 6.17. *Perturbed mesh.*

The integer n is the perturbation mode, and A_1, A_2 are constants which are determined by the perturbations ξ_1 and ξ_2 given at the internal and external interfaces, i.e., $\xi_1 = \xi_r(r_1, 0)$ and $\xi_2 = \xi_r(r_2, 0)$. We have

$$\begin{cases} A_1 = -\dfrac{1}{n} \dfrac{\xi_1 r_2^{n-1} - \xi_2 r_1^{n-1}}{r_1^{-n-1} r_2^{n-1} - r_1^{n-1} r_2^{-n-1}}, \\ A_2 = -\dfrac{1}{n} \dfrac{\xi_1 r_2^{-n-1} - \xi_2 r_1^{-n-1}}{r_1^{-n-1} r_2^{n-1} - r_1^{n-1} r_2^{-n-1}}. \end{cases}$$

The analytical form (6.1) is justified by the fact that $\boldsymbol{\xi}$ derives from a potential. We check that $\nabla \cdot \boldsymbol{\xi} = 0$, that is, that the perturbation is incompressible. We have a divergence-free perturbation.

Using the previous results, we get the Cartesian coordinates of the perturbed mesh

$$\begin{aligned} x^p &= \left(1 + \frac{a_0}{r}\xi_r\right) \left[x \cos(a_0 \xi_\theta) - y \sin(a_0 \xi_\theta)\right], \\ y^p &= \left(1 + \frac{a_0}{r}\xi_r\right) \left[x \sin(a_0 \xi_\theta) + y \cos(a_0 \xi_\theta)\right]. \end{aligned} \tag{6.2}$$

This is illustrated by Figure 6.17, which represents the perturbed mesh for the half-annulus of radii $r_1 = 0.9$, $r_2 = 1$, with the following data: perturbation mode $n = 8$, internal and external levels $\xi_1 = 0$, $\xi_2 = 1$, amplitude $a_0 = 0.02$. In [22], we show that the amplification factor $a(\theta, t)$ of the perturbation at the external interface is given by

$$a(\theta, t) = n \left[A_1 r^{-n-1} G_1(t) + A_2 r^{n-1} G_2(t)\right] \cos(n\theta),$$

where $t \in [0, 1[$ is the time.[3] The functions G_1 and G_2 are

$$\begin{aligned} G_1(t) &= \sqrt{1 - t^2} \cos\left[\tfrac{\sqrt{n}}{2} \log\left(\tfrac{1-t}{1+t}\right)\right], \\ G_2(t) &= \tfrac{1}{2}\sqrt{1 - t^2} \left[\left(\tfrac{1-t}{1+t}\right)^{\sqrt{n}/2} + \left(\tfrac{1+t}{1-t}\right)^{\sqrt{n}/2}\right]. \end{aligned}$$

[3]The time is adimensionalized by the focusing time τ.

Next, we present the results for the following cases: modes 4 and 8 with a perturbation of the external interface and mode 8 with a perturbation of the internal and external interfaces. For a given mode, we first define a mesh compatible with this mode, i.e., with enough angular sectors per wave length. Then, we compute the nonperturbed implosion and store the location of the vertex of the external face that corresponds to $\theta = 0$, namely $R(r_2, t)$. Then we perturb the mesh according to (6.2). An implosion simulation is done on this perturbed mesh. Then we store $R_p(r_2, t)$, the location of the same vertex. The amplification factor is defined by the ratio

$$a_{\text{num}}(t) = \frac{R^p(r_2, t) - R(r_2, t)}{a_0}.$$

In each of the perturbed simulations, the initial amplitude is set to $a_0 = 10^{-6}$ in order to stay in the linear phase. The final time of each simulation is $t = 0.99$. The time step follows (5.9).

6.6.1. Mode $n = 4$, $\xi_1 = 0$, $\xi_2 = 1$, $a_0 = 10^{-6}$. The computational domain is defined using the symmetries of mode 4, i.e., $(r, \theta) \in [r_1, r_2] \times [0, \frac{\pi}{2}]$ (see Figure 6.18). We use equiangular meshes defined by $(n_x, n_y) = (25, 44)$, $(n_x, n_y) = (50, 88)$, and $(n_x, n_y) = (100, 176)$.

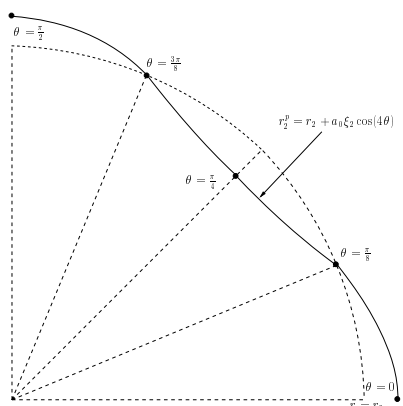
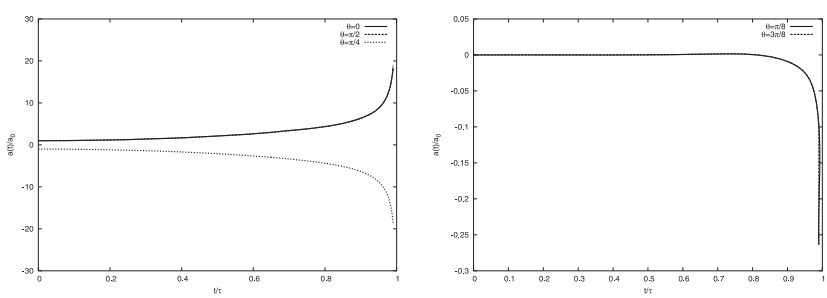


FIG. 6.18. *Mode $n = 4$, location of the vertices of interest.*



Points of maximal amplitude Points of vanishing amplitude

FIG. 6.19. *Mode $n = 4$, time evolution of the external perturbation for several vertices of interest.*

In Figure 6.19, we show the results for the coarsest mesh. The time evolution of the amplitude for the vertices defined by $\theta = 0,\ \pi/8,\ \pi/4,\ 3\pi/8,\ \pi/2$ (see Figure 6.18)

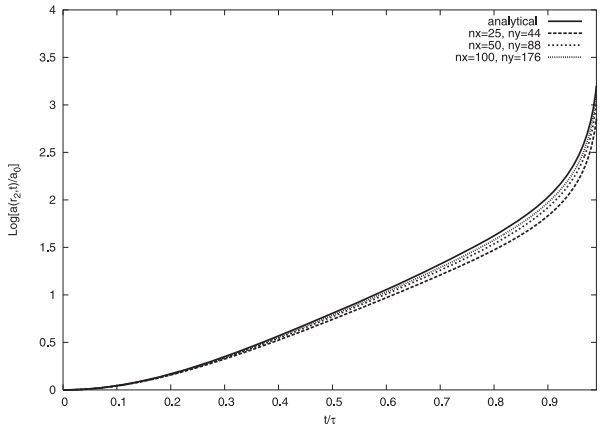
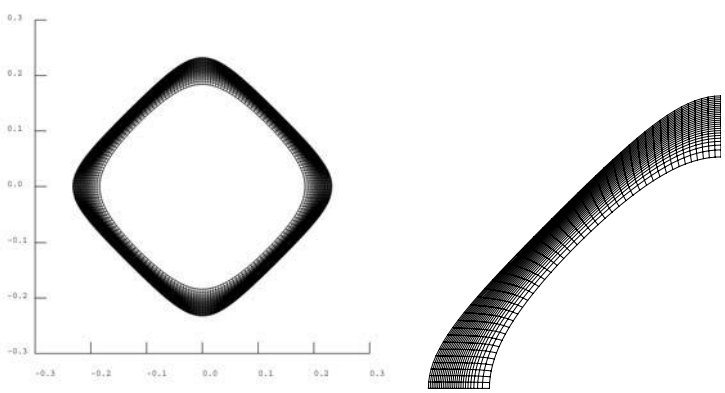


FIG. 6.20. *Mode $n = 4$, time evolution of the amplitude of the external perturbation.*

is stored. We see that the expected symmetries for the vertices $\theta = 0,\ \pi/4,\ \pi/2$ are perfectly respected. Moreover, the amplification factor for the vertices defined by $\theta = \pi/8,\ 3\pi/8$ is almost null, but we notice a very small amplification, on the order of 1%, when we are near the focusing. In Figure 6.20, we plot the time evolution of the perturbation obtained numerically for the three meshes and the analytical solution. The comparison is satisfactory since the numerical amplification is coherent with the analytical one. This comparison is even more satisfactory as the scheme is only first order in time and space.



Mesh at $t/\tau = 0.98$ Zoom

FIG. 6.21. *Mode $n = 4$, nonlinear phase $a_0 = 10^{-3}$.*

We present in Figure 6.21 the mesh that corresponds to a perturbed simulation near the focusing time. The initial mesh is defined by $(n_x, n_y) = (25, 176)$. The initial amplitude is set to $a_0 = 10^{-3}$ so that, at the final time, we are in the nonlinear regime.

6.6.2. Mode $n = 8$, $\xi_1 = 0$, $\xi_2 = 1$, $a_0 = 10^{-6}$. The computational domain is defined using the symmetries of the mode 8, i.e., $(r, \theta) \in [r_1, r_2] \times [0, \frac{\pi}{2}]$. We use equiangular meshes with $(n_x, n_y) = (25, 44)$, $(n_x, n_y) = (50, 88)$, and $(n_x, n_y) = (100, 176)$. Figure 6.22 shows a good agreement with the analytical solution for the time evolution

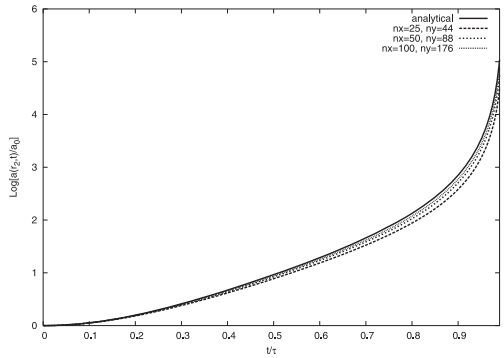
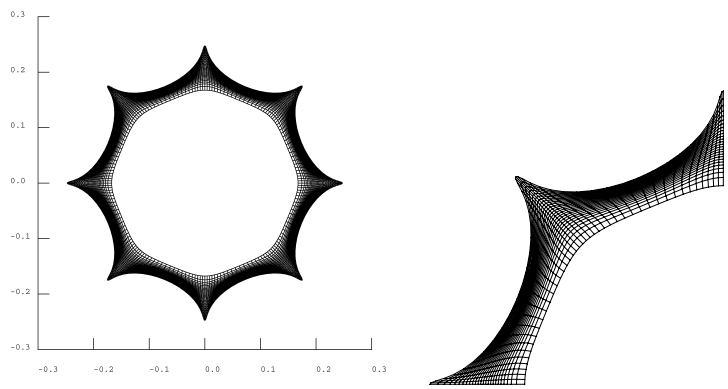


FIG. 6.22. *Mode $n = 8$, time evolution of the amplitude of the external perturbation.*

of the amplitude of the external perturbation. As for the $n = 4$ mode case, the scheme converges by lower values.



Mesh at $t/\tau = 0.98$ Zoom

FIG. 6.23. *Mode $n = 8$, nonlinear phase for $a_0 = 10^{-3}$.*

In Figure 6.23 we display the mesh obtained during the nonlinear phase for the flow perturbed on the full geometry. The initial mesh is defined by $(n_x, n_y) = (25, 176)$. The initial amplitude of the perturbation is $a_0 = 10^{-3}$.

6.6.3. Mode $n = 8$, $\xi_1 = 2.565293$, $\xi_2 = 1$, $a_0 = 10^{-6}$. In this case, we have introduced a nonzero initial perturbation on the internal interface. The level of ξ_1 is defined so that at the final time $t = 0.99$, the amplitude of the perturbation on the external face comes back to its initial value. This permits the coupling of the perturbations between the two interfaces. This coupling enables us to get a periodic growth of the perturbation on the internal face. In this case, we use the same meshes and the same domain as in the previous case.

The results are very satisfactory; see Figure 6.24. They show the ability of our scheme to reproduce the coupling of perturbations between the internal and external faces.

6.7. ICF-like test case. In this section, we present a simplified test case of a cylindrical ICF target. The target is made of two materials which have the same

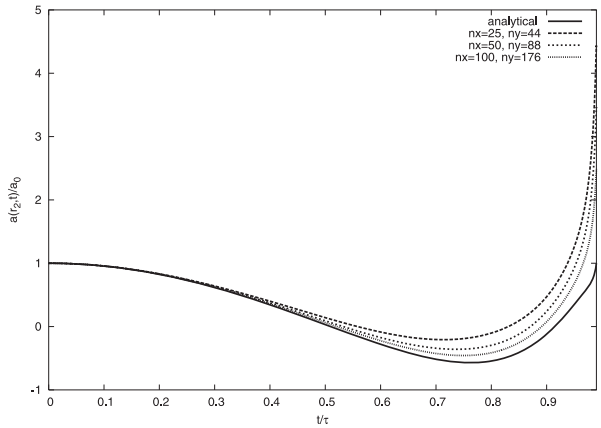


FIG. 6.24. *Coupled mode $n = 8$, growth of the perturbation on the internal interface.*

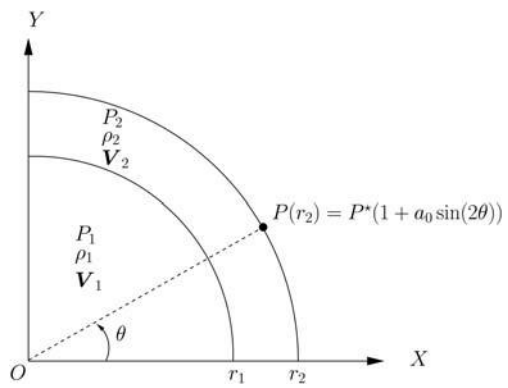


FIG. 6.25. *ICF-like target: initial conditions.*

perfect gas equation of state ($\gamma = 5/3$). The data are chosen to be representative of ICF situations. The interface between the gas and the shell is located at $r_1 = 0.1$, and the external radius of the target is $r_2 = 0.11$; see Figure 6.25. The initial state of the gas is given by $(\rho_1, P_1, \boldsymbol{V_1}) = (0.01, 2.5\ 10^9, 0)$, and the state inside the shell is $(\rho_2, P_2, \boldsymbol{V_2}) = (1, 2.5\ 10^{11}, 0)$. On the external surface of the shell, we apply a pressure law $P^\star = 10^{14}(1 + a_0 \sin(2\theta))$. We study this target in two configurations: the first is monodimensional with $a_0 = 0$, and the second is bidimensional with $a_0 = 0.2$. This second configuration may crudely represent the hydrodynamic part of an ICF flow which results from an inhomogeneous irradiation of the target.

For the monodimensional problem the mesh is made of 25 identical angular sectors on $[0, 90°]$ and 200 layers (100 uniform layers in the gas and 100 layers with a geometrical progression in the shell). A geometrical progression is applied so that we have a good representation of the contact discontinuity between the two materials (we have the same mass in the two cells around the interface). The time evolution of internal interface is displayed in Figure 6.26. This shows the two acceleration periods at $t = 1\ 10^{-9}$ and $t = 3\ 10^{-9}$ of the interface and at the end the deceleration of the interface until the stagnation time at $t = 5\ 10^{-9}$. In order to validate our results for the monodimensional configuration, we compare them to the results obtained with a

monodimensional code named CHIVAS. This code uses a von Neumann–Richtmyer-type scheme; see [16]. In Figure 6.26, we compare the trajectory of the internal interface with our scheme and with CHIVAS. The two curves can be almost superimposed, which demonstrates the quality of our scheme.

For the bidimensional case we use two types of meshes. One is a polar mesh (see Figure 6.27) with 40 angular sectors on $[0, 90°]$ and 100 layers (50 uniform layers in the gas and 50 layers with a geometrical progression in the shell). The other is a multiblock mesh in which we have a square in the central part of the target and nearly the same mesh as the polar grid for the rest of the domain (see Figure 6.27). Figure 6.28 represents the mesh at $t = 4.4\ 10^{-9}$ for both configurations. The nonuniformity of the pressure on the shell boundary leads to a bidimensional deformation of the internal interface.

Figure 6.29 illustrates one of the advantages of using a multiblock mesh. We can see that the time step in the case of the polar mesh reduces to a very small value ($\approx 1\ 10^{-14}$), whereas for the multiblock mesh we keep a reasonable time step ($\approx 1\ 10^{-13}$). The very small value of the time step for the polar mesh is due to the very small size of the triangular cells located at the center. Moreover, the multiblock mesh makes the computation more robust and enables us to deal with higher nonuniformity values for a_0.

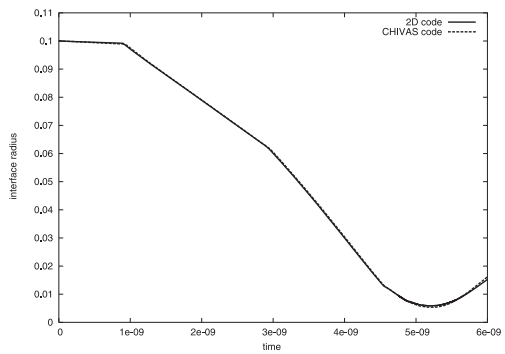
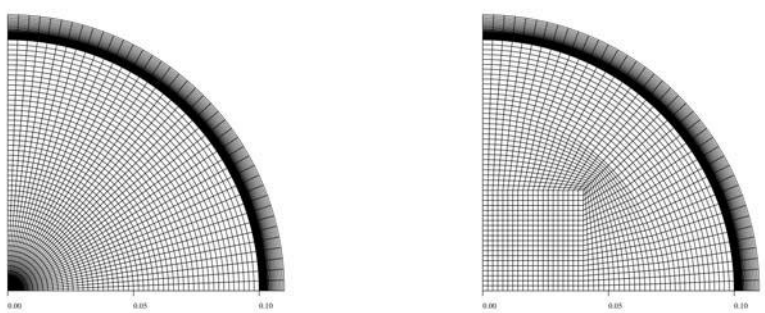


FIG. 6.26. *Monodimensional ICF-like test case: time evolution of the internal interface radius.*



Polar mesh Multiblock mesh

FIG. 6.27. *Initial mesh for bidimensional ICF-like test case.*

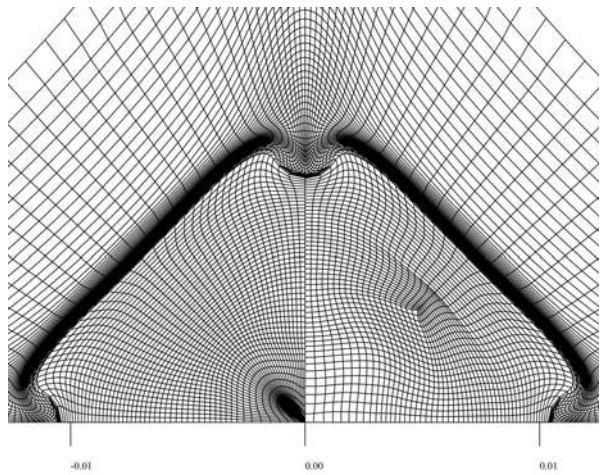


FIG. 6.28. *Result for two types of meshes at time* $t = 4.4\ 10^{-9}$.

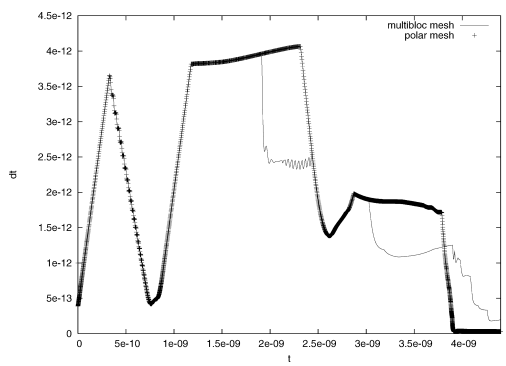


FIG. 6.29. *Time step during the simulations.*

7. Conclusion and perspectives. We have developed a new Lagrangian cell-centered conservative scheme. This scheme relies on a genuinely two-dimensional nodal solver. It can be interpreted as the two-dimensional generalization of the Godunov acoustic solver. The main new feature of the algorithm is the introduction of four pressures on each edge, two for each node on each side of the edge. This extra degree of freedom allows us to construct a nodal solver that fulfills two properties. First, the conservation of momentum and total energy is ensured. Second, a semidiscrete entropy inequality is provided. The face fluxes and the nodal velocities are all evaluated in a coherent manner. We have checked that, in the case of one-dimensional problems (with planar or cylindrical symmetry), our solver is exactly equivalent to the one-dimensional Godunov acoustic solver.

The scheme is only first order accurate; however it appears to be quite robust and versatile according to the numerical results obtained for the various test cases presented in this paper.

The next step is to develop the axisymmetric extension of our scheme. Then, in order to improve the accuracy of the scheme we will study its second order extension, probably with a MUSCL-type method. This new Lagrangian method looks promising

both as a stand-alone entity but also as a foundation for a more general method which may include ALE and/or adaptive mesh refinement techniques. The uniform cell centering of the solution variables provides a consistent basis for employing a wide range of well-proven remapping and adaptation schemes.

REFERENCES

[1] R. Abgrall, R. Loubère, and J. Ovadia, *A Lagrangian discontinuous Galerkin-type method on unstructured meshes to solve hydrodynamics problems*, Internat. J. Numer. Methods Fluids, 44 (2004), pp. 645–663.

[2] D. J. Benson, *Computational methods in Lagrangian and Eulerian hydrocodes*, Comput. Methods Appl. Mech. Engrg., 99 (1992), pp. 235–394.

[3] J. Breil, L. Hallo, P.-H. Maire, and M. Olazabal-Loumé, *Hydrodynamic instabilities in axisymmetric geometry: Self-similar models and numerical simulations*, Laser Part. Beams, 23 (2005), pp. 155–160.

[4] J. C. Campbell and M. J. Shashkov, *A tensor artificial viscosity using a mimetic finite difference algorithm*, J. Comput. Phys., 172 (2001), pp. 739–765.

[5] E. J. Caramana, M. J. Shashkov, and P. P. Whalen, *Formulations of artificial viscosity for multidimensional shock wave computations*, J. Comput. Phys., 144 (1998), pp. 70–97.

[6] E. J. Caramana, D. E. Burton, M. J. Shashov, and P. P. Whalen, *The construction of compatible hydrodynamics algorithms utilizing conservation of total energy*, J. Comput. Phys., 146 (1998), pp. 227–262.

[7] E. J. Caramana and M. J. Shashkov, *Elimination of artificial grid distortion and hourglass-type motions by means of Lagrangian subzonal masses and pressures*, J. Comput. Phys., 142 (1998), pp. 521–561.

[8] A. Chorin and J. Marsden, *A Mathematical Introduction to Fluid Mechanics*, Springer-Verlag, New York, 1993.

[9] B. Després and C. Mazeran, *Lagrangian gas dynamics in two dimensions and Lagrangian systems*, Arch. Ration. Mech. Anal., 178 (2005), pp. 327–372.

[10] J. K. Dukowicz and B. Meltz, *Vorticity errors in multidimensional Lagrangian codes*, J. Comput. Phys., 99 (1992), pp. 115–134.

[11] J. K. Dukowicz, M. C. Cline, and F. S. Addessio, *A general topology Godunov method*, J. Comput. Phys., 82 (1989), pp. 29–63

[12] S. K. Godunov, A. Zabrodine, M. Ivanov, A. Kraiko, and G. Prokopov, *Résolution numérique des problèmes multidimensionnels de la dynamique des gaz*, Mir, Moscow, 1979.

[13] S. J. Han and B. R. Suydam, *Hydrodynamic instabilities in an imploding cylindrical plasma shell*, Phys. Rev. A, 26 (1982), pp. 926–939.

[14] C. W. Hirt, A. Amsden, and J. L. Cook, *An arbitrary Lagrangian–Eulerian computing method for all flow speeds*, J. Comput. Phys., 14 (1974), pp. 227–253.

[15] W. H. Hui, P. Y. Li, and Z. W. Li, *A unified coordinate system for solving the two-dimensional Euler equations*, J. Comput. Phys., 153 (1999), pp. 596–637.

[16] S. Jacquemot and A. Decoster, *Z scaling of collisional NE-like X-ray lasers using exploding foils. Refraction effects*, Laser Part. Beams, 9 (1991), pp. 517–526.

[17] R. E. Kidder, *Laser-driven compression of hollow shells: Power requirements and stability limitations*, Nuclear Fusion, 1 (1976), pp. 3–14.

[18] L. Landau and E. Lifchitz, *Mécanique des fluides*, Mir, Moscow, 1989.

[19] J. D. Lindl, *Inertial Confinement Fusion*, Springer-Verlag, New York, 1998.

[20] R. Loubère, *Une méthode particulaire Lagrangienne de type Galerkin discontinu. Application à la mécanique des fluides et à l'interaction laser plasma*, Ph.D. thesis, Université Bordeaux 1, Talence, France, 2002.

[21] R. Loubère and M. J. Shashkov, *A subcell remapping method on staggered polygonal grids for arbitrary-Lagrangian-Eulerian methods*, J. Comput. Phys., 209 (2005), pp. 105–138.

[22] P.-H. Maire, J. Breil, L. Hallo, and M. Olazabal-Loumé, *Hydrodynamic instabilities in cylindrical geometry. Self-similar models and numerical simulations*, in Proceedings of the 31st EPS Conference on Plasma Physics, London, 2004.

[23] W. F. Noh, *Errors for calculations of strong shocks using artificial viscosity and an artificial heat flux*, J. Comput. Phys., 72 (1987), pp. 78–120.

[24] R. D. Richtmyer and K. W. Morton, *Difference Methods for Initial-Value Problems*, John Wiley, New York, 1967.

[25] Y. Saillard, *Hydrodynamique de l'implosion d'une cible FCI*, C. R. Acad. Sci. Paris Sér. IV, 1 (2000), pp. 705–718.

[26] G. A. Sod, *A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws*, J. Comput. Phys., 27 (1978), pp. 1–31.
[27] J. von Neumann and R. D. Richtmyer, *A method for the numerical calculations of hydrodynamical shocks*, J. Appl. Phys., 21 (1950), pp. 232–238.
[28] P. P. Whalen, *Algebraic limitations on two dimensional hydrodynamics simulations*, J. Comput. Phys., 124 (1996), pp. 46–54.
[29] M. L. Wilkins, *Calculation of elastic-plastic flow*, in Methods in Computational Physics, Vol. 3, Academic Press, New York, 1964, pp. 211–263.
[30] Y. B. Zel'dovich and Y. P. Raizer, *Physics of Shock Waves and High-Temperature Hydrodynamic Phenomena,* Vol. I, Academic Press, New York, 1966.

COPPER MOUNTAIN SPECIAL ISSUE ON ITERATIVE METHODS

The 9th Copper Mountain Conference on Iterative Methods was held April 2–7, 2006, in Copper Mountain, Colorado. The conference is organized by Front Range Scientific Computations, Inc., with co-organizers the Center for Applied Scientific Computing at Lawrence Livermore National Laboratory, and the University of Maryland. Traditionally, the [illegible] (SISC) hosts a special issue on iterative methods that is open to the public and is announced well in advance through both the SISC and the Copper Mountain Conference web sites.

I would like to thank the current members of the program committee for their service as guest editorial board. The committee members are
Xiao-Chuan Cai, University of Colorado
Iain Duff, RAL and CERFACS
Roland Freund, University of California, Davis
Kirk Jordan, IBM, Watson Research Center
Tim Kelley, North Carolina State University
David Keyes, Columbia University
Tom Manteuffel, University of Colorado
Steve McCormick, University of Colorado
David Silvester, University of Manchester
Ray Tuminaro, Sandia National Laboratory
Henk van der Vorst, University of Utrecht
Homer Walker, Worcester Polytechnic Institute
Carol Woodward, Lawrence Livermore National Laboratory
Irad Yavneh, Technion
The committee is chaired by Howard Elman (University of Maryland) and myself.

This issue contains 21 papers that largely reflect the topics presented at the conference, including papers that were part of the student papers competition. More specifically, we have papers covering the more traditional numerical linear algebra topic of preconditioning, including preconditioners for nonsymmetric problems in GMRES, radial basis function interpolation via preconditioned Krylov iterations, preconditioned eigensolvers, graph-based algorithms for preconditioning, and strategies for constructing block preconditioners for systems of PDEs, as well as some emerging new topics, such as preconditioners for stochastic elliptic PDEs and linear systems arising in interior point optimization methods. Most of the papers deal with some aspects of iterative methods for systems of algebraic equations; however, a paper on iterative optimization procedures with application to image registration also found its place in the issue. The overwhelming majority of the papers deal with various aspects of (mostly algebraic) multigrid (or (A)MG), ranging from theory of geometric MG applied to nontraditional formulation of Stokes problems to investigation of a number of algorithmic issues in AMG (such as the construction of the interpolation operator, coarsening strategies including for nonsymmetric problems), as well as the construction of specialized AMG algorithms for Google PageRank-like problems and for problems arising from higher-order spectral elements discretizations.

Many thanks are due to all authors for their valuable contributions. I would like also to thank the SIAM staff for their constant support and efforts to bring this issue to life. Finally, let me acknowledge the traditional support for the conference from

the US Department of Energy, the National Science Foundation, IBM, Los Alamos National Laboratory, Lawrence Livermore National Laboratory, and Sandia National Laboratory.

Panayot S. Vassilevski
Lawrence Livermore National Laboratory

SIAM J. SCI. COMPUT.
Vol. 29, No. 5, pp. 1825–1853
© 2007 Society for Industrial and Applied Mathematics

A GREEDY STRATEGY FOR COARSE-GRID SELECTION*

S. MACLACHLAN† AND YOUSEF SAAD†

Abstract. Efficient solution of the very large linear systems that arise in numerical modeling of real-world applications is often only possible through the use of multilevel techniques. While highly optimized algorithms may be developed using knowledge about the origins of the matrix problem to be considered, much recent interest has been in the development of purely algebraic approaches that may be applied in many situations, without problem-specific tuning. Here, we consider an algebraic approach to finding the fine/coarse partitions needed in multilevel approaches. The algorithm is motivated by recent theoretical analysis of the performance of two common multilevel algorithms, multilevel block factorization and algebraic multigrid. While no guarantee on the rate of coarsening is given, the splitting is shown to always yield an effective preconditioner in the two-level sense. Numerical performance of two-level and multilevel variants of this approach is demonstrated in combination with both algebraic multigrid and multilevel block factorizations, and the advantages of each of these two algorithmic settings are explored.

Key words. iterative methods, algebraic multigrid (AMG), algebraic preconditioners

AMS subject classifications. 65F10, 65N55, 68Q25

DOI. 10.1137/060654062

1. Introduction.

Much of modern scientific computing is driven by the need to solve increasingly large linear systems that arise from discretization of mathematical models of physical systems. The matrices of these linear systems are often not only large but also ill-conditioned. While classical methods may be efficient for smaller systems, the increasing demand for accuracy in numerical models requires a more efficient approach.

When full details of the driving application and its discretization are available, efficient multigrid solvers and multilevel preconditioners may be naturally defined. Coarse-level acceleration ideas were proposed as early as 1935 in the work of Southwell [33], while the potential efficiency of multilevel ideas was first demonstrated in the theoretical work of Fedorenko [16, 17] and Bakhvalov [2] in the 1960s. Geometric multigrid methods [10, 20], first used for computation in the 1970s, can be shown to be optimally efficient for simple problems on regular meshes. Hierarchical basis approaches based on detailed knowledge of the finite-element discretization also lead to effective preconditioners [3]. While techniques based on such complete knowledge are often the most efficient, it is not known how to construct optimal solution strategies for all problems. Further, a good discretization-based preconditioner for one problem may not work at all for a similar problem, without substantial modification.

Algebraic multigrid (AMG) methods, first proposed in [4] and later analyzed and tested in [29], offer performance comparable to geometric multigrid on many problems without requiring as much a priori knowledge. The goal of an AMG algorithm is to use algebraic means to create the multigrid hierarchy based solely on the linear system to be solved. While there are many variants of the original approach, in all cases this amounts to choosing a coarse set of points, and then defining intergrid

*Received by the editors March 10, 2006; accepted for publication (in revised form) August 14, 2006; published electronically September 28, 2007. This work was supported by NSF under grant ACI-0305120, and by the Minnesota Supercomputing Institute.
http://www.siam.org/journals/sisc/29-5/65406.html
†Department of Computer Science and Engineering, University of Minnesota, 200 Union Street S.E., Minneapolis, MN 55455 (maclach@cs.umn.edu, saad@cs.umn.edu).

transfer operators and a coarse-scale operator that provide a good correction to errors that are slow to be reduced by relaxation. AMG has been successfully applied to many problems but is known to be most effective for those that arise from standard discretizations of elliptic differential equations [12].

The algebraic recursive multilevel solver (ARMS) approach, introduced in [30], is a multilevel block factorization approach based on incomplete LU (ILU) factorizations. The multilevel block factorization approach relies on splitting the matrix into 2×2 block form, then approximately inverting one diagonal block, and then computing a Schur complement to form a preconditioner [1, Chap. 9]. In ARMS, the inversion is accomplished using a threshold-based incomplete LU factorization (ILUT) [31] approach to compute sparse LU factors, which can then be reused in forming an approximation to the needed Schur complement.

Both the AMG and ARMS approaches require a good splitting into fine and coarse grids in order to achieve the best possible performance. To be considered good, a splitting must give a coarse grid that is sufficiently smaller than the original mesh yet still allows an effective preconditioner to be constructed. In AMG, this means that the coarse grid is big enough that the coarse-grid correction can effectively address all errors that are prohibitively slow to be reduced by relaxation. Many heuristics have been proposed to do this, primarily based on independent set algorithms, as were originally used in [29]. For ARMS, the primary goal of the coarse-grid selection algorithm is to allow for an accurate sparse factorization of the fine-grid block, plus an accurate sparse approximation of the coarse-grid Schur complement.

Theoretical convergence analysis of ARMS and AMG give implicit suggestions of ways to choose the coarse grid, but there is no easy explicit connection between the selection of the coarse-grid points and the convergence of the overall scheme. The idea of compatible relaxation (CR) [6, 8, 15, 23] provides a relationship between the properties of the fine-scale block of the matrix and the overall performance of AMG. Similarly, the analysis in [27, 28] provides a relationship between the preconditioning of the fine-scale block within ARMS and the overall quality of the preconditioner. Brief reviews of the AMG and ARMS methodologies and their theoretical analyses are given in section 2.

Here, we use these theoretical results to motivate a coarse-grid selection criterion and algorithm. The optimal coarse-grid selection criterion is seen to lead to an NP-complete coarsening algorithm and, so, a greedy algorithm is proposed to approximately maximize the measure of a good splitting. The resulting algorithm is shown to always satisfy the conditions on the fine-scale block in both AMG and ARMS theory; however no guarantee of reduction in the coarse-grid size is possible. This greedy coarsening strategy and its theoretical analysis are presented in section 3.

Numerical results are given in section 4 that show the algorithm works well for many problems. Of particular interest is the comparison between the performance of AMG and ARMS. Here, it is necessary to distinguish between the efficiency and the robustness of an algorithm. Efficiency is easily measured in terms of error reduction versus computational cost. Robustness, on the other hand, is difficult to quantify precisely. Here, we consider the robustness of an algorithm to be indicated by similar performance on a wide variety of problems. Our numerical results demonstrate the efficiency of the AMG-based algorithm for finite-element discretization of elliptic PDEs. The ARMS-based algorithm is seen to be less efficient than AMG, when AMG works well, but more robust.

2. Background and motivation. The use of multilevel solution techniques has become ubiquitous in the numerical solution of PDEs, and has also begun to

play an important role in more general numerical linear algebraic contexts. While many of these techniques make use of more knowledge than is present in the linear system of equations to be solved (e.g., geometric multigrid [10], hierarchical basis transformations [3]), we concentrate here on purely algebraic approaches, such as the ARMS technique [30] and AMG methods [4, 29]. Such algebraic techniques presume a splitting of the overall set of degrees of freedom into two disjoint sets, typically called the fine- and coarse-level degrees of freedom that are used to partition the given matrix.

Consider the matrix equation, $A\mathbf{x} = \mathbf{b}$, and suppose that the degrees of freedom in $\mathbf{x}$ and $\mathbf{b}$ are partitioned into two disjoint sets, F and C. We consider the matrix, A, to be already reordered, so that

$$\begin{bmatrix} A_{ff} & -A_{fc} \\ -A_{cf} & A_{cc} \end{bmatrix} \begin{pmatrix} \mathbf{x}_f \\ \mathbf{x}_c \end{pmatrix} = \begin{pmatrix} \mathbf{b}_f \\ \mathbf{b}_c \end{pmatrix}. \tag{1}$$

2.1. Multilevel block factorization. Multilevel block factorization techniques [1, Chap. 9] (such as ARMS [30]), make use of this block form to factor A in terms of A_{ff}^{-1}. Writing

$$A = \begin{bmatrix} I & 0 \\ -A_{cf}A_{ff}^{-1} & I \end{bmatrix} \begin{bmatrix} A_{ff} & 0 \\ 0 & \hat{A}_{cc} \end{bmatrix} \begin{bmatrix} I & -A_{ff}^{-1}A_{fc} \\ 0 & I \end{bmatrix}, \tag{2}$$

where $\hat{A}_{cc} = A_{cc} - A_{cf}A_{ff}^{-1}A_{fc}$ denotes the Schur complement of A, we can solve for $\mathbf{x}_f$ and $\mathbf{x}_c$ as in Algorithm 1.

ALGORITHM 1 (block factorization solve).

1. $\mathbf{y}_f = A_{ff}^{-1}\mathbf{b}_f$
2. $\mathbf{y}_c = \mathbf{b}_c + A_{cf}\mathbf{y}_f$
3. $\hat{A}_{cc}\mathbf{x}_c = \mathbf{y}_c$
4. $\mathbf{x}_f = \mathbf{y}_f + A_{ff}^{-1}A_{fc}\mathbf{x}_c$

Such a solution technique is, of course, only useful if we can easily solve systems with A_{ff} and $\hat{A}_{cc}$. In practice, multilevel block factorizations can lead to good preconditioners by approximating these operators by ones that can be inverted in a more computationally efficient manner. The ARMS algorithm [30] utilizes an incomplete LU factorization to approximate A_{ff}^{-1}, both in Steps 1 and 4 of Algorithm 1, and in computing an approximate Schur complement, which is approximately inverted by a recursive application of the ARMS methodology.

The action of the two-level ARMS preconditioner on a residual, $\mathbf{r} = \left(\begin{smallmatrix} \mathbf{r}_f \\ \mathbf{r}_c \end{smallmatrix}\right)$, can be easily explained in terms of a modified block factorization, rewriting (2) as

$$A = \begin{bmatrix} L_{ff} & 0 \\ -A_{cf}U_{ff}^{-1} & I \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & \hat{A}_{cc} \end{bmatrix} \begin{bmatrix} U_{ff} & L_{ff}^{-1}A_{fc} \\ 0 & I \end{bmatrix},$$

where $A_{ff} = L_{ff}U_{ff}$. Taking ILU factors, $L \approx L_{ff}$ and $U \approx U_{ff}$, and letting S^{-1} approximate the action of the inverse of $\hat{A}_{cc}$, we can write the ARMS preconditioner as in Algorithm 2.

ALGORITHM 2 (action of ARMS preconditioner on residual, $\mathbf{r}$).

1. $\mathbf{y}_f = L^{-1}\mathbf{r}_f$
2. $\mathbf{y}_c = \mathbf{r}_c + A_{cf}U^{-1}\mathbf{y}_f$
3. $\mathbf{x}_c = S^{-1}\mathbf{y}_c$
4. $\mathbf{z}_f = \mathbf{y}_f + L^{-1}A_{fc}\mathbf{x}_c$
5. $\mathbf{x}_f = U^{-1}\mathbf{z}_f$

Note that Algorithms 1 and 2 are, in fact, equivalent, if $A_{ff} = LU$ (i.e., if $L = L_{ff}$ and $U = U_{ff}$) and $S^{-1} = \hat{A}_{cc}^{-1}$.

It is apparent that the success of the ARMS preconditioner depends on how well the action of A_{ff}^{-1} can be approximated by the ILU solve. This, in turn, depends directly on the properties of the A_{ff} block of A. Notay relates the condition number of the preconditioned system, $B^{-\frac{1}{2}}AB^{-\frac{1}{2}}$, where B represents the action of the (approximate) multilevel block factorization, as in Algorithms 1 and 2, to the spectral equivalence of A_{ff} and $\hat{A}_{cc}$ and their approximations [27, 28]. Paraphrasing Theorem 9 of [28], we have the following result.

THEOREM 1. [illegible] A [illegible] (1) [illegible] B [illegible]

$$B = \begin{bmatrix} I & 0 \\ -A_{cf}D_{ff}^{-1} & I \end{bmatrix} \begin{bmatrix} D_{ff} & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} I & -D_{ff}^{-1}A_{fc} \\ 0 & I \end{bmatrix},$$

[illegible] D_{ff} [illegible] S [illegible] $\begin{bmatrix} D_{ff} & -A_{fc} \\ -A_{cf} & A_{cc} \end{bmatrix}$ [illegible] $\lambda_{min}(D_{ff}^{-1}A_{ff}) \geq 1$ [illegible]

$$\frac{\lambda_M}{\lambda_m} \leq \kappa\left(B^{-\frac{1}{2}}AB^{-\frac{1}{2}}\right) \leq \left(1 + \sqrt{1 - \lambda_M^{-1}}\right)^2 \frac{\lambda_M^2 \nu_M}{\min(\lambda_m, \nu_m)}, \tag{3}$$

[illegible]

$$\lambda_m = \lambda_{min}(D_{ff}^{-1}A_{ff}),\ \lambda_M = \lambda_{max}(D_{ff}^{-1}A_{ff}),$$
$$\nu_m = \lambda_{min}(S^{-1}\hat{A}_{cc}),\ \nu_M = \lambda_{max}(S^{-1}\hat{A}_{cc}),$$

[illegible] $D_{ff}^{-1}A_{ff}$ [illegible] $S^{-1}\hat{A}_{cc}$

While [27, 28] provide tighter bounds than those in inequality (3), we emphasize the role that good spectral equivalence to the A_{ff} block plays in these bounds: The overall preconditioner can be no better than the equivalence between D_{ff} and the A_{ff} block, and good equivalence implies the existence of a well-conditioned block factorization preconditioner for the overall system (using $S = \hat{A}_{cc}$). Thus, constructing the partition such that we know a D_{ff} with good equivalence to the A_{ff} block is an attractive approach to realizing an efficient preconditioner for A. A much harder question is achieving good spectral equivalence between the true Schur complement, $\hat{A}_{cc}$, and its approximation, S, especially as $\hat{A}_{cc}$ is, in practice, not explicitly available within the computation.

2.2. Algebraic multigrid. Originally proposed in [4] and further explained in [29], the Algebraic Multigrid (AMG) methodology has, in fact, grown to encompass a number of algorithms focused on the theme of complementing a given relaxation procedure by an algebraically determined correction from a coarser subspace. This collection of algorithms can be roughly parameterized by the choices of the multigrid components: relaxation, intergrid transfer operators, the (nodal) coarse grid, and the coarse-grid operator. Once these choices have been made, the AMG V-cycle can be described as a preconditioner or, more commonly, through its error propagation operator,

$$(I - B_{\text{AMG}}^{-1}A) = (I - M^{-T}A)^{l_2}(I - PA_C^{-1}RA)(I - M^{-1}A)^{l_1},$$

where $I - M^{-1}A$ is the error propagation operator of the chosen relaxation scheme (e.g., Jacobi or Gauss–Seidel), l_1 and l_2 are parameters giving the number of pre- and postrelaxations to be used, R is the restriction operator transferring residuals from the fine grid to the coarse grid, A_C is the coarse-grid operator, and P is the interpolation operator, transferring corrections from the coarse grid to the fine grid.

Choices of the multigrid components can be quite varied. Typical relaxation schemes are the Jacobi and Gauss–Seidel iterations, but other schemes, including symmetric Gauss–Seidel and ILU factorizations, are also possible. Choice of the coarse grid may be done based on the classical AMG definitions of strength of connection [29], using independent set algorithms (possibly modified for parallel processing environments) [14, 21], based on modified strength measures [7], or using CR [6, 8, 15, 23] (as discussed in more detail below). Once the coarse grid has been chosen, the dimensions of the intergrid transfer operators, R and P, are set, and their nonzero structure and values can be determined. Once R and P are chosen, a coarse-grid operator, A_C, may be chosen to solve for the coarse-grid correction.

If A is assumed to be symmetric and positive definite, a variational problem may be defined to simplify the problem of choosing R, P, and A_C. Note the part of the multigrid error propagation operator that is associated with the coarse-grid correction step, $I - PA_C^{-1}RA$. Whatever coarse-grid correction is calculated, error is reduced only within the range of P. Thus, a reasonable goal is to choose the coarse-grid correction to minimize the error after correction, over all corrections in the range of P—that is, to choose $\mathbf{w}_c$, such that $\|\mathbf{e} - P\mathbf{w}_c\|_A = \min$, where $\mathbf{e}$ is the error before the coarse-grid correction step, and the norm, defined as $\|\mathbf{v}\|_A^2 = \langle A\mathbf{v}, \mathbf{v}\rangle$, is chosen so that the minimization is achievable. Choosing this variational approach requires the coarse-grid correction, $\mathbf{w}_c$, to satisfy the equation $P^TAP\mathbf{w}_c = P^TA\mathbf{e}$. Noting that $A\mathbf{e} = \mathbf{r}$, the residual before coarse-grid correction, we see that this prescribes the choices of $R = P^T$ and $A_C = P^TAP$ for any given P. Note that choosing P and exactly computing $A_C = P^TAP$ is similar to Tismenetsky's strategy for computing approximate Schur complements [35], although the usual focus in multigrid is on appropriately complementing relaxation.

Thus, the problem of defining R, P, and A_C is reduced to one of choosing P such that the range of P adequately complements the space over which relaxation is effective. The classical AMG algorithm [29] does this by collapsing certain off-diagonal connections based on the assumption that the near-null space of A is accurately represented by the constant vector. If the nature of errors that are slow to be reduced by relaxation is not known, the adaptive AMG method [9] may be used to expose prototypes of such errors to be used in the definition of interpolation. Energy-minimization principles may also be used to define the multigrid interpolation operators [36, 37].

Even in the variational setting, the overall performance of multigrid depends on the complementarity obtained between the relaxation procedure and the coarse-grid correction. In particular, the key is choosing a coarse grid such that a good definition of interpolation is possible. The principle of compatible relaxation (CR), first introduced by Brandt [6], is based on the idea that the ability to define a good interpolation operator is tied to the relaxation it must complement. In [6], Brandt defines CR as "a modified relaxation scheme which keeps the coarse-level variables invariant" and states that "a general measure for the quality of the set of coarse variables is the convergence rate of the compatible relaxation." Defined in this manner, several possible versions of CR are possible [23], the most common of which is relaxation only on the unknowns associated with the fine grid (so-called F-relaxation [34]).

Falgout and Vassilevski [15] analyze the performance of AMG methods for symmetric and positive-definite matrices, A, based on a measure that accounts for the role of relaxation in complementing the variational coarse-grid correction process. For a relaxation scheme with convergent error propagation operator $I - M^{-1}A$, define the measure

$$\mu(P, \mathbf{e}) = \frac{\langle M(M + M^T - A)^{-1} M^T (\mathbf{e} - P\mathbf{e}_c), (\mathbf{e} - P\mathbf{e}_c)\rangle}{\langle A\mathbf{e}, \mathbf{e}\rangle},$$

where $M + M^T - A$ is invertible because $\|I - M^{-1}A\|_A < 1$. This measure takes into account the performance of relaxation in measuring the effectiveness of a given interpolation operator, P, beyond what is normally considered in the standard eigenvector-approximation criterion for AMG [5, 26]. If, for a given choice of P, $\mu(P, \mathbf{e}) < k$ for all $\mathbf{e} \neq \mathbf{0}$, then it can be shown that $\|I - B_{\mathrm{AMG}}^{-1}A\|_A < 1 - \frac{1}{k}$ [15, Theorem 2.2] for $l_1 = l_2 = 1$. Thus, $\mu(P, \mathbf{e})$ can be seen as a measure of the performance of the resulting AMG scheme for a chosen relaxation, coarse-grid, and interpolation operator.

The idea of CR may be used to bound the optimal value,

$$k^\star = \min_P \max_{\mathbf{e}\neq\mathbf{0}} \mu(P, \mathbf{e}),$$

of k over all possible interpolation operators, P, from the same coarse set. Paraphrasing [15, Theorem 5.1], we have the following result.

THEOREM 2. A $I - M^{-1}A$ ($\|I - M^{-1}A\|_A < 1$) $S_M = \frac{1}{2}(M + M^T)$ M $\omega = \lambda_{max}(S_M^{-1}A)$ Δ M S_M $\mathbf{v}$ $\mathbf{w}$

$$\langle M\mathbf{v}, \mathbf{w}\rangle \leq \Delta \langle S_M\mathbf{v}, \mathbf{v}\rangle^{\frac{1}{2}} \langle S_M\mathbf{w}, \mathbf{w}\rangle^{\frac{1}{2}}.$$

$\rho_{ff} = \|I - M_{ff}^{-1}A_{ff}\|_{A_{ff}}$ M_{ff} M (1)

$$k^\star \leq \frac{\Delta^2}{2 - \omega} \frac{1}{1 - \rho_{ff}},$$

$0 < \omega < 2$

Here, the bound on $k^\star$ gives a bound on the best possible measure of a two-level AMG algorithm with a given relaxation (prescribed by M) and partition into fine and coarse grids. For symmetric smoothers ($M = M^T$), $\Delta = 1$, and this inequality gives an upper bound on the best possible convergence factor for an AMG method with a particular coarse grid, which depends on the relaxation method chosen and the performance of CR. In particular, if CR is quick to converge (and ω is bounded away from 2, meaning that the slowest to converge modes of the symmetric relaxation, $I - S_M^{-1}A$, are the lowest-energy modes of $S_M^{-1}A$), then there exists an interpolation operator, P, that gives good multigrid performance. In general, choosing such a P that also results in an easy-to-invert coarse-grid operator, P^TAP, is difficult, as we seek to retain significant sparsity in P.

The reduction-based AMG algorithm (AMGr) presumes that a partition of the degrees of freedom has been chosen so that CR is fast to converge [24]. In particular,

AMGr is based on the assumption of a close spectral equivalence between A_{ff} and a fine-scale relaxation operator, D_{ff}, as in [24, Theorem 1].

THEOREM 3. [illegible] A [illegible] (1)

[illegible]

$$\langle D_{ff}\mathbf{x}_f, \mathbf{x}_f\rangle \leq \langle A_{ff}\mathbf{x}_f, \mathbf{x}_f\rangle \leq (1+\epsilon)\langle D_{ff}\mathbf{x}_f, \mathbf{x}_f\rangle$$

[illegible] $\mathbf{x}_f$ [illegible] $\left[\begin{smallmatrix} D_{ff} & -A_{fc} \\ -A_{cf} & A_{cc} \end{smallmatrix}\right]$ [illegible]

$$(I - M^{-1}A) = \left(I - \frac{2}{2+\epsilon}\begin{bmatrix} D_{ff}^{-1} & 0 \\ 0 & 0 \end{bmatrix} A\right),$$

[illegible]

$$P = \begin{bmatrix} D_{ff}^{-1}A_{fc} \\ I \end{bmatrix}.$$

[illegible] l [illegible] l [illegible]

$$MG = (I - M^{-1}A)^l(I - P(P^TAP)^{-1}P^TA)(I - M^{-1}A)^l,$$

[illegible]

$$\|MG\|_A \leq \frac{\epsilon}{1+\epsilon}\left(1 + \left(\frac{\epsilon^{2l-1}}{(2+\epsilon)^{2l}}\right)\right) < 1.$$

Now, under the additional assumptions of Theorem 3 (over those of Theorem 2), we can not only say that there exists a P that gives good multigrid convergence, but also give the form of one such P. Notice the similarities between the assumptions for Theorems 1 and 3; both require that $\lambda_{\min}(D_{ff}^{-1}A_{ff}) \geq 1$, and that the matrix, $\left[\begin{smallmatrix} D_{ff} & -A_{fc} \\ -A_{cf} & A_{cc} \end{smallmatrix}\right]$, is semidefinite. The explicit choice of a coarse-grid operator in the AMGr algorithm gives a bound on the multigrid convergence factor dependent solely on the maximum eigenvalue of $D_{ff}^{-1}A_{ff}$, which is also the dominant factor in the bound of Theorem 1.

Thus, for both multilevel block factorizations and (nodal) AMG, tight spectral equivalence between the fine-scale preconditioner or relaxation operator, D_{ff}, and A_{ff} is needed to guarantee good performance of the overall solver. Constructing the fine-grid set so that this is always possible is the subject of the remainder of this paper.

3. The greedy coarsening algorithm. In [32], the strategy of reordering the columns and rows of a nonsymmetric matrix to ensure diagonal dominance of the A_{ff} block was introduced. There, the partitioning was accomplished by ordering the columns and rows of the matrix in order of decreasing dominance, and choosing, as A_{ff}, a subblock of A with sufficiently dominant rows. The algorithm defined here is similar in style, in that it chooses rows for A_{ff} in a greedy manner, but it also takes into account the added dominance within the A_{ff} block that occurs when points are added to the coarse set.

In order to make decisions on dominance to select the A_{ff} block, we must have some sort of measure of the dominance that is sought. We consider choosing D_{ff}

as a simple, diagonal preconditioner or relaxation method, so that we seek classical rowwise diagonal dominance of A_{ff}. Writing the partition of the degrees of freedom into fine and coarse sets as $F \cup C$, we say that row i of A_{ff} has θ dominance if

$$|a_{ii}| \geq \theta \sum_{j \in F} |a_{ij}|.$$

Note that, by this definition, classical diagonal dominance is equivalent to $\theta = \frac{1}{2}$ dominance, as the diagonal, a_{ii}, is included in the sum on the right side. Then, the diagonal dominance associated with row i is given by

$$\theta_i = \frac{|a_{ii}|}{\sum_{j \in F} |a_{ij}|}.$$

Note that the measure, θ_i, depends on the selection of F- and C-points.

Ideally, we would like to choose F as the largest set of degrees of freedom such that for every $i \in F$, $\theta_i \geq \theta$. That is, we would like F to be the largest subset possible such that A_{ff} is θ dominant. This would guarantee good equivalence of a diagonal matrix, D_{ff}, to A_{ff}, as required in the theorems of section 2, but also yield as small a coarse-scale problem as possible. Formally, we would seek to maximize $|F|$ (where $|F|$ denotes the size of the set, F) under the constraint that for every $i \in F$, $\frac{|a_{ii}|}{\sum_{j \in F} |a_{ij}|} \geq \theta$ for a chosen θ. Defining the binary variables

$$f_i = \begin{cases} 1 & \text{if } i \in F, \\ 0 & \text{if } i \in C, \end{cases}$$

the problem can be posed in optimization form:

$$\begin{aligned} &\text{maximize } \sum_{i=1}^{n} f_i, \\ &\text{subject to } f_i \in \{0,1\} \ \forall i, \\ &\qquad |a_{ii}| \geq \theta \sum_{j \in F} |a_{ij}| \text{ if } f_i \neq 0. \end{aligned}$$

Note that the sum over $j \in F$ can be converted into a sum over all j, by weighting the terms by f_j, $\sum_{j \in F} |a_{ij}| = \sum_j |a_{ij}| f_j$. Then, as this is a constrained optimization problem, we convert it to an integer programming problem by adding the term $\theta \|A\|_\infty (1 - f_i)$ to the left side of the inequality constraint,

$$\begin{aligned} &\text{maximize } \sum_{i=1}^{n} f_i, \\ &\text{subject to } f_i \in \{0,1\} \ \forall i, \\ &\qquad \theta \|A\|_\infty (1 - f_i) + |a_{ii}| \geq \theta \sum_j |a_{ij}| f_j. \end{aligned}$$

Note that the modified problem is equivalent to the original, as if $i \in F$, $1 - f_i = 0$, so the original constraint is imposed for all $i \in F$, but if $i \notin F$, $1 - f_i = 1$, and the constraint is automatically satisfied, as $\sum_{j \in F} |a_{ij}| \leq \|A\|_\infty$. Thus, the problem of finding the largest set, F, with the desired dominance is a 0-1 integer programming

problem, known to be NP-complete [38]. As such, we cannot expect to easily find an algorithm that is linear in the number of nonzero entries in A that finds the largest possible set F that has the desired level of diagonal dominance. Instead, we introduce here a greedy algorithm that partitions the unknowns into F and C by iteratively building these sets, adding any sufficiently diagonally dominant points that arise to F, and moving the least diagonally dominant points into C, one at a time.

To accomplish the partitioning, we introduce a third set of points, denoted by the set, U, as those points whose partitioning is, as yet, undecided in the process. Initially, then, U includes all of the degrees of freedom, and sets F and C are both empty. We introduce the dynamic measures,

$$\hat{\theta}_i = \frac{|a_{ii}|}{\sum_{j \in F \cup U} |a_{ij}|},$$

as measures of the diagonal dominance of row i amongst those rows that either are already designated as F-points, or could potentially be so designated. If, at any point, row i is determined to be sufficiently diagonally dominant, that is, $\hat{\theta}_i \geq \theta$ for some preselected threshold θ, then i is automatically added to the set of F-points (as it may only become more dominant as points are moved from U to C or F). If no points are attractive as F-points, then a point that is not attractive as a potential F-point is chosen to be moved to C, in order to make the remaining points in U more diagonally dominant. Thus, the partitioning algorithm, given in Algorithm 3, is a greedy algorithm that, at each step, makes a choice to increase the diagonal dominance of those points in U. In this way, the points in U can be viewed as candidate F-points that are either confirmed or discarded based on the dynamic values of $\hat{\theta}_i$. A common notation, which we use here, is to call $Adj(j)$ the set of nearest neighbors of j, i.e., the set $\{k : a_{jk} \neq 0\}$.

ALGORITHM 3 (greedy partitioning algorithm).

0. $U = \{1, 2, \ldots, n\}; \quad F = \emptyset$. Compute $\hat{\theta}_i, \ i = 1, \ldots, n$
1. For $i = 1, \ldots, n$, if $\hat{\theta}_i \geq \theta$, $\begin{cases} F := F \cup \{i\}, \\ U := U \setminus \{i\} \end{cases}$
2. While $U \neq \emptyset$
   (a) Let $j = \operatorname{argmin}_{i \in U} \{\hat{\theta}_i\}$
   (b) Move j from U to C
   (c) For $i \in U \cap Adj(j)$
   - Recompute $\hat{\theta}_i = \frac{|a_{ii}|}{\sum_{k \in F \cup U} |a_{ik}|}$
   - If $\hat{\theta}_i \geq \theta$, $\begin{cases} F := F \cup \{i\}, \\ U := U \setminus \{i\} \end{cases}$

As the only way for a point to be reclassified from U to F is if $\hat{\theta}_i \geq \theta$, and $\hat{\theta}_i$ is a nondecreasing function (since points are only removed from $F \cup U$), it is immediately apparent that each row within the A_{ff} block created by Algorithm 3 has (at least) θ dominance. This, in turn, suggests the existence of a diagonal preconditioner or relaxation matrix, D_{ff}, that achieves the tight equivalence bounds suggested by Theorems 1, 2, and 3. We now verify this for the case that A is symmetric and positive definite.

THEOREM 4. Let A be symmetric and positive definite, and let $\theta \in (\frac{1}{2}, 1]$ be given. Then

$$\theta \leq \theta_i = \frac{a_{ii}}{\sum_{j \in F} |a_{ij}|} \quad \forall i \in F,$$

$D_{ff} \equiv \mathrm{diag}(A_{ff})$

$$\lambda_m = \lambda_{min}(D_{ff}^{-1}A_{ff}) \geq 2 - \frac{1}{\theta},$$
$$\lambda_M = \lambda_{max}(D_{ff}^{-1}A_{ff}) \leq \frac{1}{\theta}\ .$$

. Let λ be an eigenvalue of $D_{ff}^{-1}A_{ff}$. Clearly, λ is real because $D_{ff}^{-1}A_{ff}$ is similar to the symmetric matrix $D_{ff}^{-\frac{1}{2}}A_{ff}D_{ff}^{-\frac{1}{2}}$. By Gerschgorin's theorem, there is a certain index i such that λ satisfies

$$|\lambda - 1| \leq \sum_{j\in F,\ j\neq i} \left|\frac{a_{ij}}{a_{ii}}\right| = \frac{1}{\theta_i} - 1 \leq \frac{1}{\theta} - 1$$

from which the result follows immediately. □

The results in Theorems 1 and 3 require that the smallest eigenvalue of $D_{ff}^{-1}A_{ff}$ be at least one. By scaling the matrix D_{ff} with the scalar $2 - 1/\theta$ the smallest eigenvalue will be scaled to 1 and the largest to $1/(2\theta - 1)$.

COROLLARY 1. 4
$D_{ff} \equiv (2 - \frac{1}{\theta})\,\mathrm{diag}(A_{ff})$

$$\lambda_m = \lambda_{min}(D_{ff}^{-1}A_{ff}) \geq 1, \qquad \lambda_M = \lambda_{max}(D_{ff}^{-1}A_{ff}) \leq \frac{1}{2\theta - 1}.$$

A similar result may be obtained in the same manner as Theorem 4 using a rowwise scaling of D_{ff} based on the level of dominance in each row.

COROLLARY 2. 4
$(D_{ff})_{ii} = \left(2 - \frac{1}{\theta_i}\right)a_{ii}$ $i \in F$

$$\lambda_m = \lambda_{min}(D_{ff}^{-1}A_{ff}) \geq 1, \qquad \lambda_M = \lambda_{max}(D_{ff}^{-1}A_{ff}) \leq \frac{1}{2\theta - 1}.$$

Recall that the the smallest and largest eigenvalues of $D_{ff}^{-1}A_{ff}$ are the minimum and maximum (resp.) of the Rayleigh quotients $\frac{\mathbf{x}_f^T A_{ff}\mathbf{x}_f}{\mathbf{x}_f^T D_{ff}\mathbf{x}_f}$, so these corollaries are equivalent to saying that, for all $\mathbf{x}_f$,

$$\mathbf{x}_f^T D_{ff}\mathbf{x}_f \leq \mathbf{x}_f^T A_{ff}\mathbf{x}_f \leq \frac{1}{2\theta - 1}\mathbf{x}_f^T D_{ff}\mathbf{x}_f.$$

This, in turn, implies that the parameter ϵ from Theorem 3 may be chosen as $\frac{2-2\theta}{2\theta-1}$. Knowing the extremal Rayleigh quotients also allows definition of an appropriately weighted relaxation scheme, $I - \sigma D_{ff}^{-1}A_{ff}$, with minimal spectral radius for use in the bound in Theorem 2.

Theorems 1 and 3 both require the additional assumption that $\left[\begin{smallmatrix} D_{ff} & -A_{fc} \\ -A_{cf} & A_{cc} \end{smallmatrix}\right]$ be positive semidefinite. In AMGr, this requirement is important as the coarse-level matrix depends on D_{ff} through the variational formulation. For block factorization preconditioners, this condition arises indirectly, in the bounding of the Cauchy–Bunyakowski–Schwarz constant associated with the factorization using D_{ff}. In either case, it is difficult to guarantee this condition for general classes of matrices, although it is easy to construct a matrix, A, for which it holds: simply choose $A = \left[\begin{smallmatrix} D_{ff} & -A_{fc} \\ -A_{cf} & A_{cc} \end{smallmatrix}\right] + \left[\begin{smallmatrix} E_{ff} & 0 \\ 0 & 0 \end{smallmatrix}\right]$ for any semidefinite matrices $\left[\begin{smallmatrix} D_{ff} & -A_{fc} \\ -A_{cf} & A_{cc} \end{smallmatrix}\right]$ and $\left[\begin{smallmatrix} E_{ff} & 0 \\ 0 & 0 \end{smallmatrix}\right]$.

For a given matrix, A, the existence of the decomposition can be proven in the case of diagonal dominance if D_{ff} is chosen as in Corollary 2, as in the following result.

THEOREM 5. A (1)
A D_{ff}
2 $(D_{ff})_{ii} = \left(2 - \frac{1}{\theta_i}\right) a_{ii}$ $\theta_i = \frac{a_{ii}}{\sum_{j \in F} |a_{ij}|}$ $\begin{bmatrix} D_{ff} & -A_{fc} \\ -A_{cf} & A_{cc} \end{bmatrix}$

. As A is diagonally dominant, $\sum_j |a_{ij}| \leq 2a_{ii}$ for every $i \in F$. Splitting the degrees of freedom into $F \cup C$, we then have

$$\begin{aligned} \sum_{j \in C} |a_{ij}| &\leq 2a_{ii} - \sum_{j \in F} |a_{ij}| \\ &= \left(2 - \frac{1}{\theta_i}\right) a_{ii} = (D_{ff})_{ii}. \end{aligned}$$

Thus, all rows in $\begin{bmatrix} D_{ff} & -A_{fc} \\ -A_{cf} & A_{cc} \end{bmatrix}$ associated with an F-point are diagonally dominant. All rows associated with a C-point are also diagonally dominant, as they already were in A. So, $\begin{bmatrix} D_{ff} & -A_{fc} \\ -A_{cf} & A_{cc} \end{bmatrix}$ is diagonally dominant. But then, by Gerschgorin's theorem, it must be positive semidefinite. ∎

1 (relationship to CR). Brandt defines compatible relaxation (CR) as "a modified relaxation scheme which keeps the coarse-level variables invariant" and states that "a general measure for the quality of the set of coarse variables is the convergence rate of the compatible relaxation" [6]. Here, we do not explicitly perform the relaxation in CR, but we achieve a similar result. The construction of the coarse/fine partition, however, is such that the existence of a quickly converging fine-level Jacobi-like relaxation is guaranteed by Theorem 4 and its corollaries. As a result, the partitioning algorithm proposed here may be considered as a new approach to CR, where a property of quickly converging CR is identified (in this case, diagonal dominance) and used to construct a partition for which CR is guaranteed to converge quickly, without actually needing to perform the iteration.

4. Numerical results. To test the approach developed in section 3, we consider a variety of problems and explore some of the different options inherent in the development of multilevel algorithms based on this approach to coarse-grid selection. The results in Theorems 1 and 3 prescribe specific two-level algorithms with bounded convergence properties based only on the fine-scale equivalence of D_{ff} and A_{ff}. Theorems 1 and 2 may also be read as giving indicators of the best possible performance of a two-level algorithm based on the partitioning prescribed. A need for more general (and more expensive) treatment of the interaction between fine and coarse scales is demonstrated and addressed by using the full power of the ARMS and AMG approaches.

4.1. AMG. The reduction-based AMG algorithm discussed in Theorem 3 prescribes a simple two-level scheme that is useful when the coarse-grid selection gives a fine-scale block that is easily relaxed. We first test the diagonal-dominance-based coarse-grid selection scheme in this setting, for PDE-based problems where AMG in general (and AMGr, in particular) is expected to perform reasonably well.

The convergence result of Theorem 3 rests on the requirement that $\begin{bmatrix} D_{ff} & -A_{fc} \\ -A_{fc}^T & A_{cc} \end{bmatrix}$ be positive semidefinite, although this condition is not explicitly enforced in the coarsening algorithm. While established only as a sufficient condition, fast convergence of the AMGr algorithm is typically not found when it is violated. For this reason, we

begin by testing problems where we know the needed definiteness occurs, when A itself is diagonally dominant (as established in Theorem 5). For these tests, we consider the matrices that arise from the bilinear finite-element discretizations of the elliptic operator, $-\nabla \cdot K(x,y)\nabla p(x,y)$, on the unit square, subject to Dirichlet boundary conditions.

There are several relevant measures of the performance of the coarsening algorithm itself and of AMGr coupled with it. We record the required setup time for the algorithm, including computing the coarse grid, partitioning the fine-scale operator, computing the coarse-grid operator, and computing any necessary parameters of the method. For AMGr, we choose a random initial guess and iterate directly on the homogeneous problem, $A\mathbf{x} = \mathbf{0}$. The number of iterations needed to reduce the ℓ^2-norm of the residual by six orders of magnitude and the time taken for these iterations are recorded. These two-level results were computed using MATLAB and, thus, the times presented are intended primarily for comparison purposes; the multilevel results that follow are computed using research C codes compiled with appropriate optimization flags. As always, the reported run times are only accurate to a few hundredths of a second, and so those presented here are rounded to the nearest tenth of a second.

As the coarse grid and D_{ff} determine the convergence bounds for these algorithms, the spectral equivalence properties for A_{ff} and D_{ff} are reported for each problem and each grid resolution. In particular, we report the number of coarse-grid points, n_c, and $\epsilon = \lambda_{\max}(D_{ff}^{-1}A_{ff}) - 1$, computed from the upper spectral equivalence bounds for the fine-scale operators (which is itself computed using a few steps of Arnoldi iteration). The asymptotic convergence factor, ρ, measured in the A-norm, is computed directly using at most 200 steps of the power method. The AMGr cycle used is one with l prerelaxations followed by a coarse-grid correction and no postrelaxation. The analysis of Theorem 3 can also be performed in this setting, yielding the bound, $\|MG\|_A \leq \left(\frac{\epsilon}{1+\epsilon}\left(1+\left(\frac{\epsilon^{2l-1}}{(2+\epsilon)^{2l}}\right)\right)\right)^{\frac{1}{2}}$, where the square-root arises as the convergence of this one-sided cycle is limited by halving the number of relaxation sweeps relative to the symmetric cycle. In the results presented here, we choose $l = 3$, which, for all examples, yielded the fastest solution time for $1 \leq l \leq 5$.

Table 1 shows the performance of the two-level AMGr scheme on matrices that come from bilinear finite element discretizations of the two-dimensional diffusion equation, $-\nabla \cdot K\nabla p$. Three different choices of the diffusion coefficient, $K(x,y)$, are chosen, corresponding to the homogeneous case ($K(x,y) = 1$, the Poisson equation), the smoothly varying case ($K(x,y) = 10^{-8} + 10(x^2 + y^2)$), and the discontinuous case, where $K(x,y)$ is 1 in 80% of the elements, chosen at random, and 10^{-8} in the remaining 20% of the elements. In all three cases, the two-level AMGr algorithm performs well, reducing the A-norm of the error by a factor of 10^6 in at most 20 iterations.

TABLE 1
Performance of two-level AMGr on test matrices from discretizations of $-\nabla \cdot K(x,y)\nabla p(x,y)$.

Coefficient	Grid	n_c	ϵ	t_{setup}	t_{solve}	# iters.	ρ
$K(x,y) = 1$	32×32	225	4.98	0.5	0.2	13	0.37
	64×64	961	4.99	7.2	0.7	13	0.37
	128×128	3969	5.00	119.3	3.1	13	0.37
$K(x,y) = 10^{-8} + 10(x^2 + y^2)$	32×32	255	4.53	0.4	0.3	17	0.51
	64×64	1012	4.69	6.4	1.2	19	0.57
	128×128	4036	4.69	106.8	5.1	18	0.55
random $K(x,y)$	32×32	277	4.61	0.5	0.3	18	0.52
	64×64	1149	4.78	6.4	1.4	19	0.59
	128×128	4715	4.86	109.2	6.7	20	0.62

Grid sizes are effectively reduced, and the resulting asymptotic convergence factors are bounded well away from unity.

In the homogeneous case, the coarsening chosen by Algorithm 3 results in uniform coarse grids. All Dirichlet boundary nodes are automatically chosen as fine-grid points (as they are already diagonally dominant). Nodes neighboring the boundary are also attractive as fine-grid points because of their inherent diagonal dominance (from the algebraic elimination of connections to the boundary nodes). Thus, the 33×33 mesh of unknowns from the 32×32 element grid is coarsened to a uniform 15×15 mesh. Similarly, the 64×64 and 128×128 element meshes are coarsened to meshes of 31×31 and 63×63 degrees of freedom. The parameter, ϵ, of the splitting stays close to 5.0 as the mesh is refined and is theoretically bounded by 9, from Corollary 2, for our choice of $\theta = 0.55$. As expected from a two-level scheme, the setup time does not scale with refinement, nor does the solve time (although this is closer). The resulting convergence factors of approximately 0.37 do not degrade with grid size, as predicted by Theorem 3, although the measured convergence factor is signficantly better than that predicted by the theory.

For the variable coefficient problems, coarsening is less uniform, as is expected given the variations in these operators. The spectral equivalence constant, ϵ, however, remains bounded below 5, leading to effective two-level solvers. Setup times again do not scale, as expected, and the lack of scaling in the iteration times is slightly more pronounced. Asymptotic convergence factors remain nicely bounded below one, and below those predicted by Theorem 3. As a result, iteration counts for reducing the A-norm of the error remain small.

4.1.1. Implementation and cost issues. For multilevel tests, we give more weight to the question of an efficient implementation of Algorithm 3. There are two potential sources of inefficiency in the partitioning algorithm—the selection of the new C-point and the updating of all neighboring F-points. In order to be able to efficiently find the absolute minimal measure, we would need to either keep a sorted list of the measures, which would need updating each time a point was put into C, or search the (unsorted) list of measures. Instead, we implement an incomplete bucket sort [13, section 9.4] based on a doubly linked-list data structure [13, section 11.2] that allows efficient identification of a point with approximately minimal measure as well as fast updates of measures.

For each potential F-point, a standard linked-list data item is created. With each linked-list item, a key for the associated node is assigned as the measure of the node. As measures of potential F-points lie in the interval $[0, \theta)$, this interval is divided into equal-sized subintervals (or buckets), and all list items whose keys lie in the given bucket are added to a single doubly linked list associated with that bucket. The nonempty bucket associated with the smallest key then contains not only the item associated with the absolute minimal measure, but also all items within some small range of the minimal measure. A single element is chosen out of this bucket as an approximately minimal measure and used as the new C-point, j, in Algorithm 3. Taking advantage of the symmetry of A, we then can look over all nonzero columns, i, in row j of A, to update $\hat{\theta}_i$ to account for the removal of j. This is easily accomplished by an indexing array that points to the linked-list item corresponding to node i. Using these structures, we no longer select the absolute minimal-measured node to remove at each iteration, but are rewarded with $O(1)$ update and find-approximate-min operations, and an $O(n)$ amount of storage for partitioning the $n \times n$ matrix, A.

As a multigrid cycle, we consider a mixture of components from the AMGr algorithm and a classical multigrid V-cycle. Full relaxation sweeps, as opposed to the

TABLE 2
Performance of multilevel AMG with AMGr-style interpolation, as in Theorem 3, on test matrices from discretizations of $-\nabla \cdot K(x,y)\nabla p(x,y)$.

Coefficient	Grid	# levels	c_G	c_A	t_{setup}	t_{solve}	# iters.	ρ
$K(x,y) = 1$	128×128	6	1.31	1.32	0.03	0.08	10	0.34
	256×256	7	1.32	1.32	0.2	0.3	10	0.36
	512×512	8	1.33	1.33	0.6	1.2	10	0.38
	1024×1024	9	1.33	1.33	2.6	4.9	10	0.40
	2048×2048	10	1.33	1.33	12.4	17.9	9	0.41
$K(x,y) = 10^{-8} + 10(x^2 + y^2)$	128×128	6	1.31	1.32	0.03	0.08	10	0.33
	256×256	7	1.32	1.32	0.2	0.3	10	0.35
	512×512	8	1.33	1.33	0.6	1.2	10	0.37
	1024×1024	9	1.33	1.33	2.6	5.0	10	0.39
	2048×2048	10	1.33	1.33	12.3	19.9	10	0.41
random $K(x,y)$	128×128	6	1.38	1.75	0.08	0.2	17	0.59
	256×256	7	1.39	1.83	0.5	0.7	17	0.62
	512×512	8	1.40	1.89	3.1	3.2	18	0.68
	1024×1024	9	1.40	1.93	33.7	14.9	21	0.77
	2048×2048	10	1.40	1.98	907.6	88.2	31	0.91

F-relaxation used in AMGr, are prevalent in AMG and, so, we use a full sweep of Gauss–Seidel relaxation for both pre- and postrelaxation within the V-cycle, ordered so that coarse points are relaxed first on the downward part of the cycle, but that F-points are relaxed first on the upward part of the cycle. Interpolation is defined as in AMGr (cf. Theorem 3), using the diagonal matrix, D_{ff}, as in Corollary 2. The problem is coarsened until the next coarsest grid would have fewer than 8 degrees of freedom, and the coarsest-grid problem is solved using a (complete) Cholesky factorization.

We first test this AMG approach on the same problems as in Table 1, again with $\theta = 0.55$. The number of subintervals in the bucket sort is taken to be 1000, regardless of the problem size. As we no longer explicitly need the parameter, ϵ, to define relaxation, we do not calculate it for these examples. In Table 2, we show the number of levels created in the multigrid hierarchy for each problem. To measure the efficiency of the multigrid coarsening process, we report both the multigrid grid complexity and operator complexity. The grid complexity, c_G, defined as the sum of the number of grid points on each level divided by the number of grid points on the finest level, is an indicator of the rate at which the problem is coarsened. The multigrid operator complexity, c_A, defined as the sum of the number of nonzero entries in the operator on each level divided by the number of nonzero entries in the finest-grid operator is a measure of both the storage needed and computational cost of a single multigrid cycle. The times required for the setup and solve phases are reported, where solution is taken to mean a reduction in the residual by a factor of 10^6 in solving the homogeneous problem, $A\mathbf{x} = \mathbf{0}$, given a random initial guess. The number of iterations needed for convergence, as well as the asymptotic convergence factor, ρ, measured after at most 200 iterations are also reported.

Performance for the two problems with smooth coefficients is typical of geometric multigrid methods applied to these problems. The number of levels grows by one for each grid refinement, and both grid and operator complexities remain relatively constant as the problem is refined. In fact, because of the approximate minimization in Step (2)(a) of Algorithm 3, a series of fully coarsened grids is chosen for both of these problems, bounding the grid complexity by a factor of $\frac{4}{3}$. That the operator complexity is also bounded by $\frac{4}{3}$ for these problems is a consequence of this regular coarsening, coupled with the fact that, in the variational definition of the coarse-grid operator, the stencil of the coarse-to-fine interpolation operator is that of A_{fc},

resulting in coarse-grid operators that are also nine-point operators on a regular mesh. Setup times roughly scale with matrix size (although there is some slowing, likely due to the indirect indexing operations needed to define interpolation), and the solve times scale nicely with problem size. The resulting convergence factors are bounded well away from unity, although are slightly worse than typical (Ruge–Stüben) AMG performance on these problems, which would lead to convergence factors in the range of 0.1 to 0.2. The simple interpolation operator defined in Theorem 3 allows for efficient computation of the AMG interpolation and coarse-grid operators, but limits its applicability. For these problems, where the AMGr interpolation is effective, setup times for the AMG algorithm can be much lower than those of classical AMG, but typically result in poorer convergence behavior, as seen here.

Performance for the randomly discontinuous coefficient problem, as seen in Table 2, degrades with refinement of the mesh. While the grid complexity does remain nicely bounded, only slightly larger than the factor of $\frac{4}{3}$ seen in the previous examples, there is a notable growth in the operator complexities. This growth is a result of the nonuniform coarsening that is naturally chosen for this problem, coupled with the use of an interpolation stencil that exactly matches that of A_{fc}. In this way, the density of nonzeros in coarse-grid operators grows as the multigrid cycle coarsens. Such growth is compounded in further coarsenings, as new nonzero entries in the stencil of A_{fc} lead to new nonzeros in the variational coarse-grid operator, as well as denser matrix-matrix-matrix products needed to compute these products. This is most noticeable in the dramatic increase in the time needed for the setup stage as the grid is refined, a result of the effectively dense matrix computations needed to coarsen the coarsest few levels of the multigrid hierarchy for the 2048×2048 element grid problem. The obvious solution to this growth is to drop some of the small (or weak) off-diagonal connections in these growing factors, either in the choice of interpolation or in the computation of the coarse-grid operators themselves. Dropping weak off-diagonal connections in the definition of interpolation is, in fact, exactly what is done in the standard AMG approach, discussed below. The slower growth in the solve times is very closely correlated to the combination of the growth in the problem size (by a factor of 4) and the number of iterations needed to reduce the residual appropriately (by a factor of $\frac{3}{2}$ between the 1024×1024 and 2048×2048 problems). This increase in the number of iterations, in turn, is closely linked to the increase in the asymptotic convergence factors as the grids are refined. It is difficult to say (without finer grids to test) if this increase is due to a lack of optimality of the overall multigrid approach (i.e., convergence factors continue to worsen as the grid is refined), or is related to the poorer treatment of the coarsest levels of the multigrid hierarchy.

4.1.2. Importance of theoretical conditions. An even poorer example of the performance of an AMGr-like algorithm can be seen when we violate the assumption that the "reduced" matrix, $\left[\begin{smallmatrix} D_{ff} & -A_{fc} \\ -A_{cf} & A_{cc} \end{smallmatrix}\right]$, be nonnegative definite. In Theorem 5, we note that such an example cannot arise when A is diagonally dominant and, so, we consider another PDE-based example, $-\nabla \cdot K(x,y)\nabla p(x,y)$, where we choose a tensor coefficient, $K(x,y) = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 0.01 \end{smallmatrix}\right]$. As with the continuum operator, the discretization has strong connections between nodes that lie on the same line of fixed y value, but a finite-element discretization of this equation has significant nonzero connections to degrees of freedom that lie off of these lines as well. The greedy coarsening algorithm effectively recognizes these properties of strength of connection and chooses the first coarse grid to be a semicoarsened version of the fine grid, known to be effective for this problem. The AMGr interpolation operator, however, attempts to define an

interpolation that connects a fine-grid point to all of its coarse-grid neighbors. In this case, this includes interpolation from points that are not strongly related to the fine-grid point in question. As a result, the multigrid complexities grow substantially (as A_{fc} is relatively dense) and the overall algorithm performs poorly. For a 32×32 element grid, a grid complexity of 4.69 and operator complexity of 48.16 were recorded, with an asymptotic convergence factor of 0.98989.

A solution to the degradation of performance seen for the randomly discontinuous problems or overall poor performance in the anisotropic case is to combine the coarsening of Algorithm 3 with classical AMG interpolation techniques. While classical AMG [29] limits interpolation to the stencil of A_{fc}, not all nonzero entries in this stencil are used. To compensate for dropping connections in A_{fc}, as well as for strong connections in A_{ff}, the nonzero entries in a row of interpolation are also not necessarily equally proportional to the original entries of A_{fc}. Interpolation is defined based on the principle that the errors remaining after relaxation yield small residuals, so that interpolation needs to be most accurate for vectors, $\mathbf{e}$, such that $A\mathbf{e} \approx \mathbf{0}$ in a pointwise sense. Considering a fine-grid point, i, this means that

$$\begin{aligned} (A\mathbf{e})_i &= 0, \\ \text{or } a_{ii}e_i &= -\sum_{k \in C_i} a_{ik}e_k - \sum_{j \in F_i} a_{ij}e_j, \end{aligned}$$

where the sets C_i and F_i are defined through the localizations of the global coarse and fine sets: $C \cap Adj(i)$, and $F \cap Adj(i)$. Interpolation from coarse neighbors of the point, i, is relatively straightforward, so the main goal is to collapse the connections in F_i onto C_i or i itself in a way to meaningfully define an interpolation operator.

4.1.3. Combination with AMG coarsening. Classical AMG [29] categorizes the off-diagonal connections in row i as strong or weak depending on their suitability for use in interpolation. When the connection between two points is said to be strong, it is treated as an important connection to account for in interpolation, while a weak connection is to be discarded. The classical definition for the set of points that i strongly depends on is $S_i = \{j : -a_{ij} \geq \beta \max_{k \neq i}\{-a_{ik}\}\}$ for a chosen strength threshold, β [10] (other possible algebraic definitions of strong connections include those in [7, 11]). The set C_i is then defined by $C_i = C \cap S_i$, while $F_i = Adj(i) \setminus C_i$. Connections in F_i are then further partitioned into strong connections, $F_i^s = F_i \cap S_i$, and weak connections, F_i^w. Weak connections are not important in interpolation, so they are collapsed directly to the diagonal. Strong connections to fine-grid points are accounted for through an indirect interpolation, based on an assumption on the errors that are slow to be reduced by relaxation (as in [29]) or, possibly, further knowledge of such errors [9]. Making choices consistent with classical AMG gives the interpolation formula [10] for $i \in F$,

$$e_i = -\sum_{k \in C_i} \left(\frac{a_{ik} + \sum_{j \in F_i^s} \left(\frac{a_{ij}a_{jk}}{\sum_{k' \in C_i} a_{jk'}} \right)}{a_{ii} + \sum_{j \in F_i^w} a_{ij}} \right) e_k. \tag{4}$$

TABLE 3
Performance of multilevel AMG with classical AMG-style interpolation and a two-stage coarsening algorithm on test matrices from discretizations of $-\nabla \cdot K(x,y)\nabla p(x,y)$.

Coefficient	Grid	# levels	c_G	c_A	t_{setup}	t_{solve}	# iters.	ρ
$K(x,y) = 1$	128×128	6	1.31	1.32	0.07	0.04	5	0.11
	256×256	7	1.32	1.32	0.3	0.2	5	0.12
	512×512	8	1.33	1.33	1.3	0.6	5	0.13
	1024×1024	9	1.33	1.33	5.3	2.5	5	0.14
	2048×2048	10	1.33	1.33	21.3	10.2	5	0.14
$K(x,y) = 10^{-8} + 10(x^2 + y^2)$	128×128	6	1.31	1.32	0.07	0.04	5	0.11
	256×256	7	1.32	1.32	0.3	0.2	5	0.12
	512×512	8	1.33	1.33	1.3	0.6	5	0.13
	1024×1024	9	1.33	1.33	5.3	2.6	5	0.14
	2048×2048	10	1.33	1.33	21.1	10.2	5	0.14
random $K(x,y)$	128×128	10	1.50	1.97	0.1	0.07	6	0.23
	256×256	12	1.51	2.02	0.6	0.3	6	0.28
	512×512	15	1.51	2.06	2.3	1.2	6	0.35
	1024×1024	19	1.51	2.08	9.7	4.7	6	0.40
	2048×2048	22	1.52	2.10	40.5	10.0	6	0.46
anisotropic $K(x,y)$	128×128	9	1.91	2.26	0.08	0.06	5	0.13
	256×256	11	1.95	2.34	0.4	0.3	5	0.13
	512×512	13	1.96	2.39	1.5	1.0	5	0.13
	1024×1024	15	1.97	2.41	6.2	4.1	5	0.20
	2048×2048	16	1.97	2.43	27.2	17.3	5	0.20

Dividing the neighbors of a node, i, into groups of strong and weak connections is necessary to ensure that AMG interpolation does not try to connect points that have no real relationship in the matrix (or governing PDE). In order to define nontrivial interpolation by (4), we now need to ensure that C_i is nonempty. To do this, we use a two-pass coarsening routine based on that used in classical AMG [29], where the coarse grid is first chosen as a maximal independent set over the strong connections of the matrix; then a second pass is made, changing F-points to C-points, to ensure that the strong connections of i are properly accounted for in interpolation. Here, we let Algorithm 3 choose an initial partition into F and C, and then use the AMG second pass algorithm that allows changing F-points to C-points to allow for better use of (4) in defining interpolation [29]. Note that removing points from F can only improve the diagonal dominance of A_{ff}, so Theorem 4 and its corollaries will still hold for the repartitioned system. Our only concern then is to ensure that we do not enlarge C by too much in seeking good approximation properties for the resulting interpolation.

Results for AMG-style interpolation (4), using Algorithm 3 as a first pass for coarsening, augmented by the second pass described above, are shown in Table 3. Here, we choose the AMG strength threshold $\beta = 0.25$ for the isotropic problems and $\beta = 0.3$ for the anisotropic problem. Operator and grid complexities are reported, as are the number of levels in the multigrid hierarchy, showing the deepening hierarchies as the coarsening becomes less uniform. This increase in the number of levels allows AMG to retain the low operator complexities seen in Table 2, yet still address the important need of all fine-grid points for adequate strong connection to the coarse grids.

For all of these problems, we see very effective solvers, with low iteration counts and convergence factors. For the smoothly varying coefficient problems, we see the same grid and operator complexities for the AMG-based approach as for the AMGr-style interpolation, as the second pass of coarsening does not add any points to the coarse grids chosen by Algorithm 3 and the nonzero pattern of interpolation is the same. The setup times for these problems are larger, due both to the second pass of coarsening (which still checks that all points have suitable connection to the coarse grid) and the more complicated definition of interpolation. The AMG-style interpo-

lation, however, produces solvers with lower convergence factors, so the extra setup time is roughly equally offset by the lower solve time. For the randomly discontinuous-coefficient problem, we see slightly higher grid and operator complexities, but these are more evenly distributed throughout the multigrid hierarchy, so there is no longer a significant slowing of the setup stage on finer grids. The resulting solvers are slightly less efficient than those of the smooth coefficient cases but still much better than those shown in Table 2. The anisotropic problem is also well handled by this algorithm, with larger grid and operator complexities consistent with geometric multigrid approaches to this problem. As with all of these examples, setup and solve times scale nicely with the increase in problem size, and the convergence factors grow slightly, but remain bounded well below unity.

4.2. ARMS. Effective two-level block factorizations, as discussed in Theorem 1, rely on good spectrally equivalent preconditioners for both the fine-scale block and Schur complement, $\hat{A}_{cc}$. To test the use of the diagonal-dominance-based coarsening scheme with approximate block-factorization preconditioners, we first consider two-level preconditioners, making use of the equivalence from Corollary 2 to handle the fine-scale preconditioning. Preconditioning of the Schur complement block is considered both exactly and using the variational coarse-grid operator as an approximation to $\hat{A}_{cc}$.

The requirement that $\left[\begin{smallmatrix} D_{ff} & -A_{fc} \\ -A_{fc}^T & A_{cc} \end{smallmatrix}\right]$ be positive semidefinite also arises in the theory for ARMS (cf. Theorem 1), so we begin by testing the two-level ARMS algorithm on the same PDE-based test problems as we did with AMGr. Some of the measures discussed above are again relevant, including the overall setup time (choosing the coarse grid, partitioning the fine-scale operator, computing the fine-scale preconditioner, and computing the coarse-scale operator). We test the two-level ARMS algorithm as a preconditioner for conjugate gradient (CG) by choosing a right-hand side, $\mathbf{b}$, to be the result of the matrix applied to the vector of all ones, $\mathbf{b} = A\mathbf{1}$, and a random initial guess. The number of iterations and time needed to reduce the A-norm of this error by a relative factor of 10^6 are also reported. Once again, these two-level results are computed using MATLAB with the reported times useful only for comparison between themselves and those in Table 1; the multilevel results that follow are computed using research C codes compiled with the appropriate optimization flags. The reported CPU times are accurate only to a few hundredths of a second and, so, are rounded to the nearest tenth of a second.

The coarse grids are selected using the same code and parameters as are used for the runs in Table 1, so the coarse-grid sizes presented here are the same as those (and are presented only for convenient reference). As in Theorem 1, the important quantities in bounding the condition number of the overall preconditioner are $\lambda_M = \lambda_{\max}(D_{ff}^{-1}A_{ff})$ and $\lambda_m = \lambda_{\min}(D_{ff}^{-1}A_{ff})$, the upper and lower spectral equivalence bounds for the fine-scale operators. We also report the condition number of the preconditioned system, $\kappa(B^{-\frac{1}{2}}AB^{-\frac{1}{2}}) = \frac{\lambda_{\max}(B^{-1}A)}{\lambda_{\min}(B^{-1}A)}$, as an indicator of the overall success of the preconditioner.

Table 4 shows details of the performance of two-level ARMS-preconditioned CG, using the exact Schur complement, on these model PDE problems. As observed with AMGr, Algorithm 3 effectively reduces the dimension of the fine-scale problem and produces a diagonal matrix, D_{ff}, that has good spectral equivalence to the fine-scale block, A_{ff}. Setup and solve times scale poorly, as expected, due to the time required to first compute the exact Schur complement and then to invert it for each

TABLE 4

Performance of two-level ARMS using exact Schur complements on test matrices from discretizations of $-\nabla \cdot K(x,y)\nabla p(x,y)$.

Coefficient	Grid	n_c	λ_m	λ_M	t_{setup}	t_{solve}	# iters.	$\kappa(B^{-\frac{1}{2}}AB^{-\frac{1}{2}})$
$K(x,y)=1$	32×32	225	1.00	5.98	0.6	0.3	19	7.45
	64×64	961	1.00	5.99	10.7	6.5	18	7.48
	128×128	3969	1.00	6.00	199.1	366.0	18	7.49
$K(x,y)=10^{-8}+10(x^2+y^2)$	32×32	255	1.00	5.53	0.7	0.4	20	8.64
	64×64	1012	1.00	5.69	13.1	8.5	21	9.33
	128×128	4036	1.00	5.69	241.7	419.8	20	9.14
random $K(x,y)$	32×32	277	1.00	5.61	1.3	0.4	19	8.16
	64×64	1149	1.00	5.78	24.2	11.5	20	9.25
	128×128	4715	1.00	5.86	253.8	696.2	21	10.16

TABLE 5

Performance of two-level ARMS using variational approximations to the Schur complements on test matrices from discretizations of $-\nabla \cdot K(x,y)\nabla p(x,y)$.

Coefficient	Grid	n_c	ν_m	ν_M	t_{setup}	t_{solve}	# iters.	$\kappa(B^{-\frac{1}{2}}AB^{-\frac{1}{2}})$
$K(x,y)=1$	32×32	225	0.63	1.00	0.3	0.2	22	10.53
	64×64	961	0.63	1.00	4.4	0.7	22	10.62
	128×128	3969	0.63	1.00	71.8	4.0	22	10.64
$K(x,y)=10^{-8}+10(x^2+y^2)$	32×32	255	0.47	1.00	0.4	0.3	25	14.13
	64×64	1012	0.43	1.00	6.3	1.3	27	16.47
	128×128	4036	0.44	1.00	109.2	6.4	26	16.02
random $K(x,y)$	32×32	277	0.48	1.00	0.4	0.5	25	13.97
	64×64	1149	0.41	1.00	6.3	1.4	27	16.83
	128×128	4715	0.37	1.00	109.6	9.8	28	18.63

iteration. Iteration counts are stable as the problem size grows, and there is only slight growth in the condition numbers as the meshes are refined. The resulting condition numbers satisfy the bounds of Theorem 1, but these bounds are clearly not sharp. In particular, the upper bounds for these condition numbers are at least 100, while the largest observed condition number is just larger than 10. The lower bound, dependant only on the spectral equivalence of D_{ff} and A_{ff}, is a much better predictor of the true performance for these problems.

Table 5 shows details of the performance of two-level ARMS preconditioned CG, using variational approximations to the Schur complement, on these model PDE problems. The variational approximation is formed, as in AMGr, by taking the interpolation operator, $P = \begin{bmatrix} D_{ff}^{-1}A_{fc} \\ I \end{bmatrix}$, and then computing the approximation, $S = P^T AP$. As in Theorem 1, the spectral equivalence between S and $\hat{A}_{cc}$ plays an important role in the upper bound on the condition number of the preconditioned system. Here, we report the upper and lower spectral-equivalence bounds, ν_M and ν_m, along with the usual measures of performance. The fine-scale spectral-equivalence bounds, λ_M and λ_m, for these problems are the same as those appearing in Tables 1 and 4. The coarse-scale spectral-equivalence bounds for these problems are nicely controlled, with the upper bounds close to 1, and the lower bounds away from 0. Setup time again does not scale, as is expected from the increasing cost of computing the partitioning (without an efficient sorted data structure) and coarse-scale operator. Iteration times are very similar to those of two-level AMGr, as in Table 1, with a slightly larger factor of increase over that of the problem size. Iteration counts are steady, as are the condition numbers of the preconditioned operators. Once again, the bounds of Theorem 1 are loose bounds on the actual performance of the preconditioner; the lower bound does not take into account the role of the approximation to the Schur complement and, so, is unchanged from the case of an exact Schur complement, whereas the upper

bound increases slightly (by a factor of $\frac{1}{\nu_m}$) but remains significantly larger than the condition numbers that are observed.

4.2.1. PDE problems. To test the effectiveness of coarsening within ARMS using Algorithm 3 in a multilevel setting, we again consider PDE-based problems where the system matrix is symmetric, positive definite, and diagonally dominant (so that we are certain to satisfy all of the conditions of Theorem 1). As exact computation of the Schur complement on each level is prohibitively expensive, we first test the variational coarsening scheme used in Table 5. Here, GMRES is preconditioned by ARMS, and the number of iterations and time to solution needed to reduce the ℓ^2-norm of the residual by a relative factor of 10^6 are recorded. The time needed to set up the ARMS preconditioner is also reported, including time needed for choosing the coarse grids, computing the variational coarse-grid operators, and computing the exact Schur complement for the coarsest-grid operator, which is taken to be any operator which has dimension less than 10, or which is itself θ dominant. The parameter, θ, is again chosen to be 0.55. The ARMS preconditioner complexity (fill factor), c_B, defined to be the storage required for the ARMS preconditioner on all levels, divided by the number of nonzeros in the matrix A (and, thus, excluding the storage needed for A itself) is also recorded, as is the number of levels in the ARMS hierarchy.

Here, we use GMRES as the Krylov accelerator instead of CG for reasons of consistency. The ARMS algorithm has been developed with a focus on general matrices and not only symmetric problems [32, 30]. As such, GMRES has been the appropriate choice of accelerator. For these symmetric problems, using variational coarse-grid operators, we have chosen to use GMRES to remain consistent with this previous work. Additionally, while effort is made to retain some degree of symmetry in the ILU-based preconditioners to follow, symmetry may be sacrificed in favor of accuracy in the treatment of the coarsest levels (which may be quite large). In these cases, as well, GMRES is the only appropriate choice of Krylov subspace method. Thus, we have chosen to use it here, as well, for ease of comparison between these results. Nonsymmetric extensions of the approaches proposed here are considered (and compared to symmetric approaches) in [25].

Results for the multilevel-ARMS-preconditioned GMRES using variational coarse-grid operators and the greedy coarsening algorithm are shown in Table 6. Coarsening for the constant and smoothly varying cases are again the same, as they were in the multilevel AMGr results of Table 2. Counting the storage needed for the fine-scale operator itself, the operator complexities for all three problems are slightly higher than those of AMGr. Coupled with the added memory requirements of GMRES (storing one fine-scale vector per iteration), the 2048×2048 element mesh problems that were possible with AMGr are no longer tractable on a workstation with 2 Gigabytes of memory. Setup times are, in general, notably larger than AMGr for the same problem, with the only exception being the 1024×1024 mesh for the randomly discontinuous permeability where AMGr performed quite poorly. In all cases, iteration times for ARMS-preconditioned GMRES are larger than those for AMGr.

4.2.2. Combination with ILU-based coarsening. Theorem 1 also requires that $\left[\begin{smallmatrix} D_{ff} & -A_{fc} \\ -A_{cf} & A_{cc} \end{smallmatrix}\right]$ be nonnegative definite for the bounds on the condition number of the preconditioned system to hold. While this is true for all of the PDE problems tested in Table 6, it is not true for an anisotropic problem, such as when $K(x,y) = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 0.01 \end{smallmatrix}\right]$. Testing the variational-coarsening-based ARMS as a preconditioner for GMRES on a 32×32 element mesh for this problem requires 911 iterations

TABLE 6
Performance of multilevel ARMS with variational coarsening on test matrices from discretizations of $-\nabla \cdot K(x,y)\nabla p(x,y)$.

Coefficient	Grid	# levels	c_B	t_{setup}	t_{solve}	# iters.
$K(x,y) = 1$	128×128	6	0.69	0.1	0.3	42
	256×256	7	0.70	0.4	1.5	46
	512×512	8	0.70	1.7	8.0	50
	1024×1024	9	0.70	6.9	35.4	53
$K(x,y) = 10^{-8} + 10(x^2 + y^2)$	128×128	6	0.69	0.1	0.3	42
	256×256	7	0.70	0.4	1.5	46
	512×512	8	0.70	1.7	7.8	49
	1024×1024	9	0.70	6.9	34.2	52
random $K(x,y)$	128×128	6	0.89	0.1	0.3	46
	256×256	7	0.92	0.6	2.0	53
	512×512	8	0.95	2.7	12.2	64
	1024×1024	9	0.96	11.9	52.8	66

to reduce the residual by a relative factor of 10^6. To achieve the robustness we would like from our preconditioner, we turn to the full power of the ARMS approach [30], using an ILU factorization in place of the diagonal D_{ff}. Because we have partitioned A specifically so that the A_{ff}-block is diagonally dominant, we expect to be able to find a suitably sparse ILU factorization that provides both better spectral equivalence to A_{ff} than is possible with a diagonal matrix and a better approximate Schur complement. This combination should then result in a more robust preconditioner than is possible with a diagonal D_{ff}.

Since the fine-scale operators from these PDE problems are symmetric, certain savings are natural in the coarse-grid selection algorithm. In particular, when a coarse-grid node, j, is selected from the set, U, we need only look at the nonzero elements in row j of the matrix to find the nodes whose measures need updating. This is particularly easy when using a compressed-sparse-row storage format, as these indices and the appropriate updates are easily available. In the general (nonsymmetric) case, the update is not as convenient (when using compressed sparse row storage), as a search for nonzero entries in column j must be done, requiring either extra storage or extra computation. To ensure this advantage is maintained, a modification of the ILUT procedure is used in order to ensure that a symmetric dropping strategy is employed. Here, the ILU factorization of A_{ff} is computed using the ILUT algorithm (dropping based both on size and rowwise fill factors [31]), but the U factor in $A_{ff} \approx LDU$ is replaced by L^T. The matrix product $L^{-1}A_{fc}$ is easily computed at the same time as the factorization of A_{ff}, by applying the elimination to the extended matrix, $[A_{ff},\ -A_{fc}]$. The Schur complement can then be computed rowwise from the matrix equation, $S = A_{fc} - (L^{-1}A_{fc})^T D^{-1}(L^{-1}A_{fc})$. In order to control sparsity directly within S, we precompute the diagonal entries of S, allowing dropping based on size, using the symmetric drop criterion, $s_{ij} \leq \alpha\sqrt{s_{ii}s_{jj}}$. Dropping based on level of fill is also possible [22] but not used in the examples presented here.

Table 7 shows the results for the ARMS-preconditioned GMRES algorithm based on symmetrized ILUs. The drop tolerance, α, is chosen to be either 0.01 or 0.1 for both the ILU factorization of A_{ff} and the approximate Schur complement calculation. Choosing α appropriately requires a balance between accuracy of the ILU approximation to A_{ff} and the fill of the resulting preconditioner. Typically, choosing α to be relatively small results in a more effective preconditioner, but a larger value of α results in a smaller preconditioner complexity, c_B, which is useful for large problems

TABLE 7

*Performance of multilevel ARMS with symmetric ILU-based coarsening on test matrices from discretizations of $-\nabla \cdot K(x,y)\nabla p(x,y)$. Results marked with a * indicate that the residual reduction criterion was a relative factor of 10^4 instead of 10^6.*

Coefficient	Grid	α	# levels	c_B	t_{setup}	t_{solve}	# iters.
$K(x,y) = 1$	128×128	0.01	2	2.59	0.3	0.3	28
	256×256	0.01	2	2.65	1.5	2.5	44
	512×512	0.01	2	2.68	12.7	24.5	82
	1024×1024	0.1	2	1.03	159.1	34.2*	46*
$K(x,y) = 10^{-8} + 10(x^2 + y^2)$	128×128	0.01	2	2.60	0.3	0.4	31
	256×256	0.01	2	2.65	1.5	3.4	56
	512×512	0.01	2	2.68	12.7	31.7	97
	1024×1024	0.1	2	1.03	159.6	40.6*	52*
random $K(x,y)$	128×128	0.01	3	1.40	0.2	0.4	32
	256×256	0.01	3	1.41	0.7	2.5	45
	512×512	0.01	3	1.42	3.1	25.1	83
	1024×1024	0.01	3	1.42	13.5	12.4*	17*
anisotropic $K(x,y)$	128×128	0.01	5	1.61	0.2	0.3	26
	256×256	0.01	5	1.62	0.8	2.3	42
	512×512	0.01	5	1.63	3.3	17.3	65
	1024×1024	0.01	5	1.63	14.9	6.9*	10*

where memory limitations are restrictive. A further dropping rule is applied to each row of L, allowing no more than twice the average number of nonzeros in a row of A. Again, $\theta = 0.55$, and the coarsening is continued until either the coarse-grid operator has fewer than 10 degrees of freedom or is itself θ dominant.

The robustness of the ILU-based approach comes at a cost in increased memory consumption, reflected in the higher operator complexities for the results in Table 7, as compared to those in Table 6. For the smooth isotropic coefficients, only one coarsening was performed, because of the resulting θ-dominance of the coarse-grid operator. This behavior is largely independent of the choices of α and the row-based fill threshold, only changing if θ was measurably changed. As a result, the operator complexities for these problems are somewhat large, over 2 when $\alpha = 0.01$. In order to perform any meaningful number of iterations on the 1024×1024 element mesh problems, it is necessary to choose $\alpha = 0.1$ to reduce the fill. Even then, memory limitations prevented reducing the residual by a relative factor of 10^6, and a smaller reduction, by a relative factor of 10^4 is used instead. GMRES performance (residual reduction per iteration) for these problems does tend to get worse as the iterations proceed and, so, the number of iterations recorded for these large problems is significantly fewer than the number that would be needed for a reduction by a relative factor of 10^6.

Comparing performance between these four problems, however, shows the robustness of the combination of the greedy coarsening algorithm with ILU-based ARMS, which will be further demonstrated in section 4.2.3. Preconditioner complexities, a primary factor in memory limitations, are controlled by the parameter, α. Setup times are generally comparable across the different problems (for the same α), as are solution times and iteration counts. In fact, the problems which caused most difficulty for the variational coarse-grid operator approach are now more efficient than the smoothly varying coefficient problems that were most efficient. The structure of these problems appears to naturally lead to deeper ARMS hierarchies, yielding lower preconditioner complexities and, thus, lower costs per iteration.

Comparing the results for AMG (Table 3) to those for ARMS (Table 7), we see that both generally perform well for these PDE-based problems, but that the AMG approach appears to be more efficient. ARMS requires storage of part of the

TABLE 8
Performance of multilevel ARMS with symmetric ILU-based coarsening on test matrices with $\theta = 0.55$, drop tolerance of 0.01, average fill factor of 5, and a maximum of 10 levels of coarsening.

Name	n	nnz	c_B	t_{setup}	t_{solve}	# iters.
bodyy4	17546	121938	1.08	0.1	0.05	6
bodyy5	18589	129281	1.06	0.1	0.04	4
bodyy6	19366	134748	1.04	0.1	0.03	3
bcsstk18	11948	149090	1.17	0.1	2.7	189
t2dah_a	11445	176117	1.23	0.2	0.9	88
obstclae	40000	197608	2.41	0.3	0.09	5
jnlbrng1	40000	199200	1.85	0.3	0.1	7
minsurfo	40806	203622	2.02	0.3	0.1	6
bcsstk25	15439	252241	1.26	0.3	3.7	190
cvxbqp1	50000	349968	6.51	7.2	1111.2	2670
bcsstk17	10974	428650	2.30	1.2	24.7	660
t3dl_a	20360	509866	1.48	0.5	20.8	465
gridgena	48962	512084	1.32	0.5	2.7	76
finan512	74752	596992	1.32	0.6	0.2	4
gyro_k	17361	1021159	1.13	1.1	19.9	397
msc10848	10848	1229778	1.54	2.9	22.0	431
cfd1	70656	1828364	3.33	10.5	391.3	1003
vanbody	47072	2336898	7.55	244.7	1131.8	2415
oilpan	73752	3597188	1.48	6.2	862.7	1582

operator on each level, plus the ILU factors of each fine-grid block, while AMG requires storage of only the operators, as relaxation on each level is determined entirely by these operators. The ability of AMG to run as an iteration by itself, instead of as a preconditioner for a Krylov subspace method, also reduces its memory overhead, although the ARMS preconditioners used here could be modified for use with CG instead of GMRES, ameliorating this effect. Overall solve times and iteration counts are also much lower for the AMG approach. This is not surprising, as multigrid techniques are largely focused on solving PDE-based problems such as these. The ARMS approach, on the other hand, is motivated by purely algebraic consideration, and, so, we expect it to not perform as well as AMG on PDE-based problems, but to exhibit robustness that the AMG approach may not have.

4.2.3. General sparse symmetric systems. To further test the robustness of the ARMS approach, we consider a subset of the matrices from the test set of [18, 19]. As we are interested primarily in matrices that are not naturally diagonally dominant, we consider only those symmetric and positive-definite matrices for which full data is available (i.e., we do not consider matrices for which only a nonzero pattern is given). In order to run all tests on a single machine (dual-processor Intel Xeon 3.0 GHz with 2 GBytes of RAM), we test all problems with fewer than 3 million nonzero entries. By using swap space, convergence results for one problem with approximately 3.5 million nonzero entries, `oilpan`, are also obtained.

Results in Table 8 show the performance of ARMS-preconditioned GMRES for these matrices. As a first test, we take the parameters for ARMS to allow significant fill in the preconditioner, B, thus looking at the case where accuracy in computing B is not a significant factor in ARMS performance. Diagonal-dominance parameter θ was chosen to be 0.55, as in the PDE-based problems above. The ILU drop tolerance, α, was taken to be 0.01, and further dropping is done so that no row of the approximate L factor has more than 5 times the average number of nonzeros per row of A_{ff} in it. A maximum of 10 levels is allowed, with the coarsest-grid operator computed using

ILUTP with $\alpha = 0.0001$ and fill per row restricted to 20 times the average number of nonzeros per row of the coarsest-grid operator. ILUTP is a pivoting variant of the ILUT algorithm discussed above [31]; elements in the elimination are again filtered by size and fill per row, but a column pivoting step is added for each row to improve the stability of the factorization. With these parameters, ARMS failed to reduce the norm of the residual by a relative factor of 10^6 before memory limitations were reached for only three of the twenty-two problems.

Table 8 reports the complexities (fill factors), c_B, of the resulting preconditioners, setup and iteration times (in seconds), and number of iterations needed to reduce the norm of the residual by a relative factor of 10^6 for the nineteen problems that did converge. As these problems arise from many different applications, it is difficult to make comparisons across different problems, in particular, between smaller and larger problems. We do see quick convergence with low preconditioner complexities for a number of problems, however. For the problems with slower convergence, those with low preconditioner complexities allow many iterations within a short period of CPU time, making the overall time to solution acceptable. For a few of the problems with preconditioner complexities greater than 3, solution was still possible before running into memory limitations. The three problems for which convergence was not obtained, `bcsstk36` ($nnz = 1143140$, $c_B = 10.63$), `msc23052` ($nnz = 1154814$, $c_B = 9.83$), and `nasasrb` ($nnz = 2677324$, $c_B = 4.80$), the large preconditioner complexities coupled with the large initial problem sizes required termination of the iterations before convergence. Slow but steady convergence was seen for `msc23052` and `nasasrb` before iterations were stopped. Only iteration on `bcsstk36` was ineffective in reducing the residual substantially in 6100 iterations before memory limitations were reached.

Experiments with larger drop tolerances (e.g., $\alpha = 0.05$ or $\alpha = 0.1$) show good decreases in preconditioner complexity, but poor performance of the resulting preconditioners. Thus, in Table 9, we look to reduce the preconditioner complexities by allowing each row of L to have only twice as many nonzeros as an average row of A_{ff}, by allowing more levels (up to 50), and by using a smaller $\theta = 0.51$ in the resulting preconditioners. For some problems, these results are equivalent to those in Table 8, up to measurement errors in the times reported. For the problems with large preconditioner complexities in Table 8, the preconditioner complexities are now lower (although still possibly large). This reduction of complexity, however, is (as expected) correlated with an increase in iteration counts and overall time-to-solution (although this is not always the case; cf. `vanbody` and `cvxbqp1`). Setup and solution times are comparable for the problems with fewer than one million nonzeros, although, in some cases, notable growth in c_B and setup time are also seen. For the larger problems, results are slightly mixed. A few more iterations are needed for `cfd1` and `oilpan`, resulting in slightly higher solution times, whereas a much lower preconditioner complexity results in lower setup and solution times for `vanbody`. The lower preconditioner complexities for `bcsstk36` and `msc23052` result in sufficient residual reduction to meet the convergence criteria before available memory (including swap space) is filled. A slightly lower preconditioner complexity for `nasasrb`, 4.36, was obtained for these parameters, but memory considerations still forced early termination of the run, with a residual reduction by a relative factor of approximately 3.35×10^4 in 2500 iterations.

4.2.4. A comparison with AMG and ILUTP. As discussed in section 4.2.2, AMG is expected to perform better than ARMS in some cases (particularly for ma-

TABLE 9
Performance of multilevel ARMS with symmetric ILU-based coarsening on test matrices with $\theta = 0.51$, drop tolerance of 0.01, average fill factor of 2, and a maximum of 50 levels of coarsening.

Name	n	nnz	c_B	t_{setup}	t_{solve}	# iters.
bodyy4	17546	121938	1.01	0.06	0.07	6
bodyy5	18589	129281	0.99	0.09	0.05	4
bodyy6	19366	134748	0.98	0.1	0.04	3
bcsstk18	11948	149090	1.16	0.1	2.1	165
t2dah_a	11445	176117	1.19	0.2	0.9	81
obstclae	40000	197608	8.40	3.9	0.1	3
jnlbrng1	40000	199200	5.10	1.3	0.1	4
minsurfo	40806	203622	3.29	0.4	0.1	6
bcsstk25	15439	252241	1.24	0.3	3.1	162
cvxbqp1	50000	349968	4.02	1.8	642.0	2018
bcsstk17	10974	428650	1.90	0.9	27.4	705
t3dl_a	20360	509866	1.33	0.5	25.5	527
gridgena	48962	512084	1.32	0.5	2.9	81
finan512	74752	596992	1.33	0.5	0.2	5
gyro_k	17361	1021159	1.12	1.0	19.9	398
bcsstk36	23052	1143140	5.79	21.6	2301.5	5648
msc23052	23052	1154814	5.72	22.4	2403.4	5754
msc10848	10848	1229778	1.37	2.3	21.9	433
cfd1	70656	1828364	2.99	8.2	515.6	1198
vanbody	47072	2336898	3.51	19.6	460.8	1413
oilpan	73752	3597188	1.33	4.6	979.1	1694

trices arising from discretization of elliptic PDEs), but ARMS is generally expected to be more robust. That is, we expect that the loss in efficiency of ARMS for certain problems will be offset by more consistent performance on a general collection of problems, such as those in Tables 8 and 9. To test this expectation, Table 10 gives the performance of classical AMG on the general problems of the previous section. Here, the two-stage coarsening algorithm using greedy coarsening as a first pass with $\theta = 0.55$ and strength threshold $\beta = 0.3$ is used to select the coarse grid and interpolation stencil, while the classical (Ruge–Stüben) algorithm is used to define the interpolation weights. A single full-grid sweep of Gauss–Seidel relaxation is used on each level descending and ascending the V-cycle. To ensure the multigrid hierarchy fits in available memory, 10 sweeps of relaxation are used to "solve" the coarsest-grid problem, instead of computing the (sometimes large, dense) LU factorization of the coarsest-grid operator.

Comparing the AMG and ARMS results for these problems, we see that AMG results in a faster overall time to solution for only five problems. For problems `obstclae`, `jnlbrng1`, and `minsurfo`, the setup time for AMG is one-third faster and solve times the same as those in Table 8. For these three problems, the AMG operator complexities are also close to the ARMS preconditioner complexities, reflecting similar overall memory costs, especially as the AMG operator complexity includes the storage for the original fine-scale operator, not counted in the ARMS fill factors. AMG outperforms ARMS on matrix `cvxbqp1`, although this really reflects better, but still bad, performance. The low AMG operator complexity for this problem allows AMG to converge in less time than ARMS but, with over ten thousand iterations required to reduce the residual by a relative factor of 10^6, AMG performance can hardly be called acceptable. Finally, the reduced storage costs of AMG compared to ARMS-preconditioned GMRES allows AMG to converge on the `nasasrb` matrix, although with a large operator complexity and many iterations.

TABLE 10
Performance of multilevel AMG V(1,1) cycles on test matrices with full-grid Gauss–Seidel relaxation, two-stage greedy coarsening using $\theta = 0.55$ and $\beta = 0.3$, a maximum of 20 levels of coarsening, with the coarsest-grid problem solved by 10 sweeps of relaxation.

Name	n	nnz	# levels	c_G	c_A	t_{setup}	t_{solve}	# iters.
bodyy4	17546	121938	8	1.50	1.44	0.03	2.8	363
bodyy5	18589	129281	8	1.57	1.51	0.04	15.8	1851
bodyy6	19366	134748	7	1.61	1.55	0.04	74.7	8385
bcsstk18	11948	149090	20	2.49	5.51	0.2	11.0	497
t2dah_a	11445	176117	20	5.11	6.89	0.4	324.3	10348
obstclae	40000	197608	11	1.73	2.47	0.2	0.08	3
jnlbrng1	40000	199200	13	2.10	3.45	0.2	0.1	4
minsurfo	40806	203622	13	1.90	2.93	0.2	0.1	4
bcsstk25	15439	252241	20	3.42	9.07	0.6	9.4	168
cvxbqp1	50000	349968	4	1.81	1.38	0.2	287.7	10694
bcsstk17	10974	428650	20	4.57	13.46	1.4	525.9	4290
gridgena	48962	512084	16	1.80	3.13	0.6	13.9	274
t3dl_a	20360	509866	20	3.69	6.31	0.8	587.9	7333
finan512	74752	596992	12	1.74	2.50	0.4	0.4	6
gyro_k	17361	1021159	20	4.08	8.17	2.4	172.3	946
bcsstk36	23052	1143140	20	6.87	26.03	7.1	2828.0	4486
msc23052	23052	1154814	20	6.95	28.00	8.5	2312.4	3357
msc10848	10848	1229778	20	8.07	20.05	6.0	10375.7	19829
cfd1	70656	1828364	20	4.52	80.16	52.1	4318.7	1426
vanbody	47072	2336898	20	6.15	21.87	13.5	2846.3	2526
nasasrb	54870	2677324	20	4.27	13.02	9.6	3586.4	4711
oilpan	73752	3597188	20	2.80	3.66	3.3	1314.2	3938

For the remaining problems, AMG is generally worse in terms of both storage factors (or cost per iteration) and iteration count. The operator complexities for AMG are often larger than ten times the storage of the fine-grid operator, and as much as eighty times this storage is needed. Iteration counts are over ten thousand for three problems and are often many times those used for ARMS. Put simply, there is no reason to expect that the assumptions made in classical AMG on the slow-to-converge modes of Gauss–Seidel relaxation are valid for these general problems and, as a result, the AMG interpolation scheme may provide an arbitrarily bad complement to this relaxation. While the parameters of the AMG methodology may be tuned to improve performance on any one of these problems, the results in Table 10 clearly show the lack of robustness of AMG when used as a black-box solver, in comparison to the relatively good results seen in Tables 8 and 9.

For comparison with a single-level preconditioning strategy, we also look at the performance of (nonsymmetrized) ILUTP on these same test problems in Table 11. In order to make fair comparisons between the multilevel ILU strategy in ARMS and that in ILUTP, we choose the parameters of ILUTP, notably α, the drop tolerance, such that the overall preconditioner complexity is roughly the same as those in Table 9. We limit the number of nonzero entries in each row of L and U^T to be no more than 20 times the average number of nonzeros per row of A. This is done so that the choice of α is used as the primary control over the preconditioner complexities; α is varied in increments of 0.0025 to give the operator complexity closest to those of Table 9. We also report the preconditioner complexities, setup and solve times, and number of iterations to reduce the residual by a relative factor of 10^6. For each problem, we allow at most twice the number of iterations that ARMS-preconditioned GMRES took to converge in Table 9, unless that would require too much memory (i.e., for the problems where swapping was required), in which case the limit is the number of iterations that

TABLE 11
Performance of ILUTP-preconditioner GMRES on test problems.

Name	n	nnz	α	c_B	t_{setup}	t_{solve}	# iters.	Reduction
bodyy4	17546	121938	0.03	1.00	0.5	0.03	6	-
bodyy5	18589	129281	0.0175	1.00	0.6	0.04	7	-
bodyy6	19366	134748	0.0125	0.97	0.7	0.03	7	-
bcsstk18	11948	149090	0.03	1.17	0.1	4.5	330	1.79×10^4
t2dah_a	11445	176117	0.1975	1.18	0.2	1.5	162	1.24×10^1
obstclae	40000	197608	0.0025	4.55	3.4	0.09	4	-
jnlbrng1	40000	199200	0.0025	4.39	3.5	0.1	6	-
minsurfo	40806	203622	0.0025	3.54	3.4	0.09	5	-
bcsstk25	15439	252241	0.04	1.21	0.2	3.2	216	-
cvxbqp1	50000	349968	0.0225	4.14	4.4	594.5	2018	2.29×10^2
bcsstk17	10974	428650	0.125	1.91	0.8	70.4	1410	5.32×10^2
t3dl_a	20360	509866	0.03	1.36	0.9	7.4	275	-
gridgena	48962	512084	0.025	1.32	1.5	4.0	118	-
finan512	74752	596992	0.0075	1.34	2.4	0.1	4	-
gyro_k	17361	1021159	0.0725	1.16	0.9	44.4	796	1.04×10^5
bcsstk36	23052	1143140	0.21	5.70	54.0	2111.8	5648	3.67
msc23052	23052	1154814	0.18	5.70	82.3	2172.8	5754	7.54
msc10848	10848	1229778	0.05	1.38	2.4	9.0	244	-
cfd1	70656	1828364	0.0175	2.97	12.0	44.9	297	-
vanbody	47072	2336898	0.3025	3.60	12.9	1266.4	2826	9.09×10^1
oilpan	73752	3597188	0.2	1.25	11.0	908.4	1694	3.58

ARMS-preconditioned GMRES took to converge. The residual reduction factors for the problems that did not converge are reported in the final column of Table 11.

Comparing the results from Tables 9 and 11, we see that the ARMS-preconditioning approach often, but not always, pays off relative to ILUTP-preconditioned GMRES. In terms of total time to solution, the two strategies tie for a single problem (`bcsstk25`), the ARMS-preconditioners are faster for sixteen problems, while ILUTP yields faster solvers for four problems. In terms of number of iterations (closely related to iteration time as the preconditioner complexities are approximately equal for each problem), ILUTP-preconditioned GMRES required fewer iterations for five problems, with one tie and the ARMS preconditioner requiring fewer iterations for fifteen problems. Of these fifteen problems, ILUTP converged before iteration and memory limits were reached for only six problems. For the remaining nine problems, ARMS was significantly better than ILUTP. On three of the test matrices, not only did ILUTP not converge in double the number of iterations that ARMS required, but it did not succeed in reducing the norm of the residual by a factor of even half that required for convergence. On the two most difficult problems for which ARMS was successful, `bcsstk36` and `msc23052`, ILUTP was unable to reduce the residual by a factor of 10 in the same number of iterations that ARMS required for a reduction by a relative factor of 10^6. For the `nasasrb` matrix that caused ARMS the most difficulty, ILUTP reduced the residual by a factor of 2.02 in the same number of iterations it took ARMS to reduce the residual by a factor of 3.4×10^4.

5. Conclusions. Motivated by a theoretical analysis, a new coarsening algorithm for algebraic multigrid and multilevel solvers is proposed. This greedy algorithm partitions the degrees of freedom so that the fine-grid block satisfies some diagonal dominance properties, and, thus, can be easily treated by relaxation or an ILU factorization. Numerical results show that two-level solvers based on this technique exhibit optimal convergence properties. Multilevel solvers based on simple grid-transfer oper-

ators perform well for simple model problems but break down when certain theoretical assumptions are violated. When used in combination with coarsening ideas from the usual AMG and ARMS methodologies, the resulting algorithms achieve improved robustness, demonstrated on both PDE-based problems and test problems from a variety of sources.

Acknowledgment. The authors would like to thank the anonymous referees for their comments that helped to improve the quality of this paper.

REFERENCES

[1] O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, Cambridge, UK, 1994.
[2] N. S. Bakhvalov, *On the convergence of a relaxation method under natural constraints on an elliptic operator*, Z. Vycisl. Mat. i. Mat. Fiz., 6 (1966), pp. 861–883.
[3] R. E. Bank, *Hierarchical bases and the finite element method*, in Acta Numerica, Acta Numer. 5, Cambridge University Press, Cambridge, UK, 1996, pp. 1–43.
[4] A. Brandt, S. F. McCormick, and J. W. Ruge, *Algebraic multigrid (AMG) for sparse matrix equations*, in Sparsity and Its Applications, D. J. Evans, ed., Cambridge University Press, Cambridge, UK, 1984.
[5] A. Brandt, *Algebraic multigrid theory: The symmetric case*, Appl. Math. Comput., 19 (1986), pp. 23–56.
[6] A. Brandt, *General highly accurate algebraic coarsening*, Electron. Trans. Numer. Anal., 10 (2000), pp. 1–20.
[7] J. Brannick, M. Brezina, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge, *An energy-based AMG coarsening strategy*, Numer. Linear Algebra Appl., 13 (2006), pp. 133–148.
[8] J. Brannick and R. Falgout, *Compatible relaxation and coarsening in algebraic multigrid*, in preparation.
[9] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge, *Adaptive algebraic multigrid*, SIAM J. Sci. Comp., 27 (2006), pp. 1261–1286.
[10] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, 2nd ed., SIAM, Philadelphia, 2000.
[11] O. Bröker, *Parallel Multigrid Methods using Sparse Approximate Inverses*, Ph.D. thesis, Swiss Federal Institute of Technology Zurich, Zurich, Switzerland, 2003.
[12] A. J. Cleary, R. D. Falgout, V. E. Henson, J. E. Jones, T. A. Manteuffel, S. F. McCormick, G. N. Miranda, and J. W. Ruge, *Robustness and scalability of algebraic multigrid*, SIAM J. Sci. Comput., 21 (2000), pp. 1886–1908.
[13] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, The MIT Electrical Engineering and Computer Science Series, MIT Press, Cambridge, MA, 1990.
[14] H. De Sterck, U. M. Yang, and J. J. Heys, *Reducing complexity in parallel algebraic multigrid preconditioners*, SIAM J. Matrix Anal. Appl., 27 (2006), pp. 1019–1039. Also available as LLNL technical report UCRL-JRNL-206780.
[15] R. D. Falgout and P. S. Vassilevski, *On generalizing the alebraic multigrid framework*, SIAM J. Numer. Anal., 42 (2004), pp. 1669–1693. Also available as LLNL technical report UCRL-JC-150807.
[16] R. P. Fedorenko, *On the speed of convergence of an iteration process*, Z. Vycisl. Mat. i. Mat. Fiz., 4 (1964), pp. 559–564 (in Russian).
[17] R. P. Fedorenko, *On the speed of convergence of an iteration process*, U.S.S.R. Comput. Math. and Math. Phys., 4 (1964), pp. 227–235.
[18] N. I. M. Gould and J. A. Scott, *Complete results for a numerical evaluation of HSL packages for the direct solution of large sparse, symmetric linear systems of equations*, Numerical Analysis Internal Report 2003-2, Rutherford Appleton Laboratory, Chilton, Didcot Oxfordshire, UK, 2003.
[19] N. I. M. Gould and J. A. Scott, *A numerical evaluation of HSL packages for the direct solution of large sparse, symmetric linear systems of equations*, ACM Trans. Math. Software, 30 (2004), pp. 300–325.
[20] W. Hackbusch, *Multigrid Methods and Applications*, Springer Ser. Comput. Math. 4, Springer-Verlag, Berlin, 1985.
[21] V. E. Henson and U. M. Yang, *BoomerAMG: A parallel algebraic multigrid solver and preconditioner*, Appl. Numer. Math., 41 (2002), pp. 155–177.

[22] L. Little, Y. Saad, and L. Smoch, *Block LU preconditioners for symmetric and nonsymmetric saddle point problems*, SIAM J. Sci. Comput., 25 (2003), pp. 729–748.
[23] O. Livne, *Coarsening by compatible relaxation*, Numer. Linear Algebra Appl., 11 (2004), pp. 205–227.
[24] S. MacLachlan, T. Manteuffel, and S. McCormick, *Adaptive reduction-based AMG*, Numer. Linear Algebra Appl., 13 (2006), pp. 599–620.
[25] S. MacLachlan and Y. Saad, *Greedy coarsening strategies for non-symmetric problems*, Tech. rep. umsi-2006-58, Minnesota Supercomputing Institute, University of Minnesota, Minneapolis, MN, 2006.
[26] S. F. McCormick and J. W. Ruge, *Multigrid methods for variational problems*, SIAM J. Numer. Anal., 19 (1982), pp. 924–929.
[27] Y. Notay, *Using approximate inverses in algebraic multilevel methods*, Numer. Math., 80 (1998), pp. 397–417.
[28] Y. Notay, *Algebraic multigrid and algebraic multilevel methods: A theoretical comparison*, Numer. Linear Algebra Appl., 12 (2005), pp. 419–451.
[29] J. W. Ruge and K. Stüben, *Algebraic multigrid (AMG)*, in Multigrid Methods, S. F. McCormick, ed., Frontiers Appl. Math. 3, SIAM, Philadelphia, 1987, pp. 73–130.
[30] Y. Saad and B. Suchomel, *ARMS: An algebraic recursive multilevel solver for general sparse linear systems*, Numer. Linear Algebra Appl., 9 (2002), pp. 359–378.
[31] Y. Saad, *ILUT: A dual threshold incomplete ILU factorization*, Numer. Linear Algebra Appl., 1 (1994), pp. 387–402.
[32] Y. Saad, *Multilevel ILU with reorderings for diagonal dominance*, SIAM J. Sci. Comput., 27 (2005), pp. 1032–1057.
[33] R. Southwell, *Stress calculation in frameworks by a method of systematic relaxation of constraints.* I, II, Proc. Roy. Soc. London. Ser. A., 151 (1935), pp. 56–95.
[34] K. Stüben, *An introduction to algebraic multigrid*, in Multigrid, U. Trottenberg, C. Oosterlee, and A. Schüller, Academic Press, San Diego, CA, 2001, pp. 413–528.
[35] M. Tismenetsky, *A new preconditioning technique for solving large sparse linear systems*, Linear Algebra Appl., 154–156 (1991), pp. 331–353.
[36] W. L. Wan, T. F. Chan, and B. Smith, *An energy-minimizing basis interpolation for robust multigrid methods*, SIAM J. Sci. Comput., 21 (2000), pp. 1632–1649.
[37] J. Xu and L. T. Zikatanov, *On an energy minimizing basis in algebraic multigrid methods*, Computing and Visualization in Science, 7 (2004), pp. 121–127.
[38] D. Zuckerman, *On unapproximable versions of NP-complete problems*, SIAM J. Comput., 25 (1996), pp. 1293–1304.

SIAM J. SCI. COMPUT.
Vol. 29, No. 5, pp. 1854–1875
© 2007 Society for Industrial and Applied Mathematics

A TRUST REGION DIRECT CONSTRAINED MINIMIZATION ALGORITHM FOR THE KOHN–SHAM EQUATION*

CHAO YANG†, JUAN C. MEZA†, AND LIN-WANG WANG†

Abstract. The self-consistent field (SCF) iteration, widely used for computing the ground state energy and the corresponding single particle wave functions associated with a many-electron atomistic system, is viewed in this paper as an optimization procedure that minimizes the Kohn–Sham (KS) total energy indirectly by minimizing a sequence of quadratic surrogate functions. We point out the similarity and difference between the total energy and the surrogate, and show how the SCF iteration can fail when the minimizer of the surrogate produces an increase in the KS total energy. A trust region technique is introduced as a way to restrict the update of the wave functions within a small neighborhood of an approximate solution at which the gradient of the total energy agrees with that of the surrogate. The use of trust regions in SCF is not new. However, it has been observed that directly applying a trust region-based SCF (TRSCF) to the KS total energy often leads to slow convergence. We propose to use TRSCF within a direct constrained minimization (DCM) algorithm we developed in [*J. Comput. Phys.*, 217 (2006), pp. 709–721]. The key ingredients of the DCM algorithm involve projecting the total energy function into a sequence of subspaces of small dimensions and seeking the minimizer of the total energy function within each subspace. The minimizer of a subspace energy function, which is computed by the TRSCF, not only provides a search direction along which the KS total energy function decreases, but also gives an optimal "step length" that yields a sufficient decrease in total energy. A numerical example is provided to demonstrate that the combination of TRSCF and DCM is more efficient than SCF.

Key words. nonlinear eigenvalue problem, Kohn–Sham total energy, constrained optimization, trust region

AMS subject classifications. 15A18, 65K10, 65F15

DOI. 10.1137/060661442

1. Introduction. The self-consistent field (SCF) iteration is the most widely used procedure for computing the ground state energy and the corresponding single particle wave functions associated with a many-electron atomistic system. The procedure is often viewed as a fixed point iteration in which a sequence of linear eigenvalue problems is solved approximately. The approximate eigenvectors computed in the jth iteration are used to construct the potential component of the Kohn–Sham (KS) matrix Hamiltonian required in the $(j+1)$th iteration. Convergence is reached when the difference between Hamiltonians formed in two consecutive iterations becomes negligible. At this point, the eigenvalues and eigenvectors of the Hamiltonian become self-consistent and the KS total energy function associated with the system reaches the global minimum.

It has long been observed that the simplest form of SCF iteration often fails to converge. As such, a number of heuristics have been developed to prevent the

*Received by the editors May 30, 2006; accepted for publication (in revised form) October 24, 2006; published electronically September 28, 2007. This work was supported by the Director, Office of Science, Division of Mathematical, Information, and Computational Sciences of the U.S. Department of Energy under contract DE-AC03-76SF00098. This work was performed by an employee of the U.S. Government or under U.S. Government contract. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights.

http://www.siam.org/journals/sisc/29-5/66144.html

†Computational Research Division, Lawerence Berkeley National Laboratory, Berkeley, CA 94720 (CYang@lbl.gov, JCMeza@lbl.gov, LWWang@lbl.gov).

SCF from diverging. These heuristics often involve combining either the potential or the charge densities computed in the previous SCF iterations to construct a new Hamiltonian so that the lack of self-consistency can be minimized. These techniques are known either as [illegible] [13, 15] in the material sciences community or as the direct inversion of iterative subspace (DIIS) extrapolation technique [24, 25] in the quantum chemistry community. Although these heuristics work remarkably well in stabilizing the SCF iteration for many problems, they can still fail in others. Even when they work, the reduction in the KS total energy is often not monotonic. Furthermore, no satisfactory theoretical foundation has been established to explain why charge mixing and DIIS work and under what circumstances they can fail. As a result, the convergence of SCF is often unpredictable for large atomistic systems with small (valence/conducting) band gaps.

In this paper, we will examine SCF from an optimization point of view. We view the SCF iteration as an indirect way to minimize the KS total energy function through the minimization of a sequence of quadratic surrogate functions. We point out the similarity and difference between the surrogate and the true objective function and introduce the concept of a [illegible] that can be used to restrict the update of the wave functions within a region in which the gradients associated with the true object and the surrogate differ very little. The trust region technique is a widely used methodology for promoting global convergence in numerical optimization procedures [5, 20]. Trust region-based SCF (TRSCF) iteration has been used in the past in the quantum chemistry community [28, 31], where it is sometimes called [illegible] SCF iteration [28].

By imposing a trust region with an appropriate radius at each SCF iteration, we can show that a monotonic reduction of the KS total energy can be achieved in the SCF procedure. However, it has been observed that for large systems, the use of TRSCF often leads to extremely slow convergence. In [31], the trust region technique is combined with DIIS to accelerate the convergence of SCF. In this paper, we propose a different approach. Instead of applying the trust region technique directly to an SCF iteration, we use it within a direct constrained minimization (DCM) algorithm developed in [36] to generate an effective search direction and step length simultaneously. This scheme is applicable to density functional theory (DFT) calculations in which the number of degrees of freedom in the discretized wave function is much larger than the number of electrons in the atomistic system.

The paper is organized as follows. In section 2, we establish some basic notation required for the discussion of SCF and DCM. In section 3, we examine the SCF iteration from an optimization point of view and provide the motivation for applying the trust region technique in SCF. A TRSCF iteration is presented in section 4. We discuss the use of TRSCF within the DCM algorithm in section 5. A numerical example is provided in section 6 to demonstrate the value of the trust region technique in KS total energy minimization.

2. Background and notation. In this section, we establish the mathematical notation required to describe the SCF and DCM algorithms. We skip the description of the continuous formulation of the KS total energy optimization problem, which has been presented in [36] and many other references [15, 16, 23]. Instead, we will focus on the finite-dimensional version of the problem.

In the following discussion, we will use A^T to denote the transpose of a matrix A, and A^* to denote the complex conjugate of A. A submatrix of A consisting of rows i through j and columns p through q will be denoted by $A(i:j, p:q)$. If the

submatrix contains all rows (columns) of A, it will be denoted by $A(:, p:q)$ (resp., $A(i:j,:)$). The Frobenius norm of a matrix A (defined as the square root of the sum of the absolute squares of all elements in A) is denoted by $\|A\|_F$.

With an appropriate discretization scheme, a single electron wave function can be approximated by a vector $x_i \in \mathbb{C}^n$, where n is the spatial degrees of freedom, i.e., the number of real space grid points. These vectors satisfy the orthonormality constraints

$$x_i^* x_j = \delta_{i,j}, \quad i,j = 1,2,\ldots,k,$$

where k is the number of occupied states. If we let $X = (x_1, x_2, \ldots, x_k)$, the matrix

$$D(X) = XX^* \tag{2.1}$$

is often known as the density matrix, and the charge density associated with the k occupied states can be expressed by

$$\rho(X) = \mathrm{diag}(XX^*), \tag{2.2}$$

where $\mathrm{diag}(A)$ denotes a column vector consisting of diagonal entries of the matrix A.

The KS total energy function consists of several components [23], i.e.,

$$E_{total}(X) = E_{kinetic}(X) + E_{ion}(X) + E_H(X) + E_{XC}(X),$$

where $E_{kinetic}$ is the kinetic energy and E_{ion}, E_H, and E_{XC} are potential energies induced by the electron-ion interaction (ionic potential), the electron-electron interaction (Hartree potential), and the exchange correlation potential, respectively.

Let $L \in \mathbb{C}^{n\times n}$ be a Hermitian matrix representing a discretized Laplacian operator. The kinetic energy is then defined by [23]

$$E_{kinetic}(X) = \frac{1}{2}\mathrm{trace}(X^* L X).$$

The ionic potential energy consists of a local and a nonlocal term. If we let D_{ion} be a real diagonal matrix representing a discretized local ionic potential function, then the local ionic potential energy is defined by [23]

$$E_{ion(local)}(X) = \mathrm{trace}(X^* D_{ion} X).$$

The contribution from the nonlocal ionic potential is defined by [23]

$$E_{ion(nonlocal)}(X) = \sum_i \sum_\ell \left| x_i^* w_\ell \right|^2,$$

where w_ℓ represents a discretized pseudopotential reference projection function.

In practice, appropriate boundary conditions are imposed so that L is nonsingular. If we use S to denote the inverse of the discrete Laplacian operator, then the Hartree potential energy, which is used to model the classical electrostatic average interaction between electrons, can be expressed [23] by

$$E_H(X) = \frac{1}{2}\rho(X)^T S\, \rho(X).$$

The exchange correlation function ϵ_{xc} is used to model the nonclassical interaction between electrons. The potential energy induced by this function is defined by [23]

$$E_{XC}(X) = e^T\big(\epsilon_{xc}[\rho(X)]\big),$$

where e is a column vector of ones.

Using the notation established above, we can state the KS total energy minimization problem as

$$\begin{aligned} \min \quad & E_{total}(X) \\ \text{s.t.} \quad & X^*X = I_k, \end{aligned} \tag{2.3}$$

where I_k denotes a $k \times k$ identity matrix.

The Lagrangian associated with (2.3) is

$$\mathcal{L}(X) = E_{total}(X) - \operatorname{trace}\big[\Lambda^T(X^*X - I_k)\big], \tag{2.4}$$

where Λ is a $k \times k$ matrix containing the Lagrange multipliers associated with the constraints specified by $X^*X = I_k$ [20].

The solution to (2.3) must satisfy the first order necessary conditions

$$\begin{aligned} \nabla_X \mathcal{L}(X) &= 0, \\ X^*X &= I_k. \end{aligned} \tag{2.5}$$

Here, $\nabla_X \mathcal{L}$ represents an $n \times k$ matrix whose (i,j)th entry is the partial derivative of $\mathcal{L}$ with respect to the (i,j)th entry of X.

It is easy to verify that

$$\nabla_X E_{kinetic} = \frac{1}{2}LX, \tag{2.6}$$

$$\nabla_X E_{ion(local)} = D_{ion}X, \tag{2.7}$$

$$\nabla_X E_{ion(nonlocal)} = \sum_{\ell}(w_\ell w_\ell^*)X, \tag{2.8}$$

$$\nabla_X E_H = \operatorname{Diag}(S\rho(X))X, \tag{2.9}$$

$$\nabla_X E_{XC} = \operatorname{Diag}(\mu_{xc}(\rho))X, \tag{2.10}$$

where

$$\mu_{xc}(\omega) \equiv \frac{d\epsilon_{xc}(\omega)}{d\omega}$$

is the derivative of the exchange-correlation function. Here the notation $\operatorname{Diag}(\rho)$ represents a diagonal matrix whose diagonal is determined by the vector ρ, and we have scaled (2.6)–(2.10) by $1/2$ to be consistent with the convention used in the electronic structure community.

Substituting (2.6)–(2.10) into (2.5), we obtain the KS equation

$$H(X)X = X\Lambda_k, \qquad X^*X = I_k, \tag{2.11}$$

where

$$H(X) = \left[\frac{1}{2}L + D_{ion} + \sum_{\ell} w_\ell w_\ell^* + \operatorname{Diag}(S\rho) + \operatorname{Diag}(\mu_{xc}(\rho))\right]. \tag{2.12}$$

Because the vector ρ in (2.12) depends on X, the eigenvalue problem defined by (2.11) is nonlinear. Also note that the solution to (2.3) is not unique. If X is a solution, then XQ is also a solution for any $Q \in \mathbb{C}^{k\times k}$ such that $Q^*Q = I_k$. That is, the solution to the constrained minimization problem, or, equivalently, the nonlinear equations defined by (2.11), is a k-dimensional invariant subspace in $\mathbb{C}^n$ rather than a specific matrix. In particular, Q can be chosen such that Λ_k is diagonal. In this case, X consists of k KS eigenvectors associated with the k smallest eigenvalues of $H(X)$.

3. The optimization view of SCF. Although the nonlinear eigenvalue problem defined by (2.11) may appear easier to solve than (2.3) because of its close connection to a linear eigenvalue problem, there is yet no robust and efficient general-purpose solver for this type of problem with guaranteed convergence. The most widely used technique for solving (2.11) is to reduce it to a sequence of linear eigenvalue problems that can be solved by many numerical linear algebra software packages such as LAPACK [1] or ARPACK [18]. This approach is often known as the SCF iteration. For completeness, we outline the major steps of the basic version of the SCF iteration procedure in Figure 3.1.

SCF ITERATION

Input: The matrices L, D_{ion}, S, and the vectors w_ℓ, $\ell = 1, 2, \ldots$. The derivative of the exchange-correlation function $\mu_{xc}(x)$; an initial guess $X^{(0)}$ for the optimal wave function $X \in \mathbb{C}^{n\times m}$;

Output: $X \in \mathbb{C}^{n\times m}$ such that $X^*X = I_m$ and $E_{tot}(X)$ is minimized, where $E_{tot}(X)$ is defined by (2.6)–(2.10).

1. for $i = 1, 2, \ldots$ until convergence
2. Form $H^{(i)} = H(X^{(i-1)})$;
3. Compute $X^{(i)}$ such that $H^{(i)}X^{(i)} = X^{(i)}\Lambda^{(i)}$, and $\Lambda^{(i)}$ contains the k smallest eigenvalues of $H^{(i)}$;
4. end for

FIG. 3.1. *The SCF iteration.*

Depending on the discretization scheme used, it may not be necessary or possible to form the Hamiltonian $H^{(i)}$ explicitly in the SCF calculation. This is particularly true when the continuous problem is discretized by a spectral method using a plane wave basis. In that case, $H^{(i)}$ exists only in the form of a matrix vector multiplication procedure, and it is not feasible to solve the linear eigenvalue problem $H^{(i)}X^{(i)} = X^{(i)}\Lambda^{(i)}$ by using subroutines provided in LAPACK [1]. Iterative methods such as the Lanczos [17, 18, 35], preconditioned conjugate gradient [14, 12, 27], the Jacobi–Davidson type of method [6, 21, 26, 29], or multigrid accelerated Rayleigh-quotient iterations [3, 8, 11] are often used in this setting.

Because computing the k smallest eigenpairs of the discrete Hamiltonian $H^{(i)}$ is equivalent to solving the trace minimization problem

$$\begin{aligned} \min \quad & q(X) = \tfrac{1}{2}\mathrm{trace}(X^*H^{(i)}X) \\ \text{s.t.} \quad & X^*X = I_k, \end{aligned} \tag{3.1}$$

the SCF iteration can be viewed as an iterative procedure that minimizes the total energy E_{total} indirectly by minimizing a sequence of quadratic functions of

the form (3.1). Note that the discrete Hamiltonian $H^{(i)}$ is formed by using a fixed set of wavefunctions, $X^{(i-1)}$, computed in the previous SCF iteration.

At $X = X^{(i-1)}$, the gradient of the quadratic surrogate agrees with that of E_{total}, i.e.,

$$\nabla E_{total}(X)_{|X=X^{(i-1)}} = \nabla q(X)_{|X=X^{(i-1)}} = H^{(i)}X^{(i-1)}, \tag{3.2}$$

even though $E_{total}(X^{(i-1)})$ may be completely different from $q(X^{(i-1)})$ in general.

The gradient match between $E_{total}(X)$ and $q(X)$ at $X = X^{(i-1)}$ should come as no surprise because such a match is achieved by construction. A direct consequence of (3.2) is that, at least within a small neighborhood of $X^{(i-1)}$, a reduction of $q(X)$ is likely to result in a reduction in $E_{total}(X)$. However, if one moves too far away from $X^{(i-1)}$, $E_{total}(X)$ may actually increase because gradient matching does not hold in general when X is away from $X^{(i-1)}$.

This optimization view of the SCF iteration suggests the danger of minimizing the quadratic surrogate (3.1) by computing the k smallest eigenpairs of $H^{(i)}$. To illustrate this possibility, we will now give a concrete two-dimensional example below to demonstrate how SCF may fail to converge. Consider a simplified total energy function of the form

$$E_{total}(x) = \frac{1}{2}x^T Lx + \frac{\alpha}{4}\rho(x)^T L^{-1}\rho(x), \tag{3.3}$$

where

$$L = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \quad \text{and} \quad \rho(x) = \begin{pmatrix} x_1^2 \\ x_2^2 \end{pmatrix}$$

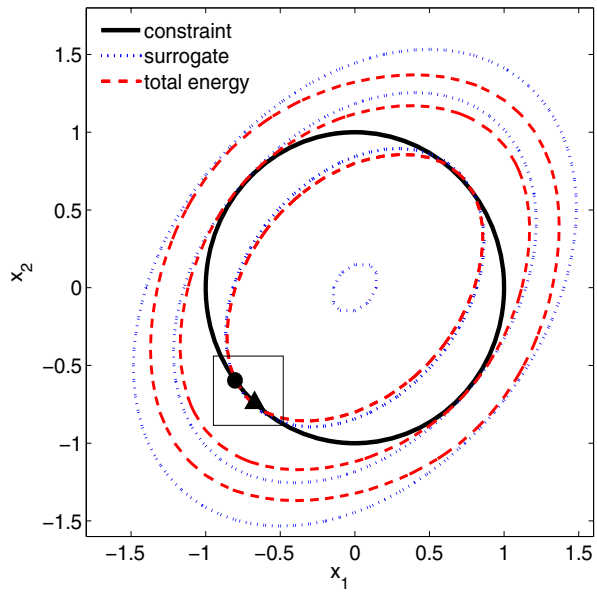
is used as our objective function, to be minimized subject to the constraint

$$x_1^2 + x_2^2 = 1. \tag{3.4}$$

Note that we deliberately ignored the ionic and exchange-correlation potentials to make this example simple enough for testing. As a result, both the total energy and the quadratic surrogate functions are convex, which may not necessarily be the case in general. Nonetheless, these examples exhibit the local convergence behavior of SCF.

When we set $\alpha = 2$ in (3.3), the simplest version of SCF algorithm shown in Figure 3.1 converges in 10 iterations. In Figure 3.2(a), we plot the contour of (3.3) (dashed contour) as well as that of the quadratic surrogate (dotted contour) constructed at the current approximation $\hat{x} = (\hat{x}_1, \hat{x}_2)^T = (-0.8033, -0.5956)^T$. The position of $\hat{x}$ is marked by a small solid circle in the figure. The minimizer of the quadratic surrogate $\hat{x}_q$ is marked by a small solid triangle. Both $\hat{x}$ and $\hat{x}_q$ lie on the large circle which corresponds to the constraint (3.4).

If we zoom into the box shown in Figure 3.2(a), we can see how the total energy changes as we move from $\hat{x}$ (the small solid circle, where the gradient of $E_{total}(x)$ matches the gradient of the quadratic surrogate) to $\hat{x}_q$ (the solid triangle, at which the dotted elliptical contour becomes tangent to the large circle). Figure 3.2(b) shows that the minimizer of the quadratic surrogate, $\hat{x}_q$, lies on the total energy contour (dashed) line that is above and to the right of the dashed contour line that passes $\hat{x}$. Because the total energy in this example is convex, this implies that moving from the current approximation $\hat{x}$ to $\hat{x}_q$ results in a reduction of the total energy. For reference purposes, we also plot the true minimizer of (3.3) as a small solid square in Figure 3.2(b).



(a)

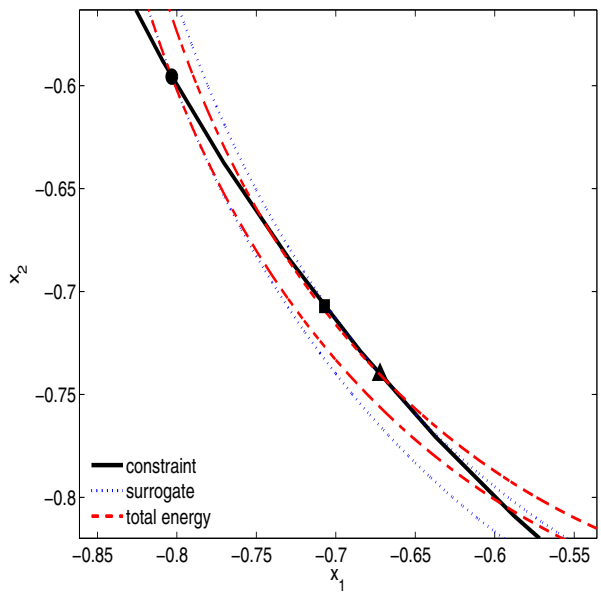
FIG. 3.2. (a) *When $\alpha = 2$ is set in* (3.3)*, the SCF iteration converges. This figure shows the change of $E_{total}(x)$ in a single SCF iteration. The contours of the total energy defined in* (3.3) *and the surrogate $q(x) = \frac{1}{2}x^T(L + \alpha \mathrm{Diag}(L^{-1}\rho(x)))x$ for $\alpha = 2$. The current iterate $\hat{x}$ is marked by the small solid circle at which $\nabla E_{total}(\hat{x}) = \nabla q(\hat{x})$. The minimizer of $q(x)$ is marked by the solid triangle. Both of these points lie on the large circle which defines the orthonormality constraint* (3.4).

When we increase α to 12, the contour of (3.3) becomes less elliptical, as we can see in Figure 3.3(a). Figure 3.3(a) also shows the quadratic surrogate constructed from the current approximation (marked by a small solid circle) $\hat{x} = (\hat{x}_1, \hat{x}_2)^T = (-0.8904,\ -0.4551)^T$, as well as the location of $\hat{x}_q$ (marked by the solid triangle). The small solid square indicates the location of the true minimizer of (3.3) for $\alpha = 12$. All of these three points lie on the orthonormality constraint shown as the large circle in this figure.

As we zoom into the rectangular box shown in Figure 3.3(a), we can see more clearly why a single SCF iteration from $\hat{x}$ leads to an increase in (3.3). In Figure 3.3(b), which provides a zoom-in view, we can clearly see that the dashed contour line that passes through $\hat{x}_q$ lies to the lower left of the curve that passes through $\hat{x}$. This implies an increase in the total energy (3.3) as we move from the current approximation $\hat{x}$ to the minimizer of the surrogate $\hat{x}_q$.

Fortunately, the optimization view of the SCF iteration also suggests at least two ways to improve the convergence of a SCF:

1. develop a better surrogate function,
2. restrict the wave function update in a small neighborhood of the current approximation.

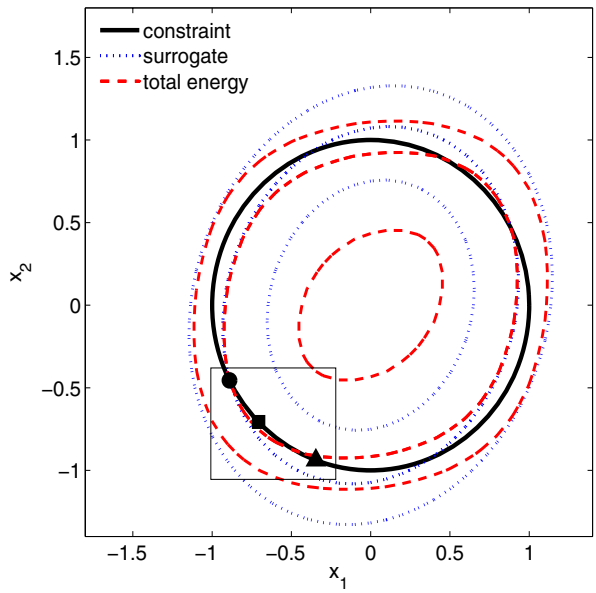


(b)

FIG. 3.2(*continued*). (b) *When* $\alpha = 2$ *is set in* (3.3), *the SCF iteration converges. This figure shows the change of* $E_{total}(x)$ *in a single SCF iteration. A zoom-in view of the boxed area in Figure* 3.2(a). *The minimizer of* $q(x)$ *(the small solid triangle) lies on the inner dashed contour line, indicating a decrease in the total energy. The solid square marks the true minimizer of* E_{total}.

One typical approach is to use the second order Taylor approximation to total energy as the surrogate model. The iterative method based on this quadratic model is the familiar Newton's method. However, due to the high cost associated with Hessian computation, this approach is not practical. Although it is possible to apply a limited memory version of a quasi-Newton method to minimize $E_{total}(X)$ subject to the orthonormality constraint, such an approach, in which only an approximate Hessian is used and updated by taking into account the gradient information computed in previous iterations, is generally not effective because of the large dimensionality of the minimization problem; hence the difficulty of obtaining a good approximation to the Hessian [33].

A technique that is widely used in the current practice of electronic structure calculation is to modify the Hessian of the quadratic surrogate (3.1) so that the [illegible] in KS Hamiltonian $H(X)$ is minimized. The lack of self-consistency in X can be measured in a number of ways. If $X^{(i+1)}$ is the approximate solution to the quadratic minimization problem (3.1), in which H is defined in terms of $X^{(i)}$, then the lack of self-consistency can be directly measured by computing $\delta_H = \|H(X^{(i+1)}) - H(X^{(i)})\|_F$, or simply $\delta_\rho = \|\rho(X^{(i+1)}) - \rho(X^{(i)})\|$, because the change in $H(X)$ is contributed by the last two terms of (2.12), which are both functions of the charge density $\rho(X)$. The minimization of δ_ρ is often achieved approximately by choosing $\rho^{(i+1)}$ to be a linear combination of the charge densities computed in

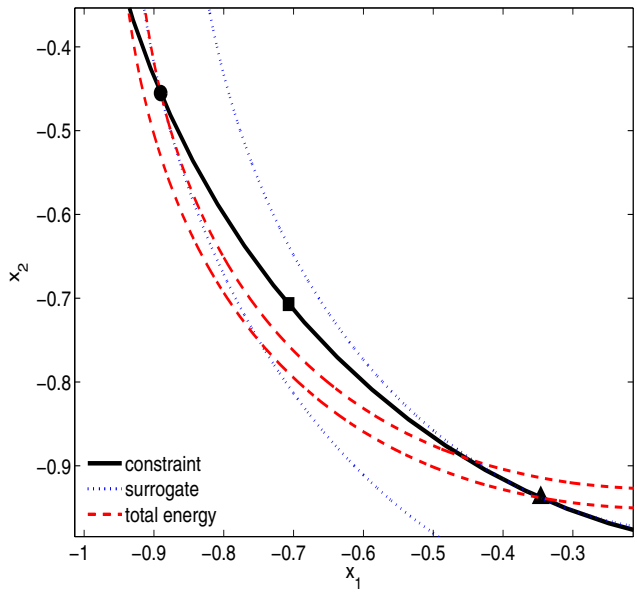


(a)

FIG. 3.3. (a) *When* $\alpha = 12$ *is set in* (3.3)*, the SCF iteration fails to converge. This figure shows the change of* $E_{total}(x)$ *in a single SCF iteration. The contours of the total energy defined in* (3.3) *and the surrogate* $q(x) = \frac{1}{2}x^T(L + \alpha \mathrm{Diag}(L^{-1}\rho(x)))x$ *for* $\alpha = 12$*. The current iterate* $\hat{x}$ *is marked by the small solid circle at which* $\nabla E_{total}(\hat{x}) = \nabla q(\hat{x})$*. The minimizer of* $q(x)$ *is marked by the solid triangle. Both of these points lie on the large circle which defines the orthonormality constraint* (3.4)*.*

the previous SCF iterations and solving an equality constrained quadratic program which returns the "optimal" linear combination. This approach is often referred to as the [illegible] scheme or the DIIS scheme [15, 24, 25]. In the material sciences community, the Pulay mixing scheme is often followed by another procedure in which selected components of the new charge density are modified or scaled. This procedure is sometimes called the [illegible] scheme [13, 15]. These mixing schemes work remarkably well for many problems but can fail for others. To our knowledge, no theoretical analysis of these schemes is yet available. For that reason, we will not go into the details of these mixing schemes but rather refer readers to [15] for more discussions.

The second way to improve the convergence of SCF, which we will discuss in the next section, is to restrict the minimization of the quadratic surrogate in (3.1) (hence the update of the single particle wave functions) to a small neighborhood of the current approximation $X^{(i)}$. This technique is known as the [illegible] technique in the numerical optimization community [5, 20]. The gradient matching (between the total energy and the surrogate) property of SCF implies that a reduction of the total energy in the SCF iteration can be guaranteed if one imposes a trust region with a sufficiently small radius to (3.1).



(b)

FIG. 3.3(*continued*). (b) *When* $\alpha = 12$ *is set in* (3.3), *the SCF iteration fails to converge. This figure shows the change of* $E_{total}(x)$ *in a single SCF iteration. A zoom-in view of the boxed area in Figure* 3.2(a). *The minimizer of* $q(x)$ *(the small solid triangle) lies on the outer dashed contour line, indicating an increase in the total energy. The solid diamond marks the true minimizer of* E_{total}.

4. Trust region SCF. The simplest type of trust region one may consider for (3.1) is

$$\|X - X^{(0)}\|_F \leq \Delta, \tag{4.1}$$

where Δ is a trust region radius that may be reduced or enlarged depending on the ratio of the actual reduction of $E_{total}(X)$ over the predicted reduction measured in terms of the change in $q(X)$. The constrained optimization problem

$$\begin{aligned} &\min q(X) \\ &X^*X = I, \\ &\|X - X^{(0)}\|_F \leq \Delta, \end{aligned} \tag{4.2}$$

associated with a particular choice of Δ is known as a [illegible].

However, there are two serious drawbacks associated with this type of constraint. First of all, (4.1) is not rotationally invariant; i.e., the inequality (4.1) is not equivalent to

$$\|XQ - X^{(0)}\|_F \leq \Delta$$

for all $Q \in \mathbb{C}^{k\times k}$ such that $Q^*Q = I_k$. As a result, the solution to the trust region subproblem (4.2) is not rotation invariant, whereas the solution to the original total energy minimization problem is.

Secondly, adding (4.1) as a constraint makes the constrained quadratic minimization problem much more difficult to solve. If we introduce the constraint (4.1) as a penalty function and solve

$$\begin{aligned} &\min \hat{q}(X;\sigma) \equiv q(X) + \sigma \|X - X^{(0)}\|_F^2, \\ &X^*X = I_k \end{aligned} \tag{4.3}$$

for an appropriately chosen penalty parameter σ, the first order necessary condition of (4.3) cannot be expressed as a linear eigenvalue problem or other simple form. Thus it cannot be solved easily.

To preserve the rotational invariance property of the solution to (2.3) in a trust region subproblem, we must define the trust region in terms of quantities that are rotationally invariant. Both the charge density $\rho(X)$ defined in (2.2) and the density matrix $D(X)$ defined in (2.1) satisfy this desirable property. However, we will show in the following that a trust region defined in terms of $D(X)$ allows us to reduce the minimizing of the constrained quadratic surrogate function to a linear eigenvalue problem.

Note that $D(X)$ is an orthogonal projector associated with the subspace spanned by columns of X when $X^*X = I_k$. It is well known [10] that

$$\|D(X) - D(X^{(0)})\|_F \tag{4.4}$$

measures the distance between the subspaces defined by columns of X and $X^{(0)}$, where $\|A\|_F$ denotes the Frobenius norm of A. Therefore, we impose the constraint

$$\|D(X) - D(X^{(0)})\|_F \le \Delta$$

on the solution to the quadratic minimization problem. Because $X^*X = {X^{(0)}}^*X^{(0)} = I_k$, it is easy to verify that

$$\begin{aligned} \|D(X) - D(X^{(0)})\|_F^2 &= \|D(X)\|_F^2 + \|D(X^{(0)})\|_F^2 + 2\mathrm{trace}\big[D(X)^*D(X^{(0)})\big] \\ &= 2k - 2\mathrm{trace}\big[X^*X^{(0)}{X^{(0)}}^*X\big]. \end{aligned}$$

If we solve the trust region subproblem by introducing (4.4) as a penalty function in the quadratic objective $q(x)$, i.e., we solve

$$\begin{aligned} &\min \hat{q}(X;\sigma) \equiv \frac{1}{2}\mathrm{trace}\big[X^*H(X^{(0)})X\big] - \frac{\sigma}{2}\mathrm{trace}\big[X^*X^{(0)}{X^{(0)}}^*X\big], \\ &\text{s.t. } X^*X = I_k, \end{aligned} \tag{4.5}$$

where σ is an appropriately chosen penalty parameter, then the first order necessary condition associated with (4.5) becomes

$$\begin{aligned} \big[H(X^{(0)}) - \sigma X^{(0)}{X^{(0)}}^*\big]X &= X\Lambda, \\ X^*X &= I_k, \end{aligned}$$

where Λ (which can be diagonalized by postmultiplying X by an unitary transformation) is a matrix of Lagrange multipliers.

Therefore, when a trust region is defined with respect to $D(X)$, solving the corresponding trust region subproblem is equivalent to computing the eigenvectors associated with the smallest k eigenvalues of the [illegible] Hamiltonian $H(X^{(0)}) -$

$\sigma X^{(0)} X^{(0)^*}$, a problem that we generally know how to solve efficiently, at least when the dimension of $H(X^{(0)})$ is small.

What remains to be determined now is the penalty or trust region parameter σ in (4.5). Choosing a large σ value has the same effect as setting a small trust region radius Δ in (4.2). When σ is sufficiently large, the convergence of the SCF iteration can be guaranteed, although the rate of convergence may be very slow. When σ is too small, the solution to (4.6) may lead to an increase in the total energy. Unfortunately, the optimal choice of σ cannot be obtained analytically in general. The standard recipe for choosing such a parameter is usually dynamic [20]. In an unconstrained optimization problem, one starts with an arbitrary guess Δ_0 bounded by the maximum step length allowed. After the trust region subproblem is solved, the trust region radius may be reduced or increased based on the ratio of reduction in the true objective E_{true} over the predicted reduction E_{pred} measured from the surrogate function. A reduction of the trust region radius implies that the trust region subproblem must be resolved.

Two special features of the SCF iteration require the selection of penalty or trust region parameters to be made in a slightly different manner. First of all, the quadratic surrogate function in (3.1) does not match the true objective, i.e., the KS total energy function, at the current iterate $X^{(i)}$. Recall that the only thing that matches between $E_{total}(X)$ and $q(X)$ at $X^{(i)}$ is their gradients. Hence, the ratio between the changes in $E_{total}(X)$ and $q(X)$ can be difficult to predict. Consequently, an adjustment of the penalty parameter based on this ratio is unlikely to be effective. Secondly, the evaluation of $E_{total}(X)$ tends to be costly. Therefore, the selection of the penalty parameter should avoid repeated evaluation of the total energy.

A heuristic for estimating the penalty parameter is developed in [31] by expressing the change in E_{total} as a function of a matrix built from a linear combination of previous density matrices. The optimal σ is estimated by applying an inexact line search to the approximate model. Such a scheme requires saving wave functions or density matrices obtained in previous SCF iterations; thus is not practical for large problems.

We propose to use a simpler heuristic in this paper. Our heuristic is based on the following observation. As $X^{(i)}$ converges to X in a trust region enabled SCF iteration, the eigenvalues of the level-shifted Hamiltonian $H(X) - \sigma X X^*$ converge to

$$\lambda_1 - \sigma, \lambda_2 - \sigma, \ldots, \lambda_k - \sigma, \lambda_{k+1}, \lambda_{k+2}, \ldots, \lambda_n, \tag{4.6}$$

where

$$\lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$$

are eigenvalues of the KS Hamiltonian $H(X)$ as defined in (2.12). Thus, adding a trust region of the form (4.4) has the effect of increasing the gap between the kth and the $(k+1)$th eigenvalues of the shifted Hamiltonian. Even though (4.6) does not hold in general for eigenvalues of the shifted Hamiltonian $H(X^{(i)})$ before $X^{(i)}$ converges to the solution of (2.3), our numerical experiments and those presented in [31] show an increased gap between the kth and $(k+1)$th eigenvalues of the shifted $H(X^{(i)})$ when σ is sufficiently large.

It is well known [22] that a larger gap between the eigenvalues associated with the desired invariant subspace and the rest of the spectrum generally makes it easier to compute the desired invariant subspace. Thus our heuristic tries to enlarge such a gap when total energy increases in an SCF iteration. To be specific, we set σ to zero

TRSCF

Input: The matrices L, D_{ion}, S, and the vectors w_ℓ, $\ell = 1, 2, \ldots$. The derivative of the exchange-correlation function $\mu_{xc}(x)$; an initial guess $X^{(0)}$ for the optimal wave function $X \in \mathbb{C}^{n\times m}$;

Output: $X \in \mathbb{C}^{n\times m}$ such that $X^*X = I_m$ and $E_{tot}(X)$ is minimized, where $E_{tot}(X)$ is defined by (2.6)–(2.10).

1. Set $\sigma = 0.0$;
2. for $i = 1, 2, \ldots$ until convergence
3. if ($\sigma = 0.0$)
4. Set $H^{(i)} = H(X^{(i-1)})$;
5. else
6. Set $H^{(i)} = H(X^{(i-1)}) - \sigma X^{(i-1)} X^{(i-1)^*}$;
7. endif
8. Compute $X^{(i)}$ such that $H^{(i)}X^{(i)} = X^{(i)}\Lambda^{(i)}$ and $\Lambda^{(i)}$ contains the k smallest eigenvalues of $H^{(i)}$;
9. if ($E_{tot}(X^{(i)}) > E_{tot}(X^{(i-1)})$), set $\sigma \leftarrow \gamma(\lambda_{k+1}^{(i)} - \lambda_k^{(i)})$ for some constant $\gamma > 1$;
10. end for

FIG. 4.1. *A trust region-based SCF iteration.*

initially. If the minimizer of $q(X)$ yields an increase in $E_{total}(X)$, we increase σ by setting it to $\gamma\eta_{\max}$, where $\eta_{\max}$ is the maximum gap defined as

$$\eta_{\max} = \max_{\ell\in\{1,2,\ldots,n-1\}} \lambda_{\ell+1}^{(i)} - \lambda_\ell^{(i)},$$

where $\lambda_\ell^{(i)}$ is the ℓth eigenvalue of $H(X^{(i)})$ and γ is a small constant. Empirically, we found a good choice of γ to be around 2 to 5. This particular strategy for choosing the penalty parameter is somewhat conservative in the sense that σ is never decreased in subsequent SCF iterations to allow a larger reduction in total energy. A more sophisticated and efficient scheme will be described in a separate paper.

The major steps of a TRSCF iteration is summarized in Figure 4.1. We now return to the simple example (3.3) and show the effect of applying the trust region technique in the SCF iteration. The eigenvalues of the Hamiltonian constructed at $\hat{x}$ are

$$\hat{\lambda}_1 = 6.4597 \quad \text{and} \quad \hat{\lambda}_2 = 9.5403.$$

We set $\sigma = \hat{\lambda}_2 - \hat{\lambda}_1$ to increase the gap between the eigenvalues of the shifted Hamiltonian by a factor of two. Figure 4.2 is almost identical to Figure 3.3(b). The only difference is that we plotted the solution to the trust region subproblem (4.5), which is marked by a solid diamond, in addition to the solution to the unpenalized surrogate (3.1), which is marked by a solid triangle. As we can see in this figure, the solution to the trust region subproblem lies inside the inner dashed contour, indicating a reduction in total energy as we move from the small solid circle to the solid diamond. Furthermore, the solid diamond is clearly closer to the minimizer of the total energy

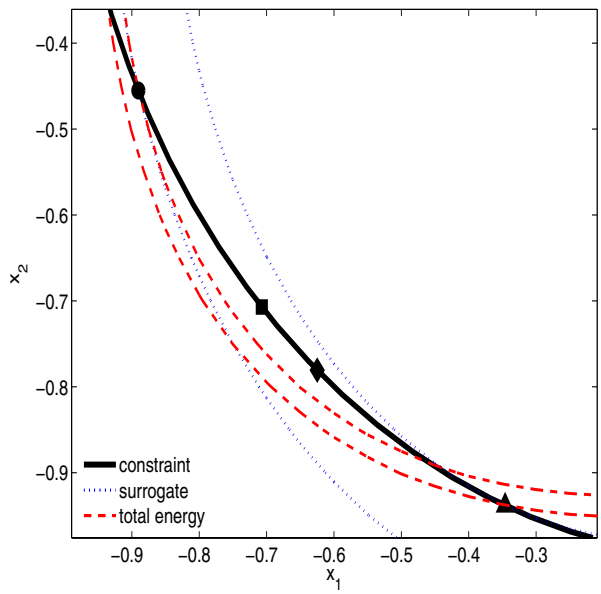


FIG. 4.2. *This is the same contour plot as shown in Figure* 3.3(b)*, except that the figure also shows the effect of applying the trust region technique in SCF. The solution to the trust region subproblem or the penalized problem* (4.5) *is marked by the solid diamond, which lies to the upper right of the inner dashed contour. Clearly, it is closer to the true minimizer (the small solid square) than* $\hat{x}$ *(the small solid circle), indicating a reduction in the total energy in the SCF iteration.*

(the solid square) than either the current approximation $\hat{x}$ (the small solid circle) or the minimizer of the unpenalized surrogate $\hat{x}_q$ (the solid triangle).

5. Direct minimization of the total energy. A monotonic reduction of the total energy can be guaranteed in TRSCF if the penalty parameter σ is sufficiently large. However, choosing a large penalty parameter may lead to a slow convergence rate for large atomistic systems [4] because one is forced to take a shorter step in each TRSCF iteration. In [31], TRSCF is combined with a DIIS-like acceleration scheme to improve the convergence rate. In this paper, we proposed to use TRSCF within a DCM that we developed in [36].

Instead of minimizing $E_{total}(X)$ by minimizing a sequence of quadratic surrogate functions, the DCM algorithm minimizes the $E_{total}(X)$ directly.

This general approach has been discussed in a number of papers [2, 9, 15, 23, 30, 32, 33]. In [23, 30], a conjugate gradient (CG) type of algorithm is used to minimize the total energy. The minimization is carried out "band-by-band"; i.e., the total energy is minimized with respect to one wave function at a time. For the jth band (wavefunction), a search direction $p_j^{(i)}$ is generated from a linear combination of the wavefunction $x_j^{(i)} = X^{(i)} e_j$ and the residual

$$r_j = H^{(i)} x_j^{(i)} - x_j^{(i)} \lambda_j,$$

where λ_j is the jth eigenvalue of the projected Hamiltonian $X^{(i)*} H^{(i)} X^{(i)}$. Note that r_j is simply the jth column of the gradient matrix $\nabla_X \mathcal{L}(X^{(i)})$. Similar to the standard CG algorithm, the linear combination of $x_j^{(i)}$ and r_j is chosen so that $p_j^{(i)}$ is

$H^{(i)}$-conjugate to the previous search direction $p_j^{(i-1)}$. The new wave function $x_j^{(i+1)}$ is then computed by minimizing the KS total energy in the subspace spanned by $x_j^{(i)}$ and $p_j^{(i)}$. To simplify this minimization problem, $p_j^{(i)}$ is first orthogonalized against $x_j^{(i)}$ and normalized so that $\|p_j^{(i)}\| = 1$. The new wave function is then parameterized by

$$x_j^{(i+1)} = x_j^{(i)} \cos\theta + p_j^{(i)} \sin\theta,$$

where the optimal θ is obtained by a standard line search procedure. Instead of using the KS function to perform the line search, Teter, Payne, and Allan [30] proposed using a surrogate function that is cheaper to evaluate. However, this approach was shown in [15] to be less efficient than the SCF iteration. We believe this is primarily due to the "band-by-band" nature of the algorithm.

The methods presented in [9, 2, 33, 32] were designed to minimize the total energy with respect to all wave functions (associated with the occupied state) simultaneously. The method developed in [9] modifies the unconstrained CG search direction so that the orthonormality constraint $X^*X = I_k$ can be satisfied. The approaches taken in [2, 32] reparameterize the search direction so that standard unconstrained minimization can be used directly. The algorithm developed in [33] first computes the search direction via a limited-memory BFGS (Broyden–Fletcher–Goldfarb–Shannon) scheme [19], and the search direction is then modified through a parallel transport technique [7] to ensure that the orthonormality constraint $X^*X = I_k$ is satisfied in the line search procedure. In all of these methods, the search direction is computed first, and a step length is then determined to reduce the total energy along the computed search direction.

The direct minimization algorithm that we presented in [36] also seeks the optimal wave functions associated with all occupied states simultaneously. However, we choose the search direction and the step length simultaneously from a subspace that consists of the existing wave functions $X^{(i)}$, the gradient of the Lagrangian (2.4), and the search direction produced in the previous iteration. A special strategy is developed to minimize the total energy within the search space while maintaining the orthonormality constraint required for $X^{(i+1)}$. This strategy requires us to solve a projected nonlinear eigenvalue problem, as we will illustrate below.

Let $R^{(i)}$ be the preconditioned gradient of the Lagrangian (2.4) with respect to X evaluated at $X^{(i)}$, and let $P^{(i-1)}$ be the search direction obtained in the $(i-1)$th iteration. In our algorithm, the wave function update is performed within the $3k$-dimensional subspace spanned by $X^{(i)}$, $R^{(i)}$, and $P^{(i-1)}$. This is in the same spirit as the locally optimal block preconditioned conjugate gradient (LOBPCG) algorithm proposed in [14] for solving large-scale linear eigenvalue problems. Note that the inclusion of $P^{(i-1)}$ is important. It prevents the search direction constructed at the ith step from being parallel to the steepest descent direction which often results in a tiny change between $X^{(i)}$ and $X^{(i+1)}$ (hence a small reduction in E_{total} from $X^{(i)}$ to $X^{(i+1)}$).

If we let

$$Y = (X^{(i)},\, R^{(i)},\, P^{(i-1)}),$$

we can then express the new approximation, $X^{(i+1)}$, by

$$X^{(i+1)} = YG, \tag{5.1}$$

where $G \in \mathbb{C}^{3k\times k}$ is chosen to minimize $\hat{E}(G) \equiv E_{total}(YG)$; i.e., we must solve

$$\begin{aligned} &\min_G E_{total}(YG) \\ &\text{s.t. } G^*Y^TYG = I_k. \end{aligned} \tag{5.2}$$

The first order necessary condition of (5.2) can be derived by examining the gradient of the Lagrangian associated with $\hat{E}(G)$ (with respect to G). It is easy to verify [36] that

$$\nabla_G \hat{E}(G) = \hat{H}(G)G, \tag{5.3}$$

where

$$\hat{H}(G) = Y^*\left[\frac{1}{2}L + D_{ion} + \sum_\ell w_\ell w_\ell^* + \mathrm{Diag}\Big(S\rho(YG)\Big) + \mathrm{Diag}\Big(\mu_{xc}(\rho(YG))\Big)\right]Y. \tag{5.4}$$

(Note that (5.4) has been scaled by 1/2 to be consistent with the convention used in the electronic structure community.)

Consequently, solving (5.2) is equivalent to solving

$$\hat{H}(G)G = BG\Omega_k, \qquad G^*BG = I_k, \tag{5.5}$$

where $B = Y^*Y$ and the $k \times k$ diagonal matrix Ω_k contains the k smallest eigenvalues of (5.5).

Note that the projected nonlinear eigenvalue problem defined by (5.5) is much smaller than the nonlinear eigenvalue solved in an SCF iteration. The reduction in size provides us with more flexibility in choosing appropriate algorithms to solve the nonlinear eigenvalue problem. In particular, we can apply the TRSCF iteration introduced in section 4 to compute the desired eigenpairs of (5.5). The presence of the mass matrix B does not pose any difficulty in defining a trust region. Through a change of variables (i.e., let $\tilde{G} = RG$, where R is the Cholesky factor of B, i.e., $G = R^H R$), we can easily show that the first order necessary condition of the penalized quadratic surrogate defined at $G^{(i)}$ can be expressed by

$$\big[\hat{H}(G^{(i)}) - \sigma BG^{(i)}{G^{(i)}}^*B\big]G = BG\Omega_k, \qquad G^*BG = I_k. \tag{5.6}$$

The generalized linear eigenvalue problem defined by (5.6) can be solved by calling an appropriate LAPACK [1] subroutine. Choosing a sufficiently large penalty parameter will guarantee that $E_{total}(YG)$ decreases monotonically. Furthermore, it should be noted that it is not necessary to solve (5.5) to full accuracy in the early stage of the direct minimization process, because all we need is a G that yields sufficient decrease in the objective function within the subspace spanned by columns of Y.

Once G is computed, we can update the wave function following (5.1). In addition, we can compute the search direction associated with this update using [14]

$$P^{(i)} \equiv X^{(i+1)} - X^{(i)}G(1:k,:) = Y(:,k+1:3k)G(k+1:3k,:).$$

Because the solution to (5.5) ensures that columns of $X^{(i+1)}$ are orthonormal, there is no need to explicitly orthogonalize $P^{(i)}$ against $X^{(i)}$ in our algorithm.

A complete description of the constrained minimization algorithm is shown in Figure 5.1. We should point out that solving the projected optimization problem

ALGORITHM: A CONSTRAINED MINIMIZATION ALGORITHM FOR TOTAL ENERGY MINIMIZATION

Input: An initial set of wave functions $X^{(0)} \in \mathbb{C}^{n\times k}$, where k is the number of occupied states; the matrices L, D_{ion}, S; the vectors w_ℓ, $\ell = 1, 2, \ldots$. The derivative of the exchange-correlation function $\mu_{xc}(x)$; a preconditioner K;

Output: $X \in \mathbb{C}^{n\times k}$ such that the KS total energy function $E_{total}(X)$ is minimized and $X^*X = I_k$.

1. Orthonormalize $X^{(0)}$ such that ${X^{(0)}}^* X^{(0)} = I_k$;

2. for $i = 0, 1, 2, \ldots$ until convergence

3. Compute $\Theta = {X^{(i)}}^* H^{(i)} X^{(i)}$;

4. Compute $R = K^{-1}\left[H^{(i)}X^{(i)} - X^{(i)}\Theta\right]$,

5. if $(i > 1)$ then
$Y \leftarrow (X^{(i)}, R, P^{(i-1)})$
else
$Y \leftarrow (X^{(i)}, R)$;
endif

6. $B \leftarrow Y^*Y$;

7. Find $G \in \mathbb{C}^{2k\times 2k}$ or $\mathbb{C}^{3k\times 3k}$ that minimizes $E_{total}(YG)$ subject to the constraint $G^*BG = I$;

8. Set $X^{(i+1)} = YG$;

9. if $(i > 1)$ then
$P^{(i)} \leftarrow Y(:, k+1:3k)G(k+1:3k, :)$;
else
$P^{(i)} \leftarrow Y(:, k+1:2k)G(k+1:2k, :)$;
endif

10. end for

FIG. 5.1. *A direct constrained minimization algorithm for total energy minimization.*

in step 7 of the algorithm requires us to evaluate the projected Hamiltonian (5.4) repeatedly as we search for the best G. However, since the first three terms of $\hat{H}$ do not depend on G, they can be computed and stored in advance. Only the last two terms of (5.4) need to be updated. These updates require that the charge density, the Hartree, and the exchange-correlation potentials be recomputed.

6. Numerical examples. In this section, we compare the performance of the DCM algorithm presented in the previous section with that of the SCF iteration implemented in the software package PEtot [34] through two numerical examples. In PEtot, single particle wave functions are discretized by a spectral method using plane waves as the basis. These basis functions are eigenfunction of the Laplacian operator (L) associated with the kinetic energy of the atomistic system. Thus, PEtot stores only the Fourier coefficients of each wave function $x_j = Xe_j$ instead of x_j itself so that $y \leftarrow Lx_j$ can be carried out in $\mathcal{O}(n)$ floating point operations (flops) in the frequency space. However, because the potential terms of the Hamiltonian (with the exception of the nonlocal ionic potential) are diagonal in the spatial domain, PEtot converts the

Fourier space representation of x_j into the real space representation before operations involving these potential terms are performed. The complexity of this conversion is $\mathcal{O}(n \log n)$ when it is carried out by a fast Fourier transform (FFT). We measure the convergence of both algorithms by examining the relative reduction of the total energy computed in each outer iteration. The relative reduction is evaluated by

$$\Delta E_i = E_{total}(X^{(i)}) - E_{min},$$

where E_{min} is a lower bound of the total energy.

In a PEtot SCF iteration, the minimization of the surrogate (3.1) is accomplished by applying a preconditioned conjugate gradient (PCG) algorithm to minimize the Rayleigh quotient $x^* H^{(i)} x / x^* x$. Explicit deflation is put in place to accelerate the convergence of the smallest k eigenpairs. Each PCG iteration requires a single matrix vector (MATVEC) multiplication followed by a preconditioning operation. When n is sufficiently large, the complexity of each MATVEC is dominated by the cost of the FFT calculation used to convert the Fourier space representation of x_j to the real space representation. The Laplacian operator L is used as the preconditioner. Because it is diagonal in the frequency space, the cost of preconditioning is relatively small compared to a MATVEC. If m PCG iterations are taken on average to compute an approximate eigenpair of $H^{(i)}$, then the total number of MATVECs used per SCF iteration is $m \times k$.

In the DCM algorithm, k MATVECs are performed in each outer iteration to obtain the gradient. When TRSCF is used to solve the projected problem (5.5), each outer DCM iteration contains a number of inner TRSCF iterations in which the projected Hamiltonian (5.4) must be updated repeatedly. The update of the projected Hartree potential requires us to compute $S\rho(YG)$. Because S is the inverse of L, this calculation is typically carried out by a fast Poisson solver. The complexity of this computation is approximately $\mathcal{O}(n \log n)$, which is equivalent to a single MATVEC used in the SCF iteration asymptotically. Thus, if p inner TRSCF iterations are taken in the DCM algorithm, the total number of MATVECs used per DCM iteration is $k + p$.

6.1. The PtNiO system. We applied both algorithms to a relatively large system consisting of 9 atoms and 86 valence electrons. It represents a thin PtNi slab with one O atom attached to the surface. The system is used for catalysis to dissociate O_2 molecules. The wave function is defined on a $96 \times 48 \times 48$ real space grid. The number of plane wave basis functions used in the Fourier representation is 15181, and the number of occupied states for this molecule is $k = 43$.

In the SCF calculation, we set the convergence tolerance of each PCG run to $\tau = 10^{-12}$, and the maximum number of PCG iterations allowed to 10. That is, we terminate the PCG iteration when

$$\|H^{(i)} x_j^{(i)} - \lambda_j^{(i)} x_j^{(i)}\| \leq 10^{-12},$$

or when the number of PCG iterations taken reaches 10. In our experiment, the PCG convergence tolerance was never reached before the maximum number of iterations were taken. Thus each outer SCF iteration consumed 43×10 MATVECs. Both Pulay [25] (DIIS) and Kerker [13] charge-mixing schemes were used in the outer SCF iteration to accelerate the convergence.

In the DCM calculation, the projected minimization problem was solved by applying TRSCF iteration to (5.5). We set the number of inner TRSCF iterations to 5.

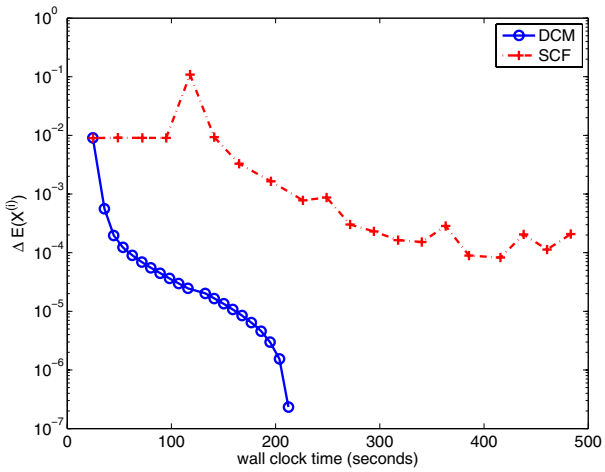


FIG. 6.1. *Comparing the convergence of SCF and DCM when they are applied to the PtNiO system.*

Thus, the number of MATVECs used in each DCM iteration is roughly $5 + 43$, which is significantly smaller than that used in SCF.

Both SCF and DCM have been parallelized using MPI (message passing interface). For the PtNiO system, we ran both codes on the IBM SP maintained at the National Energy Research Scientific Computing (NERSC) Center using 64 CPUs. Each IBM SP node contains 16 Power3 CPUs and 16 GB memory. Each Power3 CPU runs at a 375 MHz clock speed, and has 2 MB L2 cache. Figure 6.1 shows that DCM exhibits monotonic convergence whereas the total energy does not decrease monotonically in SCF even when both Pulay and Kerker charge-mixing schemes are used to stabilize the algorithm. After 20 iterations, DCM was able to reach a much lower KS total energy level.

6.2. The $Si_{29}H_{36}$ cluster. In the second example, we show that the trust region DCM algorithm is more efficient than the SCF iteration (accelerated by charge mixing) for atomistic systems that are "well behaved" (in the sense that the total energy decreases monotonically in SCF). A smaller example was presented in our earlier work [36] in which both DCM and SCF were applied to the SiH_4 system. We showed that DCM was almost four times faster than SCF when both were executed on 16 Power3 CPUs. No trust region was used in that experiment.

In this example, we apply both trust region DCM and SCF (accelerated by charge mixing) to a larger silicon cluster that contains 29 silicon atoms and 36 hydrogen atoms. The total number of valence electrons in the system is 152; i.e., the number of occupied states is $k = 76$. The system is discretized by plane waves on a real space grid of $96 \times 96 \times 96$. The number of plane wave basis functions used is 35585. We ran both DCM and SCF on 128 Power3 CPUs at NERSC. The PCG tolerance and iteration limits used in this example are the same as those set in the PtNiO example. We also set the number of inner TRSCF iterations used in DCM to 5.

Figure 6.2 shows that the total energy decreases monotonically in both the DCM and SCF runs for this problem. However, the reduction of the total energy is much faster in DCM than it is in SCF.

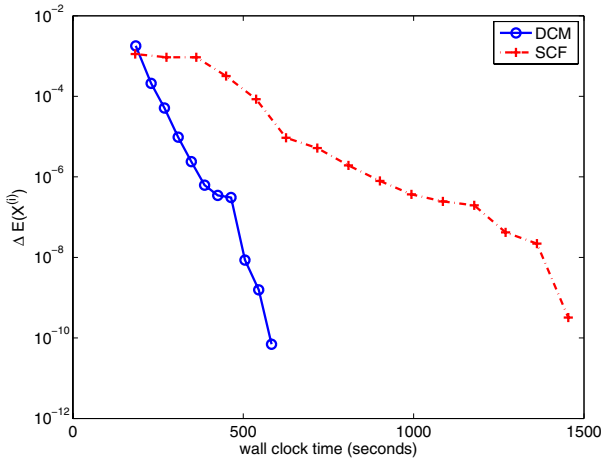


FIG. 6.2. *Comparing the convergence of SCF and DCM when they are applied to the* $Si_{29}H_{36}$ *cluster.*

7. Concluding remarks. We viewed the SCF iteration, commonly used for solving the KS equation, as an optimization procedure that minimizes the KS total energy indirectly by minimizing a sequence of quadratic surrogate. Such a viewpoint allows us to easily explain when SCF works and how it can fail. It also allows us to devise techniques that can either stabilize or accelerate the SCF iteration. We showed that the convergence of SCF can be stabilized by introducing a quadratic constraint into the surrogate minimization problem. Such a constraint restricts the wave function update to a small neighborhood of the current iterate at which the gradients of the KS total energy and the surrogate match, hence defining a "trust region." However, applying a trust region-based SCF iteration directly to the KS equation may lead to slow convergence. We proposed using the trust region technique within the DCM algorithm developed in [36] to compute optimal search directions and step lengths simultaneously. We demonstrated through two numerical examples that such a scheme outperforms the SCF iteration combined with charge mixing in terms of both efficiency and reliability. We should point out that our numerical results are still somewhat preliminary. For atomistic systems that are "well behaved," the performance difference between SCF and DCM may be less dramatic if the single vector PCG algorithm used in PEtot to solve the surrogate minimization problem (3.1) is replaced by a block algorithm. Also, it will be interesting to see how these two methods differ in performance when the number of electrons becomes a larger fraction of n. In that case, dense matrix operations will constitute a significant portion of the computational cost in both DCM and SCF. We will undertake a more systematic performance analysis in a future study that will also take into account different discretization schemes.

REFERENCES

[1] E. ANDERSON, Z. BAI, C. BISCHOF, J. DEMMEL, J. DONGARRA, J. DU CROZ, A. GREENBAUM, S. HAMMARLING, A. MCKENNEY, S. OSTROUCHOV, AND D. SORENSEN, *LAPACK Users' Guide*, 2nd ed., SIAM, Philadelphia, 1995.

[2] T. A. ARIAS, M. C. PAYNE, AND J. D. JOANNOPOULOS, *Ab initio molecular dynamics: Analytically continued energy functionals and insights into iterative solutions*, Phys. Rev. Lett., 69 (1992), pp. 1077–1080.

[3] E. L. Briggs, D. J. Sullivan, and J. Berholc, *Real-space multigrid-based approach to large-scale electronic structure calculations*, Phys. Rev. B, 54 (2001), pp. 14362–14375.
[4] C. Le Bris, *Computational chemistry from the perspective of numerical analysis*, Acta Numer., 14 (2005), pp. 363–444.
[5] A. R. Conn, N. I. M. Gould, and Ph. L. Toint, *Trust-Region Methods*, MPS-SIAM Ser. Optim. 1, SIAM, Philadelphia, 2000.
[6] E. R. Davidson, *The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real symmetric matrices*, J. Comput. Phys., 17 (1975), pp. 87–94.
[7] A. Edelman, T. A. Arias, and S. T. Smith, *The geometry of algorithms with orthogonality constraints*, SIAM J. Matrix Anal. Appl., 20 (1998), pp. 303–353.
[8] J. L. Fatteberg, *A block Rayleigh quotient iteration with local quadratic convergence*, Electron. Trans. Numer. Anal., 7 (1998), pp. 56–74.
[9] M. J. Gillan, *Calculation of the vacancy formation in aluminum*, J. Phys. Condens. Matter, 1 (1989), pp. 689–711.
[10] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins, Baltimore, MD, 1989.
[11] M. Heiskanen, T. Torsti, M. J. Puska, and R. M. Nieminen, *Multigrid method for electronic structure calculations*, Phys. Rev. B, 63 (2001), pp. 1–8.
[12] M. R. Hestenes and W. Karush, *A method of gradients for the calculation of the characteristic roots and vectors of a real symmetric matrix*, J. Res. National Bureau of Standards, 47 (1951), pp. 45–61.
[13] G. P. Kerker, *Efficient iteration scheme for self-consistent pseudopotential calculations*, Phys. Rev. B, 23 (1981), pp. 3082–3084.
[14] A. V. Knyazev, *Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method*, SIAM J. Sci. Comput., 23 (2001), pp. 517–541.
[15] G. Kresse and J. Furthmüller, *Efficiency of ab initio total energy calculations for metals and semiconductors using a plane-wave basis set*, Comput. Materials Sci., 6 (1996), pp. 15–50.
[16] G. Kresse and J. Furtthmüller, *Efficient iterative schemes for* ab initio *total-energy calculations using a plane-wave basis set*, Phys. Rev. B, 54 (1996), pp. 11169–11185.
[17] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Res. National Bureau of Standards, 45 (1950), pp. 255–282.
[18] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, Software Environ. Tools 6, SIAM, Philadelphia, 1998.
[19] D. C. Liu and J. Nocedal, *On the limited memory method for large scale optimization*, Math. Programming B, 45 (1989), pp. 503–528.
[20] J. Nocedal and S. Wright, *Numerical Optimization*, Springer-Verlag, New York, 1999.
[21] J. Olsen, P. Jorgensen, and J. Simons, *Passing the one-billion limit in full configuration-interaction (FCI) calculations*, Chem. Phys. Lett., 169 (1990), pp. 463–472.
[22] B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice–Hall, Englewood Cliffs, NJ, 1980.
[23] M. C. Payne, M. P. Teter, D. C. Allen, T. A. Arias, and J. D. Joannopoulos, *Iterative minimization techniques for* ab initio *total energy calculation: Molecular dynamics and conjugate gradients*, Rev. Modern Phys., 64 (1992), pp. 1045–1097.
[24] P. Pulay, *Convergence acceleration of iterative sequences: The case of SCF iteration*, Chem. Phys. Lett., 73 (1980), pp. 393–398.
[25] P. Pulay, *Improved SCF convergence acceleration*, J. Comput. Chem., 3 (1982), pp. 556–560.
[26] Y. Saad, A. Sathopoulos, J. Chelikowsky, K. Wu, and S. Ogut, *Solution of large eigenvalue problems in electronic structure calculations*, BIT, 36 (1996), pp. 563–578.
[27] A. H. Sameh and J. A. Wisniewski, *A trace minimization algorithm for the generalized eigenvalue problem*, SIAM J. Numer. Anal., 19 (1982), pp. 1243–1259.
[28] V. R. Saunders and I. H. Hillier, *A level-shifting method for converging closed shell Hartree-Fock wavefunctions*, Int. J. Quantum Chem., 7 (1973), pp. 699–705.
[29] G. L. G. Sleijpen and H. A. Van der Vorst, *A Jacobi–Davidson iteration method for linear eigenvalue problems*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 401–425.
[30] M. P. Teter, M. C. Payne, and D. C. Allan, *Solution of Schrödinger's equation for large systems*, Phys. Rev. B, 40 (1989), pp. 12255–12263.
[31] L. Thogersen, J. Olsen, D. Yeager, and P. Jorgensen, *The trust-region self-consistent field method: Towards a black-box optimization in Hartree-Fock and Kohn-Sham theories*, J. Chem. Phys., 121 (2004), pp. 16–27.
[32] J. VandeVondele and J. Hutter, *An efficient orbital transformation method for electronic structure calculations*, J. Chem. Phys., 118 (2003), pp. 4365–4369.

[33] T. Van Voorhis and M. Head-Gordon, *A geometric approach to direct minimization*, Molecular Phys., 100 (2002), pp. 1713–1721.
[34] L. Wang, *PETOT*, software package available online at http://hpcrd.lbl.gov/∼linwang/PEtot/PEtot.htm.
[35] K. Wu, A. Canning, H. D. Simon, and L.-W. Wang, *Thick-restart Lanczos method for electronic structure calculations*, J. Comput. Phys., 154 (1999), pp. 156–173.
[36] C. Yang, J. C. Meza, and L.-W. Wang, *A constrained optimization algorithm for total energy minimization in electronic structure calculation*, J. Comput. Phys., 217 (2006), pp. 709–721.

SIAM J. SCI. COMPUT.
Vol. 29, No. 5, pp. 1876–1899
© 2007 Society for Industrial and Applied Mathematics

FAST RADIAL BASIS FUNCTION INTERPOLATION VIA PRECONDITIONED KRYLOV ITERATION*

NAIL A. GUMEROV† AND RAMANI DURAISWAMI‡

Abstract. We consider a preconditioned Krylov subspace iterative algorithm presented by Faul, Goodsell, and Powell (*IMA J. Numer. Anal.* 25 (2005), pp. 1–24) for computing the coefficients of a radial basis function interpolant over N data points. This preconditioned Krylov iteration has been demonstrated to be extremely robust to the distribution of the points and the iteration rapidly convergent. However, the iterative method has several steps whose computational and memory costs scale as $O(N^2)$, both in preliminary computations that compute the preconditioner and in the matrix-vector product involved in each step of the iteration. We effectively accelerate the iterative method to achieve an overall cost of $O(N \log N)$. The matrix vector product is accelerated via the use of the fast multipole method. The preconditioner requires the computation of a set of closest points to each point. We develop an $O(N \log N)$ algorithm for this step as well. Results are presented for multiquadric interpolation in $\mathbb{R}^2$ and biharmonic interpolation in $\mathbb{R}^3$. A novel FMM algorithm for the evaluation of sums involving multiquadric functions in $\mathbb{R}^2$ is presented as well.

Key words. radial basis function interpolation, preconditioned conjugate gradient, cardinal function preconditioner, computational geometry, fast multipole method

AMS subject classifications. 65D07, 41A15, 41A58

DOI. 10.1137/060662083

1. Introduction.

Interpolation of scattered data of various types using radial-basis functions has been demonstrated to be useful in different application domains [23, 7, 8, 29]. Much of this work has been focused on the use of polyharmonic and multiquadric radial-basis functions (RBF) $\left[\phi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}\right]$ that fit data as

$$s(\mathbf{x}) = \sum_{j=1}^{N} \lambda_j \phi\left(|\mathbf{x} - \mathbf{x}_j|\right) + \mathcal{P}(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^d, \tag{1.1}$$

where several classes of radial-basis functions may be chosen for ϕ. Popular choices include

$$\phi(r) = \begin{cases} r & \text{biharmonic } (\mathbb{R}^3) \\ r^{2n+1} & \text{polyharmonic } (\mathbb{R}^3) \\ \left(r^2 + c^2\right)^{1/2} & \text{multiquadric } (\mathbb{R}^d). \end{cases}$$

Here λ_j are coefficients while $\mathcal{P}(\mathbf{x})$ is a global polynomial function of total degree at most $K-1$. The coefficients $\{\lambda\}$ and the polynomial functions $\mathcal{P}(\mathbf{x})$ are chosen to satisfy the N fitting conditions

$$s(\mathbf{x}_j) = f(\mathbf{x}_j), \quad j = 1, \ldots, N$$

*Received by the editors June 5, 2006; accepted for publication (in revised form) October 24, 2006; published electronically September 28, 2007.
http://www.siam.org/journals/sisc/29-5/66208.html
†Institute for Advanced Computer Studies (UMIACS), University of Maryland, College Park, MD 20742 (gumerov@umiacs.umd.edu).
‡Department of Computer Science and Institute for Advanced Computer Studies (UMIACS), University of Maryland, College Park, MD 20742 (ramani@umiacs.umd.edu).

and the constraints

$$\sum_{j=1}^{N} \lambda_j Q(\mathbf{x}_j) = 0,$$

for all polynomials Q of degree at most $K-1$.

Let $\{P_1, P_2, \ldots, P_k\}$ be a basis for the polynomials of total degree $K-1$. Then the corresponding system of equations that needs to be solved to find the coefficients λ and the polynomial $\mathcal{P}$ are

$$\begin{pmatrix} \Phi & P^T \\ P & 0 \end{pmatrix} \begin{pmatrix} \lambda \\ b \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix},$$

where

$$\Phi_{ij} = \phi(\mathbf{x}_i - \mathbf{x}_j),\ i,j = 1,\ldots,N; \quad P_{ij} = P_j(\mathbf{x}_i),\ i = 1,\ldots,N;$$
$$j = N+1,\ldots,N+K.$$

Usually the degree K of the polynomial is quite small, and for functions exhibiting linear growth at infinity it can be taken to be 1, in which case the polynomial $\mathcal{P}(\mathbf{x})$ is just a constant, and this is the case that will be considered in what follows. Micchelli [24] has shown that this system of equations is nonsingular.

For large N, as is common in many applications, the computational time and memory required for the solution of the fitting equations is a significant bottleneck. Direct solution of these equations requires $O\left(N^3\right)$ time and $O\left(N^2\right)$ memory. Iterative approaches to solving this set of symmetric equations require evaluations of dense matrix vector products $\Phi\lambda$ which have a $O(N^2)$ computational cost and require $O\left(N^2\right)$ memory. Furthermore, for rapid convergence of the iteration, the system must be preconditioned appropriately.

Considerable progress in the iterative solution of multiquadric and polyharmonic RBF interpolation was made with the development of preconditioners using cardinal functions based on a subset of the interpolating points. This approach was first proposed by Beatson and Powell [1]. Various versions of the idea have since been tried including those based on choosing a set of data near each point, a mix of near and far points, and via modified approximate cardinal functions that have a specified decay behavior [3]. An alternative group of methods that attempt to speed up the algorithm via domain decomposition have also been developed [4].

Recently, an algorithm was published by Faul, Goodsell, and Powell [13], which uses the cardinal function idea, and in which for each point a carefully selected set of q points was used to construct the preconditioner. In informal tests we found that this algorithm appears to achieve a robust performance, and was able to interpolate large problems in a few iterations, where other cardinal function based strategies sometimes took a large number of steps, or failed to converge. In subsequent discussions we will refer to this algorithm as FGP05. This algorithm reduces the number of iterations required; however, its cost remains at $O(N^2)$ and it requires $O(N^2)$ memory. The purpose of this paper is the acceleration of this algorithm.

Similar to many other iterative techniques, the FGP05 algorithm can be accelerated by the use of the fast multipole method (FMM) for the matrix vector product required at each step, making this cost $O(N \log N$ [14]. While the possibility of the use of the FMM was mentioned in [13], results were not reported. Fast multipole methods have been developed for various radial basis kernels including the polyharmonic [5] and multiquadric [11] in $\mathbb{R}^d$, the biharmonic and polyharmonic kernels in

$\mathbb{R}^3$ [19], the polyharmonic [5] kernels in $\mathbb{R}^4$ [6], and Gaussian kernels [17, 26]. Use of the FMM can thus reduce the cost of the matrix vector multiply required in each iteration of the FGP05 algorithm to $O(N \log N)$.

However, there is a further barrier to achieving a truly fast algorithm using the FGP05 algorithm, as there is a precomputation stage that requires the construction of approximate cardinal function interpolants centered at each data point and q of its carefully selected neighbors ($q \ll N$). These interpolants are used to provide a set of directions in the Krylov space for the iteration. The procedure for these preliminary calculations reported in [13] has a formal complexity of $O(N^2)$. Progress towards reducing the cost of this step to $O(N \log N)$ was reported by Goodsell in two dimensions (2D) [18]. However, his algorithm relied on Dirichlet tessellation of the plane and does not extend to higher dimensions.

In this paper we present a modification to this preliminary set up stage, use the FMM for the matrix vector product, and reduce the complexity of the overall algorithm. Section 2 provides some preliminary information on preconditioning with cardinal functions, details of the Faul, Goodsell, and Powell algorithm, and some details of the FMM that are necessary for the algorithms developed in this paper. Section 3 describes the two improvements that let us achieve improved performance in our modified algorithm: finding the q closest points to each point in the set in $O(N \log N)$ time and use of the FMM. Since the latter method is well discussed in other literature, we only provide some details that are unique to our implementation. In particular, we present novel FMM algorithms for the multiquadric (and the inverse multiquadric) in $\mathbb{R}^2$. Section 4 provides results of the use of the algorithm. Section 5 concludes the paper.

2. Background.

2.1. Preconditioning using approximate cardinal functions. A long sequence of papers [1, 3, 4] use approximate cardinal functions to create preconditioners for RBF fitting. The basic iterative method consists at updating at each iteration step the approximation to the function f, $s^{(k)}$

$$s^{(k)}(\mathbf{x}) = \sum_{j=1}^{N} \lambda_j^{(k)} \phi\left(|\mathbf{x} - \mathbf{x}_j|\right) + a^{(k)}, \quad \mathbf{x} \in \mathbb{R}^d, \tag{2.1}$$

by specifying new coefficients $\lambda_j^{(k)}$, $j = 1, \ldots, N$ and the constant $a^{(k)}$. The residual at step k is obtained as

$$r_i^{(k)} = f(\mathbf{x}_i) - s^{(k)}(\mathbf{x}_i), \quad i = 1, \ldots, N,$$

at the data points $\mathbf{x}_i$. To establish notation, and following FGP05, we denote as s^* the function that arises at the end of the iteration process.

The cardinal or Lagrange functions corresponding to any basis play a major role in interpolation theory. The cardinal function z_l corresponding to a particular data point l satisfy the Kronecker conditions

$$z_l(\mathbf{x}_j) = \delta_{lj}, \quad j, l = 1, \ldots, N, \tag{2.2}$$

i.e., z_l takes on the value unity at the lth node, and zero at all other nodes. It can itself be expressed in terms of the radial-basis functions

$$z_l(\mathbf{x}) = \sum_{j=1}^{N} \zeta_{lj} \phi\left(|\mathbf{x} - \mathbf{x}_j|\right) + a_l, \tag{2.3}$$

where the ζ_j^i are the interpolation coefficients and a_l the corresponding constant. If we now form a matrix of all the interpolation coefficients $T_{ji} = \zeta_{ij}$, then we have

$$\begin{pmatrix} \Phi & \mathbf{1}^T \\ \mathbf{1} & 0 \end{pmatrix} \begin{pmatrix} T \\ \mathbf{a} \end{pmatrix} = \begin{pmatrix} I \\ 0 \end{pmatrix}, \tag{2.4}$$

where I is the identity matrix, and $\mathbf{1}$ is a row vector of size N containing ones, and $\mathbf{a}$ a row vector containing the coefficients a_l. Thus the matrix of interpolation coefficients corresponding to the cardinal functions is the inverse of the matrix Φ, and if these coefficients were known, then it would be easy to solve the system. On the other hand, computing each of these sets of coefficients for a single cardinal function z_l (corresponding to a column of T) by a direct algorithm is an $O\left(N^3\right)$ procedure, and there are N of these functions.

The approach pursued by Beatson and Powell [1] was to compute [illegible] to these cardinal functions so that a column ζ_{lj} is approximated by a sparse set of coefficients, say q in number, corresponding to the cardinality condition being satisfied at the point l and $q-1$ other chosen points. Then the matrix $\hat{T}$ corresponding to the approximate cardinal function coefficients is used as a preconditioner in an appropriate Krylov subspace algorithm. In this case the complexity of the preconditioning step is $O(Nq^3)$ and for $q \ll N$, and independent of N, this strategy, provided it converges rapidly, can lead to a fast algorithm.

Early versions of these algorithm were applied to relatively modest sized problems and the points at which the cardinality conditions were satisfied were chosen as neighbors of the point corresponding to the row index. This strategy was observed to not work for larger sets and various modifications were tried. The algorithm of Faul, Goodsell, and Powell [13] presents a variation that works on all tried data sets. It is described in the next section.

[illegible] 2.1. The FMM applies to matrices whose entries can be characterized as radial basis functions. For smoothly varying functions, the approximate cardinal functions are likely to be good approximations to the true cardinal functions. As such, the idea of using approximate cardinal functions to construct preconditioners may be of practical use in a large number of problems where the FMM can be used to accelerate matrix vector products by using the radial basis function itself, or in the singular case, some appropriately mollified version of the FMM radial basis function.

2.2. Algorithm of Faul, Goodsell, and Powell.

2.2.1. Choice of points for fitting cardinal functions. The FGP05 algorithm implements a Krylov method for the solution of (1.1). In this algorithm the point sets on which the approximate cardinal functions are constructed are chosen in a special way, so that a rapidly converging preconditioned conjugate gradient algorithm is achieved. This algorithm was extensively tested in [13] and its convergence properties established theoretically. We also tested our implementation of it practically and found it to exhibit remarkably fast and robust convergence.

Each row of the matrix has associated with it a particular set of q or fewer points. The $N+1 \times N+1$ system of equations (2.4) can be expressed as an equivalent $N-1 \times N-1$ system by eliminating λ_n from the system using

$$\lambda_n = -\sum_{j=1}^{N-1} \lambda_j.$$

To build a preconditioner we need to consider $N-1$ sets of approximate RBF interpolants and associated points $\mathcal{L}_l$, $l = 1, \ldots, N-1$, which we call L-sets. Each set contains the point corresponding to the row index we are working with and other points. To choose the first of these sets, using the interpoint distances, the points that are closest to each other chosen, and one of these points is marked as the first point; $q-2$ further closest neighbors of the marked points are chosen, and the marked point is removed from consideration. In the remaining $N-1$ points, again the closest pair of points is chosen, one of these points is marked, and $q-2$ further closest neighbors of the marked points are chosen. At each step a marked point is selected from the closest pair, a further set of closest points chosen, and a set for the marked point built. In this way the algorithm proceeds to build sets of q points, and the procedure continues till only q points remain, which forms the $(N-q+1)$th set. A further $q-2$ sets of points are constructed from these points by taking $q-1$, $q-2, \ldots, 2$ points from the remaining points, and in each case setting one of these points as the marked point and removing it. In this way a total of $N-1$ groups of points are established.

The procedure described above, though indicated as preferred, was not what is ultimately implemented in FGP05. Rather, an approximate version was used. However, the procedure is nonetheless $O(N^2)$, as it involves computation of pairwise distances between points. A further crucial aspect of the FGP05 procedure is that at each step the algorithm requires deletion of points being considered making the process of constructing the list of closest points a dynamic one.

Now using these point sets, radial-basis function fits of the Lagrange function over these point sets, of the form

$$\hat{z}_l(\mathbf{x}) = \sum_{j \in \mathcal{L}^l} \zeta_{lj} \phi(\mathbf{x} - \mathbf{x}_j) + \text{const.}, \quad \hat{z}_l(\mathbf{x}_j) = \delta_{lj}, \quad l = 1, \ldots, N-1,$$

are constructed. The coefficients of the Lagrange function fit ζ_{lj} are used to define the search directions in a conjugate gradient algorithm that is described below. In this algorithm, the function $\tilde{z}_l$ (defined as $\hat{z}_l$ with the constant part dropped) is used,

$$\tilde{z}_l(\mathbf{x}) = \sum_{j \in \mathcal{L}^l} \zeta_{lj} \phi(\mathbf{x} - \mathbf{x}_j).$$

By construction

$$\sum_{j \in \mathcal{L}^l} \zeta_{lj} = 0.$$

2.2.2. Seminorm and inner product. Given functions s and t at the data interpolation locations $\{\mathbf{x}\}$, and having these functions interpolated using radial basis functions as

$$s(\mathbf{x}) = \sum_{j=1}^{N} \lambda_j \phi(\mathbf{x} - \mathbf{x}_j) + \alpha, \qquad t(\mathbf{x}) = \sum_{i=1}^{N} \mu_i \phi(\mathbf{x} - \mathbf{x}_i) + \beta,$$

FGP05 define the following seminorm

$$\|s\|_\phi = -\left(\lambda^T \Phi \lambda\right)^{\frac{1}{2}}, \qquad \|t\|_\phi = -\left(\mu^T \Phi \mu\right)^{\frac{1}{2}}$$

and inner product between two functions as

$$\begin{aligned}\langle s,t\rangle_\phi &= \frac{1}{2}\left(\|s+t\|_\phi^2 - \left(\|s\|_\phi^2 + \|t\|_\phi^2\right)\right)\\ &= -\frac{1}{2}\left((\lambda+\mu)^T\Phi(\lambda+\mu) - \left(\lambda^T\Phi\lambda\right) - \left(\mu^T\Phi\mu\right)\right)\\ &= -\frac{1}{2}\left(\lambda^T\Phi\mu + \mu^T\Phi\lambda\right) = -\sum_{j=1}^{N}\sum_{i=1}^{N}\lambda_i\phi(\mathbf{x}_i-\mathbf{x}_j)\mu_j = -\sum_{i=1}^{N}\lambda_i t(\mathbf{x}_i)\\ &= -\sum_{j=1}^{N}\mu_j s(\mathbf{x}_j).\end{aligned}$$

If the function values at the interpolation points are known, these inner products can be computed without any matrix-vector products, i.e., in $O(N)$ operations.

2.2.3. Krylov subspace. The Krylov subspace considered is that induced by the operator Ξ acting on functions s that are expressible via a RBF expansion. This operator is defined using the $N-1$ approximations to the cardinal functions centered at the $N-1$ points that are constructed $\tilde{z}_l$ and is given as

$$(\Xi s)(\mathbf{x}) = \sum_{l=1}^{N-1}\frac{\langle\tilde{z}_l, s\rangle_\phi}{\|\tilde{z}_l\|_\phi^2}\tilde{z}_l(\mathbf{x}).$$

2.2.4. The preconditioned conjugate gradient iteration. Let s^* be the function that is sought as the solution. Define an initial guess $s^{(1)}$ as

$$\lambda_j^{(1)} = 0, \quad j=1,\ldots,N; \qquad \alpha_1 = \frac{1}{2}(\min(f)+\max(f)),$$

which leads to the initial residual

$$r_i^{(k)} = f_i - s^{(k)}(\mathbf{x}_i), \qquad \mathbf{r} = \{r_i\}. \tag{2.5}$$

Two functions expressed as RBF sums are used in the iteration:

$$\begin{aligned}t^{(k)} &= \sum_{j=1}^{N}\tau_j^{(k)}\phi(|\mathbf{x}-\mathbf{x}_j|), \quad \sum_{j=1}^{N}\tau_j^{(k)} = 0,\\ d^{(k)} &= \sum_{j=1}^{N}\delta_j^{(k)}\phi(|\mathbf{x}-\mathbf{x}_j|), \quad \sum_{j=1}^{N}\delta_j^{(k)} = 0.\end{aligned}$$

The first function, t, is constructed using the RBF approximation to the cardinal functions on the specially chosen FGP05 points, and the value of the residual at the current iteration at these points is

$$\mu_l^{(k)} = \frac{1}{\zeta_{ll}}\sum_{i\in\mathcal{L}_l}\zeta_{li}r_i^{(k)}, \qquad \tau_j^{(k)} = \mu_l^{(k)}\zeta_{lj}.$$

It is shown that

$$t^{(k)} = \Xi\left(s^* - s^{(k)}\right),$$

while the second function, $d^{(k)}$, satisfies

$$d^{(1)} = t^{(1)}, \quad d^{(k)} = t^{(k)} - \frac{\left\langle t^{(k)}, d^{(k-1)} \right\rangle_\phi}{\left\langle d^{(k-1)}, d^{(k-1)} \right\rangle_\phi} d^{(k-1)}.$$

Because of this definition, the conjugacy criterion holds,

$$\left\langle d^{(k)}, d^{(k-1)} \right\rangle_\phi = \left\langle t^{(k)}, d^{(k-1)} \right\rangle_\phi - \frac{\left\langle t^{(k)}, d^{(k-1)} \right\rangle_\phi}{\left\langle d^{(k-1)}, d^{(k-1)} \right\rangle_\phi} \left\langle d^{(k-1)}, d^{(k-1)} \right\rangle_\phi = 0.$$

The solution is advanced by a step length γ_k along the conjugate direction $d^{(k)}$, so that

$$\lambda_j^{(k+1)} = \lambda_j^{(k)} + \gamma_k d^{(k)}, \qquad \gamma_k = \frac{\sum_{i=1}^{N} \delta_i^{(k)} r_i^{(k)}}{\sum_{i=1}^{N} \delta_i^{(k)} d^{(k)}(\mathbf{x}_i)}$$

while the constant is updated to minimize the maximum element in the new residual vector. The choice γ_k can be shown to be minimize $\left\| s^* - s^{(k+1)} \right\|_\phi$ along $d^{(k)}$.

Crucial to the implementation is the construction of the approximate cardinal functions. It is indicated in [13] that a Fortran implementation is available from Prof. Powell. Our Matlab implementation of the FGP05 algorithm is available online at http://www.umiacs.umd.edu/~ramani/fmm/FGP_RBF.

2.3. The FMM for accelerating matrix vector products. There are several papers and books that introduce the FMM in detail [14, 15, 2, 21], and our purpose is not to reproduce that literature here. However, certain details of geometric data structures in the FMM will be important in the initial computations necessary for the preconditioner. Our implementation of the FMM for polyharmonic radial-basis functions is described in [19], where we also provided operational and memory complexity.

The algorithm consists of two main parts: the preset step and the evaluation step. The preset step, which includes building the FMM data structure (building and storage of the neighbor lists, etc.) and precomputation and storage of all translation data. The data structure is generated using the bit interleaving technique described in [21], which enables spatial ordering, sorting, and bookmarking. While the FMM algorithm is designed for two independent data sets (N arbitrary located sources and M arbitrary evaluation points), for the RBF fitting we will have the same source and evaluation sets of length N. For a problem size N, the cost of building the data structure based on spatial ordering is $O(N \log N)$, where the asymptotic constant is much smaller than the constants in the $O(N)$ asymptotics of the main algorithm. The number of levels could be arbitrarily set by the user or found automatically based on the clustering parameter (the maximum number of sources in the smallest box) for optimization of computations of problems of different size.

Of particular use in developing the $O(N \log N)$ algorithm for the preliminary steps are the octree data structures that are employed in the FMM. A good introduction to these are provided in Greengard's thesis [14]. Fast algorithms for computing neighbor lists, point assignment to boxes and other necessary operation in the FMM using bit-interleaving and deinterleaving operations are described in [21].

3. Fast algorithms. The original form of the FGP05 algorithm has a complexity of $O(N^2)$ due to two reasons: First, the step of setting the data structure requires $O(N^2)$ operations due to pairwise distance search to form $O(N)$ subsets of length $O(q)$ for approximation of cardinal functions centered at each point, and, second, the step of matrix vector multiplication of $O(N \times N)$ matrix by a $O(N)$ vector also requires $O(N^2)$ operations. As mentioned before, the second step can be done using the FMM for $O(N \log N)$ operations. In the subsections below we describe efficient $O(N \log N)$ algorithms for both steps, which bring the FGP iterative method to $O(N \log N)$ complexity.

3.1. Data structure for the FGP05 algorithm. While the problem of determining the closest points in a data set is a relatively well-studied problem in computational geometry, we are unaware of any algorithms that solve the particular problem required in the FGP05 algorithm. As mentioned earlier, a reason for the dramatic performance of the algorithm, as discussed in [13], is the careful selection of the points over which the cardinal function is selected. As such a selection might be useful in the construction of preconditioners with other radial-basis function applications with the FMM, we provide details of this algorithm below.

3.1.1. Problem statement. While the selection of the point sets has been described above in words, we will provide a more precise mathematical statement here, mainly in the interest of establishing notation.

Given a set $\mathbb{X}$ of N points in d dimensional Euclidean space, $\mathbf{x}_i \in \mathbb{R}^d$, $i = 1, \ldots, N$, and $q < N$ build $N-1$ sets $\mathbb{L}_j \subset \mathbb{X}$, $j = 1, \ldots, N-1$ satisfying the following conditions:

- For $j = 1, \ldots, N-q+1$ the sets $\mathbb{L}_j$ consist of q points $\mathbf{x}_{i_{j1}}, \ldots, \mathbf{x}_{i_{jq}} \in \mathbb{X}$ while for $j = N-q+2, \ldots, N-1$ the sets $\mathbb{L}_j$ consist of $N-j+1$ points, $\mathbf{x}_{i_{j1}}, \ldots, \mathbf{x}_{i_{j,N-j+1}} \in \mathbb{X}$. The number of entries in the set is called its "power," so $Pow(\mathbb{L}_j) = q$ for $j = 1, \ldots, N-q+1$ and $Pow(\mathbb{L}_j) = N-j+1$ for $j = N-q+2, \ldots, N-1$.
- Each set $\mathbb{L}_j$ is associated with a point $\mathbf{x}_{i_{j1}} \in \mathbb{L}_j$, called the set center. All set centers are different, $i_{j1} \neq i_{k1}$, for $k \neq j$. Further, we introduce sets of set centers $\mathbb{C}_j = \left\{\mathbf{x}_{i_{11}}, \ldots, \mathbf{x}_{i_{j1}}\right\}$, $j = 1, \ldots, N-1$, $Pow\left(\mathbb{C}_j\right) = j$.
- The set $\mathbb{L}_1$ consists of points $\mathbf{x}_{i_{11}}, \ldots, \mathbf{x}_{i_{1q}} \in \mathbb{X}$ such that

$$|\mathbf{x}_{i_{11}} - \mathbf{x}_{i_{12}}| = \min_{\mathbf{x}_i, \mathbf{x}_k \in \mathbb{X}, i \neq k} |\mathbf{x}_i - \mathbf{x}_k|, \tag{3.1}$$

$$|\mathbf{x}_{i_{11}} - \mathbf{x}_{i_{1l}}| \leq \min_{\mathbf{x}_k \in \mathbb{X} \setminus \mathbb{L}_1} |\mathbf{x}_{i_{11}} - \mathbf{x}_k| \quad \text{for} \quad l = 3, \ldots, q. \tag{3.2}$$

  In other words, the distance between the center of this set and the second element is minimal among all pairwise distances in $\mathbb{X}$. Besides the set center the set consists of the $q-1$ closest points to the set center.
- The set $\mathbb{L}_j$, $j = 1, \ldots, N-1$ consists of points $\mathbf{x}_{i_{j1}}, \ldots, \mathbf{x}_{i_{jp}} \in \mathbb{X} \setminus \mathbb{C}_{j-1}$, $p = Pow(\mathbb{L}_j)$, such that

$$\left|\mathbf{x}_{i_{j1}} - \mathbf{x}_{i_{j2}}\right| = \min_{\mathbf{x}_i, \mathbf{x}_k \in \mathbb{X} \setminus \mathbb{C}_{j-1}, i \neq k} |\mathbf{x}_i - \mathbf{x}_k|, \tag{3.3}$$

$$\left|\mathbf{x}_{i_{j1}} - \mathbf{x}_{i_{jl}}\right| \leq \min_{\mathbf{x}_k \in \mathbb{X} \setminus (\mathbb{L}_j \cup \mathbb{C}_{j-1})} \left|\mathbf{x}_{i_{j1}} - \mathbf{x}_k\right| \quad \text{for} \quad l = 3, \ldots, q. \tag{3.4}$$

  In other words, the distance between the center of this set and the second element is the minimal among all pairwise distances in $\mathbb{X}$ excluding the centers of the sets $\mathbb{L}_1, \ldots, \mathbb{L}_{j-1}$. Besides the set center the set consists of $q-1$ closest

points to the set center for $j = 1, \ldots, N - q + 1$ and $N - j + 1$ closest points to the set center for $j = N - q + 2, \ldots, N - 1$.

3.1. If we construct the complete interpoint distance matrix (requiring computation of $N(N-1)/2$ distances), this problem can be solved simply. However, this requires $O\left(N^2\right)$ time and memory.

3.1.2. General algorithm. There exist fast closest neighbor search algorithms in d dimensional space of complexity $O(\log N)$ per query. These algorithms are based on hierarchical space subdivision using k-d trees or similar ideas [25]. The cost of building a tree data structure is $O(N \log N)$. Furthermore, an initial spatial ordering of the data requires $O(N \log N)$ operations. However, these structures do not enable easy deletion of points.

Deletion is enabled in the heap data structure, which also requires $O(N \log N)$ construction cost [10]. Organization of the data using heaps enables $O(\log N)$ insertion/deletion operations for a single entry. However, even with these preliminary steps the design of the actual algorithm requires several additional data structures which take care of the dynamic nature of the data (the deletions of the set center at each step). These additional structures account for the modifications of the initial set $\mathbb{X}$ due to deletion operations and enable $O(\log N)$ complexity for any operation on the modified data set. Such a performance is needed for these operations, as they must be repeated $O(N)$ times as the cardinal point sets are built for the $N - 1$ centers.

We assume the initial set $\mathbb{X}$ consists of points randomly distributed over the computational domain, which can be enclosed by a d dimensional cube of side D. In practice this set may not be random, as the data may come imbued with a certain order because of the way it was generated. Such an order may not be desirable for the performance of the preconditioner. This is why in [13] the authors suggested an initial random permutation of all initial data. We follow this approach and to avoid additional complexity in notations we assume that such a random permutation is performed as a first step and points in $\mathbb{X}$ are already arranged in random order.

Let $\mathbb{X}'$ be a spatially ordered set of points from $\mathbb{X}$. Spatial ordering in d dimensions, when d is not large (say $d = 2$ or $d = 3$, which is the case for most applications of the FMM) can be performed efficiently using the bit-interleaving technique. We use the same technique to generate the FMM quadtree or octree data structure. A detailed description of the bit-interleaving technique can be found in [22, 21]. The ordered set $\mathbb{X}'$ can be characterized by a permutation index Π. So if $\mathbf{x}_i$ is the ith element of $\mathbb{X}$, then the jth element of $\mathbb{X}'$, where $j = \Pi(i)$ corresponds to the same point. Thus, as Π is available there is no need for additional memory to store $\mathbb{X}'$. The spatially ordered list is needed to provide the following operations with a complexity $O(\log N)$ for any point:

1. deletion, and,
2. q-closest point search in the subset of remaining points after the deletion of previously considered point centers.

These procedures are described below separately.

The second virtual set which we introduce is $\mathbb{H}$, which consists of the elements of $\mathbb{X}$ partially ordered according to some ranking criterion $\mathcal{R}$ using the heap data structure [10]. This set also can be characterized completely by the permutation index H. We will operate only with first k elements of $\mathbb{H}$ which thus form a subset $\mathbb{H}^k \subset \mathbb{H}$. The top ($j = 1$) element of $\mathbb{H}^k$ has the highest rank. Elements below this element are arranged in a binary tree, with each leaf being outranked by its parent ("heap property"). The heap structure is needed to provide the following operations

of complexity $O(\log N)$:

1. find the highest ranked element,
2. delete the highest ranked element,
3. change the rank of an element.

After performing any of these operations, the heap structure (characterized by a permutation index H) can be recovered for a cost of $O(\log N)$ operations. The ranking which we introduce can be applied only to elements of $\mathbb{H}^k$. The rank $\mathcal{R}_i$ of element $\mathbf{x}_i \in \mathbb{H}^k$ is the nearness of the point to the set center. So the closest neighbor to the top element of $\mathbb{H}^k$ is closer (or at the same distance) than the closest neighbors to other points from $\mathbb{H}^k$.

We need to introduce two more lists or indices. The first list can be denoted as C and $C(i)$ is the index of the closest neighbor to $\mathbf{x}_i$ if both these points are in $\mathbb{H}^k$. The second list can be denoted as C^*, where

$$C\left(C^*(j,i)\right) = i \quad \text{for} \quad j = 1, \ldots, M_i^k. \tag{3.5}$$

In some sense this mapping is inverse to $C(i)$. However, as the closeness relationship is not unique, a point $\mathbf{x}_i$ can be the closest neighbor to several points. Since this mapping is not one-to-one, $C^*(:,i)$ provides the indices of all M_i^k points to which $\mathbf{x}_i$ is the closest neighbor. To bound the size of the array (and the memory and operations), we need to bound the quantity M_i^k. There exists a strong bound for M_i^k, which shows that for a given distribution of points in $\mathbb{R}^d$ the number of entries in C^* is $O(N)$. We present this result in the form of the following theorem.

THEOREM 3.2. [illegible] N [illegible] $\mathbb{R}^d$, $(d \geq 1)$ [illegible] $\mathbf{x}_i \in \mathbb{R}^d$ $i = 1, \ldots, N$ [illegible] C [illegible] $i = 1, \ldots, N$ [illegible] M_i [illegible] i [illegible] $M_i \leq K_{(d)}$ [illegible] $K_{(d)}$ [illegible] d [1]

[illegible]. Without any loss of generality we can assign index $i = 1$ to a given point and assign indices $2, \ldots, M+1$ to points for which this point is the closest neighbor. Denote $D_j = |\mathbf{x}_j - \mathbf{x}_1|$ and $D_{\max} = \max_{j=2,\ldots,M+1} D_j$. Consider then a sphere of radius $D_{\max}$ centered at $\mathbf{x}_1$. For each point $\mathbf{x}_j$ we will correspond point $\mathbf{y}_j$ on the sphere surface: $\mathbf{y}_j = \mathbf{x}_1 + (D_{\max}/D_j)(\mathbf{x}_j - \mathbf{x}_1)$, $j = 2, \ldots, M+1$. Let us show that $|\mathbf{y}_j - \mathbf{y}_k| \geq D_{\max}$ for $j, k = 2, \ldots, M+1$, $j \neq k$. Indeed, since for these j and k we have $|\mathbf{x}_j - \mathbf{x}_k| \geq \max(D_j, D_k)$ (otherwise $\mathbf{x}_1$ is not the closest point to $\mathbf{x}_j$ or $\mathbf{x}_k$), we obtain

$$\begin{aligned} 2(\mathbf{x}_j - \mathbf{x}_1) \cdot (\mathbf{x}_k - \mathbf{x}_1) = |\mathbf{x}_j|^2 + |\mathbf{x}_k|^2 - |\mathbf{x}_j - \mathbf{x}_k|^2 &\leq D_j^2 + D_k^2 - [\max(D_j, D_k)]^2 \\ &\leq D_j D_k. \end{aligned}$$

Therefore,

$$\begin{aligned} |\mathbf{y}_j - \mathbf{y}_k|^2 &= |\mathbf{y}_j - \mathbf{x}_1|^2 + |\mathbf{y}_k - \mathbf{x}_1|^2 - 2(\mathbf{y}_j - \mathbf{x}_1) \cdot (\mathbf{y}_k - \mathbf{x}_1) \\ &= 2D_{\max}^2 - \frac{D_{\max}^2}{D_j D_k} 2(\mathbf{x}_j - \mathbf{x}_1) \cdot (\mathbf{x}_k - \mathbf{x}_1) \geq 2D_{\max}^2 - D_{\max}^2 = D_{\max}^2. \end{aligned}$$

Now we see that there are M points on the sphere surface the pairwise distance between which is not less than the radius of this sphere. Introducing spheres of radius $R = D_{\max}/2$ centered at $\mathbf{x}_1$ and $\mathbf{y}_j$, $j = 2, \ldots, M+1$, we can see that this is equivalent

[1] The kissing number is defined as the maximum number of unit spheres which can surround a given unit sphere without intersection (excluding the touching of points on the boundary) [30].

to placing M spheres of radius R around the sphere of the same radius centered at $\mathbf{x}_1$. It is well known that this number cannot exceed the kissing number. □

[illegible] 3.3. The exact kissing number is well known for dimensions $d \leq 9$ and $d = 24$, while for other dimension bounds for $K_{(d)}$ are established. Particularly, we have $K_{(1)} = 2$, $K_{(2)} = 6$, $K_{(3)} = 12$.

The procedure of building sets satisfying the above requirements can then be reduced to procedures of q-closest neighbor search and deletion of found set centers from the data structures.

[illegible]

- Build a hierarchical space subdivision using the 2^d-trees and initialize data structure for neighbor search and deletion procedures.
- Set $k = N$; $M_i^k = 0$, $i = 1, \ldots, N$.
- For $j = 1 : N$
  1. Get the closest neighbor for point j (let this be i). This is provided by the q-neighbor search procedure for $q = 1$.
  2. Set $C(j) = i$, $d(j) = |\mathbf{x}_j - \mathbf{x}_i|$, $M_i^k = M_i^k + 1$, $C^*\left(M_i^k, i\right) = j$.
- Based on ranking $\mathcal{R} = -d$, build the heap permutation index H.

[illegible]

- For $i = 1 : N - 1$
  1. Find the size of set $\mathbb{L}_i$: $q' = \min(q, N - i + 1)$.
  2. Find the index of the set center: $j = H(1)$. This is the top ranked point in the heap. Find the index of the closest to j neighbor: $m = C(j)$.
  3. Find all the entries of $\mathbb{L}_i$. The first entry is $\mathbf{x}_j$ and the rest are $q' - 1$ closest neighbors, provided by the q-neighbor search procedure for $q = q' - 1$.
  4. Delete point j from the hierarchical data structure for neighbor search. Delete point j from the heap. The last procedure sets the heap length to $k = k - 1$ and updates the heap permutation index H.
  5. Modify M_m^k and $C^*\left(M_m^k, m\right)$. As m is the closest to j it is listed in $C^*(:, m)$. So set $M_m^k = M_m^k - 1$ and squeeze $C^*(:, m)$ to exclude j (this will change the indexing in $C^*(:, m)$).
  6. Update lists C and C^* and the heap permutation index H as follows.
  7. For $n = 1 : M_j^k$
     (a) Get the closest neighbor, s, for point $p = C^*(n, j)$. This is provided by the q-neighbor search procedure for $q = 1$.
     (b) Set $C(p) = s$, $d(p) = |\mathbf{x}_p - \mathbf{x}_s|$, $M_s^k = M_s^k + 1$, $C^*\left(M_s^k, s\right) = p$.
     (c) Update the heap permutation index for new rank of point p, $\mathcal{R}(p) = -d(p)$.

3.1.3. Data structure for deletion operation and q-closest neighbor search using 2^d-trees. For fast neighbor search we employ 2^d-tree data structures, which for $d = 2$ and $d = 3$ are known as the quadtree and octree, respectively [27]. In the 2^d-tree we have one box at level 0 (which is a cube including all points from $\mathbb{X}$), 2^d cubical boxes at level 1 obtained by division by half of the initial (parent) box along each dimension, and so on up to level $l_{\max}$, where we have $2^{l_{\max} d}$ boxes. As the data are sorted using the bit-interleaving techniques, they can be easily boxed due to the hierarchical numbering system [21], which also provides a fast way to obtain parents and neighbors for all boxes. Of course, not all the boxes may be occupied, and the "empty" boxes are not listed in the data structure. Selection of $l_{\max}$ is dictated by the parameter q, and one of the possibilities to determine $l_{\max}$ is to define it as the level at

which the finest box contains not more than $q+1$ points. For every box we maintain a list of points which reside in this box. This list is dynamic and provided by an additional data structure described below. The same relates to the list of nonempty boxes neighboring to the given box. For the neighbor search procedure we use the so called 2-neighborhoods [22], which are cubic domains consisting of at most 5^d boxes where $5^d - 1$ neighbor boxes surround the central (given) box.

The selection of 2-neighborhoods is motivated by the space dimensions $d = 2$ and $d = 3$, which are the focus of the present paper. Assume that there are $q + 1$ points located in some box, then for $d = 2, 3$ we can guarantee that for any point in the dataset, its q closest (Euclidean) neighbors are located inside the 2-neighborhood. Indeed, the maximum distance between some point and its qth closest neighbor does not exceed $Dd^{1/2}$, where D is the box size. At the same time the minimum distance from this point to the boundary of the 2-neighborhood is $2D$. As $d^{1/2} \leq 2$ (which also means that the 2-neighborhood can be used for $d = 4$) we have such a guarantee and can use the search algorithm described below.

A complication is that the general algorithm above needs an efficient deletion procedure, so the search required to form the ith set $\mathbb{L}_i$ is performed only on the set $\mathbb{X}\backslash\mathbb{C}_{i-1}$. We resolve this issue by introducing linked lists [10] for all points and boxes in the neighborhood of each box. The point linked list $L_p\,(2, N)$ is an array, where $L_p\,(1, i)$ is the index of point from the ordered set $\mathbb{X}'$ preceding point i and $L_p\,(2, i)$ is the index of point next to i. This list is initialized by setting $L_p\,(1, i) = i - 1$ and $L_p(2, i) = i + 1$ and is updated each time as some point i should be deleted. Note that all points $\{i\}$ residing in box j can be characterized by the indices of the first and last point in this box (bookmarks), as the point linked list is available.

The data structure for handling the neighborhoods is the following. For each box i we have a list of boxes in the 2-neighborhood of this box, $N_{neigh}\,(:, i)$, where the first index takes values from 1 to 5^d. The boxes in this neighborhood are linked in a similar way as the points in the original set. This is provided by the linked lists $L_n\,(:, i)$, where $L_n\,(1, i)$ is the relative index of the first element, $L_n\,(2, i)$ is the link from the first element to the second, and so on. This array is initialized as $L_n\,(m, i) = m$, $m = 1, \ldots, N_n\,(i)$, $i = 1, \ldots N_{b,tot}$, where $N_n\,(i)$ is the number of boxes in the neighborhood of box i and $N_{b,tot}$ is the total number of boxes. For conveniences we also maintain the arrays $N_p\,(i)$, which shows the number of points in box i, $B(j)$, which is the index of the box containing point j, and $N_{pn}\,(i)$, which is the total number of points in the neighborhood of box i.

With these arrays the deletion of point j (index in $\mathbb{X}'$) and the q-closest neighbor search procedures for point j (index in $\mathbb{X}'$) can be described by the following algorithms.

- Set $i = B(j)$; $n_p = N_p(i) - 1$; exit if $n_p < 0$ (nothing to delete), else:
- Update the linked list L_p: $a = L_p\,(1, L_p(j))\,, b = L_p\,(2, L_p(j))$, $L_p\,(1, L_p(j)) = b, L_p\,(2, L_p(j)) = a$ (note that this should be modified for the first and last points; also we need to update the bookmarks if the deleted point is the first or the last point in the box); $N_p(i) = n_p$.
- Set $l = l_{\max}$ and perform the following loop while $l \geq 1$ and $n_p = 0$ (this procedure excludes all boxes, which become empty due to point deletion):
  1. For $m = 1 : N_n(i)$ (go through all boxes which are in the neighborhood of a box i)
     a. $p = N_{neigh}(L_n(m, i), i)$ $(p \neq i)$; $N_{pn}\,(p) = N_{pn}\,(p) - 1$. Update the linked lists $L_n\,(:, p)$.

2. $l = l - 1$; $i = Parent(i)$; $n_p = N_p(i) - 1$; $N_p(i) = n_p$ (also check/update bookmarks for box i if $n_p > 0$).

For the sake of clarity, we drop details of our procedure for updating $L_n(:, p)$, which are rather obvious, as one should reconnect the broken links caused due to the exclusion of box i. Note then that the major complexity of this algorithm is due to the loop over all boxes, for which a given box i lie in its neighborhood. This requires at most $\left(5^d - 1\right)^2$ simple operations (repairing links). Also it may appear that this operation is needed for all parent boxes. If $l_{\max} \sim \log N$ the total cost will be $O(\log N)$, with a remark that the updates of the neighborhood data structure are needed only when the deletion of a point results in an empty box, which then should be deleted. This situation appears relatively rarely. Otherwise, the deletion of point is computationally cheap as it requires just repairing a broken link (2 operations) and a check/update of bookmarks (2 operations).

- Set $i = B(j)$, $l = l_{\max}$, $n_p = N_p(i)$.
- While $k < q + 1$ and $l > 1$ perform the following loop (determine the level at which the box containing point j has at least $q + 1$ entries):
  1. $l = l - 1$; $i = Parent(i)$; $k = N_p(i)$.
- Determine $D_{\max}$ and $D_{\min}$ which are the maximum and minimum distances from j to the boundary of box i.
- Determine all distances from point j to points in box i, which are different from j. Put them into the distance array *dist* and sort in the ascending order.
- If $dist(q) > D_{\min}$, perform the following procedure for boxes $p \neq i$ from the 2-neighborhood of i:
  1. Determine $D'_{\min}$, which is the minimum distance from j to the boundary of box p.
  2. If $D'_{\min} < D_{\max}$, determine all distances from point j to points in box p. Union them with array *dist*. Sort *dist* in the ascending order.

An informal description of the algorithm is the following. First, we are trying to find a box big enough to contain q neighbors of point j. For this purpose we consider the parent of the box, if the box does not contain a sufficient number of points. This procedure is finite (at most $\sim l_{\max} \lesssim \log N$). After such a box is found we introduce two threshold distances $D_{\max}$ and $D_{\min}$ as defined above. It may happen that the closest q points to j occupying box i are closer than $D_{\min}$. In this case there is no need to check points from the neighborhood, since they are further than the q points, which provide the solution to the problem. On the other hand, we know that there are at least q neighbors located at distance smaller than $D_{\max}$. Hence there is no need to check the boxes from the neighborhood with $D'_{\min} \geq D_{\max}$. This constraint substantially reduces the number of boxes to check. For example, for $d = 2$ we should check at most 11 neighbor boxes instead of 24, and for $d = 3$ we have 55 instead of 124 ($5^d - 1$). The numbers in practice are even smaller, as not all points are located in the worst position.

The above algorithm does not have $O(\log N)$ guaranteed complexity for a single point search and an arbitrary distribution. For very nonuniform distributions, additional research is needed as the cost of search can substantially vary from point to point and it is not clear theoretically what is the overall complexity of the above adaptive scheme.[2] However, guarantees can be provided for uniform and some other

[2]This remark applies equally to the FMM, where for carefully contrived distributions the $O(N \log N)$ complexity may not be achieved.

typical distributions. Indeed, in the case of a uniform distribution we can assume that all boxes in the neighborhood has approximately the same number of points $\sim q$. Therefore the search is performed on the set of size $O(5^d q)$, which does not depend on N, and so the $\log N$ complexity arises only from the first step of the procedure (in practice for larger q this step is cheaper than the computation of distances due to its small asymptotic constant).

3.2. Fast multipole acceleration of the matrix vector product.

3.2.1. Polyharmonic kernels in 3D. An approach for efficient computation of sums with biharmonic kernel in $\mathbb{R}^3$, $\phi(r) = r$, using the FMM is described in detail in [19], where we also show how this method can be extended to computation of polyharmonic kernels in $\mathbb{R}^3$, $\phi(r) = r^{2n+1}$. The major result there can be formulated as follows. The FMM for kernels $\phi(r) = r^{2n+1}$, $n = 0, 1, \ldots$, can be efficiently reduced to application of $n+2$ FMMs for the Laplace equation in $\mathbb{R}^3$ (with kernel $\phi(r) = r^{-1}$) for which the FMM is well developed and studied. In practice the cost for the polyharmonic kernel is even smaller, as the same data structure is needed, direct summation in the neighborhood is only needed once, and only a slight modification of the translation operators is needed.

The efficiency of the FMM is determined by the length of function representation, which for polyharmonic kernel $\phi(r) = r^{2n+1}$ can be referred to as $(n+2)\,p^2$, where p is the truncation number for the equivalent Laplace equation determined by the prescribed accuracy of computations, and the cost of a single translation, which for the fastest practical methods available is $O(p^3)$ (with different asymptotic constants [9, 20]); ($O(p^2 \log p)$ methods with larger asymptotic constant are also available, e.g., [16, 12]). In tests used in the present paper we employed our version for summations with the biharmonic kernel [19], where we used rotational-coaxial translation decomposition of the translation operators, which result in $O(p^3)$ complexity for a single translation.

3.2.2. A novel FMM for the multiquadric kernel in $\mathbb{R}^2$. An FMM for the multiquadric kernel, $\phi(r) = \left(r^2 + c^2\right)^{1/2}$, in $\mathbb{R}^n$ was presented in [11]. While this algorithm can be used for the multiquadric function in $\mathbb{R}^2$, we present a new FMM based approach to evaluate sums with the multiquadric kernel.

First, we note that representation of general functions via expansion coefficients or samples in 2D requires double sums or $O(p^2)$ function representation length for a truncation number p (e.g., via truncated Taylor series expansions). In contrast functions that are constrained to satisfy some partial differential equation, such as the harmonic function, can be expressed with just $O(p)$ coefficients in $\mathbb{R}^2$. General translation operators for functions with a $O(p^2)$ representation can then be represented via a $p^2 \times p^2$ truncated matrix, which results in a computational cost of $O(p^4)$ per translation. For large p, methods that employ the convolutional characteristics of the translation can exploit the fast Fourier transform to reduce the asymptotic complexity $O(p^2 \log p)$. However, these methods will in practice not be useful for typical problems and further suffer an inefficiency as they require zero padding. (See [16] for a discussion of this issue.) The proposed method also uses $O(p^2)$ representation; however, the translation cost can be reduced to $O(p^3)$, which is faster than the direct translation for $p \geq 3$.

The essence of the method is to extend the space dimensionality from 2 to 3 and separate the source and receiver points, even though in $\mathbb{R}^2$ these points can be the same. Indeed, let $\mathbf{x}_i$ and $\mathbf{y}_j$ be the source and receiver in 2D. Their interaction is

then described as

$$\phi\left(|\mathbf{y}_j - \mathbf{x}_i|\right) = \left(|\mathbf{y}_j - \mathbf{x}_i|^2 + c^2\right)^{1/2} = \left|\mathbf{y}'_j - \mathbf{x}'_i\right|; \quad \mathbf{y}'_j = \mathbf{y}_j + c\mathbf{i}_z,\ \mathbf{x}'_i = \mathbf{x}_i + 0\mathbf{i}_z,$$

where $\mathbf{y}'_j$ and $\mathbf{x}'_i$ are points in $\mathbb{R}^3$ and $\mathbf{i}_z$ is the unit vector orthogonal to the original plane on which $\mathbf{x}_i$ and $\mathbf{y}_j$ are located. It is seen then that this transform leaves the sources on the original plane, while it shifts the locations of all evaluation points by a constant displacement c, so $\mathbf{y}'_j$ is located on the plane parallel to the original one (the case $c = 0$ is a particular, but not special, case). Therefore the problem of computation of sums with the multiquadric kernel in $\mathbb{R}^2$ is equivalent to the computation of sums with the biharmonic kernel in $\mathbb{R}^3$ with the sources and receivers having special locations (placed on parallel planes).

REMARK 3.4. The use of an increased dimensionality should not frighten us and, in fact, this does not increase either the memory or the computational cost. Indeed, in the FMM that we use all "empty" boxes are skipped.

REMARK 3.5. As the sets of sources and receivers are separated by distance not less than c there is no sense in dividing the space up to levels where the size of the boxes are less than $c/2$ (if the division goes deeper there will be no sources in the neighborhood of the receivers and the computational cost will just increase due to additional translation operations).

REMARK 3.6. This method can also be successfully applied for computations of sums with the inverse multiquadric kernel $\phi(r) = \left(r^2 + c^2\right)^{-1/2}$ in $\mathbb{R}^2$. Sums involving this kernel can be reduced to a summation with the Laplacian kernel $1/r$ in $\mathbb{R}^3$.

4. Numerical tests. To check the accuracy and performance of the fast algorithms and compare them with [13], we implemented the described algorithms and the FGP05 algorithm in Fortran 90 in double precision arithmetic and performed some tests on an Intel Xeon 3.2 GHz processor with 3.5 GB RAM. Performance tests were conducted using two basic random point distributions for $\mathbb{R}^2$ (with the multiquadric kernel) and $\mathbb{R}^3$ (with the biharmonic kernel). The parameter c for the multiquadric kernel was varied from 0 to a value based on the characteristic distance between neighboring points. The first distribution coincides with test case A of [13]. In this case points were uniformly distributed inside a d dimensional unit sphere, we also refer to this case as "A." The second distribution is substantially nonuniform, as the points in this case were uniformly distributed over the surface of a d dimensional unit sphere and so were on manifolds of lower than d dimensions. Such distributions are typical for interpolation of curves and surfaces. We refer to this case as "E." In both cases the entries of the right-hand side vector were generated as random numbers uniformly distributed on $[-1, 1]$. We also performed several illustrative computations for some standard problems appearing in 2D and 3D computer graphics.

4.1. Preconditioner tests. There are new elements in the present algorithm which are related both to construction of the preconditioner (or approximate cardinal functions) and to the approximate FMM based matrix vector multiplication. We first studied the effect of the new preconditioner in isolation in cases where the matrix vector product was performed in the straightforward way.

The first test was to ensure that our implementation of the original FGP05 algorithm is correct. We found that we were able to produce results consistent with those reported in [13]. We next modified the algorithm to use the L-sets produced by our algorithm for selecting the cardinal function interpolation points. The difference between our algorithm and that of [13] is that we impose the selection of closest

TABLE 4.1
Number of iterations required for test problem A of [13]*, implemented using the cardinal point set algorithm of* [13] *and our algorithm.*

Problem	N	$q = 10$		$q = 30$		$q = 50$	
		Present	FGP05	Present	FGP05	Present	FGP05
$d = 2, c = 0$	200	15	16	8	8	7	7
$d = 2, c = 0$	500	18	19	9	9	8	7
$d = 2, c = 0$	1000	18	20	10	9	8	8
$d = 2, c = 0$	2000	21	22	10	10	9	9
$d = 2, c = 0$	5000	23	26	11	11	10	10
$d = 2, c = 0$	10000	25	27	13	12	11	10
$d = 2, c = N^{-1/2}$	200	20	20	8	8	7	7
$d = 2, c = N^{-1/2}$	500	23	24	11	10	9	8
$d = 2, c = N^{-1/2}$	1000	25	26	11	11	9	9
$d = 2, c = N^{-1/2}$	2000	27	28	11	12	9	10
$d = 2, c = N^{-1/2}$	5000	31	33	12	12	11	10
$d = 2, c = N^{-1/2}$	10000	35	36	13	13	11	11
$d = 3, c = 0$	200	22	23	11	12	8	9
$d = 3, c = 0$	500	29	30	14	14	10	11
$d = 3, c = 0$	1000	36	39	17	16	12	11
$d = 3, c = 0$	2000	41	44	19	19	13	13
$d = 3, c = 0$	5000	57	58	23	23	15	15
$d = 3, c = 0$	10000	68	74	26	26	17	17

points exactly where as this is done approximately in FGP05. Table 4.1 shows the number of iterations required to converge to an accuracy $\epsilon = 10^{-10}$ for the test case A for different N and q. For each case we did several tests trying to find averages, the variation from test to test was rather small (usually within 1 iteration) so we put in the table values for a single random realization. The stability of the number of iterations for different realizations was also pointed out in [13].

The results for case $d = 2, c = 0$ are very close to that reported in [13] (we also checked the $q = 20$ and $q = 40$ reported in [13], with similar results, but do not report them here to conserve space). As one can see from the table, the number of iterations for preconditioners produced by both algorithms are almost the same.

The time required to build the L-sets and compute approximate cardinal function approximation is almost insensitive to parameter c of the multiquadric kernel, while it does depend on the point distribution. Table 4.2 shows the CPU time (in seconds) required to build preconditioners using the original and present algorithm for four cases (A and E, $d = 2$ and $d = 3$) for $q = 30$ and different N. These results are also presented in Figure 4.1. As before, we report the timing for a single random realization, while several entries into the table were recomputed several times to ensure that the variations in timing are rather small (within a couple of percent).

It is clearly seen that for $N < N_b$, which is of order of thousands, the original algorithm performs better, as we incur an additional overhead due to the more complex data structure needed in our algorithm. However, for $N > N_b$ the execution time for the original algorithm scales as $O(N^2)$, while the present algorithm scales almost linearly with N. The scaling constant and the value of N at which the present algorithm starts to perform linearly depends on the problem dimensionality and point distribution. The problem dependence of the present algorithm is stronger than for the original algorithm. We also note that due to large time required by the original al-

TABLE 4.2
The time required for initial computations in the FGP05 iterative algorithm.

$N \rightarrow$	300	1000	3000	10000	30000	100000	300000	1000000
(A) $d = 2$:								
Present	0.2	0.7	2.1	7.2	22	81	256	854
FGP05	0.1	0.5	2.2	17	157	2445	(19900)	(221100)
(A) $d = 3$:								
Present	0.2	0.8	3.0	12	43	184	623	2130
FGP05	0.1	0.5	2.1	15	139	2211	(19900)	(221100)
(E) $d = 2$:								
Present	0.2	0.6	1.8	6	18	62	188	623
FGP05	0.1	0.5	2.4	18	177	2816	(19900)	(221100)
(E) $d = 3$:								
Present	0.2	0.8	2.7	9.2	32	110	330	1125
FGP05	0.1	0.6	2.4	17	155	2451	(19900)	(221100)

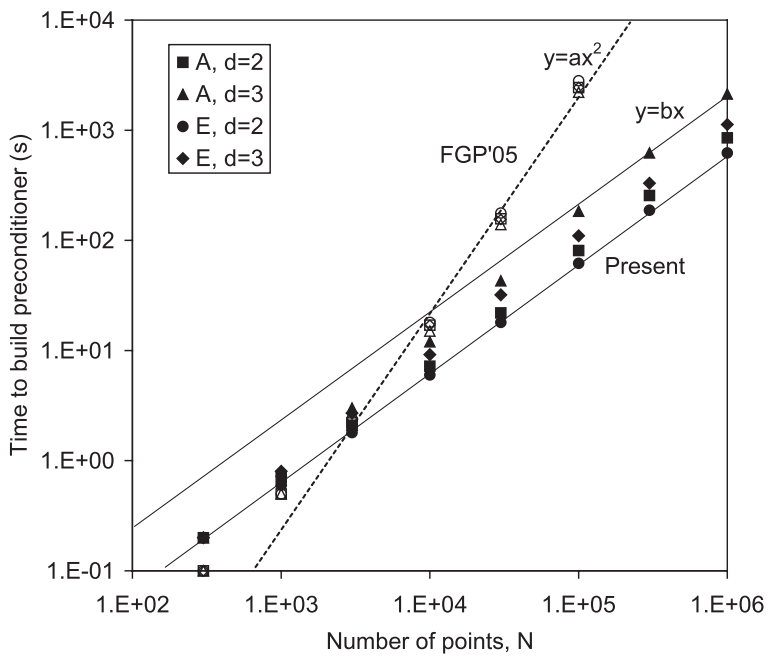


FIG. 4.1. *Dependences of the CPU time for construction of the preconditioner ($q = 30$) for algorithm FGP05 (shown by the light markers) and the present algorithm (the dark markers) for four different problems. The solid lines show the linear function and the dashed line shows the quadratic function in the log-log coordinates.*

gorithm for large N we did not compute the values for $N > 100000$ and put estimated times in parenthesis based on the fit line shown in the figure. It is seen that for large N the ratio of times for the algorithms tested can be several orders of magnitude.

Table 4.1 shows that the number of iterations decreases as the size q of the L-sets increases. Thus, it may be preferable to use a larger q to reduce the iteration time. On the other hand, the time required to build the L-sets and compute cardinal function approximations increases with increasing q, which is the limiting factor to use larger q. This time depends on N and the particular distribution. Theoretically, the time to build the L-sets for a fixed distribution and N should grow almost linearly with

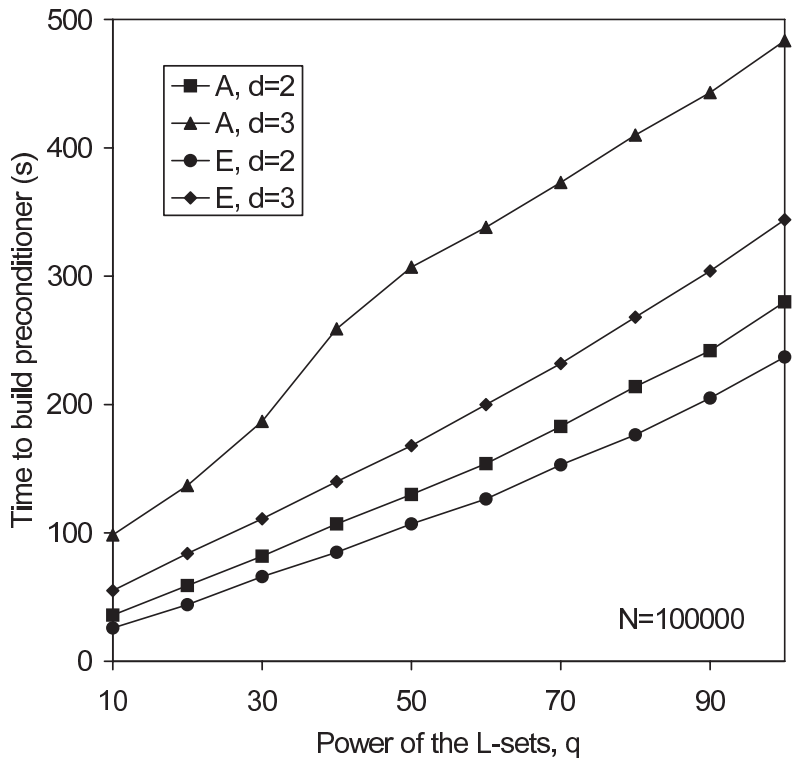


FIG. 4.2. *Dependences of the time required to construct preconditioner for problems A and E on the size of the L-sets q. The number of points for all cases is the same,* $N = 100000$.

q (asymptotically as $q \log q$; the logarithmic complexity appears due to the sorting procedure used for determination of closest neighbors). The time to solve a system of q equations to build the approximate cardinal function grows $\sim q^3$ as we used a standard LAPACK symmetric solver routine for this purpose. Figure 4.2 presents actual computation times for problems A and E and $d = 2$ and 3 at $N = 100000$. It is seen that the time depends on the problem dimensionality and the largest times obtained for $d = 3$. We also note that an effective dimensionality of problem E for large N is $d-1$ rather than d, which explains the smaller times observed for problem E compared to problem A.

4.2. Tests of the algorithm with the FMM. Speeding up the matrix vector product using approximate methods, such as the FMM, raises several issues on the accuracy, convergence, and speed of the algorithm. The accuracy of the FMM is controlled by the truncation number p. For multiquadric functions the error of matrix vector multiplication in the L^∞ norm ϵ_N decays exponentially at growing p, $\epsilon_N \sim N\eta^{-p}$, assuming that they contribute to the error in the worst way (no error cancellation). Actual errors usually are orders of magnitude lower than theoretical bounds and depend on the problem dimensionality and particular point distributions. Concerning convergence of the iterative algorithm we found for given prescribed accuracy empirically numbers $p_*(N)$, $p > p_*(N)$. Here p_* depends on the problem that the number of iterations to converge to any prescribed accuracy is the same as in the case when direct matrix vector multiplication is used. Moreover, the norm of the residuals in this process do not depend on p. Despite this, because of conditioning

TABLE 4.3
Total CPU time required by the present and FGP05 algorithms.

$N \rightarrow$	1000	3000	10000	30000	100000	300000	1000000
(A) $d = 2, c = 0$:							
Present	1.1	3.0	13	43	163	618	2512
FGP05	0.7	4.1	42	424	5477	(49320)	(547700)
(A) $d = 2, c = N^{-1/2}$:							
Present	1.1	3.7	15	50	217	740	3319
FGP05	0.8	4.6	47	430	6463	(49320)	(547700)
(A) $d = 3, c = 0$:							
Present	1.9	5.5	36	126	736	3460	18660
FGP05	0.7	5.5	61	634	5248	(49320)	(547700)
(E) $d = 3, c = 0$:							
Present	1.2	3.6	15	69	324	1152	5332
FGP05	0.8	4.1	38	384	5486	(49320)	(547700)

of the matrix, the solution may converge to a solution that is not exact. The error of this solution can be evaluated for smaller N as we can check it using one straightforward matrix vector multiplication to evaluate the residual and the results can be extrapolated to larger N as we know qualitatively the dependence of the matrix vector product on N and p. Of course, the error criterion used to stop the iterative process should be selected in some consistent way with this error, and this is an item of current research (see, e.g., [28]). This whole issue is also application dependent since it is often the data to be fit are noisy [8] and the fitting equations may themselves be regularized.

Note then the tuning (optimization) of the overall algorithm is not a simple procedure. Indeed, for some prescribed error, the truncation number should be a function of N (e.g., $p \sim \log CN$). Based on p, the optimal level of space subdivision, $l_{\max}$, for the FMM should be selected. The complexity of the FMM for optimal level depends on the complexity of translation procedures. For example, for the rotational-coaxial translation decomposition in some range of p for 3D problems, this can be $O(p^{3/2}N) = O(N \log^{3/2} N)$. Furthermore, one should select an appropriate q for the preconditioner. As the time to build the preconditioner depends on q and the number of iterations decays with q, one can expect a minimum for the cost. We found that there can be several local minima (e.g., we found four for $N = 100000$, $d = 2$, $c = 0$, problem A in range $15 \leq q \leq 30$). Dependence of the number of iterations on N is also important. An accurate addressing of these issues is outside the scope of this paper. Further fundamental studies of the FMM induced matrix perturbations on the accuracy of the overall solution are required, we will further limit ourselves by presentation of several results of the algorithm performance for some test cases. Further, Faul, Goodsell, and Powell [13] indicate that their intuition is that the choice of q should be independent of problem size.

Table 4.3 shows the CPU times required for computation of some test problems of different size. Here the CPU times include both the time to compute the preconditioner and run the iterative process. In all cases the prescribed error was $\epsilon = 10^{-3}$, and p was increasing for increasing N. For example, for test case A, $d = 2$, $c = N^{-1/2}$, we changed p from 16 to 28 as N changed from 1000 to 1000000. The parameter q in all cases was 30 (the number suggested by [13]), which provided an acceptably small number of iterations, except of case A, $d = 3$, $c = 0$, where a substantial growth in the number of iterations was observed for $N \geq 100000$. For these cases we set $q = 100$, which provided a substantially smaller number of iterations and overall savings in the computational time.

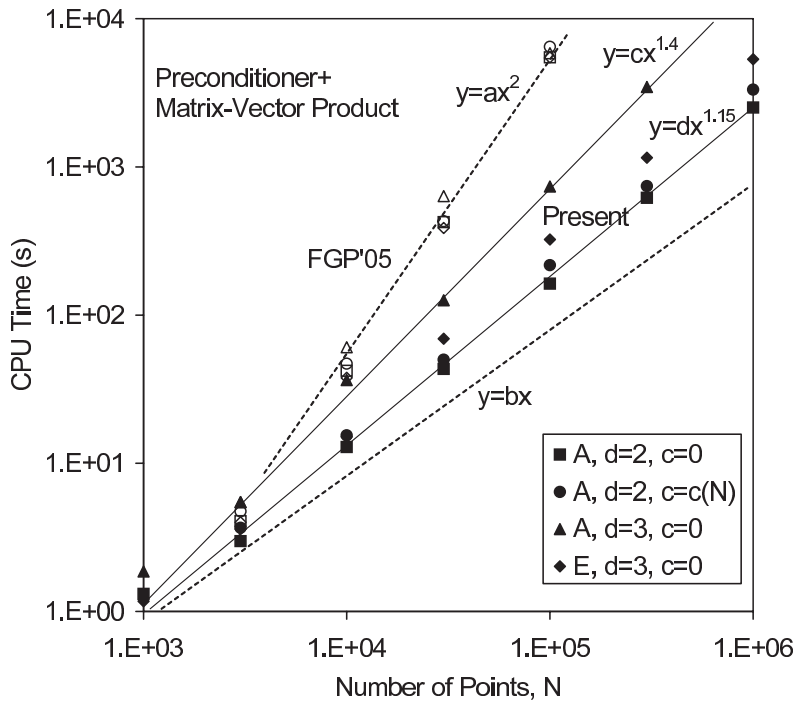


FIG. 4.3. *The total CPU time required for the solution of different problems. The empty markers show computations using algorithm FGP*05*, while the filled markers show computations using the present algorithm. The straight lines with different angular coefficients are plotted for convenience of data analysis.*

Results presented in the table are also plotted in Figure 4.3. The table shows that for all examples the crossover point between the FGP05 and the present algorithm did not exceed $N = 3000$. One can also see that for larger N the CPU time for the FGP05 in the computed range can be fitted by a $O(N^2)$ dependence (in fact, it should be larger than this estimate as the number of iterations does slightly grow with N). We used this fit to estimate the CPU time for the FGP05 algorithm for $N > 100000$, which in fact underestimates the time a bit, and these are shown in Table 4.3 in parentheses. For the present algorithms the data can be fitted by a $O(N^\alpha)$ dependence (one can also use fits of type $O(N \log^\beta N)$, which should hold if the number of iterations grows with N as some power of $\log N$). The constant α for our computations varies from $\alpha = 1.15$ to $\alpha = 1.4$, and as we mentioned there are two basic reasons for this: growth of the truncation number p and of the number of iterations. Also, some computational overhead for larger N contribute to the total CPU time reported. In any case, one can see substantial speed increases achieved using the present method, which make practical computations for million point size sets (larger problems can be computed using parallel computations and larger memory).

Performance of the algorithms for higher accuracy computations is shown in Table 4.4. Here for the FMM computations we set $p = 31$ for all N and the iterative process for both algorithms was terminated when the residual error reached $\epsilon = 10^{-8}$. Other parameters were the same as for cases reported in Table 4.3. For higher accuracy, more iterations are needed and the times reported in Table 4.4 are larger than in Table 4.3. Further, the FMM based algorithms become slower due to an increase of p. This shifts the crossover point for the algorithms towards higher values. Nonetheless,

TABLE 4.4
Total CPU time required by the present and FGP05 algorithms for higher accuracy computations.

$N \rightarrow$	1000	3000	10000	30000	100000	300000	1000000
(A) $d = 2, c = N^{-1/2}$:							
Present	2.3	6.4	28	100	390	1443	6103
FGP05	0.9	6.2	64	665	8696	(76500)	(850000)
(E) $d = 3, c = 0$:							
Present	8.5	13	38	161	841	3002	9460
FGP05	0.9	5.7	59	575	8087	(76500)	(850000)

the crossover point for the 2D case A is about $N = 3000$, while for the 3D case E it is somewhere between 3000 and 10000, so the present algorithm is more efficient in terms of time than the FGP05 for problems of size $N \geq 10000$.

The effect of the parameter c in the multiquadric RBF on the convergence of iterative processes using the present method (fast preconditioner + FMM with $p = 30$) and the original FGP05 algorithm is shown in Figure 4.4. This parameter has the dimension of length and forces the function to a constant for large values (and thus uninfluenced by neighbors), while it acts as a regularization parameter for derivative evaluation for small values of c. It is seen that, while there are small differences in the norm of the residual after a certain number of iterations, both methods perform approximately in the same manner in terms of the dependence of the number of iterations on c (while the present algorithm is faster). As c increases and q is fixed the number of iterations grows. This growth is not substantial when c is of the order of characteristic distance between the data points, while it is significant for larger c. Moreover, we found that if c is larger than some critical c_c the process, it seems, does not converge (at least after a hundred iterations or so). This critical c depends on q and one of the recipes to compute interpolants with larger c is to use a larger q, which can also be seen in Table 4.1. The same behavior is observed for the original FGP05 algorithm.

Note that for uniform distributions of N points over d dimensional manifolds of volume V the characteristic distance between the points can be evaluated as $d_c = (V/N)^{1/d}$ and the effect of c can be measured in terms of the ratio $\overline{c} = c/d_c$, which for the cases plotted in Figure 4.4 ($d = 2, V = \pi$) are approximately 0.564, 1.69, and 2.82. Also, the number of the L-sets is of order N/q, which provides some mean volume, $V_L = Vq/N$, and some mean size, $d_L = V_L^{1/d}$, of the L-set. One may guess that the convergence rate depends on c/d_L and so the critical c is of the order d_L or $\overline{c}_c \sim q^{1/d}$. The latter quantity for the illustrated case was $q^{1/2} \approx 5.5$ and this is a rough estimate for the value of $\overline{c}_c$, which we observed in our experiments ($4 < \overline{c}_c < 4.5$). However, more detailed study on optimization of parameters c, q, and p, which provide the best performance of the present algorithms is required.

Finally, Figure 4.5 provides illustrations of solutions of some problems from 3D and 2D computer graphics using multiquadric RBFs and the present algorithm. In the first example, bunny (34834 point set) was extended to size 104502 (for points along the normal inside and outside) and after interpolation coefficients were determined the interpolant was evaluated at $\sim 8 \cdot 10^6$ points (regular spatial grid $201 \times 201 \times 201$) from which isosurfaces were found using standard routines. Here we used the same settings as for problem A with $d = 3$ and $c = 0$. The computation of preconditioner took 310 seconds, and the iteration process 283 seconds. Evaluation was made using the FMM (which required one $10^5 \times 8 \cdot 10^6$ matrix-vector multiplication) and took 395 seconds.

In the second example, 86% of 375050 pixels in a 2D color image were removed randomly. The remaining ($N = 51204$) pixels were used to interpolate data back to

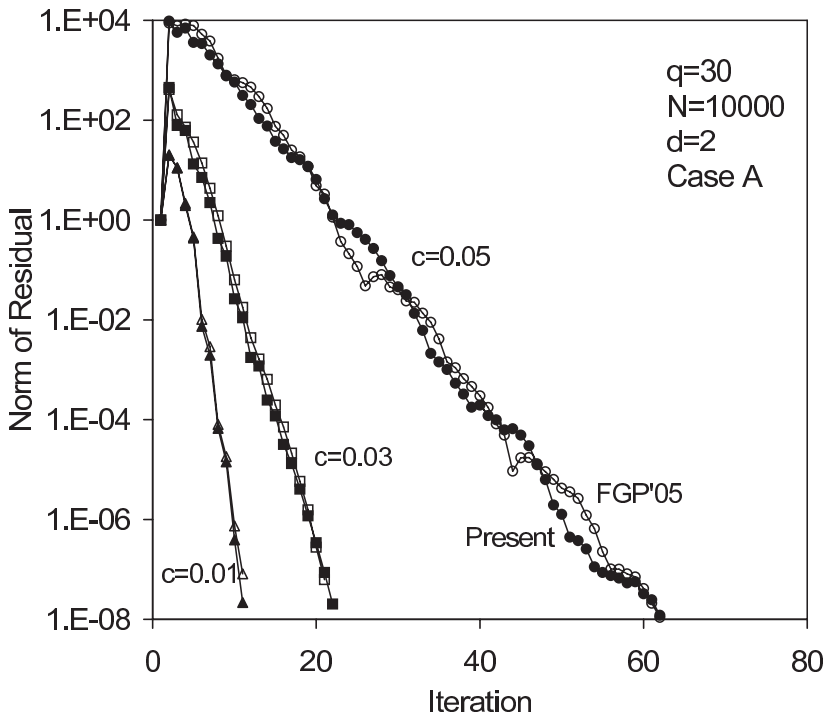


FIG. 4.4. *Convergence of the iterative processes using the present (the dark markers) and the original FGP05 algorithm (the light markers) for different values of parameter c in the multiquadric RBF.*

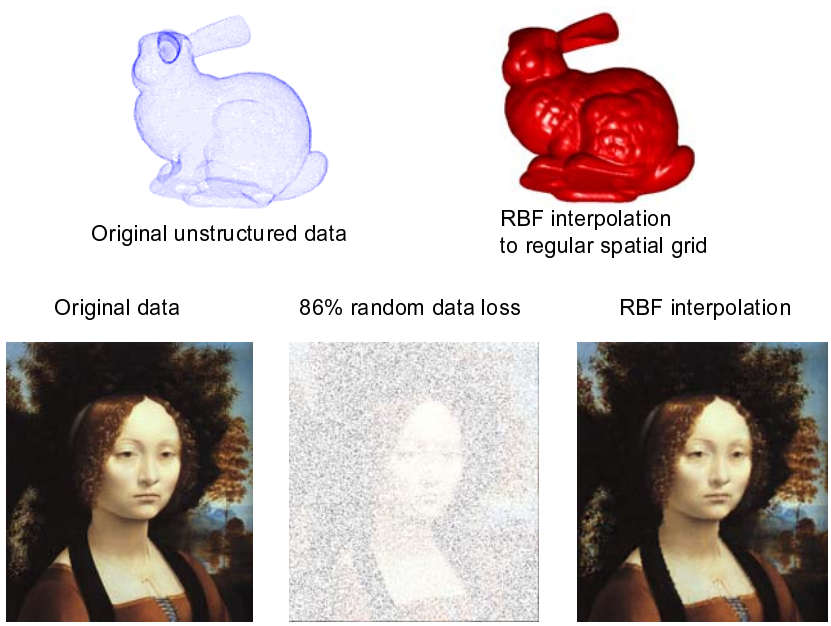


FIG. 4.5. *Illustrations of use of the RBF for 3D and 2D computer graphics problems.*

the grid using the 2D multiqudaric kernel with $c = N^{-1/2}$. The results of recovery are presented in the figure. Here we used the same setting as for problem A with $d = 2$ and $c = N^{-1/2}$. The preconditioner should be built only once as it is suitable for all three RGB channels and took about 30 seconds to perform this task. The iteration process took 75 seconds per channel and the final matrix-vector multiplication took 19 seconds. Note that the three-digit accuracy that we imposed for solution is sufficient to determine the RGB intensities as they vary between 0 to 255 for each channel.

5. Discussion and conclusion. For many dense linear systems of size the availability of an algorithm based on the FMM provides a matrix vector product with time and memory complexity that is $O(N \log N)$. This reduces the cost of the matrix vector product step in the iterative solution of linear systems, and for an iteration requiring k steps, a complexity $O(kN \log N)$ can be expected. To bound k, appropriate preconditioning strategies must be used. However, many conventional preconditioning strategies rely on sparsity in the matrix, and applying them to these dense matrices requires computations that may have a formal time or memory complexity of $O(N^2)$, which negates the advantage of the FMM. For the particular case of the FGP05 algorithm, with our construction, all steps have been accelerated, have a similar formal complexity, and in practice converge in relatively few iterations. The success of this preconditioner based on the geometry of point sets for a matrix that is capable of being accelerated using the FMM should not be surprising, as the FMM itself is based on the use of computational geometry to partition the matrix vector product.

The idea of constructing preconditioned Krylov algorithms based on the cardinal functions may have wider applicability. For any regular FMM kernel function, as the discussion in section 2.1 shows, we can construct an approximate inverse to the cardinal function expressed in terms of the kernel function. This idea bears further exploration.

Finally, we note that our implementation of the algorithm presented in this paper has been recently licensed from the University of Maryland by Fantalgo, LLC (info@fantalgo.com).

6. Acknowledgments. We would like to thank Prof. David Mount for providing us with some pointers to material on computational geometry. We also thank the reviewers for their comments and suggestions to improve the quality of the paper.

REFERENCES

[1] R. K. Beatson and M. J. D. Powell, *An iterative method for thin plate spline interpolation that employs approximations to Lagrange functions*, in Numerical Analysis 1993, D. F. Griffiths and G. A. Watson eds., Longman Sci. Tech., Harlow, UK, 1994, pp. 17–39.

[2] R. K. Beatson and L. Greengard, *A short course on fast multipole methods*, in Wavelets, Multilevel Methods and Elliptic PDEs, M. Ainsworth, J. Levesley, W. A. Light, and M. Marletta, eds., Oxford University Press, New York, 1997, pp. 1–37.

[3] R. K. Beatson, J. B. Cherrie, and C. T. Mouat, *Fast fitting of radial basis functions: methods based on preconditioned GMRES iteration*, Adv. Comput. Math., 11 (1999), pp. 253–270.

[4] R. K. Beatson, W. A. Light, and S. Billings, *Fast solution of the radial basis function interpolation equations: Domain decomposition methods*, SIAM J. Sci. Comput., 22 (2000), pp. 1717–1740.

[5] R. K. Beatson, J. B. Cherrie, and D. L. Ragozin, *Polyharmonic splines in $\mathbb{R}^d$: tools for fast evaluation*, in Curve and Surface Fitting: Saint-Malo 1999, A. Cohen, C. Rabut, and L. Schumaker, eds., Vanderbilt University Press, Nashville, TN, 2000, pp. 47–56.

[6] R. K. Beatson, J. B. Cherrie, and D. L. Ragozin, *Fast evaluation of radial basis functions: Methods for four-dimensional polyharmonic splines*, SIAM J. Math. Anal., 32 (2001), pp. 1272–1310.

[7] J. Bloomenthal, C. Bajaj, J. Blinn, M. P. Cani-Gascuel, A. Rockwood, B. Wyvill, and G. Wyvill, *Introduction to Implicit Surfaces*, Morgan Kaufmann Publishers, San Francisco, CA, 1997.

[8] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans, *Reconstruction and representation of 3D objects with radial basis functions*, ACM SIGGRAPH 2001, Los Angeles, CA, 2001, pp. 67-76.

[9] H. Cheng, L. Greengard, and V. Rokhlin, *A fast adaptive multipole algorithm in three dimensions*, J. Comput. Phys., 155 (1999), pp. 468–498.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, MA, 2004.

[11] J. B. Cherrie, R. K. Beatson, and G. N. Newsam, *Fast evaluation of radial basis functions: Methods for generalized multiquadrics in R^n*, SIAM J. Sci. Comput., 23 (2002), pp. 1549–1571.

[12] W. D. Elliott and J. A. Board, *Fast Fourier transform accelerated fast multipole algorithm*, SIAM J. Sci. Comput., 17 (1996), pp. 398–415.

[13] A. C. Faul, G. Goodsell, and M. J. D. Powell, *A Krylov subspace algorithm for multiquadric interpolation in many dimensions*, IMA J. Numer. Anal., 25 (2005), pp. 1–24.

[14] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comput. Phys., 73 (1987), pp. 325–348.

[15] L. Greengard, *The Rapid Evaluation of Potential Fields in Particle Systems*, MIT Press, Cambridge, MA, 1988.

[16] L. Greengard and V. Rokhlin, *On the Efficient Implementation of the Fast Multipole Algorithm*, Technical report RR-602, Department of Computer Science, Yale University, New Haven, CT, 1988.

[17] L. Greengard and J. Strain, *The fast Gauss transform*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 79–94.

[18] G. Goodsell, *On finding p-th nearest neighbours of scattered points in two dimensions for small p*, Comput. Aided Geom. Design, 17 (2000), pp. 387–392.

[19] N. A. Gumerov and R. Duraiswami, *Fast multipole method for the biharmonic equation in three dimensions*, J. Comput. Phys., 215 (2005), pp. 363–383.

[20] N. A. Gumerov and R. Duraiswami, *Comparison of the Efficiency of Translation Operators used in the Fast Multipole Method for the 3D Laplace Equation*, Technical report CS-TR-4701, Department of Computer Science, University of Maryland, College Park, MD, 2005.

[21] N. A. Gumerov and R. Duraiswami, *Fast Multipole Methods for the Helmholtz Equation in Three Dimensions*, Elsevier, Oxford, UK, 2005.

[22] N. A. Gumerov, R. Duraiswami, and E. A. Borovikov, *Data Structures, Optimal Choice of Parameters, and Complexity Results for Generalized Multilevel Fast Multipole Methods in d Dimensions*, Technical report CS-TR-4458, Department of Computer Science, University of Maryland, College Park, MD, 2003.

[23] R. L. Hardy, *Theory and applications of the multiquadric-biharmonic method, 20 years of discovery* 1968–1988, Comput. Math. Appl., 19 (1990), pp. 163–208.

[24] C. A. Micchelli, *Interpolation of scattered data: distance matrices and conditionally positive definite functions*, Constr. Approx., 2 (1986), pp. 11–22.

[25] S. Maneewongvatana and D. M. Mount, *Analysis of approximate nearest neighbor searching with clustered point sets*, in Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges, M. H. Goldwasser, D. S. Johnson, C. C. McGeoch, eds., DIMACS Ser. Discrete Math. Theoret. Comp. Sci. 59, AMS, Providence, RI, 2002, pp. 105–123.

[26] V. C. Raykar, C. Yang, R. Duraiswami, and N. A. Gumerov, *Fast Computation of Sums of Gaussians in High Dimensions*, Technical report CS-TR-4767, Department of Computer Science, University of Maryland, College Park, MD, 2005.

[27] H. Samet, *The Design and Analysis of Spatial Data Structures*, Addison–Wesley, New York, 1989.

[28] V. Simoncini and D. B. Szyld, *Theory of inexact Krylov subspace methods and applications to scientific computing*, SIAM J. Sci. Comput., 25 (2003), pp. 454–477.

[29] G. Turk and J. F. O'Brien, *Modelling with implicit surfaces that interpolate*, ACM Trans. Graph., 21 (2002), pp. 855–873.

[30] E. W. Weisstein, *Kissing number*, in MathWorld–A Wolfram Web Resource, http://mathworld.wolfram.com/KissingNumber.html.

SIAM J. SCI. COMPUT.
Vol. 29, No. 5, pp. 1900–1917
© 2007 Society for Industrial and Applied Mathematics

SPATIAL MULTIGRID FOR ISOTROPIC NEUTRON TRANSPORT*

B. CHANG†, T. MANTEUFFEL‡, S. MCCORMICK‡, J. RUGE‡, AND B. SHEEHAN‡

Abstract. A spatial multigrid algorithm for isotropic neutron transport is presented in x-y geometry. The linear system is obtained using discrete ordinates in angle and corner balance finite differencing in space. Spatial smoothing is accomplished by a four-color block-Jacobi relaxation, where the diagonal blocks correspond to 4-cell blocks on the spatial grid. A bilinear interpolation operator and its transpose are used for the grid transfer operators. Good convergence factors were observed for homogeneous material properties. Heterogeneous material properties prove more difficult, especially the case of a vacuum region surrounded by a thick, diffusive region. In this case, a small amount of absorption, or "effective absorption" in a time-dependent problem, restores good convergence. Numerical results are presented.

Key words. neutral particle transport, multigrid, S_N approximation, finite difference discretization

AMS subject classifications. 65N06, 65N30, 65N55

DOI. 10.1137/060661363

1. Introduction.

The purpose of this paper is to present a spatial multigrid algorithm for isotropic neutron transport. The algorithm described here is an extension to two spatial dimensions of the work presented by Manteuffel et al. in [13] and [14]. The partial differential equation (PDE) to be solved is

$$\Omega \cdot \nabla\psi + \sigma_t\psi - \frac{\sigma_s}{4\pi}\int_\Omega \psi \, d\Omega = q, \qquad (x, y, \Omega) \in D \times S^2, \tag{1.1}$$

$$\psi = g, \qquad (x, y) \in \partial D, \quad \mathbf{n} \cdot \Omega < 0. \tag{1.2}$$

$\psi = \psi(x, y, \Omega)$	the angular flux of neutrons
$\Omega \in S^2$	direction of neutron travel on unit sphere
$\sigma_t = \sigma_t(x, y)$	total macroscopic cross section
$\sigma_s = \sigma_s(x, y)$	scattering macroscopic cross section
$q = q(x, y, \Omega)$	interior source of neutrons
$g = g(x, y, \Omega)$	boundary source of neutrons
D, ∂D	unit square, boundary of the unit square
$\mathbf{n}$	outward unit normal

The primary numerical methodology currently in use for solving this PDE is source iteration combined with diffusion synthetic acceleration (DSA) [2, 4, 10, 12]. While DSA has achieved a great deal of success, it also suffers from certain limitations.

*Received by the editors May 30, 2006; accepted for publication (in revised form) December 7, 2006; published electronically September 28, 2007.
http://www.siam.org/journals/sisc/29-5/66136.html

†Center for Applied Scientific Computation, Lawrence Livermore National Lab, P. O. Box 808 L-561, Livermore, CA, 94551 (bchang@llnl.gov).

‡Department of Applied Mathematics, Campus Box 526, University of Colorado at Boulder, Boulder, CO 80309-0526 (tmanteuf@colorado.edu, stevem@colorado.edu, jruge@colorado.edu, brendan.sheehan@colorado.edu). The work of these authors was sponsored by the Department of Energy under grants DE-FC02-01ER25479 and DE-FG02-03ER25574, Lawrence Livermore National Laboratory under contract B533502, Sandia National Laboratory under contract 15268, and by the National Science Foundation under DMS-0410318.

One of these limitations is the sensitivity of the method to the spatial discretization [2]. In particular, it is difficult to implement DSA with the discontinuous finite difference method known as [illegible].[1] Fast transport algorithms for the corner balance discretization are an area of active research interest. Other limitations of DSA stem from source iteration, the relaxation step on which it is based [12]. This relaxation is not ideally suited to parallel computing and does not allow for acceleration by spatial multigrid [13, 8, 15]. In contrast, the method outlined below employs the corner balance discretization, acceleration by spatial multigrid, and is well suited to parallel computing.

To explain these concepts further, it is necessary to first discretize the PDE in angle. The angular discretization used in this algorithm is known as discrete ordinates, or S_N [12]. The idea is simply that the PDE is enforced at only a finite number of angles, $\Omega_k, k = 1, \ldots, d$. The angles are chosen so that the angular integral can be replaced by an appropriate quadrature rule, and (1.1)–(1.2) is then written as the following system of coupled PDEs in space. For $k = 1, \ldots, d$,

$$\Omega_k \cdot \nabla \psi_k + \sigma_t \psi_k - \frac{\sigma_s}{4\pi} \sum_{m=1}^{d} w_m \psi_m = q_k, \qquad (x, y) \in D, \tag{1.3}$$

$$\psi_k = g_k, \qquad (x, y) \in \partial D, \quad \mathbf{n} \cdot \Omega_k < 0, \tag{1.4}$$

$\Omega_k,\ w_k$ given direction, quadrature weight,
$\psi_k,\ q_k,\ g_k$ $\psi(x, y, \Omega_k),\ q(x, y, \Omega_k),\ g(x, y, \Omega_k)$.

The spatial discretization of (1.3)–(1.4) is discussed in the next section. For now, assume the problem has been spatially discretized, and write the discrete problem as

$$[\mathbf{D} + \mathbf{\Sigma} - \mathbf{S}]\, \mathbf{\Psi} = \mathbf{Q}, \tag{1.5}$$

where the three terms in brackets represent the discrete versions of the three terms in (1.3). Notice that $\mathbf{D} + \mathbf{\Sigma}$ contains local spatial coupling of the unknowns, and $\mathbf{S}$ contains the angular coupling. Since problems of interest are too large to be solved directly, an iterative approach based on a matrix splitting is used. In the case of transport sweeps, $\mathbf{S}$ is split from the rest of the matrix, and the iteration becomes

$$[\mathbf{D} + \mathbf{\Sigma}]\, \mathbf{\Psi^{i+1}} = [\mathbf{S}]\, \mathbf{\Psi^{i}} + \mathbf{Q}, \tag{1.6}$$

which implies that the error evolves according to

$$\mathbf{e^{i+1}} = [\mathbf{D} + \mathbf{\Sigma}]^{-1}\, [\mathbf{S}]\, \mathbf{e^{i}}. \tag{1.7}$$

This step is computationally inexpensive because $[\mathbf{D} + \mathbf{\Sigma}]$ is triangular, or block triangular, depending on the discretization. However, calculating the action of the inverse of a triangular matrix on a vector is a sequential operation, and this is why transport sweeps are not ideally suited to parallel computing.[2] To see why transport sweeps do

[1] DSA is implemented in [18] for a discretization similar to corner balance. The authors indicate that a fully consistent implementation is computationally expensive, but that similar results can be obtained when an inconsistent DSA is used as a preconditioner for a Krylov method

[2] A domain decomposition method for transport sweeps is described in [19]. This method performs sweeps on local subdomains and avoids the global sequential solve. While this improves the parallel scalability of sweeps, it also significantly changes the performance of the associated DSA algorithm. For the duration of this paper, all references to the performance and parallel scalability of DSA assume an algorithm based on global sweeps.

not allow for acceleration by spatial multigrid, consider the case where $\sigma_s \approx \sigma_t$ and $\sigma_t h \gg 1$, where h is the cell width. In this parameter regime, which is common in real world problems, $[\mathbf{D} + \mathbf{\Sigma}]^{-1} \approx [\mathbf{\Sigma}]^{-1}$. Now imagine an error, $\mathbf{e^i}$, that is independent of angle, so that $\mathbf{Se^i} \approx \mathbf{\Sigma e^i}$. Equation (1.7) now shows $\mathbf{e^{i+1}} \approx \mathbf{e^i}$. Errors that are constant in angle are not significantly changed by transport sweeps in this parameter regime, regardless of their spatial form. Since these errors have no predictable form in space after relaxation, a spatial coarse-grid correction cannot be used to accelerate convergence.

The algorithm described below is based on a different relaxation step, which is well suited to parallel computation and leaves the error with a predictable spatial form. The remainder of this paper is organized as follows. Section 2 describes the spatial discretization, and section 3 gives the details of the relaxation step. Section 4 outlines the multigrid algorithm. Sections 5 and 6 give numerical results for homogeneous and heterogeneous domains. Section 7 examines the computational cost of the method. Sections 8 and 9 compare the parallel scalability and performance of the method to that of DSA, and section 10 contains a brief conclusion.

2. Spatial discretization. As mentioned above, the spatial discretization used in this algorithm is the corner balance finite difference method.[3] The derivation of corner balance is contained in [1]. In this context, it suffices to say that in the case of square cells the solution has four spatial degrees of freedom per cell. Figures 2.1 and 2.2 illustrate the location of the spatial unknowns and a random example solution on a 4-cell grid. It is important to note that these figures represent the grid and solution in only one of the directions.

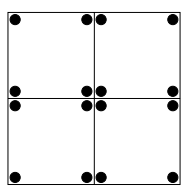


FIG. 2.1. *Location of spatial unknowns.*

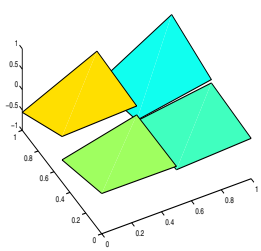
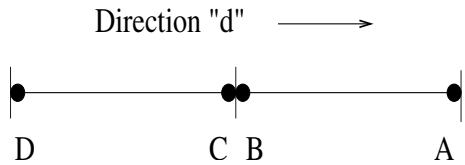


FIG. 2.2. *Bilinear discontinuous example solution.*

[3]The spatial discretization used in this algorithm is *simple* corner balance, not *upwind* corner balance, both of which are derived in [1].

To describe the relaxation step in section 3, it is necessary to examine the corner balance approximation of a first derivative. The nature of the corner balance approximation of the derivative terms in (1.3) is best explained with a one-dimensional (1-D), 2-cell example:

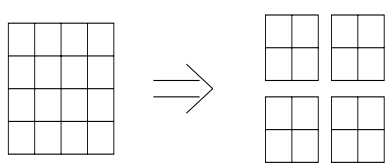


(2.1) At point A in direction d, $$\frac{d\psi}{dx} \approx \frac{\psi_A - \psi_B}{h}.$$

(2.2) At point B in direction d, $$\frac{d\psi}{dx} \approx \frac{\frac{\psi_A+\psi_B}{2} - \psi_C}{\frac{h}{2}} = \frac{\psi_A + \psi_B - 2\psi_C}{h}.$$

Equations (2.1) and (2.2) contain two kinds of spatial coupling. Equation (2.1) contains [illegible] coupling (A to B), while (2.2) contains both within-cell and [illegible] coupling (A and B to C). The distinction between these two categories of spatial coupling is the basis for the matrix splitting described below.

3. Relaxation method. Imagine splitting the two-dimensional domain into 4-cell blocks:



The matrix $\mathbf{D}$ is split as follows: $\mathbf{D} = [\mathbf{D_{wb}} + \mathbf{D_{bb}}]$, where $\mathbf{D_{wb}}$ contains the [illegible] coupling (elements of $\mathbf{D}$ which couple unknowns in a given block to other unknowns in that same block), and $\mathbf{D_{bb}}$ contains the [illegible] coupling (elements of $\mathbf{D}$ which couple unknowns in a given block to unknowns in a separate block). This splitting gives the new relaxation step

(3.1) $$[\mathbf{D_{wb}} + \mathbf{\Sigma} - \mathbf{S}]\, \mathbf{\Psi^{i+1}} = [-\mathbf{D_{bb}}]\, \mathbf{\Psi^{i}} + \mathbf{Q}.$$

The inversion of $[\mathbf{D_{wb}} + \mathbf{\Sigma} - \mathbf{S}]$ is straightforward because, with the appropriate ordering of the unknowns, it is block diagonal and represents a set of unrelated 4-cell transport problems. From this perspective, it is clear that this relaxation is a form of block Jacobi. Since each 4-cell problem can be handled by a different processor, the step is well suited to parallel computing. Equation (3.1) implies that the unknowns corresponding to all the directions within a 4-cell block are solved for simultaneously. This means that the size of the matrix corresponding to each 4-cell block grows with the size of the quadrature set being used. However, since the scattering operator is a low-rank perturbation of the streaming-collision operator, the Sherman–Morrison–Woodbury formula allows the 4-cell inversions to be accomplished at a cost that rises

only linearly with the number of angles. This formula and a more detailed discussion of computational cost are included in section 7.

This relaxation step is designed to spatially smooth the error in the thick diffusive regime, where $\sigma_s \approx \sigma_t$ and $\sigma_t h \gg 1$. In order to see how the smoothing is accomplished, consider applying (3.1) in a diffusive material, with a zero right-hand side and random initial guess. The first iteration calls for the solution of many unrelated 4-cell problems, each with a zero interior source and random inflow boundary conditions. Intuition suggests that in the continuous case, the solution would be random near the boundary but smooth away from the boundary. Therefore, in the discrete 4-cell case, we expect the solution to have the four spatial unknowns at the center of the domain roughly equal. Figures 3.1 and 3.2 show the corner balance solution to a problem with zero interior source, random inflow, and $\sigma_t = \sigma_s = 10^4$. Figure 3.1 is a domain with 64 cells and is intended to be an approximation to the continuous case. Figure 3.2 is a 4-cell domain and shows that the center variables are roughly equal. Once again, these plots show the solution in only one of the directions.

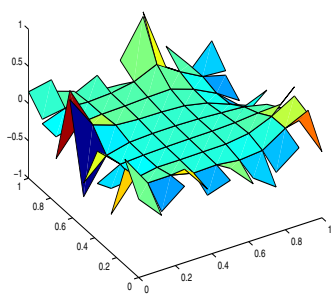


FIG. 3.1. *Zero source, random inflow, on* 64 *cells with* $\sigma_t = \sigma_s = 10^4$.

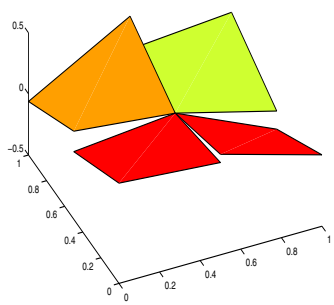


FIG. 3.2. *Zero source, random inflow, on* 4 *cells with* $\sigma_t = \sigma_s = 10^4$.

After one relaxation, the error is no longer random. At the center of each 4-cell block, the four spatial errors are roughly equal in any given direction. Therefore, the error is now "partially continuous." This relaxation step is the extension to two spatial dimensions of the relaxation used by Manteuffel et al. [13, 14], and has been

presented and analyzed in two spatial dimensions by Lansrud [9]. However, the error is not spatially smoothed unless the 4-cell blocks are shifted and the block-Jacobi step is repeated, a total of four times. If the arrangement of the 4-cell blocks is now shifted by one cell, a second relaxation step makes the error continuous at the centers of the new 4-cell blocks. This process of shifting the arrangement of the 4-cell blocks is repeated, and a total of four different arrangements are used. The different 4-cell block patterns are illustrated in Figure 3.3. The iterate is updated before the blocks are shifted, so each of the arrangements below represents a separate block-Jacobi iteration.

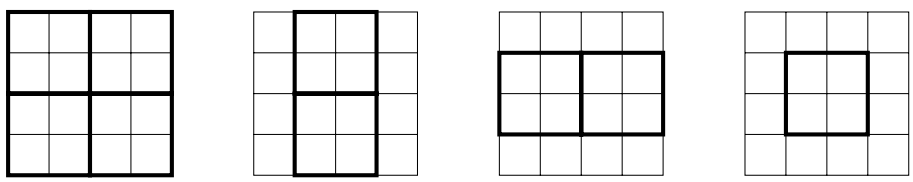


FIG. 3.3. *4-cell block pattern.*

At the end of the complete red-black-orange-white (RBOW) pass, every vertex on the original grid has been at the center of a 4-cell solve once, and the error is nearly continuous over the entire domain. In addition to becoming nearly continuous, the error also tends to become smooth over the entire domain and bilinear over the 4-cell blocks defined by the last color pass. This happens because the center lines of the 4-cell blocks in a given color pass become the boundary conditions for the blocks in the next color pass. Continuous centers become continuous boundary conditions, and continuous boundary conditions produce smoother solutions. Smoother solutions become smoother boundary conditions, and the cycle pushes the error into a form that is not only continuous, but is increasingly smooth and bilinear over the blocks of the last pass.

In practice, it is not necessary to include all spatial points within a 4-cell block in the block solve. The computational cost can be reduced by including only the points which are spatially coupled to the points at the center of the block. Figure 3.4 shows the [illegible] on the left and the [illegible] on the right. Only the 12 points along the "plus sign" of the 4-cell block are included in the reduced block.[4] For the remainder of this paper, the term "4-cell inversion" is taken to mean that the unknowns at the spatial locations of the reduced block, and in all the directions, are solved for simultaneously. Furthermore, the term "RBOW relaxation" implies that the reduced block is being used.

Figure 3.5 shows the evolution of random error on a 256 cell grid, where $\sigma_t = \sigma_s = 10^4$, over five RBOW relaxations. The important feature of the RBOW step is not that it reduces the size of the error, but that it puts the error in a predictable spatial form. The desired spatial form is bilinear over the "red" blocks. This allows a coarse-grid correction based on bilinear interpolation. To leave the error bilinear over the red blocks, the order of the color pass is arranged so that the red pass is last. In

[4]In three spatial dimensions, there would be 64 spatial locations in an 8-cell block. However, only 32 of these would lie along the "three-dimensional plus sign," making the cost savings of using reduced blocks more dramatic than it is in two spatial dimensions.

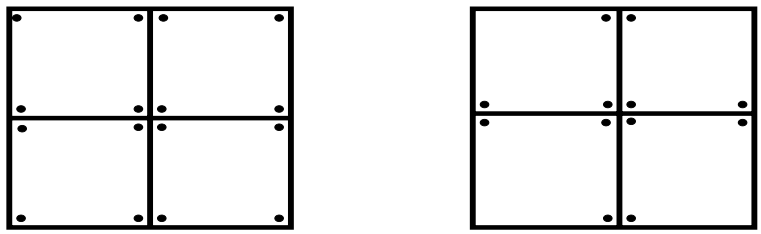


FIG. 3.4. *Spatial unknowns in the full and the reduced block.*

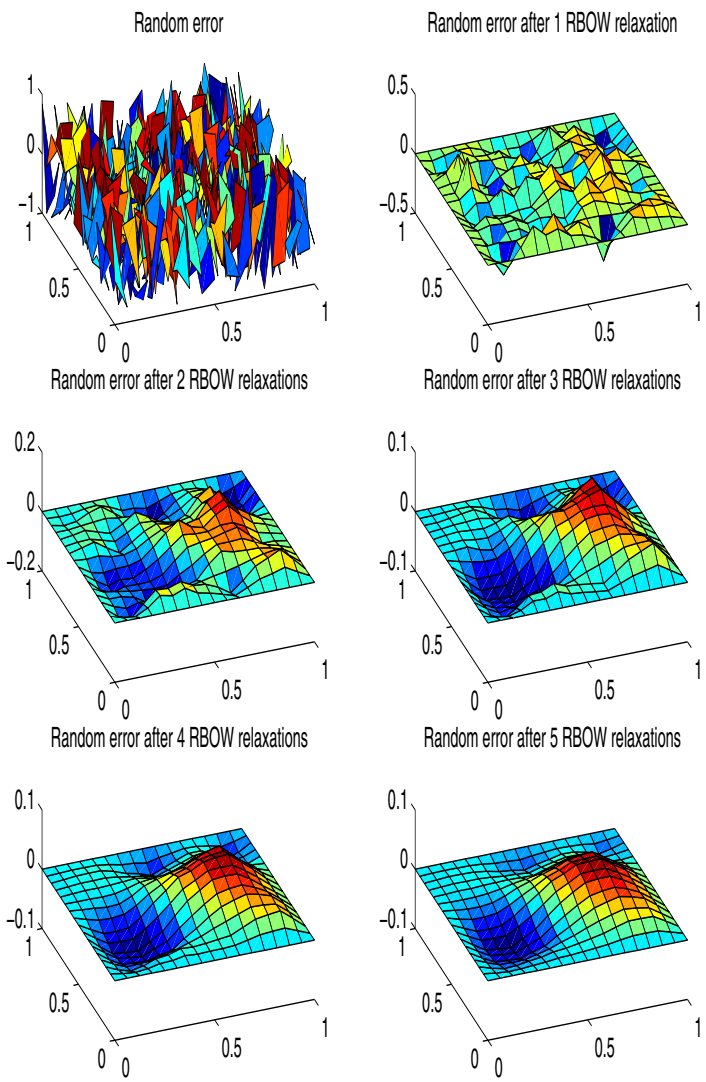


FIG. 3.5. *Evolution of random error.*

Figure 3.5, it is easy to see that the error is nearly bilinear over the red blocks after five RBOW relaxations, and a coarse-grid correction would be very effective at this point. Experiments have shown, however, that the coarse-grid correction is surprisingly effective after just one relaxation.

4. Multigrid algorithm. Since the error after RBOW relaxation is close to being bilinear over the red blocks, it can be approximately represented on a coarse grid. To accomplish this, it is necessary to define the coarse grid, the fine grid, and grid transfer operators that map vectors from one grid to the other. Consider a fine grid with a total of N unknowns, and a coarse grid on which the fine-grid red blocks are single cells. This coarse grid has a total of $\frac{N}{4}$ unknowns. The [illegible] operator, $\mathbf{P}$, is an $N \times \frac{N}{4}$ matrix that maps coarse vectors, denoted by $\mathbf{v^{2h}}$, to fine vectors, denoted by $\mathbf{v^h}$ (the notation h and 2h represent a doubling of the cell width). The interpolation is bilinear, meaning that the spatial unknowns on any coarse cell, which are bilinear by definition, are mapped to the same bilinear over the corresponding fine-grid red block. The [illegible] operator, $\mathbf{R}$, is simply the transpose of $\mathbf{P}$, up to a constant. This restriction operator is often called [illegible].

For the purpose of illustrating the concept of coarse-grid correction, assume that relaxation leaves the error exactly bilinear over the red blocks. This means the fine error lies in the range of bilinear interpolation, and we can write $\mathbf{e^h} = \mathbf{Pe^{2h}}$. The coarse-grid correction then proceeds as follows:

- After relaxation, calculate the fine residual, $\mathbf{r^h} = \mathbf{b} - \mathbf{A\Psi^i}$ ($\mathbf{A}, \mathbf{b}, \mathbf{\Psi^i}$ represent the entire discrete transport operator, the discrete right-hand side, and the current iterate, respectively).
- Write the residual equation $\mathbf{Ae^h} = \mathbf{r^h}$ and use the assumption above to get $\mathbf{APe^{2h}} = \mathbf{r^h}$.
- Solve the $\frac{N}{4} \times \frac{N}{4}$ system $[\mathbf{RAP}]\,\mathbf{e^{2h}} = \mathbf{Rr^h}$.
- Correct the current iterate: $\mathbf{\Psi^{i+1}} = \mathbf{\Psi^i} + \mathbf{Pe^{2h}}$.

In the ideal situation outlined above, the coarse-grid correction would be perfect, and the problem would be solved exactly. In reality, the coarse-grid correction is not perfect because the fine error is not exactly in the range of interpolation. The idea is that if the fine error is close to the range of interpolation, then the coarse-grid correction improves the current iterate at a low computational cost. There are, of course, several other important differences between the ideal example above and the actual algorithm. Since an $\frac{N}{4} \times \frac{N}{4}$ system is still far too large to be solved directly, the procedure is applied recursively. The exact solve does not take place until the coarsest grid, which can be as small as four cells total. The sequence of relaxations and coarse-grid corrections is called a V-cycle [7]. A V-cycle can feature any given number of relaxation steps, called prerelaxations, on each grid level as the grid is being coarsened. It is also common to include some number of additional relaxations, called postrelaxations, after each coarse-grid correction. The optimal number of pre- and postrelaxations to use is often determined experimentally. The notation V(n,m) indicates a V-cycle with n prerelaxations and m postrelaxations. In summary, the spatial multigrid algorithm is a V-cycle with the following features:

- RBOW relaxation,
- bilinear interpolation operator ($\mathbf{P_{BL}}$),
- restriction is full weighting ($\mathbf{R_{FW}}$),
- coarse-grid operator is $\mathbf{R_{FW}AP_{BL}}$.

Notice that the operator for the coarse-grid problem is $\mathbf{R_{FW}AP_{BL}}$. When the corner balance transport operator is multiplied on the left and right by $\mathbf{R_{FW}}$ and

$\mathbf{P_{BL}}$, respectively, its stencil is altered. The evolution of this stencil on coarser and coarser grids is fairly easy to track and heads rapidly to a limit that is equivalent to the bilinear discontinuous Galerkin (BLDG) finite element discretization. The BLDG discretization is very similar to corner balance [1, 2, 3, 16], and the sequence of intermediate stencils also has a similar character. Since the evolution of the stencil is not dramatic, RBOW relaxation is an appropriate smoother for the coarse-grid problems.[5]

5. Results for problems with homogeneous material. The method described above has been coded and tested in two spatial dimensions. In this section, convergence factors are given for the zero right-hand side problem, with zero boundary conditions and random initial guess. The convergence factor is measured by taking a ratio of the vector two-norm of the residual before and after a given V-cycle:

$$\rho^{i+1} = \frac{\|\mathbf{r^{i+1}}\|_2}{\|\mathbf{r^i}\|_2}. \tag{5.1}$$

In most cases, the convergence factor settles to a nearly constant value after a few V-cycles. For consistency, the factors given below were calculated by running fifteen V-cycles and taking the geometric mean of the last five,

$$\rho = \left(\prod_{i=11}^{15} \rho^i \right)^{\frac{1}{5}}. \tag{5.2}$$

All cycles are V(1,1) cycles. In all cases, the domain is the unit square, which is divided into 4,096 square cells. Four directions are used, giving a total of 65,536 unknowns. Table 5.1 gives results for homogeneous, pure scattering domains. In all cases, the results improve with absorption ($\sigma_s < \sigma_t$).

TABLE 5.1
Convergence factors for domains with homogeneous material and pure scattering.

$\sigma_t h$	ρ
10^4	.06
1	.03
10^{-4}	.1

In the case of the thick and medium problem (top two lines), the convergence factors in Table 5.1 do not depend on problem size; using more cells or more directions has no effect on the convergence. The same cannot be said for the thin problem (bottom line). Due to the nature of the thin operator, successive V-cycles create a "marching-style" near exact solve in a pure thin domain. Once the marching process crosses the domain, there is a large drop in the residual. For this reason, thin convergence factors on domains with few cells are better than they would be on domains with many cells. Also, increasing the number of directions degrades the thin convergence factor because, as directions become more grid-aligned, bilinear interpolation is less appropriate for thin regions. While the pure thin problem by itself is not of much interest, it is important to understand how the method might behave in thin regions of domains with heterogeneous material. Domains of this type are discussed further in the next section.

[5]The BLDG stencil is not altered by the coarsening process, i.e., $\mathbf{R_{FW}} \mathbf{A^h_{BLDG}} \mathbf{P_{BL}} = \mathbf{A^{2h}_{BLDG}}$. The algorithm described here can be applied with BLDG instead of corner balance with very similar results.

6. Results for problems with heterogeneous material. This method has been tested on a variety of heterogeneous domains. The domains and coarse grids are set up in such a way that there is no "mixed coarsening." In other words, the case of a 4-cell block composed of multiple materials being coarsened into a single cell does not arise. The idea is to deal with the difficulties of a heterogeneous domain without the added complication of mixed coarsening.[6]

In some cases, the convergence factors for heterogeneous domains are no worse than those for homogeneous domains. The most notable exception to this rule is the case of a thin region surrounded by a thick region, as indicated in Figure 6.1. Tables 6.1 and 6.2 give the convergence factors for this domain with several variations to the cross sections. In Table 6.1, both regions are pure scattering and the values of $\sigma_t h$ are listed for the thick (thin) regions, respectively. While the convergence is still good for fairly large jumps in cross section, it is clear that performance deteriorates completely as the thin region approaches a true vacuum. In Table 6.2, the thin region is always a true vacuum, while the thick region always has $\sigma_t h = 10^4$, with some absorption, as indicated by the scattering ratio ($scat.\ ratio = \frac{\sigma_s}{\sigma_t}$). Clearly, the convergence is improved with the introduction of a small amount of absorption.

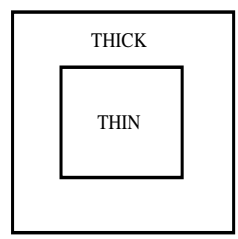


FIG. 6.1. *Heterogeneous model domain.*

It is important to note that a small amount of "effective absorption" will always be present if the steady state problem is being solved at each time step of a time-dependent problem with an implicit time discretization. To see this, let c be the particle speed, and consider the time-dependent problem

$$\frac{1}{c}\frac{\partial \psi}{\partial t} + \Omega \cdot \nabla \psi + \sigma_t \psi - \frac{\sigma_s}{4\pi}\int_\Omega \psi \, d\Omega = q. \tag{6.1}$$

A backward Euler approximation in time yields

$$\frac{1}{c}\frac{\psi^{n+1} - \psi^n}{\Delta t} + \Omega \cdot \nabla \psi^{n+1} + \sigma_t \psi^{n+1} - \frac{\sigma_s}{4\pi}\int_\Omega \psi^{n+1} \, d\Omega = q^{n+1} \tag{6.2}$$

or

$$\Omega \cdot \nabla \psi^{n+1} + \left(\sigma_t + \frac{1}{c\Delta t}\right)\psi^{n+1} - \frac{\sigma_s}{4\pi}\int_\Omega \psi^{n+1} \, d\Omega = q^{n+1} + \frac{1}{c\Delta t}\psi^n. \tag{6.3}$$

The effective absorption cross section is $\sigma_a^* \geq (\frac{1}{c\Delta t})$. This parameter prevents any region from being either pure vacuum or pure scattering. Larsen [11] details a method

[6]Preliminary experiments with mixed coarsening show that the convergence is not adversely affected, provided that the bilinear interpolation is replaced with a kinked-linear interpolation similar to that used by Manteuffel et al. in [14].

TABLE 6.1

Convergence factors for the heterogeneous model domain with a thin pure scattering center and a thick pure scattering perimeter.

$\sigma_t h$ thick (thin)	ρ
10^4 (10^4)	.06
10^4 (10^{-1})	.11
10^4 ($5 * 10^{-2}$)	.23
10^4 (10^{-2})	.61
10^4 (0)	.97

TABLE 6.2

Convergence factors for the heterogeneous model domain with vacuum center and thick perimeter. The thick region always has $\sigma_t h = 10^4$ with some absorption, as indicated by the scattering ratio.

$scat.\ ratio = \frac{\sigma_s}{\sigma_t}$	ρ
.999	.09
.9999	.24
.99999	.63
1	.97

TABLE 6.3

Convergence factors for the heterogeneous model domain with vacuum center, thick pure scattering perimeter, and time-step-based effective absorption. The thick region always has $\sigma_t h = 10^4$. Parentheses indicate convergence factor for a W-cycle instead of a V-cycle.

$\frac{1}{c\Delta t}$	$\frac{c\Delta t}{\Delta x}$	ρ
nx	1	$7 * 10^{-4}$
$\frac{nx}{10}$	10	.12
$\frac{nx}{100}$	100	.56 (.38)
$\frac{nx}{1000}$	1000	.88 (.69)

for time- and energy-dependent problems that requires the solution of an equation similar to (6.3) in the inner iteration. The algorithm described here has been applied to (6.3) for the model domain with pure vacuum and pure scattering ($\sigma_t h = 10^4$) materials. Naturally, convergence improves with larger values of $\frac{1}{c\Delta t}$. To put this another way, the convergence factor depends on the value of the Courant number, $\frac{c\Delta t}{\Delta x}$, where $\Delta x = \frac{1}{nx}$ and nx is the number of cells in a single dimension (i.e., in two dimensions on a 64×64 grid, $nx = 64$). Table 6.3 shows the convergence factors for various values of $\frac{1}{c\Delta t}$ between $\frac{nx}{1000}$ and nx. The value of $\frac{1}{c\Delta t}$ must be proportional to nx for the results to be independent of grid size. This is because it is the amount of absorption , , , , that affects the convergence of the method. The second column in Table 6.3 gives the Courant number, $\frac{c\Delta t}{\Delta x}$. The values listed in parentheses are for a W-cycle, which requires slightly more work than a V-cycle [7].

7. Computational cost. To put the performance of the method in perspective, it is necessary to discuss computational cost. As mentioned previously, all 4-cell inversions in the relaxation step are accomplished via the Sherman–Morrison–Woodbury formula

$$(\mathbf{H} + \mathbf{U}\mathbf{V}^{\mathbf{T}})^{-1} = \mathbf{H}^{-1} - \left[\mathbf{H}^{-1}\mathbf{U}(\mathbf{I} + \mathbf{V}^{\mathbf{T}}\mathbf{H}^{-1}\mathbf{U})^{-1}\mathbf{V}^{\mathbf{T}}\mathbf{H}^{-1}\right]. \tag{7.1}$$

To see how this formula applies to the discrete transport operator, let $\mathbf{H} = \mathbf{D} + \mathbf{\Sigma}$ be the streaming-collision operator and let $-\mathbf{S} = \mathbf{U}\mathbf{V}^{\mathbf{T}}$ be the scattering operator, where

$$\mathbf{U} = -\begin{bmatrix} \mathbf{\Sigma_s} \\ \mathbf{\Sigma_s} \\ \vdots \\ \mathbf{\Sigma_s} \end{bmatrix} \tag{7.2}$$

and

$$\mathbf{V}^{\mathbf{T}} = \frac{1}{4\pi}\begin{bmatrix} w_1\mathbf{I} & w_2\mathbf{I} & \cdots & w_d\mathbf{I} \end{bmatrix}. \tag{7.3}$$

In (7.3), w_i is a quadrature weight and $\mathbf{I}$ is an $N_{sp} \times N_{sp}$ identity, where N_{sp} is the number of spatial unknowns in the domain. In (7.2), $\mathbf{\Sigma_s}$ is a diagonal matrix of size $N_{sp} \times N_{sp}$, with the values of σ_s at each spatial point in the domain on the diagonal. As before, d is the number of directions, so the total number of unknowns in the problem is $N = N_{sp} * d$, and the sizes of $\mathbf{U}$ and $\mathbf{V}^{\mathbf{T}}$ are $N \times N_{sp}$ and $N_{sp} \times N$, respectively. Notice that $\mathbf{V}^{\mathbf{T}}$ is the discrete integration operator, and $\mathbf{V}^{\mathbf{T}}\mathbf{\Psi} = \mathbf{\Phi}$, where $\mathbf{\Phi}$ is the scalar flux. Now that the discrete system can be written as $(\mathbf{H} + \mathbf{U}\mathbf{V}^{\mathbf{T}})\mathbf{\Psi} = \mathbf{q}$, consider the following algebraic manipulations:

- Multiply $(\mathbf{H} + \mathbf{U}\mathbf{V}^{\mathbf{T}})\mathbf{\Psi} = \mathbf{q}$ by $\mathbf{H}^{-1}$ and rearrange to get

$$\mathbf{\Psi} = \mathbf{H}^{-1}\left[\mathbf{q} - \mathbf{U}\mathbf{\Phi}\right]. \tag{7.4}$$

- Multiply (7.4) by $\mathbf{V}^{\mathbf{T}}$ and rearrange to get

$$(\mathbf{I} + \mathbf{V}^{\mathbf{T}}\mathbf{H}^{-1}\mathbf{U})\mathbf{\Phi} = \mathbf{V}^{\mathbf{T}}\mathbf{H}^{-1}\mathbf{q}. \tag{7.5}$$

In practice, the Sherman–Morrison–Woodbury formula is applied by solving (7.5) and then solving (7.4). The cost of this process is twice the cost of calculating the action of $\mathbf{H}^{-1}$ on a vector, plus the cost of forming and inverting $(\mathbf{I} + \mathbf{V}^{\mathbf{T}}\mathbf{H}^{-1}\mathbf{U})$. By calculating the cost of a single 4-cell inversion this way, the cost of a complete RBOW relaxation can easily be obtained. To minimize the cost per iteration, an LU factorization of the matrix $(\mathbf{I}+\mathbf{V}^{\mathbf{T}}\mathbf{H}^{-1}\mathbf{U})$ corresponding to each 4-cell block is formed and stored in a separate setup phase. To facilitate the calculation of the action of $\mathbf{H}^{-1}$, LU factorizations of the single-cell diagonal blocks of $\mathbf{H}$ are also stored during the setup phase. Table 7.1 gives both the computational cost and storage requirement of one fine-grid RBOW relaxation, including the setup phase. One [illegible] (CU) is the number of operations necessary to compute the fine-grid residual, and one [illegible] (SU) is the memory required to store the current iterate. Notice the cost and storage requirement for the setup phase are given as upper bounds. The setup costs 36 CUs only if every cell in the domain has a different size and/or cross section. Regions of identical cells significantly reduce the setup cost. Similarly, the setup storage is 11 SUs only in the case of a minimal quadrature set (4 directions in two dimensions). Using more directions reduces the setup storage requirement in terms of SUs. The range of 4–6 CUs for the iteration cost is also a function of the quadrature set. Again, if a minimal set of 4 angles is chosen, the iteration cost is 6 CUs. This value decreases to 4 CUs as the number of angles rises.[7]

[7]The cost and storage requirements of the algorithm have a component that is independent of the number of angles, which is trivialized as the number of angles rises. This is why the values in the table are higher for fewer angles, in terms of CU/SU.

TABLE 7.1
Computational cost of one RBOW relaxation including setup phase. One CU is the cost of computing the fine-grid residual. One SU is the memory required to store the current iterate.

Phase	Operations	Storage
Setup	$\leq$ 36 CU	$\leq$ 11 SU
Iteration	4 – 6 CU	1 SU

The cost of the entire V-cycle can easily be obtained based on the cost of one fine-grid relaxation [7]. Let the cost of one fine-grid RBOW relaxation be one work unit (WU). Neglecting the cost of intergrid transfers (10% to 20% of total), the cost of a V(1,1)-cycle is

$$2(1 + 2^{-2} + 2^{-2*2} + \cdots + 2^{-n*2}) < \frac{2}{1 - 2^{-2}} = \frac{8}{3} \text{ WU}. \tag{7.6}$$

Using Table 7.1 and (7.6), the approximate cost of a V(1,1)-cycle in CUs is $\frac{8}{3}(4 -$ 6 CU) $\approx (11 -$ 16 CU).

8. Parallel scalability. The purpose of this section is to compare the time required to complete one RBOW relaxation to the time required to complete one global transport sweep on a parallel machine. In terms of the computational units defined above, a transport sweep requires only 1 CU, while the RBOW relaxation requires 4–6 CUs. Therefore, on a serial machine, the RBOW relaxation takes 4–6 times as long as a sweep. However, the picture is different on a parallel machine. In order to make a time-to-completion comparison, it is necessary to outline a model for the way in which information propagates through the processor grid during a sweep. First, imagine conducting a sweep in only one of the S_N directions, on a domain with only one spatial dimension, using a machine with P processors. The first processor, starting with the inflow boundary conditions, must perform its calculations and send the results to the next processor. Define the time required to complete this number of operations to be one single processor time (SPT). The time required to complete the sweep in this single S_N direction is simply P SPTs. In two spatial dimensions, the time required to complete a sweep in a single S_N direction is $2P^{\frac{1}{2}}$ SPTs. This is because, if one imagines the P processors to be arranged in a $P^{\frac{1}{2}} \times P^{\frac{1}{2}}$ array, the information can be passed through the processor grid on "diagonal bands" in a total of approximately $2P^{\frac{1}{2}}$ steps, as shown in Figure 8.1.

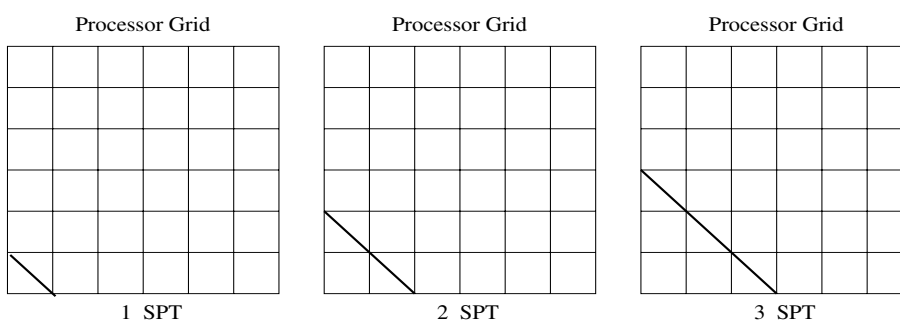


FIG. 8.1. *Transport sweep in one S_N direction and two spatial dimensions through a processor grid. One SPT is the time required for one processor to complete its computations.*

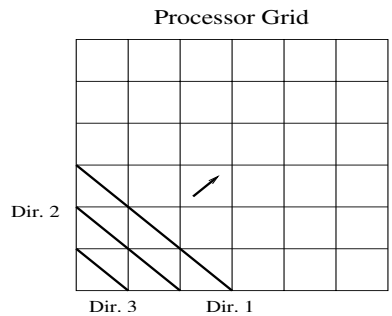


FIG. 8.2. *Transport sweep in multiple S_N directions and two spatial dimensions. The directional sweeps follow each other through the processor grid.*

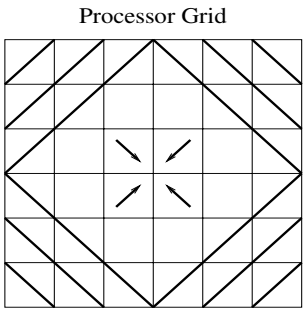


FIG. 8.3. *Transport sweep in multiple S_N directions and two spatial dimensions. Directional sweeps are initiated from the appropriate place in the processor grid.*

In general, if i is the number of spatial dimensions, the time required to complete the sweep in a single S_N direction is $iP^{\frac{1}{i}}$ SPTs. Now consider performing the sweep in all the S_N directions instead of just one. Returning to the 1-D example, once the first processor has completed its calculations and passed the results on to the second processor, it is free to begin a sweep in a second S_N direction. Naturally, this is true with more spatial dimensions as well, and the sweeps for the various S_N directions can follow each other through the processor grid like ducks in a row, as shown in Figure 8.2. If it were possible to begin all of the directional sweeps from the same starting point in the processor grid, it would be easy to see that the total time required to complete a sweep in M directions is $[iP^{\frac{1}{i}} + M]$ SPTs. In reality, in one dimension, half the directional sweeps must begin from the other side of the processor grid, and in two-dimensions, one-quarter of the directional sweeps begin in each of the four corners of the processor grid, as shown in Figure 8.3. This reduces the number of "ducks" in each row, but also creates a delay when processors begin to receive information from multiple directions. A careful analysis reveals that the total time-to-completion is still $[iP^{\frac{1}{i}} + M]$ SPTs even when the various directional sweeps are initiated from the appropriate place in the processor grid.

The number of SPTs required to complete one RBOW relaxation can be obtained more easily by using the serial result given above. In parallel, the number of operations each processor must do to complete its share of the RBOW step is simply (4–6) CU_{local}, where one CU_{local} is the number of operations required to compute the portion of the residual that one processor is responsible for. This is roughly the same number of operations that would be required to do a sweep in $\,,,$ directions on that

TABLE 8.1
Summary of computation time and communication time for one RBOW step and one global sweep on a parallel machine.

	RBOW	Sweep
Computation time	(4-6)M SPT	$[iP^{\frac{1}{i}} + M]$ SPT
Communication time	$4(t_l + k_{rbow}t_b)$	$[iP^{\frac{1}{i}} + M](t_l + k_{sw}t_b)$

processor's portion of the domain, and therefore requires M single processor times, where again, M is the total number of S_N directions. Therefore, the time to complete the RBOW step in parallel is (4–6)M SPTs, regardless of the number of processors being used[8].

The time-to-completion models described above consider only the time required to complete the necessary operations. However, the time required to complete the processor-to-processor communications might also be significant, and the nature of that communication is quite different for the RBOW step than it is for a transport sweep. While numerical tests on a parallel machine are beyond the scope of this paper, it is worthwhile to give a brief outline of the different communication requirements. Basic models of processor communication time describe the total time required to pass a message containing k units of data as [17]

$$t_l + kt_b. \tag{8.1}$$

The time t_l is called the message [illegible], and the reciprocal of t_b is called the [illegible]. In the case of a transport sweep, many small communications are required. Each time information is passed to the next group of processors; the message size is equal to the number of spatial unknowns on half of the processor boundary in only one S_N direction. The total time for all communication in a complete sweep is therefore $[iP^{\frac{1}{i}} + M](t_l + k_{sw}t_b)$, where k_{sw} indicates the relatively small message size associated with sweeps. In the case of the RBOW step, communication only needs to occur four times, once at the beginning of each color pass. However, the message size is larger, including the number of spatial unknowns on half of the processor boundary in all of the inflow directions (half of all the S_N directions). The total time for all communication in an RBOW step is $4(t_l + k_{rbow}t_b)$, where k_{rbow} indicates the larger message size. Clearly, the parallel time-to-completion for a transport sweep or an RBOW relaxation depends on the number of processors and the number of S_N directions, as well as the value of t_l and t_b on a given machine. Table 8.1 summarizes computation and communication time for the models described above.

While the above discussion sheds some light on the parallel scalability issue, it is only a first step towards a parallel performance comparison with DSA. In order to compare performance, one must consider two additional issues. First, the total iteration cost is a multiple of the relaxation cost. As discussed above, the V-cycle cost in two dimensions is approximately $\frac{8}{3}$ times the cost of the RBOW step. In the context of parallel computing, this factor of $\frac{8}{3}$ assumes perfect parallel scalability of the coarse-grid work. In reality, the coarse-grid work does not scale perfectly. It is

[8]This time-to-completion model for the RBOW step assumes no overlap in the spatial regions for which each processor is responsible. In reality, the shifting block process requires that a small overlap exist to facilitate the inversion of 4 cell blocks that straddle the line between two processors. If a significant number of cells are assigned to each processor, the effect of the overlap on time-to-completion is very small.

likely that at some point in the V-cycle, the number of unknowns in the coarse-grid problem will drop below the number of processors, and therefore certain processors will be idle. However, in most applications, the percentage of computational work that takes place while processors are idle is very small, and the effect on parallel scalability is insignificant. This issue is examined in [6]. In the case of an efficient DSA algorithm, the total iteration cost may in fact be close to the cost of the transport sweep. However, if the diffusion solve is accomplished via multigrid, the parallel scalability of the coarse-grid work affects the DSA algorithm just as it does the spatial multigrid algorithm described here. Furthermore, in the case of corner balance and other similar discretizations, a fully consistent DSA cannot be implemented at a computational cost near that of a single sweep. If an inconsistent DSA is used as a preconditioner for a Krylov method [18], the Krylov method itself represents additional computational cost. Second, the convergence factor obtained by the spatial multigrid method for a given problem is different from the convergence factor a DSA algorithm would obtain. Convergence factors for particular problems are discussed below.

9. Comparison to DSA. In order to properly compare spatial multigrid to DSA for a particular problem, one must consider not only the time required to complete an iteration, but also the convergence factor. In the case of a homogeneous thin domain, DSA gives excellent convergence due to the sweeps. For these domains, spatial multigrid struggles, especially if the number of S_N directions is large. For a homogeneous thick domain dominated by scattering, both methods have impressive convergence factors. As indicated above, the spatial multigrid algorithm produces a convergence factor of about .06 in the thick diffusive case. While the exact convergence factor obtainable from a DSA algorithm for the thick diffusive case may depend on the particular implementation, it is likely that it would be near the theoretical value of about 23 obtained from Fourier analysis. Finally, there is the heterogeneous case to consider, especially the case of the thin center. While tests have been done on thin center domains with DSA [18], convergence factors as defined by (5.1) and (5.2) are not given. Therefore, for the purpose of comparison, a DSA algorithm was coded and tested on the heterogeneous model domain described above.

In multigrid terms, DSA can be thought of as a 2-grid scheme in angle. The relaxation method is transport sweeps, and the coarse-grid problem is formed by taking a Galerkin P_1 closure[9] of the residual equation. The simplest way to implement a "fully consistent" DSA scheme is to simply solve the coarse grid system exactly, and this is what we have done. While this strategy is computationally expensive, the goal here is only to get an idea of what convergence factors DSA is capable of producing for certain heterogeneous problems. In [18], Warsa, Wareing, and Morel indicate that an inconsistent DSA used as a preconditioner for a Krylov method is a viable alternative to the fully consistent DSA. Therefore, for the purpose of this comparison, we do not consider the cost of the fully consistent DSA we have implemented, but only use it to get an idea of what convergence factors DSA is capable of producing for the thin center case. Table 9.1 shows the same tests as Table 6.3. The convergence factors for the spatial multigrid method are listed again, and the convergence factors for our implementation of DSA are also given. Clearly the performance of DSA degrades in much the same way as the performance of the spatial multigrid method does for the thin center problem.

[9] P_1 refers to a first order spherical harmonic expansion in angle. One forms the coarse-grid system by assuming the angular flux has a first order spherical harmonic expansion in angle, and then integrating the transport equation against the zeroth and first order spherical harmonics.

TABLE 9.1

Convergence factors for spatial multigrid and DSA on the heterogeneous model domain with vacuum center, thick pure scattering perimeter, and time step–based effective absorption. The thick region always has $\sigma_t h = 10^4$. Parentheses indicate convergence factor for a W-cycle instead of a V-cycle.

$\frac{1}{c\Delta t}$	$\frac{c\Delta t}{\Delta x}$	$\rho_{sp.mg}$	ρ_{dsa}
nx	1	$7 * 10^{-4}$	.03
$\frac{nx}{10}$	10	.12	.41
$\frac{nx}{100}$	100	.56(.38)	.84
$\frac{nx}{1000}$	1000	.88(.69)	.98

The reason for the degraded performance is the same in both cases; the error after relaxation is not sufficiently close to the range of interpolation. Our spatial multigrid method is based on the idea that RBOW relaxation leaves behind bilinear error. For the pure thick domain, this is true, and convergence is excellent. However, in the case of the thin center problem, the error after the RBOW step is not as close to the range of bilinear interpolation, and convergence suffers. Similarly, DSA is based on the idea that a transport sweep leaves behind error that is P_1 in angle. Again, for a pure thick problem, this is true, and convergence is excellent. In the thin center case, the error after the transport sweep in the thin region is much more complicated than P_1 in angle, and therefore the coarse-grid correction loses its effectiveness, and convergence suffers.

In conclusion, while a parallel performance comparison of a state-of-the-art DSA algorithm with our spatial multigrid method is beyond the scope of this paper, the discussion above allows for a rough comparison from which one can identify the types of problems and machines on which one method would likely prove faster than the other. The time required to complete one iteration of either algorithm depends on the number of processors, the number of S_N angles being used, and the speed of processor-to-processor communication. The convergence factors obtained by the two methods are also different and highly problem dependent.

10. Conclusions and future work. The multigrid algorithm presented here gives good convergence factors at a realistic cost for most homogeneous domains and is appropriate for heterogeneous domains in time-dependent applications. Sequential operations associated with global transport sweeps are avoided, making the method suitable for large parallel machines. Future work will focus on improving the convergence factors for heterogeneous domains, reducing computational cost, and applying the methodology to other discretizations.

REFERENCES

[1] M. L. ADAMS, *Subcell balance methods for radiative transfer on arbitrary spatial grids*, Transp. Theory Stat. Phys., 26 (1997), pp. 385–432.

[2] M. L. ADAMS AND E. W. LARSEN, *Fast iterative methods for discrete-ordinates particle transport calculations*, Progress in Nuclear Energy, 40 (2002), pp. 3–159.

[3] M. L. ADAMS AND W. R. MARTIN, *Diffusion-synthetic acceleration of discontinuous finite-element transport iterations*, Nucl. Sci. Eng., 111 (1992), pp. 145–167.

[4] S. F. ASHBY, P. N. BROWN, M. R. DORR, AND A. C. HINDMARSH, *A linear algebraic analysis of diffusion synthetic acceleration for the Boltzmann transport equation*, SIAM J. Numer. Anal., 32 (1995), pp. 128–178.

[5] A. BARNETT, J. E. MOREL, AND D. R. HARRIS, *A multigrid acceleration method for the* 1-*D* S_N *equations with anisotropic scattering*, Nucl. Sci. Eng., 102 (1989), pp. 1–21.

[6] W. L. Briggs, L. Hart, S. F. McCormick, and D. Quinlan, *Multigrid methods on a hypercube*, in Multigrid Methods: Theory, Applications, and Supercomputing, Lecture Notes in Pure and Appl. Math. 110, S. F. McCormick, ed., Dekker, New York, 1988, pp. 63–83.

[7] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, 2nd ed., SIAM, Philadelphia, 2000.

[8] A. Hoisie, O. Lubeck, and H. Wasserman, *Performance and scalability analysis of teraflop-scale parallel architectures using multidimensional wavefront applications*, Int. J. of High Performance Computing Applications, 14 (2000), pp. 330–346.

[9] B. D. Lansrud, *A Spatial Multigrid Iterative Method for Two-Dimensional Discrete-Ordinates Transport Problems*, Ph.D. thesis, Department of Nuclear Engineering, Texas A&M University, College Station, TX, 2005.

[10] E. Larsen, *Unconditionally stable diffusion-synthetic acceleration methods for the slab geometry discrete-ordinates equations. Part* I: *Theory*, Nucl. Sci. Eng., 82 (1982), p. 47.

[11] E. Larsen, *A grey transport acceleration method for time-dependent radiative transfer problems*, J. Comput. Phys., 78 (1988), pp. 459–480.

[12] E. E. Lewis and W. F. Miller, *Computational Methods of Neutron Transport*, American Nuclear Society, La Grange Park, IL, 1993.

[13] T. Manteuffel, S. McCormick, J. Morel, S. Oliveira, and G. Yang, *A fast multigrid algorithm for isotropic transport problems* I: *Pure scattering*, SIAM J. Comput., 16 (1995), pp. 601–635.

[14] T. Manteuffel, S. McCormick, J. Morel, and G. Yang, *A fast multigrid algorithm for isotropic transport problems* II: *With absorption*, SIAM Sci. J. Comput., 17 (1996), pp. 1449–1474.

[15] M. M. Mathis, N. M. Amato, and M. L. Adams, *A general performance model for parallel sweeps on orthogonal grids for particle transport calculations*, in Proceedings of the ACM International Conference on Supercomputing (ICS), Santa Fe, NM, 2000, pp. 255–263.

[16] J. E. Morel, J. E. Dendy, Jr., and T. A. Wareing, *Diffusion-accelerated solution of the two-dimensional S_N equations with bilinear-discontinuous differencing*, Nucl. Sci. Eng., 115 (1993), pp. 304–319.

[17] P. S. Pacheco, *Parallel Programming with MPI*, Morgan Kaufmann, San Francisco, 1997.

[18] J. S. Warsa, T. A. Wareing, and J. E. Morel, *Krylov iterative methods and the degraded effectiveness of diffusion synthetic acceleration for multidimensional S_N calculations in problems with material discontinuities*, Nucl. Sci. Eng., 147 (2004), pp. 218–248.

[19] M. Yavuz and E. W. Larsen, *Iterative methods for solving x-y geometry S_N problems on parallel architecture computers*, Nucl. Sci. Eng., 112 (1992), pp. 32–42.

SIAM J. SCI. COMPUT.
Vol. 29, No. 5, pp. 1918–1941
© 2007 Society for Industrial and Applied Mathematics

EFFICIENT PRECONDITIONING OF SEQUENCES OF NONSYMMETRIC LINEAR SYSTEMS*

JURJEN DUINTJER TEBBENS† AND MIROSLAV TŮMA†

Abstract. We present a new approach for approximate updates of factorized nonsymmetric preconditioners for solving sequences of linear algebraic systems. This approach is algebraic and it is theoretically motivated. It generalizes diagonal updates introduced by Benzi and Bertaccini [*BIT*, 43 (2003), pp. 231–244] and Bertaccini [*Electron. Trans. Numer. Anal.*, 18 (2004), pp. 49–64]. It is shown experimentally that this approach can be very beneficial. For example, it is successful in significantly decreasing the number of iterations of a preconditioned iterative method for solving subsequent systems of a sequence when compared with freezing the preconditioner from the first system of the sequence. In some cases, the updated preconditioners offer a rate of convergence similar to or even higher than the rate obtained when preconditioning with recomputed preconditioners. Since the updates are typically cheap and straightforward, their use is of practical interest. They can replace recomputing preconditioners, which is often expensive, especially in parallel and matrix-free environments.

Key words. preconditioned iterative methods, sparse matrices, sequences of linear algebraic systems, incomplete factorizations, factorization updates, Gauss–Jordan transformations, minimum spanning tree

AMS subject classifications. Primary, 65F10, 65F50, 65N22, 65H10; Secondary, 15A06

DOI. 10.1137/06066151X

1. Introduction.

We consider the solution of sequences of linear systems

$$A^{(i)}x = b^{(i)},\ i = 1, \ldots, \tag{1.1}$$

where $A^{(i)} \in \mathbb{R}^{n\times n}$ are general nonsingular sparse matrices and $b^{(i)} \in \mathbb{R}^n$ are corresponding right-hand sides. Such sequences arise in many applications such as computational fluid dynamics, structural mechanics, numerical optimization as well as in solving non-PDE problems. For example, a system of nonlinear equations $F(x) = 0$ for $F : \mathbb{R}^n \to \mathbb{R}^n$ solved by a Newton- or Broyden-type method leads to a sequence of problems

$$J(x_i)(x_{i+1} - x_i) = -F(x_i),\ i = 1, \ldots, \tag{1.2}$$

where $J(x_i)$ is the Jacobian evaluated in the current iteration x_i or its approximation [31], [32].

The solution of sequences of linear systems is the main bottleneck in many applications mentioned above. For instance, the solvers may need powerful preconditioners in order to be efficient, and computing preconditioners $M^{(1)}, M^{(2)}, \ldots$ for individual systems separately can be very expensive. There is a strong need for reduction of costs by sharing some of the computational effort among the subsequent linear systems.

A way to reduce the overall costs for solving systems of the type (1.2) is to modify Newton's method by skipping some Jacobian evaluations as in the Shamanskii

*Received by the editors May 31, 2006; accepted for publication (in revised form) January 8, 2007; published electronically September 28, 2007. This work was supported by the National Program of Research "Information Society" under project 1ET400300415.

http://www.siam.org/journals/sisc/29-5/66151.html

†Institute of Computer Science, Czech Academy of Sciences, Pod Vodárenskou věží 2, 18207 Praha 8, Czech Republic (tebbens@cs.cas.cz, tuma@cs.cas.cz). The work of the first author was supported by project KJB100300703 of the Grant Agency of the Academy of Sciences of the Czech Republic.

combination of Newton's method and the Newton-chord method [11], [54]. In this way we get a sequence of systems with identical matrices, and techniques for solving systems with more right-hand sides may be applied provided that the right-hand sides are available a priori; see, e.g., [45], [25], [55], [60]. However, combinations of Newton's method and the Newton-chord method have much weaker nonlinear convergence properties than the standard Newton method.

A different approach to reducing the overall costs, which is usually more efficient, is based on reusing the preconditioner (using the same preconditioner for a sequence of linear systems), but recomputing (approximate) Jacobians $A^{(i)}$ [12], [37], [38]. This approach is very natural in the context of a matrix-free environment, where the system matrices $A^{(i)}$ may be available only in the form of matrix-vector products (matvecs); see also the overview of matrix-free Newton–Krylov methods in [36].

Another way to avoid efficiency and/or memory related problems connected to algebraic preconditioning is to use conceptually simpler preconditioners derived from the physics of the problem. In some PDE problems the original operator can be replaced by a simpler one. Early results related to preconditioning by fast solvers can be found in [16], [24]. For instance, the simpler operator can be a scaled diffusion operator for a PDE with variable coefficients or a convection-diffusion operator [12], [34], [36]. In the algebraic setting, simple preconditioners derived from stationary iterative methods can be used. Preconditioning by the symmetric part of a nonsymmetric matrix was proposed in [17], [62]; see also [14]. Another popular preconditioning technique for general convection-diffusion-reaction models is based on generalizations of ADI splitting from [48]; see, e.g., [34]. Note that we restrict ourselves here to linear preconditioners; for nonlinear preconditioning techniques we refer, e.g., to [13] and the references therein. In order to make the preconditioning more efficient and to simplify the preconditioner setup even more, reformulations based on nested iterations were introduced; see, e.g., [59]. For instance, the flexible Krylov-subspace framework enables theoretically sound implementations of inner-outer Krylov-subspace methods [51], [56].

Freezing the preconditioner or using simple preconditioning techniques may not be enough for fast convergence in practice. Our contribution proposes new and efficient approximate updates of a preconditioner which is factorized as $LDU \approx A$. The updated preconditioners are then used for solving the subsequent members of the sequence. We do not assume any simple relation among the systems of the sequence. Note that straightforward approximate small rank preconditioner updates can be obtained in case of a sequence of linear systems from a quasi-Newton method, as shown in the symmetric and positive definite case in [44], [8]. It is well known how to compute the updates of sparse decompositions [19], [20], [21]; the techniques for updates starting in early papers, e.g., [27], and having mainly the intent of being applied to the simplex method of linear programming and its extensions are a classical part of numerical mathematics. Another algebraically motivated strategy used in preconditioning sequences of systems is to use adaptive information generated by Krylov-subspace methods [2]. Recent work on recycling explicit information from Krylov subspaces can be found in [41], [47].

In this paper we directly generalize the approximate diagonal updates which are useful for solving the parabolic PDEs proposed in [3]; see also [9]. This generalization consists in modifying general off-diagonal entries. Our numerical experiments show that the generalizations are competitive with recomputing the factorized nonsymmetric preconditioners in terms of achieving similar convergence rates for subsequent systems. Moreover, forming the updates can be significantly cheaper than recomput-

ing the preconditioner. As far as we know, there are no theoretical or experimental results in this direction. We give a couple of theoretical explanations for the good performance of the updates and discuss some unexpected effects which help to improve the convergence and, as far as we know, have not been communicated before. The strategy which we use forms the updated preconditioner from two separate layers: entries of the original factorized preconditioner and scaled entries of the matrix update. For the sake of quality and efficiency we typically need to exploit only a part of the update. This part may result from a Gauss–Seidel type of splitting, or it may be found in a more sophisticated way. In this paper we treat both cases.

The paper is organized as follows. In section 2 we present a brief introduction into preconditioner updates and motivate the basic form of our updated factorizations. In section 3 we describe the new techniques for approximate updating. The results of numerical experiments with the new algorithms are presented and discussed in section 4. Directions for current and future research are given in section 5. Throughout the paper, $\|\cdot\|$ denotes an arbitrary matrix norm.

2. The ideal updated preconditioner. Some of the strategies for updating preconditioners that we mentioned in the introduction are linked with specific classes of linear solvers (e.g., recycling Krylov subspaces) and nonlinear solvers (e.g., Broyden-type methods) or they were designed for symmetric matrices. In this paper we wish to consider sequences of general, nonsymmetric systems that are solved by preconditioned iterative methods. We address here the following problems: First, how can we update, in theory, a preconditioner in such a way that the updated preconditioner is likely to be as powerful as the original one? And second, how can we approximate, in practice, such an update in order to obtain a preconditioner that is inexpensive to apply and yet useful?

In order to simplify the notation, we consider two linear systems of dimension n denoted by $Ax = b$ and $A^+x^+ = b^+$. Denote the difference matrix $A - A^+$ by B and let M be a preconditioner approximating A. Some information about the quality of the preconditioner M can be taken from a norm of the matrix

$$A - M \tag{2.1}$$

or from some norm of the matrix

$$I - M^{-1}A \quad \text{or} \quad I - AM^{-1} \tag{2.2}$$

if we consider preconditioning from the left or right, respectively (see, e.g., [3]). If preconditioners are in factorized form, both (2.1) and (2.2) should be considered in practice since the preconditioners can suffer from two types of deteriorations. While the norm of the matrix (2.1) expresses [illegible] of the preconditioner, the norms of the matrices (2.2) relate to its [illegible] [15]; see also [5]. We will define updated preconditioners M^+ for A^+ whose accuracy and stability are close to the accuracy and stability of M for A. For their derivation we concentrate on the norm of the matrix (2.1) because of its simplicity. Later in this section we present theoretical results demonstrating that both accuracy and stability of the derived updates are comparable to or even better than those of M for A.

We immediately obtain

$$\|A - M\| = \|A^+ - (M - B)\|.$$

Hence $M^+ \equiv M - B$ represents an updated preconditioner for A^+ of the same "level" of accuracy as M represents for A. We will call it the [illegible] updated preconditioner.

Note that there may very well exist different preconditioners that are ideal with respect to a norm of $A^+ - M^+$. Just consider $M^+ = M - C$ for some matrix $C \neq B$ with

$$\|A - M\| = \|A^+ - M^+\| = \|A^+ - M + C\|.$$

Because B is often readily available, we will concentrate on $M^+ = M - B$.

If we want to use M^+ as a preconditioner, we need to multiply vectors with the inverse of M^+ in every iteration of the linear solver. In some problems, the difference matrix B is such that $(M - B)^{-1}$ can be obtained from M^{-1} with low costs. For instance, if B has small rank, M^+ can be easily inverted using the Sherman–Morrison formula; see, e.g., [44, 8]. In general, however, the ideal updated preconditioner cannot be used since multiplication of vectors with $(M - B)^{-1}$ is expensive. Instead, we will consider cheap approximations of $(M - B)^{-1}$.

In this paper we will assume that M is given in the form of a triangular decomposition as $M = LDU \approx A$, where L and U have unit main diagonal. The approximate updates of factorized preconditioners which we will describe below typically assume that the matrices have a strong diagonal. Note that this assumption is very similar to theoretical assumptions which are generally required to get simple incomplete factorizations without a breakdown. For example, standard ILU(0) and AINV preconditioners are proved to be breakdown-free if the system matrix is an H-matrix [43], [6]. In order to extend the breakdown-free property to more general matrices, we need to change the decomposition by modifications which make the diagonal stronger, e.g., by a preliminary shift [43], [40] (see also [33], [1]) or by global modification of the decomposition [57], [35], [4]. The transfer from diagonal dominance of the matrix to diagonal dominance of the factors is discussed, for example, in [7] (cf. [3]) or in the practical reordering strategies based on strong transversals [46], [22], [23]. In the following we tacitly assume matrices are given in such form that the factors L and U more or less approximate the identity matrix.

If $M - B$ is invertible, we can approximate its inverse by a product of more factors which are easier to invert. For example, we can replace $(M - B)^{-1}$ by a product of inverses of triangular matrices and by an inverse of a difference of matrices where a diagonal matrix is used instead of M, as in

$$(M - B)^{-1} = U^{-1}(D - L^{-1}BU^{-1})^{-1}L^{-1} \approx U^{-1}(D - B)^{-1}L^{-1}, \tag{2.3}$$

provided that $D - B$ is nonsingular. Now assume $\overline{D - B}$ is a nonsingular approximation of $D - B$ that can be inverted inexpensively. Then we can define a preconditioner M^+ via the last expression in (2.3) as

$$M^+ = L(\overline{D - B})U. \tag{2.4}$$

In the symmetric case, this preconditioner changes to $M^+ = L(\overline{D - B})L^T$; hence symmetry is preserved if we choose $\overline{D - B}$ appropriately. Here we are primarily interested in the nonsymmetric case, and in this case we can further simplify the update. For example, we can approximate as

$$(M - B)^{-1} = (DU - L^{-1}B)^{-1}L^{-1} \approx (DU - B)^{-1}L^{-1} \tag{2.5}$$

if $DU - B$ is nonsingular. If $\overline{DU - B}$ denotes a nonsingular and easily invertible approximation of $DU - B$, then we define M^+ by

$$M^+ = L(\overline{DU - B}). \tag{2.6}$$

In comparison with (2.4), it seems to be much easier to deal only with two factors. An analogue of (2.5) is approximation through

$$(M-B)^{-1} = U^{-1}(LD - BU^{-1})^{-1} \approx U^{-1}(LD-B)^{-1}. \tag{2.7}$$

In our experiments we choose between approximation with (2.5) or (2.7) adaptively (we explain this later on). We describe our theoretical results for the case (2.5) only.

A first question is whether the update (2.6) has the potential to be more powerful than the frozen preconditioner $M = LDU$ for A^+. In the following simple lemma we express the relation of the frozen preconditioner to the updated form quantitatively.

LEMMA 2.1. [illegible] $\|A - LDU\| = \varepsilon\|A\| < \|B\|$ [illegible] (2.6)

[illegible]

$$\begin{aligned}\|A^+ - M^+\| &\le \frac{\|L(DU - \overline{DU-B}) - B\| + \varepsilon\|A\|}{\|B\| - \varepsilon\|A\|} \cdot \|A^+ - LDU\| \\ &\le \frac{\|L\|\,\|DU - B - \overline{DU-B}\| + \|L-I\|\,\|B\| + \varepsilon\|A\|}{\|B\| - \varepsilon\|A\|} \cdot \|A^+ - LDU\|.\end{aligned}$$

[illegible] We get directly

$$\begin{aligned}\|A^+ - M^+\| &= \|A - B - L(\overline{DU-B})\| = \|(A - LDU) + L(DU - \overline{DU-B}) - B\| \\ &\le \left(\varepsilon\|A\| + \|L(DU - \overline{DU-B}) - B\|\right) \frac{\|B\| - \varepsilon\|A\|}{\|B\| - \varepsilon\|A\|} \\ &\le \left(\varepsilon\|A\| + \|L(DU - \overline{DU-B}) - B\|\right) \frac{\|(A - LDU) - B\|}{\|B\| - \varepsilon\|A\|} \\ &\le \|A^+ - LDU\| \frac{\|L(DU - \overline{DU-B}) - B\| + \varepsilon\|A\|}{\|B\| - \varepsilon\|A\|} \\ &= \|A^+ - LDU\| \frac{\|L(DU - \overline{DU-B} - B) + (L-I)B\| + \varepsilon\|A\|}{\|B\| - \varepsilon\|A\|} \\ &\le \|A^+ - LDU\| \frac{\|L\|\,\|DU - B - \overline{DU-B}\| + \|L-I\|\,\|B\| + \varepsilon\|A\|}{\|B\| - \varepsilon\|A\|}. \qquad \square\end{aligned}$$

The multipliers of $\|A^+ - LDU\|$ in Lemma 2.1 can be smaller than one if $\overline{DU-B}$ is close to $DU - B$ and if $\|L - I\|$ tends to be small. In practice, taking into account preconditioner modifications to improve diagonal dominance, this is often realistic. Note that the assumption $\|A - LDU\| = \varepsilon\|A\| < \|B\|$ is satisfied as soon as we have a strong preconditioner $M = LDU$.

The lemma states, apart from showing a relation to the frozen preconditioner, that for $\varepsilon\|A\|$ small enough a good approximation to $\overline{DU-B}$ combined with a close to diagonal factor L yields an accurate preconditioner which [illegible] be as powerful as a recomputed preconditioner. If we have a recomputed preconditioner M^R with say, $\|A^+ - M^R\| = \delta = \|A - M\|$, then based on (2.5) we expect $\|A^+ - M^+\| \ge \delta$. But the previous lemma shows $\|A^+ - M^+\| < \delta$ is not at all excluded. In section 4 we will show experimentally that the update (2.6) in some cases gives a higher convergence rate than if the preconditioner is recomputed.

The following theorem shows in a different way that, under the given assumptions, the quality of the update may be better than that of recomputed preconditioners if the approximation $\overline{DU-B}$ is favorably chosen. Since Lemma 2.1 is related to the accuracy according to (2.1), the theorem considers its quality with respect to (2.2).

The result is a straightforward generalization of a result from [9]. To simplify the description, the scaled updated approximate factor $D^{-1}(\overline{DU-B})$ will be denoted by $\overline{U-D^{-1}B}$, and $U^{-1}(\overline{U-D^{-1}B})$ will be denoted by $I-\overline{U^{-1}D^{-1}B}$.

THEOREM 2.2. $LDU+E=A$ E $\|\overline{U^{-1}D^{-1}B}\|_2 \le 1/c < 1$ $\|\cdot\|_2$ σ_i

$$(I-L)B + L\left(\overline{DU-B} - (DU+L^{-1}E-B)\right)$$

$$\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_t \ge \delta \ge \sigma_{t+1} \ge \cdots \ge \sigma_n$$

t $t \ll n$ $\delta > 0$ $(\overline{DU-B})$ $D=\operatorname{diag}(d_1,\ldots,d_n)$ F Δ

$$(\overline{DU-B})^{-1}L^{-1}A^+ = I+\Delta+F, \tag{2.8}$$

$(\Delta) \le t$

$$\|F\|_2 \le \frac{c}{c-1}\max_i \frac{\delta}{|d_i|}\|L^{-1}\|_2\|U^{-1}\|_2.$$

We have

$$\begin{aligned} L(\overline{DU-B}) - A^+ &= L(DU+L^{-1}E-B+\overline{DU-B}-(DU+L^{-1}E-B)) - A^+ \\ &= (I-L)B + L\left(\overline{DU-B}-(DU+L^{-1}E-B)\right). \end{aligned}$$

By assumption, the SVD of the latter matrix can be written as

$$(I-L)B + L\left(\overline{DU-B}-(DU+L^{-1}E-B)\right) = W\Sigma V^T$$
$$= W\operatorname{diag}(\sigma_1,\ldots,\sigma_t,0,\ldots,0)V^T + W\operatorname{diag}(0,\ldots,0,\sigma_{t+1},\ldots,\sigma_n)V^T \equiv \Delta_1 + F_1,$$

where $\operatorname{rank}(\Delta_1) \le t$ and $\|F_1\|_2 \le \delta$. Hence

$$L(\overline{DU-B}) - A^+ = \Delta_1 + F_1$$

and

$$(\overline{DU-B})^{-1}L^{-1}A^+ = I - (\overline{DU-B})^{-1}L^{-1}\Delta_1 - (\overline{DU-B})^{-1}L^{-1}F_1.$$

By setting

$$F \equiv -(\overline{DU-B})^{-1}L^{-1}F_1, \qquad \Delta \equiv -(\overline{DU-B})^{-1}L^{-1}\Delta_1,$$

we get (2.8), where $\operatorname{rank}(\Delta) \le t$. The matrix F can be bounded by

$$\|F\|_2 \le \|L^{-1}\|_2 \left\|\left(D(\overline{U-D^{-1}B})\right)^{-1}\right\|_2 \delta;$$

hence

$$\begin{aligned} \|F\|_2 &\le \max_i \frac{\delta}{|d_i|}\|L^{-1}\|_2\|(\overline{U-D^{-1}B})^{-1}\|_2 \\ &\le \max_i \frac{\delta}{|d_i|}\|L^{-1}\|_2\|U^{-1}\|_2\|(I-\overline{U^{-1}D^{-1}B})^{-1}\|_2. \end{aligned}$$

By assumption, $\|\overline{U^{-1}D^{-1}B}\|_2 \leq 1/c < 1$, and, consequently,

$$\begin{aligned}\|F\|_2 &\leq \max_i \frac{\delta}{|d_i|}\|L^{-1}\|_2\|U^{-1}\|_2 \left(1 - \|\overline{U^{-1}D^{-1}B}\|_2\right)^{-1} \\ &\leq \frac{c}{c-1}\max_i \frac{\delta}{|d_i|}\|L^{-1}\|_2\|U^{-1}\|_2. \qquad \square\end{aligned}$$

Note that if the matrix F in (2.8) is zero, then the preconditioned system is a rank t update of the identity, and Krylov-subspace methods converge, in exact arithmetics, in at most $t+1$ iterations.

In the following section we propose approximations $\overline{DU-B}$ of $DU-B$ that can be efficiently computed and that lead to preconditioners that are inexpensive to apply. All techniques we present can be analogously formulated for updates of the form $(\overline{LD-B})U$ corresponding to (2.7).

3. Approximate preconditioner updates. We propose the following strategies to approximate $DU-B$ by an easily invertible matrix $\overline{(DU-B)}$. A first obvious but effective strategy is to set $\overline{DU-B} \equiv triu(DU-B)$, where $triu$ denotes the possibly sparsified upper triangular part (including the main diagonal). This results in the preconditioner

$$M^+ = L(DU - triu(B)), \tag{3.1}$$

which can be obtained entirely for free. The additional cost for applying this preconditioner is one triangular sweep with the triangular part of B if we store B and U separately. We may also merge them; then the additional sweep can be virtually free if the sparsity patterns of $triu(B)$ and U are close enough. We will call the update constructed by considering entries only from one triangular part the [illegible]. A trivial structured sparsification is given by

$$\overline{DU-B} \equiv \mathrm{diag}(DU-B),$$

which is a straightforward application of an approach from [3] to nonsymmetric problems.

As we show in the experiments, the simple update (3.1) and its analogue

$$M^+ = (LD - tril(B))U \tag{3.2}$$

seem to be powerful in many problems. One expects them to be particularly suited when one triangular part of B clearly dominates the other. The typical situation of that kind arises when matrices come from upwind/downwind discretization schemes. Nevertheless, as they take into account only one triangular part of the difference matrix B, there may be applications where important information is lost, leading to weak convergence. In the following we present a technique to replace $DU-B$ by an easily invertible matrix which is in general not triangular.

Denote the matrices $\mathrm{diag}(\overline{DU-B})$ by $\tilde{D}$, and $\tilde{D}^{-1}(\tilde{D} - \overline{DU-B})$ by $\tilde{B}$, respectively. Then $\tilde{B}$ has zero diagonal and we can write

$$\overline{DU-B} = \tilde{D}(I - \tilde{B}). \tag{3.3}$$

To motivate the scaling transformation in (3.3) consider for a moment the case when $\tilde{B} = \beta e_i e_j^T$ for some $1 \leq i,j \leq n, i \neq j$, and recall that we assume $\overline{DU-B}$ is nonsingular; hence so is $I - \tilde{B}$. Then we get, with the Sherman–Morrison formula,

$$(I-\tilde{B})^{-1} = I + \beta e_i e_j^T/(1-\beta e_j^T e_i) = I + \beta e_i e_j^T = I + \tilde{B}. \tag{3.4}$$

The matrix in (3.4) is equal to the identity modified by an off-diagonal entry β at the position (i, j). That is, $(I - \tilde{B})$ is a special Gauss–Jordan transformation [28], it is inverted without costs, and it has a fill-in free inverse.

Based on this well-known fact, in the following we will try to find approximations $\overline{DU - B}$ of $DU - B$ such that the scaled matrix $I - \tilde{B}$ can be written as a product of Gauss–Jordan transformations

$$(I - e_{i_1}\tilde{b}_{i_1*})(I - e_{i_2}\tilde{b}_{i_2*}) \dots (I - e_{i_K}\tilde{b}_{i_K*}), \qquad K \leq n-1, \tag{3.5}$$

where $\tilde{B} = (\tilde{b})_{ij}$. Denote the sparsity structure of a row i of $\tilde{B}$ (with zero diagonal) by $row(i)$, that is, $row(i) = \{k | i \neq k \wedge \tilde{b}_{ik} \neq 0\}$. The multiplication $(I - \tilde{B})^{-1}v$ for a given vector v is very cheap, as stated in Observation 3.1.

OBSERVATION 3.1. [illegible] (3.5) [illegible] $2\sum_{j=1}^{K} |row(i_j)|$

It is well known that any unit upper triangular matrix $I - \tilde{B}$ from (3.3) can be trivially written as the product $R_{n-1} \dots R_1$ of $n-1$ elementary triangular matrices $R_i = I - e_i\tilde{b}_{i*}$ for $i = 1, \dots, n-1$. Hence using (3.1) may be considered a special case of (3.5). The following theorem shows a necessary and sufficient condition for the existence of a decomposition of $I - \tilde{B}$ of the form (3.5).

THEOREM 3.1. $I - \tilde{B} = I - \sum_{j_l : l=1,\dots,K} e_{j_l}\tilde{b}_{j_l*}$ [illegible]

$$I - \tilde{B} = (I - e_{i_1}\tilde{b}_{i_1*})(I - e_{i_2}\tilde{b}_{i_2*}) \dots (I - e_{i_K}\tilde{b}_{i_K*}) \tag{3.6}$$

[illegible]

$$i_l \notin \bigcup_{k=1}^{l-1} row(i_k) \text{ for } 2 \leq l \leq K \tag{3.7}$$

[illegible] $i_1, \dots, i_K$ [illegible] $\{j_1, \dots, j_K\} = \{i_1, \dots, i_K\}$ [illegible]. The equivalence of (3.6) and (3.7) follows from the orthogonality of the unit vector e_{i_l} with respect to all $\tilde{b}_{i_k*}$ for $k < l$, $1 \leq l \leq K$. □

Based on Theorem 3.1 we first propose a greedy procedure to find a suitable approximation $\overline{DU - B}$ with $I - \tilde{B}$ satisfying (3.6). Consider a sequential choice of indices $i_1, \dots, i_K$, where $K \leq n-1$ are determined by the algorithm. In each step we keep and update a set of [illegible] $\mathcal{R}$ initialized by $\{1, \dots, n\}$. After choosing a row i we remove from $\mathcal{R}$ all the rows $j \in \mathcal{R}$ for which $\tilde{b}_{ij} \neq 0$.

ALGORITHM 3.1. We use this algorithm to approximate $DU - B$ by a matrix which, scaled by its diagonal, can be written in the form (3.6)

(1) [illegible] $\mathcal{R} = \{1, \dots, n\}$ $K = 0$
(2) [illegible] $k = 1, \dots, n$
(3) [illegible] $row(k) = \{i | i \neq k \wedge |(DU - B)_{ki}| \neq 0\}$
(4) [illegible] $p_k = \sum_{j \in row(k)} |(DU - B)_{kj}|$
(5) [illegible]
(6) [illegible] $\mathcal{R} \neq 0$
(7) [illegible] $i \in \mathcal{R}$ [illegible] $p_i - \sum_{j \in \mathcal{R} \cap row(i)} p_j$
(8) [illegible] $K = K + 1$ $i_K = i$ $\mathcal{R} = \mathcal{R} \backslash \{row(i_K) \cup i\}$
(9) [illegible]

The row indices $i_1, \dots, i_K$ provided by Algorithm 3.1 then determine the approximation in (3.3) with $I - \tilde{B}$ equal to the product (3.5). The heuristic criterion in step (7) aims, on the one hand, to choose the row of $DU - B$ with the largest entries. On

the other hand, it stimulates the choice of a row which results, based on condition (3.7), in removal of candidate rows with small entries. To balance between the two heuristics one may want to introduce a weighting parameter ω and use

(7′) $i \in \mathcal{R}$ $p_i - \omega \cdot \sum_{j \in \mathcal{R} \cap row(i)} p_j$.

Clearly, the algorithm may find more factors of (3.6) if there are fewer nonzero entries in the searched rows. Therefore it may be reasonable to perform some dropping strategy on-the-fly when running the algorithm by substituting step (3) with

(3′) *set* $row(k) = \{i | i \neq k \wedge |(DU - B)_{ki}| > tol\}$

for a predefined drop tolerance *tol*. Apart from tolerance-based dropping, sparsification based on the given mask may enhance the effectiveness of our strategy. Note that sparsification not only helps in covering as many rows as possible by Gauss–Jordan transformations, but it also leads to less expensive matvecs with the inverse of (3.5).

A more elegant and systematic way to get an unstructured update based on Gauss–Jordan transformations can be described by the following bipartite graph model. Let us define the bipartite graph of $(DU - B)$ as $G(DU - B) = (R, C, E)$, where $R = \{1, \ldots, n\}$, $C = \{1', \ldots, n'\}$ and $E = \{(i, j') | (DU - B)_{ij} \neq 0\}$. Then we have the following result.

THEOREM 3.2. $T = (V_T, E_T)$ $G(DU - B)$ $\{(i, i') | 1 \leq i \leq n\} \subseteq E_T$ $\overline{DU - B} \in \mathbb{R}^{n \times n}$

$$\overline{(DU - B)}_{ij} = \begin{cases} (DU - B)_{ij} & \text{if } (i, j') \in E_T, \\ 0 & \text{otherwise,} \end{cases}$$

(3.3) (3.5)

. First consider the case when the spanning forest T is not connected. Components of T induce a block-diagonal splitting of $\overline{DU - B}$, and matrices corresponding to individual blocks can be mutually multiplied in any order without causing any fill-in. Consequently, we can assume without loss of generality that T is connected and that T is a spanning tree. In the following we will show how to form the sequence of Gauss–Jordan transformations from the left to the right.

Our assumption implies that T contains at most $n - 1$ edges (i, j') with $i \neq j$. There exists a free row vertex $i \in R$ in T which is in T incident only to the edge (i, i') such that there is an edge $(k, i') \in E_T$ for some k. Set $i_1 = i$. Then remove from T the vertices $i \in R$, $i' \in C$ and all edges incident to them. Clearly, the updated tree T contains a free row vertex again. By repeating the choice of free row vertices and updates T in this way we get the sequence $i_1, \ldots, i_{n-1}$. If we rewrite as $I - \tilde{B}$ the matrix $\overline{DU - B}$ scaled by its diagonal, we have $I - \tilde{B} = (I - e_{i_1} \tilde{b}_{i_1 *})(I - e_{i_2} \tilde{b}_{i_2 *}) \ldots (I - e_{i_{n-1}} \tilde{b}_{i_{n-1} *})$ which proves the theorem. □

Theorem 3.2 implies the following algorithmic strategy to find a matrix $\overline{DU - B}$ which would approximate $DU - B$ and could be expressed as a product of Gauss–Jordan transformations.

ALGORITHM 3.2. We use this algorithm to find $\overline{DU - B}$ such that (3.6) is satisfied based on a bipartite graph of $DU - B$.

(1) Find a spanning forest $T = (V_T, E_T)$ of $G(DU - B)$ of maximum weight with edge weights $w_{ij} = |(DU - B)_{ij}|$ for $(i, j') \in E_T$ such that $\{(i, i') | 1 \leq i \leq n\} \subseteq E_T$.
(2) Find the entries of $\tilde{B}$ (and corresponding entries of $\overline{DU - B}$) as well as a feasible ordering of Gauss–Jordan factors for $i_1, \ldots, i_{n-1}$ in (3.5) with Theorem 3.2.

(3) For each $k = 2, \ldots, n$ add to $\overline{DU - B}$ all entries $(DU - B)_{i_k l}$ of $DU - B$ such that $l \in \{i_1, \ldots, i_{k-1}\}$.

Note that in the last step of Algorithm 3.2 we possibly put into $\overline{DU - B}$ many more nonzero entries than the $2n - 1$ entries provided by the weighted spanning forest. This is possible because of Theorem 3.1. The complexity of the weighted minimum spanning forest (here we need, in fact, a weighted maximum forest) is $O(m \log m)$ for the Kruskal algorithm [39] and $O(n + m \log m)$ for the Prim algorithm [50], where m is the number of edges in the graph G. Note, in addition, that we start with the partial spanning tree with the set of edges $\{(i, i')|1 \leq i \leq n\}$. While in some cases the algorithms may seem time consuming, this procedure can provide useful updates. As in Algorithm 3.1, we can sparsify $DU - B$ by discarding entries smaller than a certain drop tolerance *tol*, which reduces the value of m and therefore also the computational complexity.

From Lemma 2.1 it is clear that the quality of the approximation of $DU - B$ may play a decisive role in the power of the preconditioner $M^+ = L(\overline{DU - B})$. In practice, the way that the original incomplete decomposition is constructed (scaling L during the construction, pivoting) can strongly support the quality of $\overline{DU - B}$. In order to use the most powerful type of update, in our experiments we switch adaptively between (3.1) and (3.2) based on the weighting of both triangular parts of B and use an unstructured update based on Algorithm 3.1 or 3.2 if its weighting is the most important. More precisely, we compute sums of magnitudes of entries in the triangular parts of the matrices and simulate runs of Algorithms 3.1 and 3.2 to get the sum of magnitudes of entries covered by the unstructured update. We then use the strategy which corresponds to the maximum value among these sums.

It can and often does happen that, in spite of the fact that the updated preconditioner loses some information about the system matrix, it yields a better convergence rate than if the preconditioner would be recomputed from the scratch. There are several possible explanations for this phenomenon. First, note that we showed theoretically in Lemma 2.1 and Theorem 2.2 that our updated preconditioners have the potential to be stronger than recomputed factorizations. In practice, it frequently happens that by updating the preconditioner we relate it to a previous decomposition which is more diagonally dominant than a recomputed decomposition. A part of the stable triangular factors is inherited and the update may even stabilize less stable factors of the initial factorization. Note that a modified old decomposition might be useful in general, but, e.g., in the related strategy [43], the size of the modification should be typically rather small to get a useful preconditioned iterative method. This is exactly what happens when modifying with entries of difference matrices B that are typically small compared to those of $A^{(i)}$. In addition, updates appear to perform better also in cases where there is no instability. We presume this is so because the preconditioner may be favorably modified by the additional structural information given by the update. To our knowledge, this conjecture is stated for the first time. An overlooked fact is that the most powerful dual-threshold incomplete decompositions and inverse decompositions can be very memory-efficient, but they may discard the structure of the problem. Our updates can add to a memory-efficient decomposition cheap and useful information about the structure, as seems to be clear from our experiments. We believe that such a strategy might be used to improve constructing general preconditioners in some cases. We might consider the update as a simple and efficient way to modify off-diagonal entries of the preconditioner, thus getting a generalization of diagonal modifications from [43] or forced diagonal modifications introduced in [33]. It is not unusual that level-based incomplete decompositions are

much better than their sophisticated counterparts. Such a behavior has been observed on some VENKAT matrices from the Harwell–Boeing collection, where powerful and compact dual threshold ILUT [52] preconditioners are less efficient than often very dense but reasonably structured ILU preconditioners using the concept of levels [61], [29].

The next section is devoted to numerical experiments with the most promising updates introduced in the paper.

4. Numerical experiments. In this section we present results of numerical experiments with preconditioned Krylov-subspace methods for solving sequences of systems of linear algebraic equations, where updated preconditioners are compared with recomputed and frozen preconditioners. We consider the sequences in three application problems. The first and second problems were generated with the optimization software UFO [42]. The last application is based on [10]. Software for the problem was kindly provided by Philipp Birken. We present results with several kinds of ILU preconditioners to show that the introduced techniques are quite general. In order to show a larger spectrum of various results, some of the computations were done in MATLAB using its ILU decomposition script. We used MATLAB version 7.0. Most of the tests, in particular for larger problems, were written in Fortran 90 and were compiled by Compaq Visual Fortran 6.6a. The codes were run on a computer with Intel Pentium 4, 3GHz processor, 1GB RAM memory, and 512k L2 cache.

As an accelerator, the BiCGSTAB [58] iterative method with right preconditioning was used. We also performed some experiments with the restarted GMRES [53] method and the transpose-free QMR [26] method. The results were similar, and we do not report on them here. Iterations were stopped when the Euclidean norm of the residual was decreased by seven orders of magnitude. Nevertheless, in our experiments we observed close to linear behavior of convergence curves of the preconditioned iterative method. Therefore, we expect qualitatively the same results for weaker or nonuniform stopping criteria used in nonlinear solvers.

Our first test problem is a two-dimensional nonlinear convection-diffusion model problem which we use to illustrate various aspects of the proposed strategies (general behavior of the strategies, choice of parameters, values of the bounds in Lemma 2.1). It has the form (see, e.g., [31])

$$-\Delta u + Ru\left(\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y}\right) = 2000x(1-x)y(1-y) \tag{4.1}$$

on the unit square, discretized by 5-point finite differences on a uniform 70×70 grid. The initial approximation is the discretization of $u_0(x, y) = 0$. We choose the modest Reynolds number $R = 50$ in order to avoid potential discretization problems which may ask for adding stabilization terms. We obtain a small sequence of 7 matrices with 24220 nonzeros each (in the tables we denote the number of nonzeros by nnz).

Our update techniques are particularly beneficial when recomputing preconditioners is expensive. We start with a typical example given by the so-called ILU(0) incomplete decomposition which has the same sparsity pattern as the matrix it preconditions. This has the obvious advantage that it enables straightforward a priori allocation, but its computation may be time-consuming. In Table 1 we display the total time to solve the whole sequence and the numbers of BiCGSTAB iterations needed to solve the individual linear systems for several preconditioning strategies. In the first, denoted by "Recomp," the ILU(0) preconditioner was computed for each

TABLE 1
Nonlinear convection-diffusion model problem with $R = 50$, $n = 4900$, $nnz = 24220$, $ILU(0)$.

ILU(0), psize $\approx$ 24000					
Matrix	Recomp	Freeze	Str.	Unstr. GJ	Unstr. Kr.
$A^{(0)}$	40	40	40	40	40
$A^{(1)}$	29	36	32	39	30
$A^{(2)}$	21	39	27	34	30
$A^{(3)}$	20	48	26	33	24
$A^{(4)}$	17	55	26	31	26
$A^{(5)}$	16	58	29	29	30
$A^{(6)}$	15	50	22	24	26
$A^{(7)}$	15	62	26	28	29
$A^{(8)}$	17	68	28	30	31
$A^{(9)}$	15	71	27	28	28
$A^{(10)}$	15	51	24	29	28
Overall time	11 s	7.5 s	5 s	8.5 s	12.5 s

matrix separately. The strategy "Freeze" used a fixed preconditioner. The strategy denoted by "Str" used structured updates, "Unstr. GJ" stands for unstructured updates based on Gauss–Jordan transformations obtained from Algorithm 3.1, and "Unstr. Kr." stands for those obtained from Algorithm 3.2, where the spanning tree is computed with the Kruskal algorithm. We see that the recomputed ILU(0) decompositions yield powerful preconditioners for our problem, but they are rather slowly computed in MATLAB. Freezing the initial ILU(0) decomposition avoids these slow computations, and although it yields much higher numbers of BiCGSTAB iterations, the overall time to solve the sequence is shorter. Excellent behavior of the structured updates is demonstrated by this table. Here the triangular parts were chosen adaptively based on the magnitudes of their entries. While iteration numbers are nearly as low as with recomputation, significant time savings are achieved by avoiding the recomputation of preconditioners. The iteration counts for unstructured updates from Algorithm 3.1 are a little higher than for structured updates, but they are clearly lower than with the frozen preconditioner. Unstructured updates from Algorithm 3.2 yield iteration numbers comparable to those of structured updates.

Of course, running Algorithm 3.1 or 3.2 to compute the unstructured updates adds a time penalty. However, the timings displayed in Table 1 are pessimistic because they include solving with nontriangular factors of the form (3.5), which cannot compete with the highly optimized implementation of backward and forward solves in MATLAB. The complexity of Algorithm 3.1 or 3.2 alone is not very high for sparse matrices since it is linear in the number of matrix nonzeros. In this context, note that using a drop tolerance in Algorithms 3.1 and 3.2 has an influence on the number of nonzeros and hence also on computational time. We computed the unstructured updates with $tol = 0.3$ in Algorithms 3.1 and 3.2. In practice this parameter should be chosen according to the following considerations for the individual algorithms.

In Algorithm 3.2 we first construct a maximum spanning forest of at most $2n - 1$ entries. Hence we need a value of tol selecting the $2n - 1$ largest entries and as few other entries as necessary to be able to build the spanning forest. We could have optimized the choice of tol according to this rationale, leading to $tol = 0.35$ and an overall time of 10.5 seconds. For Algorithm 3.1 the situation is quite different. Here, an interesting fact is that if we significantly overestimate the parameter, then the

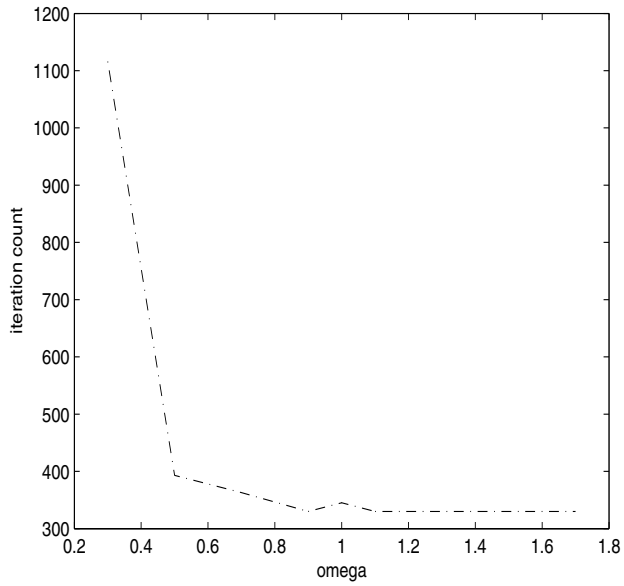


FIG. 1. *Nonlinear convection-diffusion model problem: Iteration counts for Unstr. GJ in dependency of* ω.

unstructured update may be very sparse since a smaller number of nonzeros can be covered by Gauss–Jordan transforms. If we underestimate it, then the update may be very sparse as well since we get only a small number of factors in the unstructured update of the form (3.1). In our case we did not optimize its choice, but a value $tol = 0.1-0.4$ for a reasonably scaled system matrix in order to keep only a few, say up to k, nonzeros in a row, and thus to cover by the unstructured update approximately $k \cdot n$ off-diagonal entries, is fine. This type of behavior is different from what we can sometimes observe in the field of algebraic preconditioners. As for the choice of ω in Algorithm 3.1, its value does not seem to have a crucial influence on the performance of the update either. In Figure 1 we display the total number of BiCGSTAB iterations needed to solve the whole sequence for different values of ω. If the values of ω are smaller than 0.5, criterion $(7')$ of Algorithm 3.1 starts to overemphasize the weight of the chosen candidate row, resulting in bad approximations of $DU - B$. In the other experiments presented here, we always used the choice $\omega = 1$.

In Table 2 the accuracies $\|A^{(i)} - M^+\|$ (in the Frobenius norm) of the preconditioners M^+ for the individual strategies are displayed. For this sequence, where stability of the preconditioners is not an issue, the accuracies correspond nicely to the numbers of BiCGSTAB iterations. We also present some information about the quality of the approximations $\overline{DU - B}$. Table 3 contains the values of the approximations $\|DU - B - \overline{DU - B}\|$ in the Frobenius norm for the considered update techniques. Also these values correspond to the numbers of BiCGSTAB iterations.

In Table 4 we take a closer look at the various update techniques we introduced. Whereas Table 1 suggests that structured updates provide more efficient preconditioners than unstructured updates, this is not apparent from Table 4. Here we use as initial preconditioner the ILU implemented in MATLAB with drop tolerance 0.01. The tol-

TABLE 2
Nonlinear convection-diffusion model problem, accuracies $\|A^{(i)} - M^{+}\|$.

ILU(0), psize ≈ 24000					
Matrix	Recomp	Freeze	Str.	Unstr. GJ	Unstr. Kr.
$A^{(0)}$	28.5	28.5	28.5	28.5	28.5
$A^{(1)}$	27.8	34.6	29.2	50.2	37.3
$A^{(2)}$	26.8	42.3	41.7	51.0	42.1
$A^{(3)}$	25.5	51.0	48.5	55.8	48.9
$A^{(4)}$	24.1	60.4	55.8	64.0	56.5
$A^{(5)}$	23.6	63.5	58.3	63.9	59.1
$A^{(6)}$	23.1	66.6	60.6	64.9	61.6
$A^{(7)}$	23.1	66.6	60.6	64.9	61.5
$A^{(8)}$	23.1	66.5	60.6	64.9	61.5
$A^{(9)}$	23.1	66.5	60.6	64.9	61.5
$A^{(10)}$	23.1	66.5	60.6	64.9	61.5

TABLE 3
Nonlinear convection-diffusion model problem, approximation qualities $\|DU - B - \overline{DU - B}\|$.

ILU(0), psize ≈ 632000			
Matrix	Str.	Unstr. GJ	Unstr. Kr.
$A^{(1)}$	13.89	37.01	18.19
$A^{(2)}$	22.1	36	22.7
$A^{(3)}$	29.78	40.1	30.34
$A^{(4)}$	37.46	48.32	38.47
$A^{(5)}$	39.92	47.39	41.2
$A^{(6)}$	42.29	47.91	43.69
$A^{(7)}$	42.27	47.9	43.66
$A^{(8)}$	42.23	47.86	43.62
$A^{(9)}$	42.23	47.86	43.63
$A^{(10)}$	42.23	47.86	43.63

TABLE 4
Nonlinear convection-diffusion model problem with $R = 50$, $n = 4900$, $nnz = 24220$, $ILU(0.01)$.

ILU(10^{-2}), psize ≈ 52000				
Matrix	Freeze	Unstr. GJ	Struct	Unstr. Kr.
$A^{(0)}$	17	17	17	17
$A^{(1)}$	34	57	21	48
$A^{(2)}$	49	43	24	36
$A^{(3)}$	77	39	34	33
$A^{(4)}$	102	36	54	29
$A^{(5)}$	140	37	69	28
$A^{(6)}$	142	30	76	25
$A^{(7)}$	154	35	77	28
$A^{(8)}$	144	36	91	33
$A^{(9)}$	152	35	91	29
$A^{(10)}$	123	31	90	28
Overall time	14.5 s	7.5 s	9 s	9 s

erance in Algorithms 3.1 and 3.2 for unstructured updates is 0.3. Clearly, unstructured updates are more powerful than structured updates with this kind of initial factorization. This is caused by the fact that the approximations $\overline{DU - B}$ in (2.6) cover more large entries when we use unstructured updates. In the following we quantify this property for a difference matrix B from the middle of the sequence, $B = A^{(0)} - A^{(4)}$. For other difference matrices from the sequence we would obtain similar numbers. With $B = A^{(0)} - A^{(4)}$, nonzero entries in $DU - B$ are quite evenly distributed over both triangular parts. We have $\|striu(DU - B)\| \approx 80$ and $\|stril(DU - B)\| \approx 38$ in the Frobenius norm. Here $stril(\cdot)$ and $striu(\cdot)$ denote the strict lower and upper triangular matrix part, respectively. Hence the upper triangular part is dominating, but important entries may be found in the lower part, too, and they are lost with structured updates. The unstructured updates take into account both triangular parts. This is reflected by the Frobenius norms $\|striu(\overline{DU - B})\| \approx 70$ and $\|stril(\overline{DU - B})\| \approx 16$ for the approximation $\overline{DU - B}$ from Algorithm 3.1. With Algorithm 3.2 we obtain $\|striu(\overline{DU - B})\| \approx 58$ and $\|stril(\overline{DU - B})\| \approx 32$. Note that Algorithm 3.2 yields more nonzeros, which is explained by step (3) of the algorithm. In this context, also note that the number of nonzeros of the initial factorization and structured updates is about 52000, whereas unstructured updates have smaller numbers of nonzeros, about 39000–46000, which makes application of the unstructured updated preconditioner less expensive. This is one of the reasons why the unstructured updates are competitive, even with respect to timing, with structured ones, in spite of the time penalty to run Algorithm 3.1 or 3.2. The other reason is, of course, lower BiCGSTAB iteration numbers.

In situations as in Table 1, recomputing preconditioners is outperformed by our updates because of the high expenses of recomputing. When, on the other hand, recomputation is straightforward, updates need not be more effective. An example is given in Table 5 with as initial preconditioner the dual-threshold ILUT$(0.1, 5)$ decomposition, implemented in Fortran 90. The number of nonzeros in the incomplete LU decomposition is about 38000 (slightly differing for different matrices). Here the time spent for recomputation is very small due to the simple discretization stencil, and by far the most time is spent while solving with BiCGSTAB. Still, concerning iteration counts, the (adaptively chosen) structured updates perform only slightly worse than recomputation. Note that there is a strong overlap between the location of the nonzeros in B and in the preconditioner, but as above, we did not merge the triangular parts of the updated preconditioner. Table 6 shows similar behavior for a much larger problem with ILUT$(0.1, 3)$ as initial decomposition. Here we discretized (4.1) on a 282×282 grid, the matrices having dimension 79524. While evaluating Tables 5 and 6, it is important to realize that the timings may provide here only partial information. In case of matrix-free implementation we typically need to estimate the matrices first using, for example, graph coloring techniques [18], [49]. Our matrices have five diagonals and this implies that they can be estimated by at most seven matvecs. Namely, the number of matvecs corresponds to the number of colors needed to color the undirected graph of $A^T A$, the so-called intersection graph. Computing some of the standard preconditioners both [illegible] based on matvecs is a state-of-the-art challenging problem and can be very time-consuming. When using updates in a matrix-free environment, only part of the difference matrix needs to be estimated. In our cases the needed part of the difference matrix was always available from at most three matvecs, because the intersection graph of the (possibly permuted) triangular part of the matrix could be colored by only three colors.

TABLE 5
Nonlinear convection-diffusion model problem with $R = 50$, $n = 4900$, $nnz = 24220$, $ILUT(0.1, 5)$.

ILUT$(0.1, 5)$, timep ≈ 0.01, psize ≈ 38000			
Matrix	Recomp	Freeze	Struct. update
$A^{(0)}$	25	25	25
$A^{(1)}$	25	33	26
$A^{(2)}$	23	47	27
$A^{(3)}$	19	58	27
$A^{(4)}$	18	83	27
$A^{(5)}$	17	88	28
$A^{(6)}$	16	119	28
$A^{(7)}$	16	114	27
$A^{(8)}$	17	107	27
$A^{(9)}$	17	111	28
$A^{(10)}$	17	123	27
Overall time	0.20 s	0.78 s	0.25 s

TABLE 6
Nonlinear convection-diffusion model problem with $R = 50$, $n = 79524$, $nnz = 615997$, $ILUT(0.1, 3)$.

ILUT$(0.1, 3)$, timep ≈ 0.05, psize ≈ 632000			
Matrix	Recomp	Freeze	Struct. update
$A^{(0)}$	82	82	82
$A^{(1)}$	86	85	82
$A^{(2)}$	73	97	82
$A^{(3)}$	72	91	76
$A^{(4)}$	66	97	73
$A^{(5)}$	68	113	77
$A^{(6)}$	71	140	75
$A^{(7)}$	68	139	70
$A^{(8)}$	70	137	76
$A^{(9)}$	69	136	83
$A^{(10)}$	65	217	72
Overall time	17.4 s	31.0 s	19.4 s

In addition to the experiments presented here we also performed some experiments where the nonlinear problems were discretized by upwind schemes, leading to triangular difference matrices. As one can guess from the pattern, the results for solving the linear problems were rather good, but we typically needed more nonlinear iterations. Consequently, discretization by central differences was preferable.

Our second test problem is a smaller but rather difficult problem of dimension 2500. It consists of the two-dimensional driven cavity problem of the form

$$\Delta\Delta u + R\left(\frac{\partial u}{\partial y}\frac{\partial \Delta u}{\partial x} - \frac{\partial u}{\partial x}\frac{\partial \Delta u}{\partial y}\right) = 0$$

on the unit square, discretized by 13-point finite differences on a shifted uniform grid with 50×50 inner nodes [30]. The boundary conditions are given by $u = 0$ on $\partial\Omega$ and $\partial u(0, y)/\partial x = 0$, $\partial u(1, y)/\partial x = 0$, $\partial u(x, 0)/\partial x = 0$, and $\partial u(x, 1)/\partial x = 1$. The initial approximation is the discretization of $u_0(x, y) = 0$.

For the same reason as before, we choose modest Reynolds numbers. Even with

TABLE 7
Driven cavity problem with $R = 50$, $n = 2500$, $nnz = 31504$, $ILU(0.01)$.

ILU(0.01), psize ≈ 47000					
Matrix	Recomp	Freeze	Str.	Unstr. GJ	Unstr. Kr.
$A^{(0)}$	93	93	93	93	93
$A^{(1)}$	269	93	88	337	81
$A^{(2)}$	> 500	> 500	156	324	58
$A^{(3)}$	> 500	164	179	265	60
$A^{(4)}$	> 500	288	298	206	74
$A^{(5)}$	> 500	> 500	144	184	71
$A^{(6)}$	> 500	> 500	132	190	70
Overall time	∞	∞	8 s	17 s	6.5 s

TABLE 8
Driven cavity problem with $R = 10$, $n = 2500$, $nnz = 31504$, $ILU(0.01)$.

ILU(0.01), psize ≈ 47000					
Matrix	Recomp	Freeze	Str.	Unstr. GJ	Unstr. Kr.
$A^{(0)}$	84	84	84	84	84
$A^{(1)}$	84	87	95	91	91
$A^{(2)}$	312	183	119	95	113
$A^{(3)}$	261	198	119	103	134
$A^{(4)}$	352	> 500	190	149	164
$A^{(5)}$	259	> 500	163	204	164
$A^{(6)}$	291	183	150	217	144
Overall time	12.5 s	∞	7 s	12 s	11 s

modest Reynolds numbers we obtain sequences of linear systems that are hard to solve for the BiCGSTAB accelerator. As system matrices have 31504 nonzeros, we needed a relatively dense initial ILU preconditioner with 47000 nonzeros and with drop tolerance 0.01 from MATLAB to be able to solve the linear systems at all. Sparser preconditioners caused BiCGSTAB to stagnate for the initial linear system. In Tables 7 and 8 we show experiments executed in MATLAB with the initial ILU(0.01) preconditioner for $R = 50$ and $R = 10$, respectively. As before, by "overall time" we mean the time needed to solve the whole sequence, including preconditioner computations. In the columns "Unstr." we display the performance of unstructured updates computed with Algorithm 3.1 ($tol = 0.05$ for $R = 50$ and $tol = 0.02$ for $R = 10$) and Algorithm 3.2 ($tol = 0.7$ for $R = 50$ and $tol = 0.02$ for $R = 10$).

This problem represents the case where recomputing should be avoided for stability reasons. For instance, with $R = 50$, the recomputation of the incomplete factorization failed for the last 5 linear systems (giving the MATLAB warning "Incomplete upper triangular factor had 1 zero diagonal replaced by local drop tolerance"). In order to quantify instability we computed estimates of the 2-norms of the inverses of the factors of the used factorizations. For the initial decomposition we have $\|U^{-1}\|_2 \approx 41$ and $\|(LD)^{-1}\|_2 \approx 264$, but these norms grow rapidly for subsequent recomputed factorizations. In the second column of Table 9 the norms for $(LD)^{-1}$ are displayed; norms for U^{-1} grow similarly. Clearly, forward and backward substitution have become unstable. In the columns corresponding to updated factorizations we estimated $\|(\overline{LD - B})^{-1}\|_2$. We see that higher estimates correspond in the majority of cases to higher iteration numbers. In the frozen preconditioner strategy, however, instability

TABLE 9
Driven cavity problem with $R = 50$, estimated Euclidean norms of inverses of first factor.

ILU(0.01), psize ≈ 47000					
Matrix	Recomp	Freeze	Str.	Unstr. GJ	Unstr. Kr.
$A^{(0)}$	264	264	264	264	264
$A^{(1)}$	$2 \cdot 10^3$	264	203	1069	185
$A^{(2)}$	$9 \cdot 10^5$	264	227	99	101
$A^{(3)}$	$8 \cdot 10^4$	264	326	291	130
$A^{(4)}$	$3 \cdot 10^5$	264	327	290	131
$A^{(5)}$	$2 \cdot 10^5$	264	327	290	131
$A^{(6)}$	$4 \cdot 10^5$	264	327	290	131

is not the cause of stagnation. We guess the frozen preconditioner fails to provide the structural information contained in updated factorizations. The results for $R = 10$ reflect similar phenomena in a weaker form. Structured and unstructured updates from Algorithm 3.2 yield the best results. In the case $R = 50$ the optimal choice $tol = 0.7$ results in particularly good performance of Algorithm 3.2, with respect to both time and iteration count.

We conclude this section with an application which leads to very large sequences of linear systems. They arise from numerical computation of steady vertical air flow through a level tunnel at a low Mach number subject to the gravitational force. The domain is a two-dimensional longitudinal section of the tunnel with the pressure and density varying only in the horizontal direction such that the gravitational term is balanced out by the pressure gradient. Neumann boundary conditions and Lax–Friedrichs fluxes were used. The gravitation term and the Euler equations were separated by a first-order operator splitting. For the discretization, the implicit Euler method combined with the first-order finite volume discretization in space was used. In every time step, one Newton step is performed in the flow solver only. More details can be found in [10], in particular in section 6.2. Our results were very similar for more variations of the problem.

Table 10 contains the results for two sequences from the linear systems for the described problem with a relatively coarse discretization grid. We used the dual-threshold ILUT$(0.001, 5)$ preconditioner, where the parameters were chosen in order to have a preconditioner size (that is, number of nonzeros) close to the size of the original matrix and such that the total number of matvecs (two in each iteration) to solve the initial system is reasonably small.

Here we show results only for some linear systems from the beginning of the sequences (as given by the superscripts); the whole sequence has more than 1000 linear systems. Three preconditioning strategies were tested: recomputation, freezing and updating. Updates were always related to the first matrix of the sequence. In the first sequence of Table 10, the preconditioner that is being frozen or updated was computed for the matrix $A^{(0)}$, and in the second sequence it was taken from the 30th linear system. The update strategy was implemented as a black-box routine which decides which of the updates (unstructured update from Algorithm 3.1 or 3.2, structured update based on the upper triangular part of the difference matrix, or structured update based on the lower triangular part of the difference matrix) is used, based on the sum of magnitudes of strong matrix entries. The structured updates store the update separately, although merging with the decomposition could provide even better timings. The results are characterized by the number of iterations

TABLE 10
Air flow in a tunnel, $n = 4800$*,* $nnz = 138024$*,* $ILUT(0.001, 5)$.

ILUT(0.001, 5), timep ≈ 0.05, psize ≈ 135798						
Matrix	Recomp		Freeze		Update	
	Its	Time	Its	Time	Its	Time
$A^{(5)}$	29	0.57	19	0.33	19	0.34
$A^{(10)}$	30	0.55	17	0.27	17	0.27
$A^{(15)}$	33	0.64	21	0.39	19	0.34
$A^{(20)}$	32	0.64	19	0.34	17	0.31
$A^{(25)}$	33	0.56	20	0.33	19	0.33
$A^{(30)}$	34	0.66	24	0.44	21	0.34
$A^{(35)}$	33	0.66	23	0.42	19	0.36
$A^{(40)}$	39	0.72	31	0.52	24	0.39
$A^{(45)}$	44	0.78	33	0.55	27	0.45
$A^{(50)}$	40	0.75	39	0.63	24	0.44
$A^{(55)}$	40	0.74	47	0.78	25	0.42
$A^{(60)}$	47	0.85	80	1.41	31	0.56
$A^{(65)}$	47	0.80	107	1.64	27	0.42
$A^{(70)}$	38	0.75	72	1.28	28	0.51
$A^{(75)}$	114	2.03	230	4.06	105	1.96
$A^{(80)}$	63	1.19	87	1.51	80	1.42
$A^{(35)}$	33	0.66	36	0.63	35	0.67
$A^{(40)}$	39	0.72	37	0.64	35	0.59
$A^{(45)}$	44	0.78	42	0.67	35	0.59
$A^{(50)}$	40	0.75	43	0.67	29	0.45
$A^{(55)}$	40	0.74	57	0.95	31	0.53
$A^{(60)}$	47	0.85	84	1.37	33	0.54
$A^{(65)}$	47	0.80	102	1.55	34	0.52
$A^{(70)}$	38	0.75	87	1.47	34	0.58
$A^{(75)}$	114	2.03	163	2.65	147	2.45
$A^{(80)}$	63	1.19	81	1.38	93	1.64

of the BiCGSTAB method and by the timings of the preconditioned iterative method required to solve the individual linear systems, including the time required to compute the preconditioner. The average time to compute the preconditioner is denoted by *timep*, and its average number of nonzeros is denoted by *psize*. These last two characteristics differ slightly in individual computations of a sequence of problems. Note that preconditioning this problem was necessary; the unpreconditioned method worked rather poorly.

From Table 10 we can see once more that freezing the preconditioner may not be enough for getting efficiently preconditioned iterative methods for all the systems. Freezing with updating is typically better in terms of the number of matvecs. The additional solve with the update may add a time penalty, but its influence seems to be limited. Clearly, by changing the matrix more and more the gap between the efficiency of freezing and updating gets larger up to some point where, of course, also the update is not sufficient anymore. We included this point in our table, but in practice this would be the moment to recompute a factorization. As in the previous problem, it seems that the update is even more powerful than the recomputed preconditioners in the sense of giving the smallest number of iterations among all three preconditioning strategies. This must be mainly caused by the fact that recomputation becomes less

stable as the sequence proceeds, as can be seen from the iteration numbers around the 75th linear system. However, the role of additional structural information provided by updates should not be underestimated. In Table 12 we will consider a sequence without instability regions where updates are still more powerful than recomputed factorizations.

TABLE 11
Air flow in a tunnel, $n = 9600$*,* $nnz = 277224$*,* $ILUT(10^{-7}, 30)$.

ILUT($10^{-7}, 30$), timep ≈ 0.1, psize ≈ 283751						
Matrix	Recomp		Freeze		Update	
	Its	Time	Its	Time	Its	Time
$A^{(0)}$	3	0.13	3	0.13	3	0.13
$A^{(5)}$	3	0.13	3	0.03	3	0.03
$A^{(10)}$	4	0.15	4	0.05	5	0.05
$A^{(15)}$	4	0.15	5	0.06	6	0.06
$A^{(20)}$	5	0.15	6	0.06	7	0.09
$A^{(30)}$	7	0.18	7	0.08	8	0.11
$A^{(40)}$	8	0.23	14	0.16	14	0.17
$A^{(45)}$	9	0.23	18	0.17	20	0.23
$A^{(46)}$	11	0.24	22	0.23	16	0.18
$A^{(47)}$	11	0.23	18	0.19	16	0.18
$A^{(48)}$	15	0.29	23	0.25	22	0.26
$A^{(49)}$	15	0.30	23	0.25	22	0.29
$A^{(50)}$	16	0.33	24	0.23	19	0.23
$A^{(51)}$	27	0.48	31	0.38	25	0.33
$A^{(52)}$	47	0.69	33	0.34	27	0.31
$A^{(53)}$	44	0.73	33	0.39	23	0.29
$A^{(54)}$	67	1.12	54	0.61	32	0.43
$A^{(55)}$	92	1.49	196	2.23	56	0.84
$A^{(56)}$	76	1.21	131	1.48	40	0.54
$A^{(57)}$	79	1.33	81	1.05	51	0.80
$A^{(58)}$	52	0.91	45	0.59	34	0.51
$A^{(59)}$	50	1.02	40	0.63	38	0.65
$A^{(60)}$	32	0.74	961	15.3	440	7.98

Table 11 presents qualitatively the same results for a larger matrix. As above, a powerful ILUT preconditioner was chosen in order to provide small iteration counts and to have the number of nonzeros of the preconditioner similar to the number of nonzeros of the original matrix. Note that for most of the more difficult problems, the time needed to solve the linear system is the best for our updates. While, as above, there is a similar behavior of the iteration counts we also show results for more matrices around the point where the original frozen preconditioner stops being useful. Note that for some matrices the updated preconditioner behaves . . . better than the other strategies.

In Table 12 we consider discretization leading to matrices of a dimension about 60000. Most of the remarks on the previous two tables can be made here too, though we note that there are no instability regions anymore. As before, updates achieve an acceleration compared to recomputing of up to 90%. The relation to the freezing strategy is the same as for the corresponding problems of smaller dimension. A noteworthy difference with smaller dimensions is that the ratio of the average time to recompute the preconditioner ("timep") the time to solve the systems is much larger.

Hence avoiding recomputation becomes more important with larger dimensions. To conclude, let us mention the problem of recomputing related to a different preconditioner. This large air flow problem with the standard AINV(0.1) preconditioner with a number of nonzeros close to the number of nonzeros in the first matrix of the sequence converges in 12 iterations on average, the time to compute the preconditioner is 1.67 s, and time for the BiCGSTAB iterations is 0.25 s! We may assume that the role of avoiding frequent recomputations will be significantly increased in this case, but we did not follow this line.

TABLE 12
Air flow in a tunnel, $n = 59392$*,* $nnz = 1127211$*,* $ILUT(10^{-8}, 8)$.

ILUT($10^{-8}, 8$), timep ≈ 0.45, psize $\approx$ 1307000–1490000						
Matrix	Recomp		Freeze		Update	
	Its	Time	Its	Time	Its	Time
$A^{(0)}$	24	1.25	24	1.25	24	1.25
$A^{(2)}$	21	1.13	27	0.95	23	0.88
$A^{(4)}$	22	1.15	27	0.90	23	0.89
$A^{(6)}$	21	1.15	27	0.90	23	0.90
$A^{(8)}$	21	1.14	26	0.93	23	0.89
$A^{(10)}$	22	1.15	26	0.91	23	0.91
$A^{(12)}$	24	1.23	27	0.97	23	0.88
$A^{(14)}$	23	1.20	27	1.01	23	0.90
$A^{(16)}$	24	1.23	27	0.95	22	0.89
$A^{(18)}$	24	1.27	27	0.92	22	0.89
$A^{(20)}$	25	1.23	28	0.90	21	0.83
$A^{(22)}$	25	1.24	28	0.92	22	0.86
$A^{(24)}$	26	1.29	28	0.98	22	0.84
$A^{(26)}$	29	1.60	28	1.00	22	0.85
$A^{(28)}$	30	1.43	29	0.95	22	0.84
$A^{(30)}$	28	1.37	28	0.97	23	0.89
$A^{(32)}$	31	1.53	33	1.06	22	0.81
$A^{(34)}$	28	1.42	28	0.95	23	0.89
$A^{(36)}$	31	1.51	30	1.02	22	0.91
$A^{(38)}$	30	1.51	29	1.01	23	0.95

5. Conclusions. In this paper we proposed a couple of algebraic procedures which may be useful for solving sequences of systems of linear equations. The numerical experiments show that our updated preconditioners can be rather successful in practice, and the updates can often replace recomputation of preconditioners. In many cases, one would like to make the overall number of operations smaller with simple updates, and our experiments confirm that this is possible. In particular, the preconditioner update seems to be more advantageous than the other approaches if one of the following cases applies: if preconditioner computation is not cheap, if its recomputation is unstable, or if the update is structurally dominant, that is, if it covers a significant part of the difference matrices from subsequent problems. Nevertheless, there can be also different, and sometimes very strong, reasons for avoiding preconditioner recomputations. In matrix-free and/or parallel environments, which are currently quite common, any recomputation of a preconditioner may be expensive. This is especially true for strong algebraic preconditioners which are used for solving difficult problems. We intentionally used structured updates based on only one trian-

gular part. Part of our motivation was that we concentrated on finding methods for problems where the nonsymmetry is apparent. In addition, we are interested in the structured update since we expect possible cheap estimation of sparsified triangular matrices. This may be important in a matrix-free environment. Note that our unstructured updates are very close to permuted (and sparsified) triangular updates. We intend to present fully matrix-free results in the near future. Another issue which we are currently investigating is combination of approximate factorizations with various Gauss–Seidel type preconditioners to define updates.

An interesting problem which we would like to consider in the future is to find first a nonsymmetric permutation that transforms the system matrices into a form more suitable for one particular structured or unstructured update. In particular, this permutation may make one triangular part of the matrices heavier (in the sense of the sum of magnitudes of its entries) than the other triangular part. This may have a connection to the combinatorial method in Algorithm 3.2 for finding an unstructured update. The use of a weighted spanning tree strongly brings to mind the popular strategy of matchings-based nonsymmetric permutations which has significantly improved algebraic preconditioning in recent years [23], [5].

Acknowledgments. The authors thank Philipp Birken and Ladislav Lukšan for providing the software for solving the nonlinear problems and for useful instructions on working with it. They thank Andreas Meister for initiating application of the proposed techniques to the tunnel problem.

REFERENCES

[1] M. A. Ajiz and A. Jennings, *A robust incomplete Choleski-conjugate gradient algorithm*, Internat. J. Numer. Methods Engrg., 20 (1984), pp. 949–966.

[2] J. Baglama, D. Calvetti, G. H. Golub, and L. Reichel, *Adaptively preconditioned GMRES algorithms*, SIAM J. Sci. Comput., 20 (1998), pp. 243–269.

[3] M. Benzi and D. Bertaccini, *Approximate inverse preconditioning for shifted linear systems*, BIT, 43 (2003), pp. 231–244.

[4] M. Benzi, J. K. Cullum, and M. Tůma, *Robust approximate inverse preconditioning for the conjugate gradient method*, SIAM J. Sci. Comput., 22 (2000), pp. 1318–1332.

[5] M. Benzi, J. C. Haws, and M. Tůma, *Preconditioning highly indefinite and nonsymmetric matrices*, SIAM J. Sci. Comput., 22 (2000), pp. 1333–1353.

[6] M. Benzi, C. D. Meyer, and M. Tůma, *A sparse approximate inverse preconditioner for the conjugate gradient method*, SIAM J. Sci. Comput., 17 (1996), pp. 1135–1149.

[7] M. Benzi and M. Tůma, *Orderings for factorized sparse approximate inverse preconditioners,* SIAM J. Sci. Comput., 21 (2000), pp. 1851–1868.

[8] L. Bergamaschi, R. Bru, A. Martínez, and M. Putti, *Quasi-Newton preconditioners for the inexact Newton method*, Electron. Trans. Numer. Anal., 23 (2006), pp. 76–87.

[9] D. Bertaccini, *Efficient preconditioning for sequences of parametric complex symmetric linear systems*, Electron. Trans. Numer. Anal., 18 (2004), pp. 49–64.

[10] P. Birken, *Numerical Simulation of Flows at Low Mach Numbers with Heat Sources*, Ph.D. thesis, University of Kassel, Kassel, Germany, 2005.

[11] R. P. Brent, *Some efficient algorithms for solving systems of nonlinear equations*, SIAM J. Numer. Anal., 10 (1973), pp. 327–344.

[12] P. N. Brown and Y. Saad, *Hybrid Krylov methods for nonlinear systems of equations*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 450–481.

[13] X.-C. Cai and D. E. Keyes, *Nonlinearly preconditioned inexact Newton algorithms*, SIAM J. Sci. Comput., 24 (2002), pp. 183–200.

[14] K. Chen, *Matrix Preconditioning Techniques and Applications*, Cambridge Monogr. Appl. Comput. Math. 19, Cambridge University Press, Cambridge, UK, 2005.

[15] E. Chow and Y. Saad, *Experimental study of ILU preconditioners for indefinite matrices*, J. Comput. Appl. Math., 86 (1997), pp. 387–414.

[16] P. Concus and G. H. Golub, *Use of fast direct methods for the efficient numerical solution of nonseparable elliptic equations*, SIAM J. Numer. Anal., 10 (1973), pp. 1103–1120.

[17] P. Concus and G. H. Golub, *A generalized conjugate gradient method for nonsymmetric systems of linear equations*, in Computing Methods in Applied Sciences and Engineering, Lecture Notes in Econom. Math. Systems 134, R. Glowinski and J. L. Lions, eds., Springer-Verlag, Berlin, New York, 1976, pp. 56–65.

[18] A. R. Curtis, M. J. D. Powell, and J. K. Reid, *On the estimation of sparse Jacobian matrices*, J. Inst. Math. Appl., 13 (1974), pp. 117–119.

[19] T. A. Davis and W. W. Hager, *Modifying a sparse Cholesky factorization*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 606–627.

[20] T. A. Davis and W. W. Hager, *Multiple-rank modifications of a sparse Cholesky factorization*, SIAM J. Matrix Anal. Appl., 22 (2001), pp. 997–1013.

[21] T. A. Davis and W. W. Hager, *Row modifications of a sparse Cholesky factorization*, SIAM J. Matrix Anal. Appl., 26 (2005), pp. 621–639.

[22] I. S. Duff and J. Koster, *The design and use of algorithms for permuting large entries to the diagonal of sparse matrices*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 889–901.

[23] I. S. Duff and J. Koster, *On algorithms for permuting large entries to the diagonal of a sparse matrix*, SIAM J. Matrix Anal. Appl., 22 (2001), pp. 973–996.

[24] H. C. Elman and M. H. Schultz, *Preconditioning by fast direct methods for nonself-adjoint nonseparable elliptic equations*, SIAM J. Numer. Anal., 23 (1986), pp. 44–57.

[25] P. F. Fischer, *Projection techniques for iterative solution of $Ax = b$ with successive right-hand sides*, Comput. Methods Appl. Mech. Engrg., 163 (1998), pp. 193–204.

[26] R. W. Freund, *A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems*, SIAM J. Sci. Comput., 14 (1993), pp. 470–482.

[27] P. E. Gill, W. Murray, and M. A. Saunders, *Methods for computing and modifying the LDV factors of a matrix*, Math. Comp., 29 (1975), pp. 1051–1077.

[28] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, London, 1996.

[29] D. Hysom and A. Pothen, *A scalable parallel algorithm for incomplete factor preconditioning*, SIAM J. Sci. Comput., 22 (2001), pp. 2194–2215.

[30] I. E. Kaporin and O. Axelsson, *On a class of nonlinear equation solvers based on the residual norm reduction over a sequence of affine subspaces*, SIAM J. Sci. Comput., 16 (1995), pp. 228–249.

[31] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, Frontiers Appl. Math. 16, SIAM, Philadelphia, 1995.

[32] C. T. Kelley, *Solving Nonlinear Equations with Newton's Method*, Fundam. Algorithms 1, SIAM, Philadelphia, 2003.

[33] D. S. Kershaw, *The incomplete Cholesky-conjugate gradient method for the iterative solution of systems of linear equations*, J. Comput. Phys., 26 (1978), pp. 43–65.

[34] D. Keyes, *Terascale implicit methods for partial differential equations*, in Contemp. Math. 306, AMS, Providence, RI, 2001, pp. 29–84.

[35] S. A. Kharchenko, L. Yu. Kolotilina, A. A. Nikishin, and A. Yu. Yeremin, *A reliable AINV-type preconditioning method for constructing sparse approximate inverse preconditioners in factored form*, Numer. Linear Algebra Appl., 8 (2001), pp. 165–179.

[36] D. A. Knoll and D. E. Keyes, *Jacobian-free Newton-Krylov methods: A survey of approaches and applications*, J. Comput. Phys., 193 (2004), pp. 357–397.

[37] D. A. Knoll and P. R. McHugh, *Newton-Krylov methods applied to a system of convection-reaction-diffusion equations*, Comput. Phys. Comm., 88 (1995), pp. 141–160.

[38] D. A. Knoll, P. R. McHugh, and D. E. Keyes, *Newton-Krylov methods for low Mach number compressible combustion*, AIAA J., 34 (1996), pp. 961–967.

[39] J. B. Kruskal, *On the shortest spanning subtree of a graph and the traveling salesman problem*, Proc. Amer. Math. Soc., 2 (1956), pp. 48–50.

[40] I. Lee, P. Raghavan, and E. G. Ng, *Effective preconditioning through ordering interleaved with incomplete factorization*, SIAM J. Matrix Anal. Appl., 27 (2006), pp. 1069–1088.

[41] D. Loghin, D. Ruiz, and A. Touhami, *Adaptive preconditioners for nonlinear systems of equations*, J. Comput. Appl. Math., 189 (2006), pp. 326–374.

[42] L. Lukšan, M. Tůma, J. Vlček, N. Ramešová, M. Šiška, J. Hartman, and C. Matonoha, *UFO* 2006. *Interactive System for Universal Functional Optimization*, Technical report V-977, ICS AS CR, Prague, 2006.

[43] T. A. Manteuffel, *An incomplete factorization technique for positive definite linear systems*, Math. Comp., 34 (1980), pp. 473–497.

[44] J. L. Morales and J. Nocedal, *Automatic preconditioning by limited memory quasi-Newton updating*, SIAM J. Optim., 10 (2000), pp. 1079–1096.

[45] D. P. O'Leary, *The block conjugate gradient algorithm and related methods*, Linear Algebra Appl., 29 (1980), pp. 293–322.

[46] M. Olschowka and A. Neumaier, *A new pivoting strategy for Gaussian elimination*, Linear Algebra Appl., 240 (1996), pp. 131–151.
[47] M. L. Parks, E. de Sturler, G. Mackey, D. D. Johnson, and S. Maiti, *Recycling Krylov subspaces for sequences of linear systems*, SIAM J. Sci. Comput., 28 (2006), pp. 1651–1674.
[48] D. W. Peaceman and H. H. Rachford, Jr., *The numerical solution of parabolic and elliptic differential equations*, SIAM J. Appl. Math., 3 (1955), pp. 28–41.
[49] A. Pothen, F. Manne, and A. H. Gebremedhin, *What color is your Jacobian? Graph coloring for computing derivatives*, SIAM Rev., 47 (2005), pp. 629–705.
[50] R. C. Prim, *Shortest connection networks and some generalizations*, Bell System Tech. J., 36 (1957), pp. 1389–1401.
[51] Y. Saad, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM J. Sci. Comput., 14 (1993), pp. 461–469.
[52] Y. Saad, *ILUT: A dual threshold incomplete ILU decomposition*, Numer. Linear Algebra Appl., 1 (1994), pp. 387–402.
[53] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.
[54] V. E. Shamanskii, *On a modification of the Newton method*, Ukrain. Mat. Ž., 19 (1967), pp. 133–138.
[55] V. Simoncini and E. Gallopoulos, *An iterative method for nonsymmeric systems with multiple right-hand sides*, SIAM J. Sci. Comput., 16 (1995), pp. 917–933.
[56] V. Simoncini and D. B. Szyld, *Flexible inner-outer Krylov subspace methods*, SIAM J. Numer. Anal., 40 (2003), pp. 2219–2239.
[57] M. Tismenetsky, *A new preconditioning technique for solving large sparse linear systems*, Linear Algebra Appl., 154/156 (1991), pp. 331–353.
[58] H. A. van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 631–644.
[59] P. Vassilevski, *Preconditioning nonsymmetric and indefinite finite element matrices*, Numer. Linear Algebra Appl., 1 (1992), pp. 59–76.
[60] B. Vital, *Etude de quelques méthodes de résolution de problèmes linéaires de grande taille sur multiprocesseur*, Ph.D. thesis, Université de Rennes I, Rennes, France, 1990.
[61] J. W. Watts III, *A conjugate gradient truncated direct method for the iterative solution of the reservoir simulation pressure equation*, Soc. Petr. Eng. J., 21 (1981), pp. 345–353.
[62] O. Widlund, *A Lanczos method for a class of nonsymmetric systems of linear equations*, SIAM J. Numer. Anal., 15 (1978), pp. 801–812.

SIAM J. SCI. COMPUT.
Vol. 29, No. 5, pp. 1942–1958
© 2007 Society for Industrial and Applied Mathematics

SPECTRAL ANALYSIS OF THE DISCRETE HELMHOLTZ OPERATOR PRECONDITIONED WITH A SHIFTED LAPLACIAN*

M. B. VAN GIJZEN†, Y. A. ERLANGGA‡, AND C. VUIK†

Abstract. Shifted Laplace preconditioners have attracted considerable attention as a technique to speed up convergence of iterative solution methods for the Helmholtz equation. In this paper we present a comprehensive spectral analysis of the Helmholtz operator preconditioned with a shifted Laplacian. Our analysis is valid under general conditions. The propagating medium can be heterogeneous, and the analysis also holds for different types of damping, including a radiation condition for the boundary of the computational domain. By combining the results of the spectral analysis of the preconditioned Helmholtz operator with an upper bound on the GMRES-residual norm, we are able to provide an optimal value for the shift and to explain the mesh-dependency of the convergence of GMRES preconditioned with a shifted Laplacian. We illustrate our results with a seismic test problem.

Key words. Helmholtz equation, shifted Laplace preconditioner, iterative solution methods, GMRES, convergence analysis

AMS subject classifications. 65N55, 65F10, 65N22, 78A45, 76Q05

DOI. 10.1137/060661491

1. Introduction.

In this paper we investigate the spectral behavior of iterative methods applied to the time-harmonic wave equation in heterogeneous media. The underlying equation governs wave propagation and scattering phenomena arising in acoustic problems in many areas, e.g., aeronautics, marine technology, geophysics, and optical problems. In particular, we look for solutions of the Helmholtz equation discretized by using finite difference, finite volume, or finite element discretizations. Since the number of grid points per wavelength should be sufficiently large to result in acceptable solutions, for very high frequencies the discrete problem becomes extremely large, prohibiting the use of direct solution methods. Krylov subspace iterative methods are an interesting alternative. However, Krylov subspace methods are not competitive without a good preconditioner.

Finding a suitable preconditioner for the Helmholtz equation is still an area of active research; see, for example, [7]. A class of preconditioners that has recently attracted considerable attention is the class of shifted Laplace preconditioners. Preconditioning of the Helmholtz equation using the Laplace operator without shift was first suggested in [1]. This approach has been enhanced in [8, 9] by adding a positive shift to the Laplace operator, resulting in a positive definite preconditioner. In [2, 3, 4, 13] the class of shifted Laplace preconditioners is further generalized by also considering general complex shifts.

It is well known that the spectral properties of the preconditioned matrix give important insight in the convergence behavior of the preconditioned Krylov subspace methods. Spectral analyses for the Helmholtz equation preconditioned by a shifted

*Received by the editors May 31, 2006; accepted for publication (in revised form) January 10, 2007; published electronically September 28, 2007. Part of this research was funded by the Dutch BSIK/BRICKS project.

http://www.siam.org/journals/sisc/29-5/66149.html

†Delft Institute of Applied Mathematics, Delft University of Technology, Mekelweg 4, 2628 CD, The Netherlands (M.B.vanGijzen@tudelft.nl, C.Vuik@tudelft.nl).

‡Institut für Mathematik, Technische Universität Berlin, Strasse des 17. Juni 136, D-10623 Berlin, Germany (erlangga@math.tu-berlin.de).

Laplace operator have previously been given in [2, 3, 4]. The analysis in [2], however, is restricted to the homogeneous physical parameters case, for a purely imaginary shift preconditioner. This analysis concerns the singular values of the preconditioned matrix rather than the eigenvalues. Furthermore, only reflecting and pressure release boundary conditions are considered. In [3], a convergence analysis of GMRES is discussed, under the same restriction as in [2]. A more thorough spectral analysis is presented in [4] for the case when the preconditioning operation is performed approximately by using multigrid. The analysis is based on rigorous Fourier analysis (RFA) for homogeneous physical parameters. Results from RFA, however, give little insight into the convergence of Krylov subspace methods.

This paper gives a spectral analysis from an algebraic point of view. Therefore, the results are valid under the following rather general conditions:

- They do not depend on the discretization method;
- inhomogeneous physical parameters (such as sound speed, density, and damping) are allowed;
- the analysis is also valid for various types of boundary conditions (reflecting, radiation, pressure release, and perfectly matched layer (PML) as well).

These generalizations are new, and they allow us to analyze shifted-Laplace preconditioners for a much wider class of discrete Helmholtz problems than in previous publications.

By combining the results of our analysis with a bound on the norm of the GMRES-residual, we are able to derive a "quasi" optimal value for the shift. This result is also new. The shift is derived under the assumption that the preconditioning operations with the shifted-Laplace preconditioner are performed sufficiently accurately. In practice, the preconditioning operations are performed only approximately, for example, by using a multigrid method or by making an ILU decomposition of the shifted-Laplace operator. Of course, our results do not hold unconditionally in these cases. However, the results that are reported in [2, 3, 4], where the preconditioning operations are performed approximately using either a multigrid method or incomplete LU factorization (ILU), are obtained using shifts that are close to the optimal shift that follows from our analysis. This indicates that the optimal shift that we derive in this paper also gives strong guidelines for choosing the shift in the case when the preconditioning operations are performed approximately.

This paper is organized as follows. In section 2 we describe the acoustic wave equation and its discretization. We specify some properties of the matrices (symmetry, complex valued, positive definiteness etc.), which form the coefficient matrix of the resulting linear system. In section 3 we show that for the damped Helmholtz equation with Dirichlet and Neumann boundary conditions, the eigenvalues are located on a line or on a circle with a given parameterization for a special type of damping. For radiation boundary conditions and for general viscous media we show that the eigenvalues are located on one side of the line or within the circle. In section 4 we use a simple bound on the GMRES-residual norm. Using this bound and the results of our spectral analysis we are able to derive the optimal value of the shift in the shifted Laplacian preconditioner. We also show for a number of applications that the convergence of GMRES is independent of the grid size. Section 5 contains numerical experiments to illustrate and verify the theoretical results derived in sections 3 and 4. Finally, section 6 contains the conclusions of this paper.

2. The Helmholtz equation.

2.1. The acoustic wave equation. An acoustic medium with space-varying density $\rho(\mathbf{x})$ and sound speed $c(\mathbf{x})$ occupies the volume Ω, bounded by the boundary $\Gamma = \Gamma_1 \cup \Gamma_2 \cup \Gamma_3$. In addition, the medium is assumed to be viscous with damping coefficient $\gamma(\mathbf{x})$. The wave equation for the acoustic pressure $p(\mathbf{x}, t)$ (with $\mathbf{x}$ spatial coordinates and t time) in such a medium is

$$\frac{1}{\rho c^2}\frac{\partial^2 p}{\partial t^2} + \frac{\gamma}{\rho}\frac{\partial p}{\partial t} - \nabla \cdot \frac{1}{\rho}\nabla p = \frac{s(\mathbf{x}, t)}{\rho} \quad \text{in } \Omega, \tag{1}$$

where ∇ denotes the gradient operator and $\nabla\cdot$ the divergence. Realistic conditions on the physical boundaries of an acoustic medium can be reflecting boundaries, which are described by the homogeneous Neumann condition

$$\frac{\partial p}{\partial n} = 0 \quad \text{on } \Gamma_1; \tag{2}$$

pressure release boundaries, which are described by the homogeneous Dirichlet condition

$$p = 0 \quad \text{on } \Gamma_2; \tag{3}$$

and radiating boundaries, which can be described by

$$\frac{\partial p}{\partial n} = -\frac{1}{\rho c}\frac{\partial p}{\partial t} \quad \text{on } \Gamma_3, \tag{4}$$

where n is the outward pointing normal unit vector.

We will assume that the right-hand side function $s(\mathbf{x}, t)$ is a harmonic point source $s(\mathbf{x}, t) = ae^{2\pi i f t}\delta(\mathbf{x} - \mathbf{x}_s)$, located at $\mathbf{x}_s$, which transmits a signal with amplitude a and frequency f. Here, $i = \sqrt{-1}$.

2.2. The Helmholtz equation. If the source term is harmonic, then the pressure field has the factored form

$$p(\mathbf{x}, t) = \hat{p}(\mathbf{x})e^{2\pi i f t}. \tag{5}$$

Substitution of (5) into (1) yields the so-called Helmholtz equation

$$\left(\frac{-(2\pi f)^2}{\rho c^2} + 2\pi i f\frac{\gamma}{\rho}\right)\hat{p} - \nabla \cdot \frac{1}{\rho}\nabla\hat{p} = \frac{a}{\rho}\delta(\mathbf{x} - \mathbf{x}_s) \quad \text{in } \Omega, \tag{6}$$

with boundary conditions

$$\frac{\partial \hat{p}}{\partial n} = 0 \quad \text{on } \Gamma_1, \tag{7}$$

$$\hat{p} = 0 \quad \text{on } \Gamma_2, \tag{8}$$

and

$$\frac{\partial \hat{p}}{\partial n} = -\frac{2\pi i f}{\rho c}\hat{p} \quad \text{on } \Gamma_3. \tag{9}$$

This latter condition is also known as a Sommerfeld condition of the first kind.

If the damping parameter has the special form

$$\gamma(\mathbf{x}, f) = 2\pi f \frac{\nu}{c^2}, \tag{10}$$

with ν a nonnegative constant, the Helmholtz equation (6) simplifies to

$$\left(-\frac{(2\pi f)^2}{\rho c^2}(1 - i\nu) - \nabla \cdot \frac{1}{\rho}\nabla\right)\hat{p} = a\frac{\delta(\mathbf{x} - \mathbf{x}_s)}{\rho(\mathbf{x}_s)} \quad \text{in } \Omega. \tag{11}$$

Clearly, the assumption holds for nonviscous media, i.e., with $\gamma = 0$.

The above equation can be discretized with a discretization method such as the finite element method, finite volume method, or finite difference method. Discretization of the above equation plus boundary conditions with any of these methods yields a discrete Helmholtz equation of the form

$$(L + iC - z_1 M)x = b \tag{12}$$

in which L is the discretization of $-\nabla \cdot \frac{1}{\rho}\nabla$, M corresponds to the discretized zeroth order term $\frac{1}{\rho c^2}$, C corresponds to the Sommerfeld condition and/or to damping that does not satisfy (10), and b to the source term. The complex number z_1 is defined by

$$z_1 = (2\pi f)^2(1 - i\nu). \tag{13}$$

Both L and C are real symmetric and positive semidefinite, and the matrix M is real symmetric and positive definite. The matrix $L + iC - z_1 M$, however, is complex symmetric and indefinite.

For high frequencies, system (12) can be very large, in particular in three dimensions. This is a consequence of the fact that each wavelength has to be sampled with sufficient grid points. Numbers of unknowns in excess of 10^6 for realistic models are quite common. Fortunately, system (12) is sparse. Krylov-type iterative solvers such as GMRES [12] or Bi-CGSTAB [14] are among the most popular techniques for solving large and sparse linear systems. They have proved to be particularly efficient for systems with an Hermitian positive definite matrix, or more generally, for systems with a matrix with all eigenvalues in the right half of the complex plane. Helmholtz-type systems such as (12), however, are highly indefinite, which means that the system matrix has eigenvalues with both negative and positive real parts, a characteristic that can result in a very slow convergence. In order to overcome this problem a suitable preconditioner has to be applied. A class of promising preconditioners that has attracted a lot of attention is the class of shifted Laplace preconditioners [1, 8, 2, 3, 4]. In the next section we will analyze these preconditioners by locating in the complex plane the spectrum of the preconditioned discrete Helmholtz operator.

3. Spectral analysis of the Helmholtz operator preconditioned with the shifted Laplace preconditioner. Shifted Laplace preconditioners are preconditioners of the form

$$P = L + iC - z_2 M,$$

i.e., the same form as the discrete Helmholtz operator $A = L + iC - z_1 M$. The shift parameter z_2 has to be chosen such that the convergence of GMRES (or another suitable iterative method) applied to the preconditioned system

$$(L + iC - z_2 M)^{-1}(L + iC - z_1 M)x = (L + iC - z_2 M)^{-1} b$$

is considerably faster than GMRES applied to the original system. Moreover, z_2 has to be chosen such that operations with the inverse of $(L + iC - z_2M)$ are easy to perform. In practice this means that z_2 is chosen such that operations with the inverse of the preconditioner can be computed using a fast multigrid method [4]. Note that $L + iC$ is the operator that is being shifted. This means that all boundary conditions, including the Sommerfeld condition, are included in the preconditioner, as recommended in [10].

The complex numbers z_1 and z_2 can be written as

$$z_1 = \alpha_1 + i\beta_1, \qquad z_2 = \alpha_2 + i\beta_2, \tag{14}$$

in which $\alpha_1, \beta_1, \alpha_2$, and β_2 are real. Recall that for our application, z_1 is defined by (13), and hence

$$\alpha_1 > 0, \qquad \beta_1 \leq 0.$$

Choices for the shift z_2 that are considered in the literature are $z_2 = 0$ [1], $z_2 = -\alpha_1$ [8], $z_2 = -i\alpha_1$ [2, 3], and $z_2 = (1 - 0.5i)\alpha_1$ [4].

In this section we will study the spectrum of the preconditioned system. The spectrum governs, to a large extent, the convergence of iterative methods as long as the matrix of the eigenvectors is well conditioned.

3.1. The spectrum of the preconditioned Helmholtz operator without the Sommerfeld condition. We will first assume that $C = 0$, and hence that the damped Helmholtz operator is given by $L - z_1M$. We recall that L is symmetric positive semidefinite, M symmetric positive definite, and z_1 is a complex number. We will consider a shifted Laplace preconditioner, i.e., a matrix of the form $L - z_2M$, as a preconditioner for the Helmholtz operator and analyze how the location of the eigenvalues σ of the preconditioned system depends on the parameters z_1 and z_2. The eigenvalues σ of the preconditioned matrix are solutions of the generalized eigenproblem

$$(L - z_1M)x = \sigma(L - z_2M)x. \tag{15}$$

It is easy to see that the matrices $(L - z_1M)$ and $(L - z_2M)$ share the same eigenvectors, which are the eigenvectors of

$$Lx = \lambda Mx. \tag{16}$$

Since for our problem L is symmetric positive semidefinite and M symmetric positive definite, the eigenvalues λ are real and nonnegative. Substitution of λMx for Lx in (15) yields

$$(\lambda - z_1)Mx = \sigma(\lambda - z_2)Mx,$$

and hence

$$\lambda - z_1 = \sigma(\lambda - z_2), \tag{17}$$

which, if $z_2 \neq \lambda$, gives

$$\sigma = \frac{\lambda - z_1}{\lambda - z_2}. \tag{18}$$

Note that if z_2 coincides with an eigenvalue of (16), i.e., if $z_2 = \lambda$, the preconditioner $P = L - z_2 M$ will be singular, which is a situation that has to be avoided. So in the following we assume that $z_2 \neq \lambda$ for all eigenvalues of (16). The eigenvalues λ can be considered as a real parameterization of the curves (18) in the complex plane on which the eigenvalues σ of the preconditioned system are located.

To determine these curves we write $\sigma = \sigma^r + i\sigma^i$ and substitute this into (17), which yields

$$\lambda - \alpha_1 - i\beta_1 = \sigma^r(\lambda - \alpha_2) - i\sigma^r\beta_2 + i\sigma^i(\lambda - \alpha_2) + \sigma^i\beta_2.$$

We can split this equation into an equation for the real terms and one for the imaginary terms:

$$\lambda - \alpha_1 = \sigma^r(\lambda - \alpha_2) + \sigma^i\beta_2,$$

$$-\beta_1 = -\sigma^r\beta_2 + \sigma^i(\lambda - \alpha_2).$$

If $\beta_1 = \sigma^r\beta_2$, the second equation reduces to $\sigma^i = 0$. If this is not the case, we get for λ that

$$\lambda = \alpha_2 + \frac{\sigma^r\beta_2 - \beta_1}{\sigma^i}.$$

Substitution of λ in the equation for the real terms yields the following result:

$$\beta_2(\sigma^r)^2 - (\beta_1 + \beta_2)\sigma^r + \beta_2(\sigma^i)^2 + (\alpha_1 - \alpha_2)\sigma^i = -\beta_1. \tag{19}$$

This equation is valid for all values of $\alpha_1, \beta_1, \alpha_2$, and β_2, including the case $\beta_1 = \sigma^r\beta_2$.

In the following we will distinguish between the cases $\beta_2 = 0$ and $\beta_2 \neq 0$. Theorem 3.1 deals with the case $\beta_2 = 0$.

THEOREM 3.1. [illegible] $\beta_2 = 0$ [illegible] $\sigma = \sigma^r + i\sigma^i$ [illegible] (15) [illegible]

$$-\beta_1\sigma^r + (\alpha_1 - \alpha_2)\sigma^i + \beta_1 = 0. \tag{20}$$

[illegible] The result follows directly from substituting $\beta_2 = 0$ in (19). ∎

The next theorem characterizes the spectrum in the case when $\beta_2 \neq 0$.

THEOREM 3.2. [illegible] $\beta_2 \neq 0$ [illegible] $\sigma = \sigma^r + i\sigma^i$ [illegible] (15) [illegible]

$$\left(\sigma^r - \frac{\beta_2 + \beta_1}{2\beta_2}\right)^2 + \left(\sigma^i - \frac{\alpha_2 - \alpha_1}{2\beta_2}\right)^2 = \frac{(\beta_2 - \beta_1)^2 + (\alpha_2 - \alpha_1)^2}{(2\beta_2)^2}. \tag{21}$$

[illegible] c [illegible]

$$c = \left(\frac{\beta_2 + \beta_1}{2\beta_2}, \frac{\alpha_2 - \alpha_1}{2\beta_2}\right)$$

[illegible]

$$R = \sqrt{\frac{(\beta_2 - \beta_1)^2 + (\alpha_2 - \alpha_1)^2}{(2\beta_2)^2}}.$$

Proof. Divide (19) by $(2\beta_2)$ and complete the square. ∎

To understand the convergence of iterative methods it is important to know if the origin is enclosed by the circle given in Theorem 3.2. The following theorem gives a simple condition that determines this.

THEOREM 3.3. If $\beta_1\beta_2 > 0$ then (21) does not enclose the origin for the circle given in Theorem 3.2

Proof. The origin is not enclosed by the circle if the distance of the center to the origin is larger than the radius. Hence

$$\frac{(\beta_2+\beta_1)^2+(\alpha_2-\alpha_1)^2}{(2\beta_2)^2} > \frac{(\beta_2-\beta_1)^2}{(2\beta_2)^2}+\frac{(\alpha_2-\alpha_1)^2}{(2\beta_2)^2},$$

which is clearly the case. ∎

The center of the circle can also be written as

$$c = \frac{z_1-\overline{z}_2}{z_2-\overline{z}_2}$$

and the radius as

$$R = \left|\frac{z_2-z_1}{z_2-\overline{z}_2}\right|.$$

3.2. The spectrum of the preconditioned Helmholtz operator with the Sommerfeld condition. We will now study the general damped Helmholtz operator $L+iC-z_1M$. As before, L and C are symmetric positive semidefinite matrices, M is a symmetric and positive definite matrix, and z_1 is a complex number. For our problem, the matrix C stems either from the discretization of the Sommerfeld boundary condition or from damping that does not satisfy (10). This means, for example, that a damping matrix that stems from an absorbing layer is also covered by the theory below. We consider a shifted Laplace preconditioner, i.e., a matrix of the form $L+iC-z_2M$, to precondition the Helmholtz operator. The eigenvalues of this matrix are given by

$$(L+iC-z_1M)x = \sigma_S(L+iC-z_2M)x. \tag{22}$$

Let λ_S be an eigenvalue of the generalized problem

$$(L+iC)x = \lambda_S Mx. \tag{23}$$

As in the previous section it is straightforward to show that (22) and (23) share the same eigenvectors x and that the eigenvalues σ_S of the preconditioned system are related to the eigenvalues λ_S by

$$(\lambda_S-z_2)\sigma_S = \lambda_S - z_1. \tag{24}$$

The main difference from the previous section is that λ_S is *complex*, whereas λ in the previous section was real, which allowed us to consider λ as a real valued parameterization of a curve in the complex plane. Although the eigenvalues σ_S will in general not be located on a straight line or on a circle in the complex plane if λ_S is complex, it is still possible to establish useful results regarding the location of σ_S. To this end, we will distinguish between the three cases $\beta_2 = 0$, $\beta_2 > 0$, and $\beta_2 < 0$. Before we proceed we will formulate the following lemma that we will need in the remainder of this section.

LEMMA 3.1. *L C M* $\lambda_S = \lambda_S^r + i\lambda_S^i$ (23)

We use the fact that any matrix can be split into two Hermitian matrices:

$$A = \frac{1}{2}(A + A^H) + i\frac{1}{2i}(A - A^H) = \Re(A) + i\Im(A), \tag{25}$$

where

$$\Re(A) = \frac{1}{2}(A + A^H) \text{ and } \Im(A) = \frac{1}{2i}(A - A^H)\ . \tag{26}$$

According to Bendixon's theorem (see, e.g., [6, page 69]), we have

$$\lambda_{\min}^{\Re(A)} \le \mathrm{Re}(\lambda^A) \le \lambda_{\max}^{\Re(A)},$$

$$\lambda_{\min}^{\Im(A)} \le \mathrm{Im}(\lambda^A) \le \lambda_{\max}^{\Im(A)}.$$

The eigenvalues λ_S of the generalized problem (23) are also solutions of the standard eigenproblem

$$U^{-1}(L + iC)U^{-T}y = \lambda_S y$$

in which $M = UU^T$. This means that we can take $A = U^{-1}(L + iC)U^{-T}$, in which case $\mathrm{Im}(A) = U^{-1}CU^{-T}$. This latter matrix is positive semidefinite, so by Bendixon's theorem we have $\lambda_S^i \ge 0$. □

As in the previous section we first consider the case $\beta_2 = 0$.

THEOREM 3.4. *$\beta_2 = 0$ L C M* $\sigma_S = \sigma_S^r + i\sigma_S^i$ (22)

$$-\beta_1\sigma_S^r + (\alpha_1 - \alpha_2)\sigma_S^i + \beta_1 \ge 0.$$

Since $\beta_2 = 0$ we have

$$(\lambda_S - \alpha_2)\sigma_S = \lambda_S - z_1.$$

Splitting this equation into an equation for the real terms and one for the imaginary terms yields

$$\lambda_S^r\sigma_S^r - \lambda_S^i\sigma_S^i - \alpha_2\sigma_S^r = \lambda_S^r - \alpha_1$$

and

$$\lambda_S^r\sigma_S^i + \lambda_S^i\sigma_S^r - \alpha_2\sigma_S^i = \lambda_S^i - \beta_1.$$

The second equation gives that either $\sigma_S^i = 0$ or

$$\lambda_S^r = \alpha_2 - \frac{\beta_1}{\sigma_S^i} + \lambda_S^i\frac{1 - \sigma_S^r}{\sigma_S^i}.$$

Substitution in the first equation and some straightforward manipulations yield

$$-\beta_1\sigma_S^r + (\alpha_1 - \alpha_2)\sigma_S^i + \beta_1 = \lambda_S^i((\sigma_S^r - 1)^2 + {\sigma_S^i}^2).$$

By Lemma 3.1, $\lambda_S^i \geq 0$, and hence the right-hand side term is greater than or equal to zero. ∎

If $\beta_2 < 0$, the spectrum of the preconditioned matrix is characterized by Theorem 3.5.

THEOREM 3.5. $\beta_2 < 0$ L C M σ_S (22) $c = \frac{z_1 - \overline{z}_2}{z_2 - \overline{z}_2}$ $R = |\frac{z_2 - z_1}{z_2 - \overline{z}_2}|$

PROOF. We have to prove that $|\sigma_S - c| \leq R$ if $\beta_2 < 0$:

$$
\begin{aligned}
|\sigma_S - c| &= \left| \frac{\lambda_S - z_1}{\lambda_S - z_2} - \frac{z_1 - \overline{z}_2}{z_2 - \overline{z}_2} \right|, \\
&= \left| \frac{(\lambda_S - z_1)(z_2 - \overline{z}_2) - (\lambda_S - z_2)(z_1 - \overline{z}_2)}{(\lambda_S - z_2)(z_2 - \overline{z}_2)} \right|, \\
&= \left| \frac{\lambda_S(z_2 - z_1) + (z_1 - z_2)\overline{z}_2}{(\lambda_S - z_2)(z_2 - \overline{z}_2)} \right|, \\
&= \left| \frac{\lambda_S - \overline{z}_2}{\lambda_S - z_2} \frac{z_2 - z_1}{z_2 - \overline{z}_2} \right|, \\
&= \left| \frac{\lambda_S - \overline{z}_2}{\lambda_S - z_2} \right| R. \qquad (27)
\end{aligned}
$$

What is left to prove is that $|\frac{\lambda_S - \overline{z}_2}{\lambda_S - z_2}| \leq 1$. Writing

$$\lambda_S = \lambda_S^r + i\lambda_S^i$$

we get

$$\left| \frac{\lambda_S - \overline{z}_2}{\lambda_S - z_2} \right|^2 = \frac{(\lambda_S^r - \alpha_2)^2 + (\lambda_S^i + \beta_2)^2}{(\lambda_S^r - \alpha_2)^2 + (\lambda_S^i - \beta_2)^2}. \qquad (28)$$

Since $\beta_2 < 0$ and by Lemma 3.1, $\lambda_S^i \geq 0$, we have

$$\frac{(\lambda_S^r - \alpha_2)^2 + (\lambda_S^i + \beta_2)^2}{(\lambda_S^r - \alpha_2)^2 + (\lambda_S^i - \beta_2)^2} \leq 1,$$

and hence the above condition is satisfied. ∎

If $\beta_2 > 0$, the spectrum of the preconditioned matrix is characterized by Theorem 3.6.

THEOREM 3.6. $\beta_2 > 0$ L C M σ_S (22) $c = \frac{z_1 - \overline{z}_2}{z_2 - \overline{z}_2}$ $R = |\frac{z_2 - z_1}{z_2 - \overline{z}_2}|$

PROOF. This is analogous to the proof of Theorem 3.5. ∎

The results presented above specify regions in the complex plane where the eigenvalues of the preconditioned matrix are located. These regions are completely determined by the parameters z_1 and z_2. Given the definition of z_1, (13), these regions depend only on the frequency f, on the damping parameter ν, and of course on the shift for the preconditioner z_2. It is important to note that the regions in the complex plane where the eigenvalues are located do not depend on other physical parameters, such as the sound speed or density, nor on computational parameters, such as the size of the matrix, or on the mesh size h.

4. Combination of the results of the spectral analysis with an upper bound on the GMRES-residual norm. In this section we will combine the results of the spectral analysis presented in the previous section with a well-known upper bound on the GMRES-residual norm. This upper bound assumes that the spectrum is enclosed by a circle, and hence this bound can be naturally combined with the circle specified in Theorems 3.2 and 3.5.

Let the eigenvalues of the preconditioned matrix be enclosed by a circle with radius R and center c as in Theorem 3.5. Then the GMRES-residual norm after k iterations $\|r^k\|$ satisfies (see, e.g., [12])

$$\frac{\|r^k\|}{\|r^0\|} \leq c_2(X) \left(\frac{R}{|c|}\right)^k . \tag{29}$$

In this equation X is the matrix of eigenvectors and $c_2(X)$ its condition number in the 2-norm. If this condition number is large, the upper bound gives no information about the convergence, since in that case there is no relation between the location of the eigenvalues and the convergence behavior of the preconditioned Krylov method [5]. Fortunately, in our application we may expect that the condition number of the eigenvector matrix is relatively small. If $C = 0$, the eigenvectors of the preconditioned matrix are the same as in (16), and hence independent of the shift parameters. Moreover, since the eigenvectors of (16) are M-orthogonal and M is a (scaled) mass matrix. which is in general well conditioned, we expect that $c_2(X)$ will be small in practice. This can be seen from

$$X^T M X = I \;\Leftrightarrow\; c_2(X^T M X) = 1 \;\Leftrightarrow\; c_2(M^{\frac{1}{2}} X) = 1.$$

Since

$$c_2(X) = c_2(M^{-\frac{1}{2}} M^{\frac{1}{2}} X) \leq c_2(M^{-\frac{1}{2}}) c_2(M^{\frac{1}{2}} X),$$

we get

$$c_2(X) \leq \sqrt{c_2(M)}.$$

If $C \neq 0$, the eigenvectors of the preconditioned system are the same as of (23). These are unfortunately not M-orthogonal, but for many problems we can consider (23) as a relatively small perturbation of (16), in which case we can still expect that $c_2(X)$ is small.

4.1. Optimization of the shift. Although (29) gives only an upper bound on the GMRES-residual norm, it allows us to derive a "quasi" optimal choice for the shift. We derive this shift by minimizing the upper bound. For this it is sufficient to minimize the ratio $\frac{R}{|c|}$, or, using Theorem 3.5, the function

$$f(\alpha_2, \beta_2) = \frac{R^2}{|c|^2} = \frac{(\alpha_2 - \alpha_1)^2 + (\beta_2 - \beta_1)^2}{(\alpha_2 - \alpha_1)^2 + (\beta_2 + \beta_1)^2}.$$

To analyze this function we differentiate with respect to α_2,

$$\frac{\partial f}{\partial \alpha_2} = \frac{8(\alpha_2 - \alpha_1)\beta_1\beta_2}{((\alpha_2 - \alpha_1)^2 + (\beta_2 + \beta_1)^2)^2},$$

and with respect to β_2,

$$\frac{\partial f}{\partial \beta_2} = \frac{4\beta_1((\beta_2^2 - \beta_1^2) - (\alpha_2 - \alpha_1)^2)}{((\alpha_2 - \alpha_1)^2 + (\beta_2 + \beta_1)^2)^2}.$$

Clearly, both derivatives are zero at $\alpha_1 = \alpha_2, \beta_1 = \beta_2$. This choice for the shift minimizes of course the upper bound since this corresponds to using the original operator as a preconditioner, which means that performing the preconditioning operation is as hard as solving the original system.

We are interested in the case when the preconditioning operation is relatively cheap. In particular, we have in mind the situation when preconditioning operations can be efficiently carried out using a fixed number of cycles of a multigrid method for the whole range of shifts under consideration. We therefore restrict our analysis to values for the shift for which multigrid is known to work well. We first consider the purely imaginary shift [2, 3, 4]; this means that α_2 equals zero. In this case, the derivative with respect to β_2 is zero if

$$(\beta_2^2 - \beta_1^2) - \alpha_1^2 = 0, \tag{30}$$

yielding

$$\beta_2 = \pm|z_1|.$$

Since by (13) $\beta_1 = -(2\pi f)^2\nu \leq 0$, we must choose by Theorem 3.3 $\beta_2 \leq 0$, and hence $z_2 = -|z_1|i$ as the shift that minimizes the upper bound (29). This choice is also optimal if we consider all possible shifts for which $\alpha_2 \leq 0$, meaning all possible shifts for which the preconditioner has all its eigenvalues in the right half-plane. By (13), $\alpha_1 = (2\pi f)^2 > 0$, and by Theorem 3.3, $\beta_1\beta_2 \geq 0$, so $\frac{\partial f}{\partial \alpha_2}$ is negative for $\alpha_2 \leq 0$. Therefore $f(\alpha_2, \beta_2)$ takes its minimum on the edge $\alpha_2 = 0$. We conclude that the choice

$$z_2 = -|z_1|i \tag{31}$$

minimizes the upper bound (29) for all $z_2 \in \mathbb{C}$, with $\alpha_2 \leq 0$.

The same methodology for deriving an optimal shift can still be used if we do not restrict ourselves to the case $\alpha_2 \leq 0$. Such a shift still (approximately) minimizes the number of GMRES iterations. However, the performance of a multigrid method for the preconditioning operations will deteriorate if z_2 is too close to z_1, and hence such a shift would no longer minimize the total work of the whole solution process. How to find a shift that minimizes the total work, if the performance of multigrid depends on the shift, is of great practical importance, but is outside the scope of this paper.

4.2. Discussion. The upper bound (29) is meaningful only if the circle does not enclose the origin. This is the case if $\beta_1 < 0$, or equivalently if $\nu > 0$. However, because of continuity arguments, result (31) for the "quasi" optimal shift is still valid if $\beta_1 = 0$.

As was remarked in the previous section, the circle around the spectrum of the preconditioned matrix depends only on z_2, on f, and on ν. Consequently, if $\beta_1 < 0$, inequality (29) yields an upper bound on the GMRES-residual norm that also depends only on the frequency f and on the damping parameter ν. Because of this, the number of GMRES iterations should be bounded from above by a constant that is independent of the mesh size.

By scaling the shift z_2 with the frequency we can make the upper bound on the number of GMRES iterations also independent of frequency. To this end we introduce the scaled shift

$$\tilde{z_2} = \tilde{\alpha}_2 + i\tilde{\beta}_2 = \frac{z_2}{(2\pi f)^2}.$$

Applying Theorem 3.5 and substituting $z_2 = (2\pi f)^2 \tilde{z_2}$ and the definition for z_1 (13) into (29) yield

$$\frac{\|r^k\|}{\|r^0\|} \leq c_2(X) \left(\frac{R}{|c|}\right)^k = c_2(X) \sqrt{\frac{(\tilde{\alpha}_2 - 1)^2 + (\tilde{\beta}_2 + \nu)^2}{(\tilde{\alpha}_2 - 1)^2 + (\tilde{\beta}_2 - \nu)^2}}. \tag{32}$$

Clearly, this upper bound depends only on the damping parameter ν and on the choice for the parameter $\tilde{z}_2$.

5. Experiments. In this section we describe a typical test problem with a variable sound velocity. The location of the eigenvalues of the discretized operators are compared with the theoretically predicted locations. The value of the optimal shift is validated by numerical experiments. Finally, it appears that the convergence behavior of GMRES is independent of the mesh size.

5.1. Description of the test problem. The test problem that we consider mimics three layers with a simple heterogeneity and is taken from [11].

For $\nu \in \mathbb{R}$, find $p \in \mathbb{C}^N$ satisfying

$$\begin{cases} -\Delta p - (1-\nu)(\frac{2\pi f}{c(\mathbf{x})})^2 p = s \quad \text{in } \Omega = (0,600) \times (0,1000) \text{ meter}^2, \\ s = \delta(x_1 - 300, x_2), \quad x_1 = (0,600), \quad x_2 = (0,1000) \\ \text{with Sommerfeld conditions or Neumann conditions on } \Gamma \equiv \partial\Omega. \end{cases} \tag{33}$$

The local sound velocity is given as in Figure 1. The density is assumed to be constant.

We have discretized the above problem with the finite element method using linear triangular elements. The computations that are described in this section have been performed with MATLAB.

5.2. Location of the eigenvalues. The first experiments validate the theorems that are presented in section 3. To this end we have taken as source frequency $f = 2$ and we have discretized the problem with mesh size $h = 100/2$. We have calculated all the eigenvalues of the preconditioned matrix for four typical combinations of values of the scaled parameters $\tilde{z}_1$ and $\tilde{z}_2$. (See Figure 2.) The upper left-hand side subplot shows the spectrum of the preconditioner if a real shift is chosen, as suggested in [8]. As was predicted by Theorem 3.1, the eigenvalues for the Neumann problem, which are indicate with the symbol *, are located on a line. Since the example contains damping, the line does not pass through the origin. The eigenvalues of the Sommerfeld problem, which are indicated with the symbol o, are all on one side of the line, as is predicted by Theorem 3.4. Note that the eigenvalues move away from the origin if the Neumann problem is replaced with the Sommerfeld problem.

The upper right-hand side subplot shows the spectrum of the preconditioner if a purely imaginary shift is chosen. The eigenvalues of the Neumann problem are located on the circle that is given by Theorem 3.2, and the eigenvalues of the Sommerfeld problem are, as predicted by Theorem 3.5, either on or inside this circle. This example does not contain damping (apart from the radiation condition): $\tilde{z}_1$ is real. Consequently, the circle contains the origin.

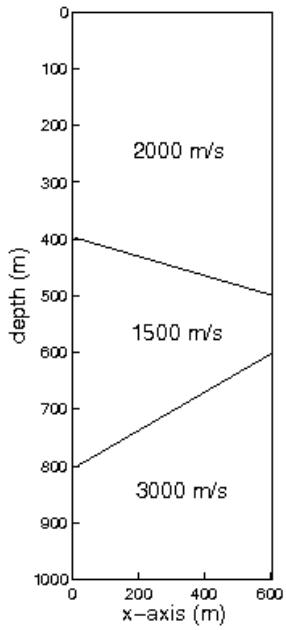


FIG. 1. *Problem geometry with sound velocity profile.*

The lower left-hand side subplot shows another picture of the spectrum of the preconditioner if a purely imaginary (negative) shift is chosen. In this case the problem contains damping since $\tilde{z}_1$ is complex. As a result, the circle is smaller than for the previous example, and the origin is outside the circle.

The lower right-hand side subplot shows for the same z_1 what happens if the sign of the complex shift is wrongly (i.e., positively) chosen. According to Theorem 3.6 the eigenvalues of the Sommerfeld problem should in this case be either on or outside the circle. This is confirmed by the numerical results. Moreover, by Theorem 3.3, the origin should be enclosed by the circle, which is the case.

5.3. Optimization of the shift. The second group of experiments validates the optimal value for the shift z_2 that was found in section 4. This value is given by (31).

The optimal value was determined by minimizing the ratio $\frac{R}{|c|}$. Using the scaled variables $\tilde{z}_2 = z_2/(2\pi f)^2$, this ratio can be written as

$$\frac{R}{|c|} = \sqrt{\frac{(\tilde{\alpha}_2 - 1)^2 + (\tilde{\beta}_2 + \nu)^2}{(\tilde{\alpha}_2 - 1)^2 + (\tilde{\beta}_2 - \nu)^2}}. \tag{34}$$

This function takes values between 0 and 1. A small value of $\frac{R}{|c|}$ indicates fast convergence and a value close to 1 slow convergence. Figure 3 shows, for three different damping parameters ν, how the value of $\frac{R}{|c|}$ depends on $\tilde{\alpha}_2$ and $\tilde{\beta}_2$. The values on the contour lines correspond to the value of $\frac{R}{|c|}$. The three plots show clearly that (34) takes its minimum when $\tilde{\alpha}_2 = 0$, and that the optimal $\tilde{\beta}_2$ becomes more negative when the damping parameter is increased. These observations are of course consistent with the optimal value for the shift parameter (31) that was derived in section 4.

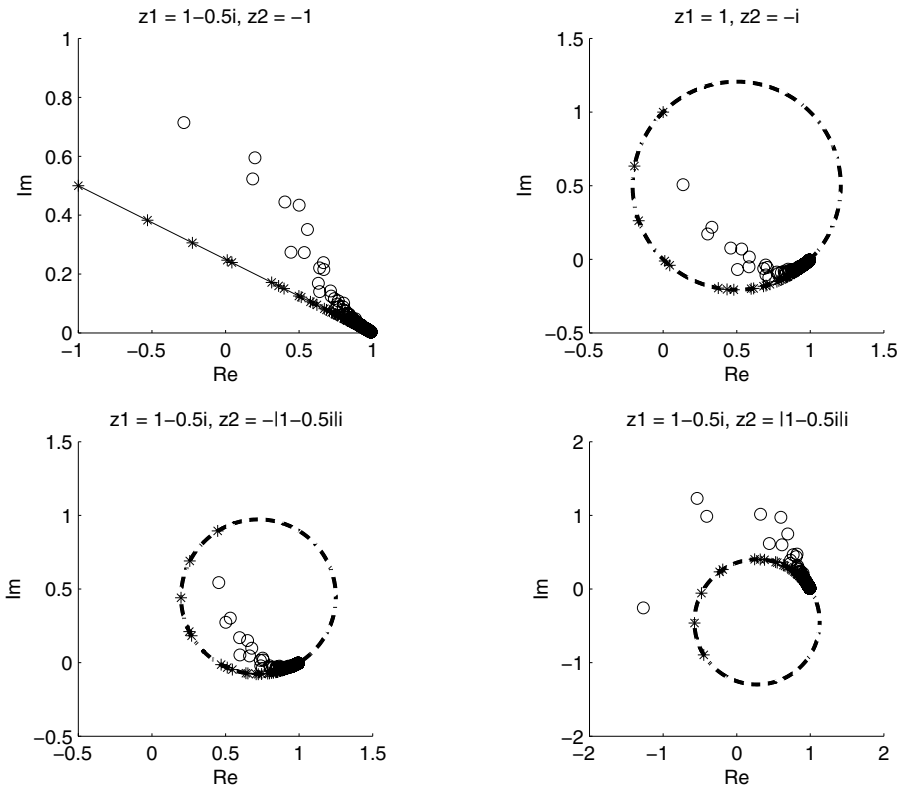


FIG. 2. *Spectra for different values of the complex shift in the preconditioner;* $h = 100/2, f = 2$. *The symbol * denotes eigenvalues for the Neumann problem; o denotes eigenvalues for the Sommerfeld problem.*

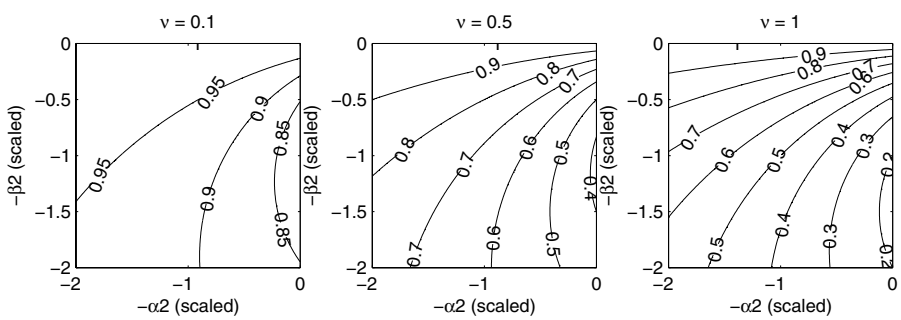


FIG. 3. *Contour plot of the convergence factor as a function of the complex shift.*

To validate that this value is really (sub-) optimal in actual computations we solve the Sommerfeld problem with scaled imaginary shifts ranging from 0 to -2. For the mesh size we take $h = 100/8$ and perform the experiment for four different damping parameters. The result is shown in Figure 4. Clearly, the more damping, the fewer the number of GMRES iterations, and the larger (more negative) the optimal imaginary shift.

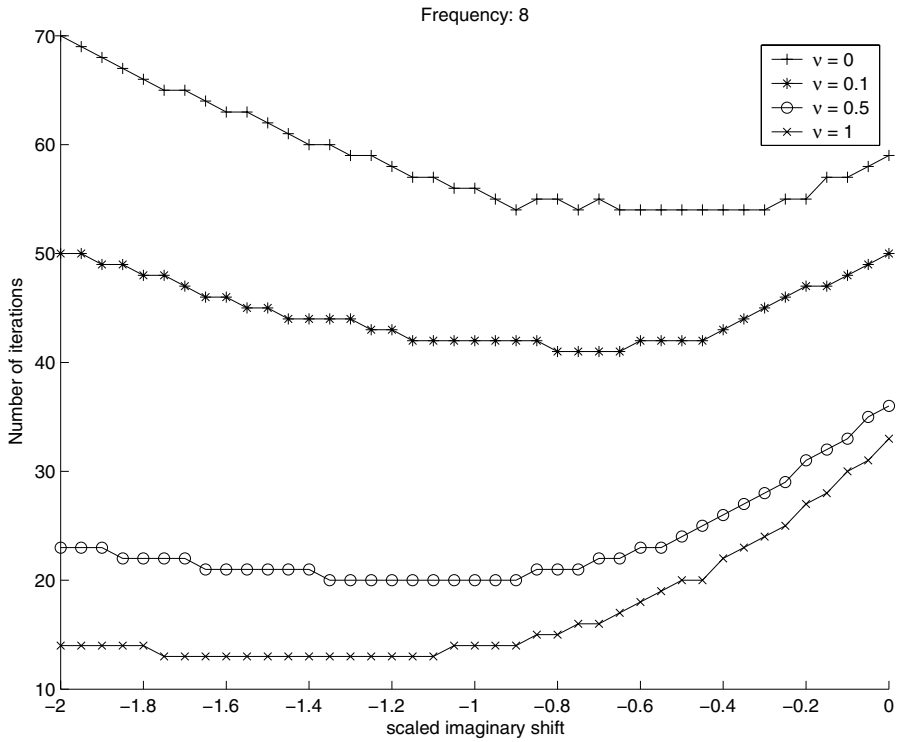


FIG. 4. *Actual number of iteration as a function of the imaginary shift* ($h = 100/8$).

TABLE 1
Number of iterations for "optimal" shift and minimum number of iterations.

Damping	Optimal shift	Iterations "optimal" shift	Min. number of iterations
$\nu = 0$	1	56	54
$\nu = 0.1$	1.005	42	41
$\nu = 0.5$	1.118	20	20
$\nu = 1$	1.4142	13	13

Table 1 shows the minimum number of iterations and compares this with the number of iterations when the "optimal" shift (31) is used. The results show that the shift (31) is nearly optimal with respect to the number of GMRES iterations.

5.4. Mesh dependency. The last set of experiments examines the dependency of the number of iterations of preconditioned GMRES on the mesh size.

Table 2 shows for an increasingly fine step size h the number of iterations for the Sommerfeld problem. The experiment is performed for four different damping parameters, and the frequency is kept fixed to $f = 2$. The results show that for all four different values of the damping parameter the number of iterations is independent of the step size h. Based on the discussion at the end of section 4, this could be expected. The theory that is presented in section 4, however, does not make any predictions about the mesh-dependent performance of preconditioned GMRES for problems with a zero damping parameter.

The results of the same type of experiments, but now with a frequency that scales

TABLE 2
Number of iterations under mesh refinement for a fixed frequency.

	Number of iterations				
h :	100/2	100/4	100/8	100/16	100/32
f :	2	2	2	2	2
$\nu = 0$	14	13	13	13	13
$\nu = 0.1$	13	12	12	12	13
$\nu = 0.5$	11	10	11	11	11
$\nu = 1$	9	9	9	9	9

TABLE 3
Number of iterations under mesh refinement for increasingly high frequencies.

	Number of iterations				
h :	100/2	100/4	100/8	100/16	100/32
f :	2	4	8	16	32
$\nu = 0$	14	25	56	116	215
$\nu = 0.1$	13	22	42	63	80
$\nu = 0.5$	11	16	20	23	23
$\nu = 1$	9	11	13	13	13

with the mesh size, are tabulated in Table 3. These results confirm that if the damping parameter is nonzero, the number of GMRES iterations is bounded by a number that is independent of the mesh size. This is most apparent in the results for $\nu = 0.5$ and $\nu = 1$. The results for $\nu = 0$ seem to indicate that the number of GMRES iterations more or less doubles if the step size is halved. As was remarked above, the theory presented in section 4 does not make any predictions for the case when $\nu = 0$.

To check that $c_2(X)$ is actually small for the above test cases we have also computed the condition numbers of the mass matrices on the five meshes. These condition numbers are equal to 24 for all meshes; hence we have that

$$c_2(X) \leq \sqrt{c_2(M)} = 2\sqrt{3}.$$

6. Conclusions. We have presented a spectral analysis of the Helmholtz operator that is preconditioned with a shifted Laplace operator. We have shown that, depending on the value of the shift, the eigenvalues of the preconditioned matrix are located either in or on a circle, or in a half-plane. Combining these results concerning the spectrum of the preconditioned matrix with a well-known bound on the GMRES-residual norm allowed us to determine a close-to-optimal shift. Furthermore, we have shown for problems with a nonzero damping parameter that there is an upper bound on the number of GMRES iterations that depends only on the damping parameter and hence is independent of the mesh size, the frequency, the sound speed, and the density.

We have derived the close-to-optimal shift for the shifted-Laplace preconditioner in combination with GMRES under the assumption that preconditioning operations are performed exactly. In practice, however, preconditioning operations are performed approximately, for example, using a multigrid method, and another Krylov method, such as Bi-CGSTAB [14], may be used instead of GMRES. In this case the analysis that has been presented in this paper no longer holds. However, experimental results reported in [3], where Bi-CGSTAB is used as a Krylov solver and preconditioning operations are performed approximately with one multigrid cycle, use values for the shift that are close to the predicted value for the optimal shift we present in this

paper. The experimental results are also in these cases quite satisfactory. We therefore conclude that our results provide strong guidelines on how to select the shift parameter for all Krylov methods, as well as for approximate preconditioners.

Acknowledgments. The authors thank Kees Oosterlee for valuable discussions.

REFERENCES

[1] A. Bayliss, C. I. Goldstein, and E. Turkel, *An iterative method for the Helmholtz equation*, J. Comput. Phys., 49 (1983), pp. 443–457.
[2] Y. A. Erlangga, C. Vuik, and C. W. Oosterlee, *On a class of preconditioners for the Helmholtz equation*, Appl. Numer. Math., 50 (2005), pp. 409–425.
[3] Y. A. Erlangga, C. W. Oosterlee, and C. Vuik, *Comparison of multigrid and incomplete LU shifted-Laplace preconditioners for the inhomogeneous Helmholtz equation*, Appl. Numer. Math., 56 (2006), pp. 648–666.
[4] Y. A. Erlangga, C. W. Oosterlee, and C. Vuik, *A novel multigrid based preconditioner for heterogeneous Helmholtz problems*, SIAM J. Sci. Comput., 27 (2006), pp. 1471–1492.
[5] A. Greenbaum, V. Pták, and Z. Strakoš, *Any nonincreasing convergence curve is possible for GMRES*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 465–469.
[6] A. S. Householder, *The Theory of Matrices in Numerical Analysis*, Blaisdell Publishing Company, New York, 1964.
[7] R. Kechroud, A. Soulaimani, and Y. Saad, *Preconditioning techniques for the solution of the Helmholtz equation by the finite element method*, in Computational Science and Its Applications (ICCSA 2003), Part II, Lecture Notes in Comput. Sci. 2668, V. Kumar et al., eds., Springer, Berlin, 2003, pp. 847–858.
[8] A. L. Laird and M. B. Giles, *Preconditioned Iterative Solution of the* 2*D Helmholtz Equation*, Tech. Report 02/12, Oxford Computer Laboratory, Oxford, UK, 2002.
[9] A. L. Laird and M. B. Giles, *Preconditioning harmonic unsteady potential flow calculations*, AAIA J., 44 (2006), pp. 2654–2662.
[10] T. A. Manteuffel and S. V. Parter, *Preconditioning and boundary conditions*, SIAM J. Numer. Anal., 27 (1990), pp. 656–694.
[11] R. E. Plessix and W. A. Mulder, *Separation-of-variables as a preconditioner for an iterative Helmholtz solver*, Appl. Numer. Math., 44 (2003), pp. 385–400.
[12] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.
[13] E. Turkel, *Numerical methods and nature*, J. Sci. Comput., 28 (2006), pp. 549–570.
[14] H. A. van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 631–644.

SIAM J. SCI. COMPUT.
Vol. 29, No. 5, pp. 1959–1981
© 2007 Society for Industrial and Applied Mathematics

AN EFFICIENT SOLVER FOR THE INCOMPRESSIBLE NAVIER–STOKES EQUATIONS IN ROTATION FORM*

MICHELE BENZI† AND JIA LIU‡

Abstract. We consider preconditioned iterative methods applied to discretizations of the linearized Navier–Stokes equations in two- and three-dimensional bounded domains. Both unsteady and steady flows are considered. The equations are linearized by Picard iteration. We make use of the rotation form of the momentum equations, which has several advantages from the linear algebra point of view. We focus on a preconditioning technique based on the Hermitian/skew-Hermitian splitting of the resulting nonsymmetric saddle point matrix. We show that this technique can be implemented efficiently when the rotation form is used. We study the performance of the solvers as a function of mesh size, Reynolds number, time step, and algorithm parameters. Our results indicate that fast convergence independent of problem parameters is achieved in many cases. The preconditioner appears to be especially attractive in the case of low viscosity and for unsteady problems.

Key words. fluid mechanics, Navier–Stokes, Krylov methods, preconditioning, rotation form, Oseen problem, Schur complement, Hermitian and skew-Hermitian splitting, generalized Stokes problem

AMS subject classifications. Primary, 65F10, 65N22, 65F50; Secondary, 76D07

DOI. 10.1137/060658825

1. Introduction.

Iterative Krylov subspace algorithms are popular methods for solving large and sparse linear systems arising in many areas of computational science and engineering [29, 34]. It is well known that in order to be effective, iterative methods must be coupled with preconditioning techniques. In the last several years, much progress has been made in the development of preconditioners tailored to the linear systems arising from linearizations and discretizations of incompressible flow problems; see [5, 36] and especially the recent monograph [12]. As a result of this effort, several effective solvers now exist for important problems like the generalized Stokes problem and the Oseen problem. Nevertheless, the performance of even the best solvers tends to deteriorate for small values of the viscosity. Furthermore, most existing methods for the Oseen problem rely on the availability of efficient solvers for convection-dominated convection-diffusion equations, which are nontrivial to develop for complex flow pattern, especially in three dimensions.

In this paper we study a preconditioning technique that is applicable to various incompressible flow problems. The preconditioner is especially well suited for the so-called [illegible] of the Navier–Stokes equations [14, 21, 27, 37]. We present numerical results on both steady and unsteady problems in two and three space dimensions aimed at assessing the performance of the solver with respect to problem parameters such as mesh size, time step, and viscosity. These experiments indicate that the solver is effective for a wide range of problems. Implementation of the preconditioner is fairly straightforward, and it can take advantage of existing solvers

*Received by the editors May 3, 2006; accepted for publication (in revised form) January 16, 2007; published electronically September 28, 2007. This work was supported by National Science Foundation grants DMS-0207599 and DMS-0511336.

http://www.siam.org/journals/sisc/29-5/65882.html

†Department of Mathematics and Computer Science, Emory University, Atlanta, GA 30322 (benzi@mathcs.emory.edu).

‡Department of Mathematics and Statistics, West Florida University, Pensacola, FL 32514 (jliu@uwf.edu).

(and software) for scalar elliptic problems. In addition, the preconditioner does not require the solution of convection-diffusion problems.

The paper is organized as follows. In section 2 we recall the standard and rotation forms of the incompressible Navier–Stokes equations in primitive variables. In section 3 we describe the discrete form of the linearized equations and briefly discuss the structure and spectral properties of the coefficient matrices. Section 4 contains a description of the preconditioning techniques used in this paper and a brief overview of related work. Numerical experiments on a selection of steady and unsteady model problems in two and three space dimensions are presented in section 5. Here we also report on the convergence of the Picard linearization for the rotation form. Some conclusions are given in section 6.

2. Incompressible fluid flow problems. We are concerned with the primitive variables formulation of the Navier–Stokes equations for a viscous, incompressible Newtonian fluid in a bounded domain $\Omega \subset \mathbb{R}^d$ $(d = 2, 3)$ with boundary $\partial\Omega$:

$$\frac{\partial \mathbf{u}}{\partial t} - \nu \quad \mathbf{u} + (\mathbf{u} \cdot \nabla)\,\mathbf{u} + \nabla p = \mathbf{f} \quad \text{in } \Omega \times (0,T]\,, \tag{2.1}$$

$$\operatorname{div} \mathbf{u} = 0 \quad \text{in } \Omega \times [0,T]\,, \tag{2.2}$$

$$\mathcal{B}\mathbf{u} = \mathbf{0} \quad \text{on } \partial\Omega \times [0,T]\,, \tag{2.3}$$

$$\mathbf{u}(\mathbf{x},0) = \mathbf{u}_0 \quad \text{in } \Omega\,, \tag{2.4}$$

where $\mathbf{u} = \mathbf{u}(\mathbf{x},t) \in \mathbb{R}^d$ is the velocity field, $p = p(\mathbf{x},t) \in \mathbb{R}$ is the pressure, $\mathbf{f} \in \mathbb{R}^d$ is a known forcing term, $\nu > 0$ is the kinematic viscosity coefficient ($\nu = O(\mathrm{Re}^{-1})$, where Re denotes the Reynolds number), $\quad$ is the (vector) Laplace operator in d dimensions, ∇ is the gradient, div is the divergence, and $\mathcal{B}$ is some type of boundary operator (e.g., a trace operator for Dirichlet boundary conditions). Equation (2.1) will be referred to as the $\quad$ of the momentum equation. The pressure field, p, is determined up to an additive constant. To uniquely determine p we may impose some additional condition, such as

$$\int_\Omega p\, d\mathbf{x} = 0\,. \tag{2.5}$$

Discretization in time with a fully implicit method (such as backward Euler, although more sophisticated methods are often used in practice) leads to a sequence of semidiscrete systems of the form

$$\sigma\mathbf{u} - \nu \quad \mathbf{u} + (\mathbf{u} \cdot \nabla)\,\mathbf{u} + \nabla p = \mathbf{f} \quad \text{in } \Omega\,, \tag{2.6}$$

$$\operatorname{div} \mathbf{u} = 0 \quad \text{in } \Omega\,, \tag{2.7}$$

$$\mathcal{B}\mathbf{u} = \mathbf{0} \quad \text{on } \partial\Omega\,. \tag{2.8}$$

Here $\sigma \geq 0$ is a constant, with $\sigma > 0$ in the unsteady case and $\sigma = 0$ in the steady case. Due to the presence of the inertial term $(\mathbf{u} \cdot \nabla)\,\mathbf{u}$ in the momentum equation (2.6), the Navier–Stokes equations are nonlinear. They can be linearized in various ways. A widely used scheme is Picard (fixed-point) linearization [12], which leads to a sequence of $\quad$, i.e., linear problems of the form

$$\sigma\mathbf{u} - \nu \quad \mathbf{u} + (\mathbf{v} \cdot \nabla)\,\mathbf{u} + \nabla p = \mathbf{f} \quad \text{in } \Omega\,, \tag{2.9}$$

$$\operatorname{div} \mathbf{u} = 0 \quad \text{in } \Omega\,, \tag{2.10}$$

$$\mathcal{B}\mathbf{u} = \mathbf{0} \quad \text{on } \partial\Omega\,. \tag{2.11}$$

In the momentum equation (2.9), the vector field $\mathbf{v}$ is the approximation of the solution from the previous Picard iteration. For $\mathbf{v} = \mathbf{0}$, equations (2.9)–(2.11) are known as the [illegible] if in addition $\sigma = 0$, they reduce to the usual (steady) Stokes problem.

An alternative linearization can be derived based on the identity

$$(\mathbf{u} \cdot \nabla)\mathbf{u} = (\nabla \times \mathbf{u}) \times \mathbf{u} + \frac{1}{2}\nabla(\mathbf{u} \cdot \mathbf{u}); \tag{2.12}$$

see [14, 21]. The linearization consists in approximating the right-hand side of (2.12) with the linearized expression $(\nabla \times \mathbf{v}) \times \mathbf{u} + \frac{1}{2}\nabla(\mathbf{u} \cdot \mathbf{u})$, where $\mathbf{v}$ represents the approximation of the velocity field from the previous Picard iteration. The linearized equations are

$$\sigma\mathbf{u} - \nu \quad \mathbf{u} + \mathbf{w} \times \mathbf{u} + \nabla P = \mathbf{f} \quad \text{in } \Omega, \tag{2.13}$$

$$\operatorname{div} \mathbf{u} = 0 \quad \text{in } \Omega, \tag{2.14}$$

$$\mathcal{B}\mathbf{u} = \mathbf{0} \quad \text{on } \partial\Omega. \tag{2.15}$$

In (2.13) we have set $P = p + \frac{1}{2}\mathbf{u} \cdot \mathbf{u}$ (the [illegible]). Moreover, in the two-dimensional (2D) case

$$(\mathbf{w}\times) = \begin{pmatrix} 0 & w \\ -w & 0 \end{pmatrix},$$

with $w = \nabla \times \mathbf{v} = -\frac{\partial v_1}{\partial x_2} + \frac{\partial v_2}{\partial x_1}$, where $\mathbf{v}$ is the approximation of the solution from the previous Picard iteration. Similarly, for three-dimensional (3D) problems

$$(\mathbf{w}\times) = \begin{pmatrix} 0 & -w_3 & w_2 \\ w_3 & 0 & -w_1 \\ -w_2 & w_1 & 0 \end{pmatrix},$$

where w_i denotes the ith component of the vector $\nabla \times \mathbf{v}$. Notice that no first-order terms in the velocities appear in (2.13); on the other hand, the velocities in the d scalar equations comprising (2.13) are now coupled due to the presence of the term $\mathbf{w} \times \mathbf{u}$. The disappearance of the convective terms suggests that the [illegible] (2.13) of the momentum equations may be advantageous over the standard form (2.9) from the linear solution point of view; see also [21]. For a discussion of the excellent conservation properties of the linearization based on the rotation form, see [22]; another relevant reference is [27].

3. The discrete equations. In this paper we assume $\Omega = [0,1]^d$, and we use a standard marker-and-cell (MAC) scheme [15] with a uniform grid to discretize either the momentum form of the linearized equations (2.9)–(2.11) or the rotation form (2.13)–(2.15). However, most of what follows holds for more complicated geometries and for a variety of spatial discretization schemes, and in particular for finite elements; see [12].

The discrete system has the form

$$\begin{bmatrix} A & B^T \\ B & O \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ p \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ 0 \end{bmatrix}, \qquad \text{or} \qquad \mathcal{A}\,u = b, \tag{3.1}$$

where now $\mathbf{u}$, p, and $\mathbf{f}$ are finite-dimensional counterparts of the functions denoted by the same symbols. System (3.1) is often referred to as a generalized saddle point

problem; see [5]. When pressure stabilization is used (e.g., to circumvent the Babuška–Brezzi condition), a nonzero (2,2) block will generally be present in $\mathcal{A}$. Here we assume that a div-stable method (like MAC) is being used, so that no stabilization is needed. However, much of what follows admits a straightforward extension to the case where the (2,2) block of $\mathcal{A}$ consists of a symmetric negative semidefinite matrix.

As is well known, for small values of the viscosity ν velocity stabilization becomes necessary in order to prevent spurious (nonphysical) oscillations in the discrete solution corresponding to all but the finest meshes. For the time being, we sidestep the issue and assume that a sufficiently fine mesh is used. Note that the same assumption is made (for Galerkin FEM discretizations) in [12, p. 327]. We return to this topic in the section on numerical experiments.

The B block in (3.1) represents a discrete divergence operator. In the 2D case, $B = \begin{bmatrix} B_1 & B_2 \end{bmatrix}$, where B_1 is a discretization of $\frac{\partial}{\partial x}$ and B_2 is a discretization of $\frac{\partial}{\partial y}$. The form of the (1,1) block A depends on whether the convection form or the rotation form of the momentum equation is used. For the convection form we have, in the 2D case,

$$A = \begin{bmatrix} \sigma I + \nu L_1 + S_1 & O \\ O & \sigma I + \nu L_2 + S_2 \end{bmatrix},$$

where L_1, L_2 are discrete (negative) Laplacians and S_1, S_2 represent discretizations of the components of the convection term $(\mathbf{v} \cdot \nabla)\,\mathbf{u}$. In this paper we assume that the convective terms are skew-symmetric: $S_1 = -S_1^T$, $S_2 = -S_2^T$. This assumption is satisfied, for example, if the boundary conditions are of Dirichlet type only, or if Neumann boundary conditions are present on the outflow and characteristic boundaries only; see [12] for a detailed discussion of this issue, especially pp. 120–121 and 151. In the 3D case, there will be three diagonal blocks of the same form rather than two.

If the rotation form is used we will have, in the 2D case,

$$A = \begin{bmatrix} \sigma I + \nu L_1 & D \\ -D & \sigma I + \nu L_2 \end{bmatrix},$$

where the matrix D represents multiplication by the discretized function $w = \nabla \times \mathbf{v}$. For MAC discretization, D is a just a diagonal matrix containing on the main diagonal the values of w on the grid edges. In the 3D case,

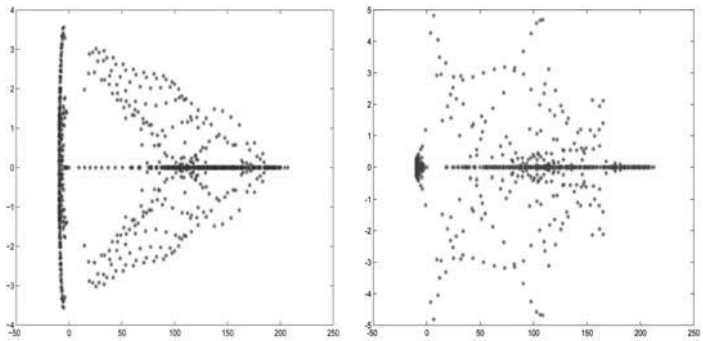
$$A = \begin{bmatrix} \sigma I + \nu L_1 & -D_3 & D_2 \\ D_3 & \sigma I + \nu L_2 & -D_1 \\ -D_2 & D_1 & \sigma I + \nu L_3 \end{bmatrix},$$

where now D_i is the matrix that corresponds to multiplication by w_i, the ith component of $\nabla \times \mathbf{v}$. For finite element discretizations, the I and D matrices will be replaced by appropriate (weighted) mass matrices which may no longer be diagonal.

We observe that for both the convection and the rotation form we can write $A = H + K$, where (2D case)

$$H = \frac{1}{2}(A + A^T) = \begin{bmatrix} \sigma I + \nu L_1 & O \\ O & \sigma I + \nu L_2 \end{bmatrix}$$

is symmetric positive definite for all $\sigma \geq 0$. The matrix $K = \frac{1}{2}(A - A^T)$ is skew-symmetric, and its structure will depend on the form of the momentum equation. In the case of the (generalized) Stokes problem, $K = O$.



(a) Convection form (b) Rotation form

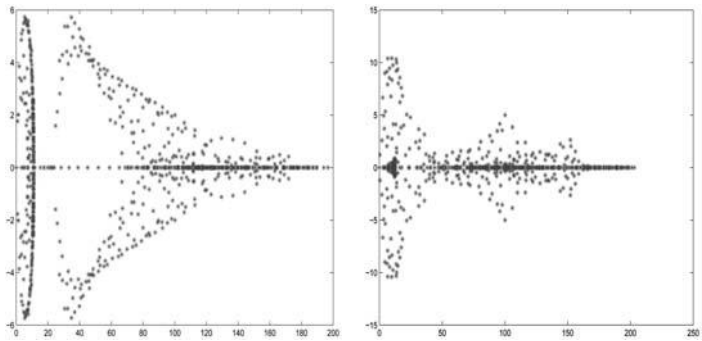
FIG. 3.1. *Eigenvalues for 2D linearized Navier–Stokes problem using MAC discretization (indefinite form).*

With either form of the momentum equation the saddle point matrix $\mathcal{A}$ is highly indefinite, in the sense that it has many eigenvalues on either side of the imaginary axis. Figure 3.1(a) displays the eigenvalues of the discrete steady Oseen operator (convection form, $\nu = 0.01$) obtained from a MAC discretization on $\Omega = [0,1]\times[0,1]$ on a 16×16 grid. As the wind function, we used $\mathbf{v} = (8x(x-1)(1-2y), 8(2x-1)y(y-1))$. Figure 3.1(b) displays the eigenvalues of the discrete operator corresponding to the rotation form on the same grid, with $\nu = 0.01$ and $w = 16x(x-1) + 16y(y-1) = \nabla \times \mathbf{v}$. As can be seen, both matrices have many eigenvalues with negative real part. Additional algebraic properties of saddle point matrices are discussed in [5, sect. 3].

Eigenvalue distributions such as those shown in Figures 3.1(a)–3.1(b) are generally unfavorable for solution by Krylov subspace methods, and, indeed, without preconditioning, Krylov subspace methods converge very slowly (or not at all) when applied to the corresponding linear systems. It has been observed by several authors (see [5, sect. 3.4]) that a simple transformation can be used to obtain an equivalent linear system with a coefficient matrix whose spectrum is entirely contained in the half-plane $\Re(z) > 0$. (Here we use $\Re(z)$ and $\Im(z)$ to denote the real and imaginary parts of $z \in \mathbb{C}$.) Indeed, we can rewrite the saddle point system in the equivalent form

$$\begin{bmatrix} A & B^T \\ -B & O \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ p \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ 0 \end{bmatrix}, \qquad \text{or} \qquad \hat{\mathcal{A}}\, u = \hat{b}\,. \tag{3.2}$$

Figures 3.2(a)–(b) display the eigenvalues of the matrix $\hat{\mathcal{A}}$ corresponding to the same linearized problems as before (convection and rotation form, respectively). As can be seen, all the eigenvalues lie in the right half-plane. Matrices with this property are said to be [illegible]. A good preconditioner should preserve this property while at the same time clustering the eigenvalues away from the origin. Ideally, most of the eigenvalues should be in the cluster, and the cluster itself should be bounded away from zero independently of the mesh size h and the viscosity ν. A few outliers do not typically constitute a problem. In general, ν-independence is harder to achieve than h-independence; see [12].



(a) Convection form (b) Rotation form

FIG. 3.2. *Eigenvalues for 2D linearized Navier–Stokes problem using MAC discretization (positive definite form).*

4. Preconditioner description. In this section we introduce and motivate our proposed preconditioning strategy for the linearized Navier–Stokes equations in rotation form, including inexact variants. We also (briefly) mention related work by other authors.

4.1. HSS preconditioner and its variants. The Hermitian/skew-Hermitian splitting (HSS) preconditioner is based on the nonsymmetric formulation (3.2). Letting $H \equiv \frac{1}{2}(A + A^T)$ and $K \equiv \frac{1}{2}(A - A^T)$, we have the following splitting of $\hat{\mathcal{A}}$ into its symmetric and skew-symmetric parts:

$$\hat{\mathcal{A}} = \begin{bmatrix} A & B^T \\ -B & O \end{bmatrix} = \begin{bmatrix} H & O \\ O & O \end{bmatrix} + \begin{bmatrix} K & B^T \\ -B & O \end{bmatrix} = \mathcal{H} + \mathcal{K}\,. \tag{4.1}$$

Note that $\mathcal{H}$, the symmetric part of $\hat{\mathcal{A}}$, is symmetric positive semidefinite since H is. Let $\alpha > 0$ be a parameter. Similar in spirit to the classical alternating-direction implicit (ADI) method, we consider the following two splittings of $\hat{\mathcal{A}}$:

$$\hat{\mathcal{A}} = (\mathcal{H} + \alpha\mathcal{I}) - (\alpha\mathcal{I} - \mathcal{K}) \qquad \text{and} \qquad \hat{\mathcal{A}} = (\mathcal{K} + \alpha\mathcal{I}) - (\alpha\mathcal{I} - \mathcal{H})\,.$$

Here $\mathcal{I}$ denotes the identity matrix of order $n + m$. Note that

$$\mathcal{H} + \alpha\mathcal{I} = \begin{bmatrix} H + \alpha I_n & O \\ O & \alpha I_m \end{bmatrix}$$

is symmetric positive definite with eigenvalues bounded below by $\alpha > 0$. Similarly,

$$\mathcal{K} + \alpha\mathcal{I} = \begin{bmatrix} K + \alpha I_n & B^T \\ -B & \alpha I_m \end{bmatrix}$$

is nonsingular and has positive definite symmetric part. Its eigenvalues are of the form $\alpha \pm \mathrm{i}\mu$ with $\mu \in \mathbb{R}$, where $\mathrm{i} = \sqrt{-1}$.

Alternating between these two splittings leads to the stationary HSS iteration:

$$\begin{cases} (\mathcal{H} + \alpha\mathcal{I})\, u_{k+\frac{1}{2}} = (\alpha\mathcal{I} - \mathcal{K})\, u_k + \hat{b}, \\ (\mathcal{K} + \alpha\mathcal{I})\, u_{k+1} = (\alpha\mathcal{I} - \mathcal{H})\, u_{k+\frac{1}{2}} + \hat{b} \end{cases} \tag{4.2}$$

($k = 0, 1, \dots$). Here $\hat{b}$ denotes the right-hand side of (3.2); the initial guess u_0 is chosen arbitrarily. Elimination of $u_{k+\frac{1}{2}}$ from (4.2) leads to a stationary (fixed-point) iteration of the form

$$u_{k+1} = \mathcal{T}_\alpha u_k + c\,, \quad k = 0, 1, \dots,$$

where $\mathcal{T}_\alpha = (\mathcal{K} + \alpha\mathcal{I})^{-1}(\alpha\mathcal{I} - \mathcal{H})(\mathcal{H} + \alpha\mathcal{I})^{-1}(\alpha\mathcal{I} - \mathcal{K})$ is the iteration matrix and c is a certain vector. The iteration converges for arbitrary initial guesses u_0 and right-hand sides $\hat{b}$ to the solution $u_* = \hat{\mathcal{A}}^{-1}\hat{b}$ if and only if $\varrho(\mathcal{T}_\alpha) < 1$, where $\varrho(\mathcal{T}_\alpha)$ denotes the spectral radius of $\mathcal{T}_\alpha$. This stationary iteration was first introduced in [2], where convergence was proved for systems of the form $Ax = b$ with A [illegible] (i.e., the symmetric part of A is positive definite). In the same paper, the authors studied the optimal choice of the parameter α and the effect of inexact solves in (4.2). The extension of this method to saddle point systems (for which the symmetric part is usually singular) was first given in [4]; in this paper it was also observed that the rate of convergence of the stationary HSS iteration is often very slow (even with the optimal choice of the parameter α) and that the method should be used as a preconditioner for a Krylov subspace method rather than as a solver.

It follows from the general theory in [7] that there is a unique splitting $\hat{\mathcal{A}} = \mathcal{P} - \mathcal{Q}$ with $\mathcal{P}$ nonsingular such that the iteration matrix $\mathcal{T}_\alpha$ is the matrix induced by that splitting, i.e., $\mathcal{T}_\alpha = \mathcal{P}^{-1}\mathcal{Q} = \mathcal{I} - \mathcal{P}^{-1}\hat{\mathcal{A}}$. An easy calculation shows that the HSS iteration (4.2) can be written in [illegible] as

$$u_{k+1} = u_k + \mathcal{P}^{-1} r_k, \quad r_k = \hat{b} - \hat{\mathcal{A}} u_k\,,$$

where the preconditioner $\mathcal{P}$ is given by

$$\mathcal{P} \equiv \mathcal{P}_\alpha = \tfrac{1}{2\alpha}(\mathcal{H} + \alpha\mathcal{I})(\mathcal{K} + \alpha\mathcal{I})\,. \tag{4.3}$$

Note that as a preconditioner we can use $\mathcal{P}_\alpha = (\mathcal{H} + \alpha\mathcal{I})(\mathcal{K} + \alpha\mathcal{I})$ instead of the expression given in (4.3), since the factor $\frac{1}{2\alpha}$ has no effect on the preconditioned system. It is just a normalization factor that allows us to conclude that $\varrho(\mathcal{T}_\alpha) < 1$; hence, the eigenvalues of the preconditioned matrix $\mathcal{P}_\alpha^{-1}\hat{\mathcal{A}}$ (or $\hat{\mathcal{A}}\,\mathcal{P}_\alpha^{-1}$, which has the same spectrum) are all contained in the disk $\{z \in \mathbb{C} : |z - 1| < 1\}$. In particular, the spectrum of the preconditioned matrix, like that of $\hat{\mathcal{A}}$, lies entirely in the right half-plane: the preconditioned matrix is positive stable.

Spectral properties of the preconditioned matrix as a function of α have been studied, under different sets of assumptions, in [3, 33]. A Fourier analysis of HSS preconditioning for saddle point formulations of Poisson's equation (including the anisotropic case) was given in [3]. The analysis showed that using a sufficiently small value of α results in h-independent convergence. Furthermore, as $\alpha \to 0+$ the eigenvalues of the preconditioned matrix are all real and fall within two small intervals $(0, \varepsilon_1)$ and $(2 - \varepsilon_2, 2)$, with $\varepsilon_1, \varepsilon_2 > 0$ and $\varepsilon_1, \varepsilon_2 \to 0$ as $\alpha \to 0+$. This clustering result was generalized in [33] to general saddle point systems with $A = A^T$ positive definite using purely algebraic arguments; see also [6]. Numerical experiments show that with an appropriate scaling of the system (such that the nonzero diagonal entries of $\hat{\mathcal{A}}$ are equal to 1), there is a unique value α_* of α for which the number of preconditioned iterations is minimized, and this α_* is usually a small number, between 0 and 1. However, h-independent convergence is not always guaranteed; for example, it does not occur for the [illegible] Stokes problem, as we shall see.

No rigorous analysis exists for the nonsymmetric ($A \neq A^T$) case. Here the only available result is that $\varrho(\mathcal{T}_\alpha) < 1$ for all $\alpha > 0$. Nevertheless, the following informal argument suggests that good performance of the HSS preconditioner applied to linearized Navier–Stokes problems should be expected for sufficiently small ν and α. Let us assume that $A = \nu L + K$, where $\frac{1}{2}(A + A^T) = H = \nu L$ is positive definite and K is skew-symmetric; note that this is certainly the case for the linearizations of the steady Navier–Stokes equations discussed in section 2, where L is a discrete vector Laplacian in $\mathbb{R}^d$, and K contains the skew-symmetric contribution from the convective or rotation terms. We have

$$\mathcal{P}_\alpha = \begin{bmatrix} \nu L + \alpha I & O \\ O & \alpha I \end{bmatrix} \begin{bmatrix} K + \alpha I & B^T \\ -B & \alpha I \end{bmatrix} = \begin{bmatrix} \alpha A + \nu L K + \alpha^2 I & \alpha B^T + \nu L B^T \\ -\alpha B & \alpha^2 I \end{bmatrix}.$$

Therefore, as $\nu \to 0$ the (scaled) preconditioner $\alpha^{-1}\mathcal{P}_\alpha$ approaches the matrix $\hat{\mathcal{A}} + \alpha\mathcal{I}$. For small α this is close to $\hat{\mathcal{A}}$, and we conclude that the HSS preconditioner may be interpreted as an [illegible] of the coefficient matrix $\hat{\mathcal{A}}$, in which the approximation error $\|\alpha^{-1}\mathcal{P}_\alpha - \hat{\mathcal{A}}\|$ decreases as $\nu, \alpha \to 0+$. (This holds true for both the convection and the rotation form of the linearized Navier–Stokes equations.) In practice, however, the shift α should not be taken too small, as the preconditioner becomes singular for $\alpha = 0$. The choice of α is an important issue which we defer until section 5.

In the unsteady case ($\sigma > 0$ in (2.9)), direct computation of $\alpha^{-1}\mathcal{P}_\alpha - \hat{\mathcal{A}}$ shows that the HSS preconditioner does not yield, as ν and α become smaller, a good approximation, except when σ is very small. However, it is possible to remedy the situation by making a small modification to the preconditioner. Observing that $H = \sigma I_n + \nu L$, we consider the following splitting of $\hat{\mathcal{A}}$:

$$\hat{\mathcal{A}} = \begin{bmatrix} \nu L & O \\ O & O \end{bmatrix} + \begin{bmatrix} K + \sigma I & B^T \\ -B & O \end{bmatrix} = \mathcal{H}' + \mathcal{K}'. \tag{4.4}$$

Note that $\mathcal{H}'$ is still symmetric, but $\mathcal{K}'$ is no longer skew-symmetric unless $\sigma = 0$. Now let $\alpha > 0$ be a shift and consider the preconditioner $\mathcal{P}_\alpha = (\mathcal{H}' + \alpha\mathcal{I})(\mathcal{K}' + \alpha\mathcal{I})$. Expanding the product shows that as $\nu, \alpha \to 0+$, the scaled preconditioner $\alpha^{-1}\mathcal{P}_\alpha$ approaches $\hat{\mathcal{A}}$. It can be shown (see [19]) that for this splitting as well, the spectral radius of the corresponding iteration matrix satisfies $\varrho(\mathcal{T}_\alpha) < 1$ for all $\alpha > 0$ and for all $\sigma \geq 0$; this result generalizes Theorem 3.1 in [4]. Furthermore, for $\sigma > 0$ sufficiently large the preconditioner yields a good approximation even for the generalized (i.e., unsteady) Stokes and Oseen-type problems. These observations are corroborated by the numerical results in section 5.

Since for $\sigma = 0$ the preconditioner based on the splitting (4.4) reduces to the standard HSS preconditioner, from now on we will use the name "HSS preconditioner" for the method based on the splitting (4.4), and we will write $\mathcal{H}$, $\mathcal{K}$ instead of $\mathcal{H}'$, $\mathcal{K}'$.

4.1.1. Application of the preconditioner. Application of the HSS preconditioner within a Krylov subspace method (e.g., GMRES [30]) requires solving a linear system of the form $\mathcal{P}_\alpha z_k = r_k$ at each iteration. This is done by first solving the system

$$(\mathcal{H} + \alpha\mathcal{I})\, w_k = r_k \tag{4.5}$$

for w_k, followed by

$$(\mathcal{K} + \alpha\mathcal{I})\, z_k = w_k\,. \tag{4.6}$$

The first system requires solving systems with coefficient matrix $H+\alpha I$. This consists of d $(= 2, 3)$ decoupled subsystems (one for each spatial direction) with symmetric positive definite coefficient matrix, regardless of whether the rotation or the convection form of the equations is used. A number of standard methods can be deployed, including sparse Cholesky factorization, preconditioned conjugate gradient schemes, or multigrid methods, either geometric or algebraic. Iterative methods can be expected to work quite well, since each system consists of a shifted Poisson-type equation. Note that owing to the diagonal scaling used, the addition of a positive shift to the main diagonal of H leads to a condition number

$$\kappa(H+\alpha I) = \frac{\lambda_{\max}(H)+\alpha}{\lambda_{\min}(H)+\alpha} < 1 + \frac{\lambda_{\max}(H)}{\alpha} = O(\alpha^{-1}),$$

since $\lambda_{\max}(H)$ is bounded independently of h. A typical value of α is usually around 0.1 or larger, resulting in well-conditioned systems and consequently fast rates of convergence of iterative methods applied to (4.5).

Solving (4.6) is somewhat more involved. It requires the solution of a linear system of the form

$$\begin{bmatrix} K+\gamma I & B^T \\ -B & \alpha I \end{bmatrix} \begin{bmatrix} x_k \\ y_k \end{bmatrix} = \begin{bmatrix} f_k \\ g_k \end{bmatrix}, \tag{4.7}$$

where we have set $\gamma = \sigma+\alpha$. This system can be solved in several ways. One approach is to eliminate x_k from the second equation using the first one (Schur complement reduction), leading to a smaller (order m) linear system of the form

$$[B(K+\gamma I)^{-1}B^T + \alpha I]y_k = B(K+\gamma I)^{-1}f_k + \alpha g_k. \tag{4.8}$$

Once the solution y_k to (4.8) has been computed, the vector x_k is given by

$$(K+\gamma I)x_k = f_k - B^T y_k\,.$$

When $K = O$, system (4.8) simplifies to

$$(BB^T + \alpha\gamma I)\, y_k = Bf_k + \alpha\gamma g_k, \tag{4.9}$$

and $x_k = \gamma^{-1}(f_k - B^T y_k)$. Note that BB^T is essentially a discrete Laplacian, and that $\gamma > 0$; therefore, system (4.9) can be solved efficiently. When $K \neq O$ the matrix in (4.8) is dense for the convective form of the Navier–Stokes equations but sparse for the rotation form. As this is a crucial point, we discuss it in some detail. Recall that in the 2D case the system matrix for the linearization of the rotation form is

$$\begin{bmatrix} A_1 & D & B_1^T \\ -D & A_2 & B_2^T \\ -B_1 & -B_2 & O \end{bmatrix} = \begin{bmatrix} \nu L_1 & O & O \\ O & \nu L_2 & O \\ O & O & O \end{bmatrix} + \begin{bmatrix} \sigma I & D & B_1^T \\ -D & \sigma I & B_2^T \\ -B_1 & -B_2 & O \end{bmatrix} = \mathcal{H} + \mathcal{K}\,.$$

Here the L_i are discrete Laplace operators with appropriate boundary conditions, and $B = \begin{bmatrix} B_1 & B_2 \end{bmatrix}$; $D = D^T$ is a matrix that corresponds to multiplication by $w = \nabla \times \mathbf{v}$. For finite difference schemes (like MAC) D is a diagonal matrix; for finite elements, it will be a scaled mass matrix. When D is diagonal, the Schur complement $B(K+\gamma I)^{-1}B^T + \alpha I$ is a sparse matrix and can be formed explicitly. This follows from the fact that

$$(I+\gamma^{-1}K)^{-1} = \begin{bmatrix} I & \gamma^{-1}D \\ -\gamma^{-1}D & I \end{bmatrix}^{-1} = \begin{bmatrix} E_1 & -E_2 \\ E_2 & E_3 \end{bmatrix},$$

where E_1, E_2, and E_3 are diagonal matrices given by $E_1 = I - \gamma^{-2} D(I + \gamma^{-2} D^2)^{-1} D$, $E_2 = \gamma^{-1} D(I + \gamma^{-2} D^2)^{-1}$, and $E_3 = (I + \gamma^{-2} D^2)^{-1}$. This can be verified directly, or also by noting that $I + \gamma^{-1} K = PGP^T$, where G is a block diagonal matrix with two-by-two blocks and P is the odd-even permutation matrix. Thus, $(I + \gamma^{-1} K)^{-1} = PG^{-1}P^T$ has exactly the same sparsity structure as $I + \gamma^{-1} K$.

When D is not diagonal, we can replace it with a (spectrally equivalent) diagonal approximation and still have a sparse Schur complement; since we are constructing a preconditioner, the action of $(\mathcal{K} + \alpha \mathcal{I})^{-1}$ need not be computed exactly. Hence, the Schur complement $B(K + \gamma I)^{-1} B^T + \alpha I$ (or the approximation of it obtained by replacing D with a diagonal matrix) is sparse, and system (4.6) can be efficiently solved via (4.8) by explicitly forming the sparse Schur complement system and solving it by appropriate sparse matrix techniques. Standard options include sparse LU factorization or nonsymmetric Krylov subspace methods with an appropriate choice of the preconditioner. It is also possible to use multigrid methods, since the Schur complement can be interpreted as a discretization of a second-order elliptic operator with variable coefficients. Indeed, the matrix $B(I + \gamma^{-1} K)^{-1} B^T$ can be regarded as a discretization of the differential operator

$$\mathcal{L} = -\mathrm{div}\,(k^{-1}\,\nabla)\,, \quad \text{where} \quad k = \begin{pmatrix} 1 & \gamma^{-1} w \\ -\gamma^{-1} w & 1 \end{pmatrix}, \quad w = w(\mathbf{x}). \tag{4.10}$$

This operator is not self-adjoint (except in the case $w =$ constant), but has positive definite symmetric part; in other words, it is strongly elliptic. Efficient multigrid methods exist for this type of problem; see [22] for an approach especially well suited for this problem. Clearly, the larger γ is, the easier it is to solve the Schur complement system.

While we have focused our description on the 2D case, the 3D case can be treated along similar lines. The system matrix is now

$$\hat{\mathcal{A}} = \begin{bmatrix} \nu L_1 & O & O & O \\ O & \nu L_2 & O & O \\ O & O & \nu L_3 & O \\ O & O & O & O \end{bmatrix} + \begin{bmatrix} \sigma I & -D_3 & D_2 & B_1^T \\ D_3 & \sigma I & -D_1 & B_2^T \\ -D_2 & D_1 & \sigma I & B_3^T \\ -B_1 & -B_2 & -B_3 & O \end{bmatrix} = \mathcal{H} + \mathcal{K}\,.$$

The Schur complement is now given by

$$\alpha I + B(I + \gamma^{-1} K)^{-1} B^T = \alpha I + B \begin{bmatrix} F_1 & -E_3 & E_2 \\ E_3 & F_2 & -E_1 \\ -E_2 & E_1 & F_3 \end{bmatrix} B^T,$$

where F_1, F_2, F_3, E_1, E_2, and E_3 are easily computed diagonal matrices. We refer to [19] for additional details. Again, the Schur complement system is sparse and can be solved using standard sparse matrix solvers or a multigrid method.

4.1.2. Inexact variants. In practice, the solves (4.5) and (4.6)–(4.8) need not be performed to high accuracy. Rather than "exact" solves, inexact solves can be performed by inner iterative schemes applied to (4.5) and (4.8). Our experience is that the rate of convergence of the outer Krylov subspace iteration is scarcely affected by the use of inexact inner solves, and that a fairly low accuracy in the inner solves is sufficient to retain good rates of convergence of the outer iteration. This is especially important for very large problems, where exact inner solves would not be feasible.

A distinction has to be made depending on the type of solvers used in the inner iterations. If the inexact inner solves consist of a fixed number of steps of a stationary iterative method (such as multigrid), then the (inexact) HSS preconditioner remains constant from outer iteration to outer iteration and a standard Krylov subspace method, such as GMRES, can be used for the outer iteration. On the other hand, if the inner iterations are terminated when a reduction of the inner residual norms by a prescribed amount has been reached, or if the inner iterations consist of a few steps of a nonstationary method (such as conjugate gradients), then the inexact HSS preconditioner changes from one outer iteration to the next, and a flexible method (such as FGMRES [28]) must be used for the outer iteration.

We reiterate that the linear systems arising from (4.5) are well conditioned and are easy to solve approximately. In particular, approximate solutions of (4.5) can be obtained in $O(n)$ operations using multigrid or preconditioned conjugate gradients. For the Schur complement system (4.9) arising in the case of Stokes and generalized Stokes problems, similar techniques can be used to obtain an approximate solution in $O(m)$ work. For the nonsymmetric (Oseen-type) case, the Schur complement system is the nonsymmetric problem (4.8). In this case one could use GMRES with ILU preconditioning, but multigrid is a viable option here as well; see [22].

The above discussion assumes that the rotation form of the equations is used. For the standard (convective) form of the Navier–Stokes equations, the solution of (4.6), whether exact or inexact, appears to be problematic.

4.2. Related work. Olshanskii has proposed an effective block triangular preconditioner for the rotation form of the (linearized) Navier–Stokes equations; see [21, 22, 23] as well as [24] and [20]. His method is quite robust with respect to both h and ν. The preconditioner has the form

$$\mathcal{P} = \begin{bmatrix} \hat{A} & B^T \\ O & \hat{S} \end{bmatrix}, \tag{4.11}$$

where $\hat{A}$ and $\hat{S}$ are not explicitly computed matrices but implicit approximations to the (1,1) block of the saddle point system and of the Schur complement, respectively. Application of the preconditioner within an outer Krylov subspace iteration requires evaluating the action of $\hat{A}^{-1}$ and of $\hat{S}^{-1}$ on given vectors. For the action of $\hat{A}^{-1}$, a suitable multigrid method has been developed and analyzed in [24]. Note that the (1,1) block A can be interpreted as a coupled system of reaction-diffusion equations for the two (three in three dimensions) components of the velocity field. Clearly, this is a more complicated problem than inverting the uncoupled shifted velocity Laplacians in the HSS approach; see (4.5).

The approximate Schur complement step requires inexact solves for the elliptic problem (4.10) already mentioned in the previous subsection. We refer the reader to [21, 23] for details. The cost of this operation is comparable to the cost of solving the Schur complement system (4.8) in the HSS preconditioner; in the latter method, however, the system to be solved is somewhat easier due to the presence of the shift parameter α, which results in linear systems with eigenvalues that are better separated from the imaginary axis. In particular, for small σ the elliptic problem (4.10) tends to become singular and must be regularized by a suitable diagonal shift in the tensor k appearing in (4.10). In our experience, the performance of the preconditioner can be affected rather strongly by the choice of the shift parameter.

In summary, Olshanskii's preconditioner requires specialized solvers for the subproblems corresponding to both the (1,1) block and the approximate Schur comple-

ment, and applying this preconditioner within one step of an outer Krylov method is somewhat more expensive than with the HSS preconditioner.

It must be noted that several other preconditioners for the discrete Navier–Stokes equations can be cast as approximate (block) factorizations of the system matrix. Furthermore, many algebraic splitting methods for the unsteady Navier–Stokes equations can be interpreted in terms of approximate block factorizations; this is the case, for instance, for the classical fractional step method and for the popular SIMPLE scheme and its variants; see, e.g., the recent references [9, 18, 26, 31, 35, 36]. These splittings naturally lead to preconditioners for Krylov methods, for both steady and unsteady problems [12, Chapter 8.3.5]. While these preconditioners have been designed for the convection form, some of them may be formally extended to the rotation form of the linearized Navier–Stokes equations; this is the case, for instance, for Elman's so-called BFB^T preconditioner [10]. Regardless of which form of the equations is used, however, these preconditioning schemes do not yield better approximations as the viscosity ν approaches zero; on the contrary, the quality of the approximation tends to deteriorate as $\nu \to 0$, especially for stationary problems. This is an important difference between these earlier approaches and the HSS and Olshanskii-type preconditioners.

Furthermore, most of the block preconditioners for the standard form of the Navier–Stokes equations depend on the availability of effective solvers (exact or approximate) for convection-dominated convection-diffusion equations, which are not easily developed. In contrast, no such solvers are needed when the rotation form of the equations is used.

5. Numerical experiments. In this section we report on several numerical experiments meant to illustrate the behavior of the HSS preconditioner on a wide range of model problems. We consider both Stokes and Oseen-type problems, steady and unsteady, in two and three dimensions. The use of inexact solves is also discussed. Our results include iteration counts, as well as some timings and eigenvalue plots. We further include a few experiments illustrating the convergence of Picard's iteration for the rotation form. All results were computed in MATLAB 7.1.0 on one processor of an AMD Opteron with 32 GB of memory. Many additional experimental results can be found in [19]; the results presented here are but a representative sampling of those in [19].

In all experiments, a symmetric diagonal scaling was applied before forming the preconditioner, so as to have all the nonzero diagonal entries of the saddle point matrix equal to 1. We found that this scaling is beneficial to convergence and makes finding (nearly) optimal values of the shift α easier. Of course, the right-hand side and the solution vector were scaled accordingly. We found, however, that without further preconditioning Krylov subspace solvers converge extremely slowly on all the problems considered here. Right preconditioning was used in all cases.

5.1. Oseen-type equations in rotation form. Here we consider linear systems arising from the discretization of the linearized Navier–Stokes equations in rotation form. For lack of a better name we will refer to the linear problem (2.13)–(2.15) as the [illegible]. The computational domain is the unit square $\Omega = [0,1] \times [0,1]$ (or the unit cube $\Omega = [0,1] \times [0,1] \times [0,1]$ for 3D problems). Homogeneous Dirichlet boundary conditions are imposed on the velocities; experiments with different boundary conditions were also performed, with results similar to those reported below. We experimented with different forms of the divergence-free field $\mathbf{v}$ appearing (via $\mathbf{w} = \nabla \times \mathbf{v}$) in (2.13). Here we present results for the choice

TABLE 5.1
Iteration counts for 2D steady Oseen problem with different values of ν (exact solves).

Grid	n	m	$\nu = 0.1$	$\nu = 0.01$	$\nu = 0.001$	$\nu = 0.0001$
16×16	480	256	20	16	14	14
32×32	1,984	1,024	24	25	20	14
64×64	8,064	4,096	31	36	30	22
128×128	33,024	16,384	43	50	50	34
256×256	130,560	65,536	58	66	79	52

$w = 16x(x-1) + 16y(y-1)$ (2D case) and $\mathbf{w} = (-4z(1-2x),\ 4z(2y-1),\ 2y(1-y))$ (3D case). The 2D case corresponds to the choice of $\mathbf{v}$ used in [11]. The equations were discretized with a MAC scheme with a uniform mesh size h. The outer iteration (full GMRES) was stopped when

$$\frac{||r_k||_2}{||r_0||_2} < 10^{-6},$$

where r_k denotes the residual vector at step k. For the results presented in this section, the symmetric positive definite systems (4.5) were solved "exactly" by means of the sparse Cholesky factorization available in MATLAB, in combination with an approximate minimum degree ordering [1] to reduce fill-in. For the sparse, nonsymmetric Schur complement system (4.8) we used the sparse LU solver available in MATLAB with the original (lexicographic) ordering. We found this to be faster than a minimum degree ordering, probably because the need for pivoting makes the fill-reducing ordering ineffective or even harmful. We defer a discussion of the effect of inexact solves to section 5.3.

When working with the usual (convection) form of the Oseen problem, it is well known that care must be taken to avoid potential instabilities in the discrete velocities, either by taking h sufficiently small (compared to ν and $\|\mathbf{v}\|_\infty$), or by adding suitable stabilization terms to the discrete equations. Velocity stabilization is an issue for the rotation form of the equations as well; see [20, 22] for a discussion within the finite element context. For the experiments in this section we ignore the issue of velocity stabilization and assume that a sufficiently fine mesh is used; the same point of view is taken in [12, Chapter 7.3]. Whenever we present results on a sequence of grids, we do so only as a means of investigating the scalability of the linear solver as $h \to 0$, assuming that the finest mesh is sufficiently fine for the computed solution to be meaningful. This has indeed been verified by using test problems with a known solution.

First we consider a 2D steady ($\sigma = 0$) Oseen-type problem. In Table 5.1 we report iteration counts for a sequence of increasingly refined grids and for different values of the viscosity ν. The number of velocity and pressure unknowns (n and m, respectively) is also included. A value of the parameter α close to the optimal one was used in all cases. We found that for steady problems, the best value of α is of the form $\alpha = ch$, where c is independent of h but depends (weakly) on ν. A good rule of thumb is $c = -4 \log_{10} \nu$; that is, $c = 4$ for $\nu = 0.1$, and $c = 16$ for $\nu = 0.0001$. Thus, it is possible to determine a good choice of the shift parameter in this way: first one finds the optimal α by solving a few small problems on a coarse grid with mesh size h; then the optimal value of α for a fine grid with mesh size h/ℓ can be estimated by dividing the value of α by ℓ. We note that for every problem we have solved, there is a unique optimal value of α.

TABLE 5.2
Results for 2D unsteady Oseen problem with $\sigma = 40$ and different values of ν (exact solves).

Grid	$\nu = 0.1$	$\nu = 0.01$	$\nu = 0.001$	$\nu = 0.0001$
16×16	16	17	17	16
32×32	16	17	17	17
64×64	22	21	21	17
128×128	30	21	20	16
256×256	38	23	22	16

TABLE 5.3
Iteration count as a function of α for 2D Oseen problem, $\sigma = 40$, $\nu = 0.001$, 256×256 grid.

α	0.01	0.05	0.08	0.10	0.20	0.25	0.50	1.00
Its	65	32	27	25	22	22	28	50

TABLE 5.4
Iteration count for 3D generalized Oseen problem (exact solves, $\alpha = 0.1$).

σ	$\nu = 0.1$	$\nu = 0.01$	$\nu = 0.001$	$\nu = 0.0001$
1	34	31	32	26
10	25	22	22	23
20	24	22	22	23
50	23	22	23	24
100	24	24	24	24

It is clear from the results in Table 5.1 that for this set of (steady) problems, the rate of convergence with HSS preconditioning suffers from some degradation as $h \to 0$; on the other hand, the convergence in all cases is pretty fast and tends to improve (albeit not monotonically) as ν becomes smaller. The robustness of HSS preconditioning with respect to ν has been observed in all the tests we performed, and in our opinion this is enough to justify interest in this preconditioning technique.

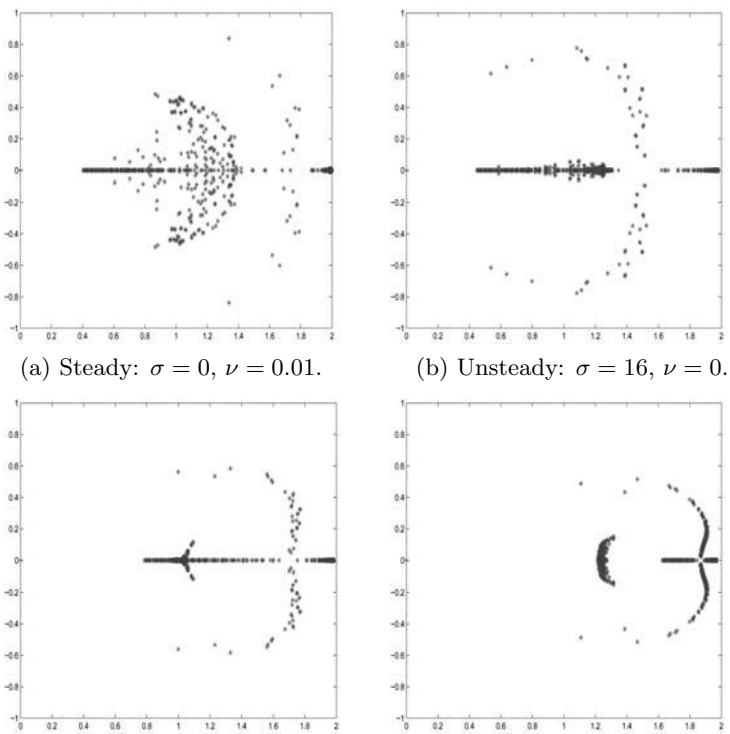
In Table 5.2 we present results for a quasi-steady problem with $\sigma = 40$ and $\nu = 0.001$. We can see from this table that HSS preconditioning results in fast convergence in all cases, and that the rate of convergence is virtually h-independent, especially for $\nu \leq 0.01$. Here as in all other unsteady (or quasi-steady) problems that we have tested, the rate of convergence is not overly sensitive to the choice of α, especially for small ν. A good choice is $\alpha \approx 0.5$ for the two coarser grids, and $\alpha \approx 0.25$ for the finer grids. Again, the optimal value of α is a few times larger for $\nu = 0.0001$ than for $\nu = 0.1$. The behavior of the iteration count as a function of α is displayed in Table 5.3 for the case $\nu = 0.001$ on the 256×256 grid.

Table 5.4 shows iteration counts for a set of 3D generalized Oseen problems on a $40 \times 40 \times 40$ grid, for several values of the parameter σ. No attempt was made to find the optimal value of α for each problem; instead, the same value $\alpha = 0.1$ was used in all cases. The discrete problem has $n = 187,200$ and $m = 64,000$, for a total of $N = 251,200$ unknowns. It can be seen that the iteration counts are remarkably stable. When the optimal value of α is used, the rate of convergence improves for increasing σ and decreasing ν.

Virtually optimal (i.e., grid-independent) rates of convergence are obtained when σ is not held constant but is taken to be inversely proportional to h, as in the case of genuinely unsteady flow problems; see [25]. To show this, we use a set of 3D, unsteady Oseen problems with $\nu = 0.001$ and $\sigma = h^{-1}$. The results for HSS preconditioning (with the fixed value $\alpha = 0.5$) are reported in Table 5.5. The total number

TABLE 5.5
Results for 3D unsteady Oseen problem with $\sigma = h^{-1}$, $\nu = 0.001$ (exact solves, $\alpha = 0.5$).

Grid	n	m	Iterations	CPU time
$16 \times 16 \times 16$	11,520	4,096	12	0.80
$32 \times 32 \times 32$	95,232	32,768	13	9.88
$64 \times 64 \times 64$	774,144	262,144	15	155.
$80 \times 80 \times 80$	1,516,800	512,000	16	563.



(a) Steady: $\sigma = 0$, $\nu = 0.01$. (b) Unsteady: $\sigma = 16$, $\nu = 0.1$.

(c) Unsteady: $\sigma = 16$, $\nu = 0.01$. (d) Unsteady: $\sigma = 16$, $\nu = 0.001$.

FIG. 5.1. *Eigenvalues of discrete generalized Oseen problem preconditioned with HSS.*

of unknowns ranges from 15,616 for the coarsest grid to 2,028,800 for the finest one.

These experiments show the good behavior of HSS preconditioning on Oseen-type problems, in particular for unsteady or quasi-steady problems. The performance is especially good for small values of the viscosity ν. This is in contrast to most other solvers, which tend to deteriorate for small values of ν.

Figures 5.1(a)–(d) display the eigenvalues of the HSS-preconditioned discrete Oseen operator on a 16×16 grid for different values of σ and ν. As predicted by the theory in [4, 19], the eigenvalues lie in the interior of the disk of center $(1, 0)$ and radius 1, except for the zero eigenvalue corresponding to the one-dimensional null space of the discrete gradient B^T. Since this lone zero eigenvalue has no effect on the convergence of preconditioned GMRES, we did not remove it by imposing (2.5) or other auxiliary conditions; see the discussion in [12, Chapter 2.3]. Note that the clustering of the spectrum away from the zero eigenvalue becomes more pronounced as ν gets smaller.

TABLE 5.6
Results for 2D steady Stokes problem with $\sigma = 0$, $\nu = 1$ (exact solves).

Grid	Iterations	CPU time
16×16	30	0.13
32×32	39	0.96
64×64	49	8.33
128×128	62	79.7
256×256	81	813.

TABLE 5.7
Results for 2D unsteady Stokes problem with $\sigma = 40$, $\nu = 0.001$ (exact solves).

Grid	Iterations	CPU time
16×16	8	0.06
32×32	9	0.32
64×64	11	2.63
128×128	15	26.6
256×256	20	292.

TABLE 5.8
Results for 3D steady Stokes problem with $\sigma = 0$, $\nu = 1$ (exact solves).

Grid	Iterations	CPU time
$16 \times 16 \times 16$	28	1.15
$32 \times 32 \times 32$	35	16.3
$64 \times 64 \times 64$	45	246.
$80 \times 80 \times 80$	51	772.

5.2. Stokes flow. Here we consider the generalized Stokes problem on the unit square (or unit cube in three dimensions). The boundary conditions, discretization scheme, and GMRES stopping criterion are the same as in the previous section. HSS preconditioning was carried out with "exact" solves using MATLAB's sparse Cholesky decomposition with an approximate minimum degree reordering; inexact solves are considered in section 5.3.

For the first two sets of experiments, we fix σ and ν and investigate the behavior of the preconditioned iteration for different values of h for a 2D problem. Iteration counts and CPU times (in seconds) are reported in Table 5.6 for $\sigma = 0$ and $\nu = 1$ and in Table 5.7 for $\sigma = 40$ and $\nu = 0.001$. For the steady ($\sigma = 0$) problem the best value of α depends on h; our rule of thumb is to take $\alpha \approx 5h$. The constant value $\alpha = 0.25$ was used for the unsteady problem. As for the Oseen-type case, our experience is that for unsteady problems the best value of α does not depend on h; hence, it can be determined for a coarse grid and used on finer grids with good results. Overall, the rate of convergence is not overly sensitive to the choice of α.

These results show that the rate of convergence with HSS preconditioning suffers some deterioration as $h \to 0$. On the other hand, while not h-independent, the rate convergence is quite good, especially for $\sigma > 0$. (A Fourier analysis of HSS preconditioning for the Stokes case will be presented elsewhere.)

Next we consider a steady 3D Stokes problem. Here we found that a good rule of thumb for the choice of the HSS parameter is $\alpha \approx 2h$. The results are shown in Table 5.8. Again, the convergence rate is reasonable, but not independent of the mesh size.

As in the 2D case, the performance of HSS preconditioning tends to improve as ν gets smaller and as σ gets larger. Table 5.9 shows iteration counts for a set of 3D generalized Stokes problems on a $40 \times 40 \times 40$ grid, for several values of the

TABLE 5.9
Iteration count for 3D generalized Stokes problem (exact solves).

σ	$\nu = 0.1$	$\nu = 0.01$	$\nu = 0.001$	$\nu = 10^{-6}$
1	45	27	16	13
10	32	19	15	12
20	30	18	14	11
50	28	15	13	11
100	25	14	12	10

TABLE 5.10
Results for 3D unsteady Stokes problem with $\sigma = h^{-1}$, $\nu = 0.001$ (exact solves).

Grid	Iterations	CPU time
$16 \times 16 \times 16$	12	0.78
$32 \times 32 \times 32$	12	8.78
$64 \times 64 \times 64$	12	126.
$80 \times 80 \times 80$	14	455.

parameter σ. The fixed value $\alpha = 0.5$ was used in all cases. The discrete problem has $n = 187,200$ and $m = 64,000$, for a total of $N = 251,200$ unknowns. It can be seen that the iteration counts improve for increasing σ and decreasing ν.

Optimal (i.e., grid-independent) rates of convergence are obtained when σ is not held constant but is taken to be inversely proportional to h (see, e.g., [25]). To show this, we use a set of 3D, unsteady Stokes problems with $\nu = 0.001$ and $\sigma = h^{-1}$. The results for HSS preconditioning (with the fixed value $\alpha = 0.5$) are reported in Table 5.10.

These experiments show the excellent behavior of HSS preconditioning for nonstationary Stokes problems, especially for small values of the viscosity ν and for a wide range of values of σ. On the other hand, the behavior of the preconditioner is suboptimal for steady Stokes problems (as can be verified by Fourier analysis).

It is interesting to note that for the Stokes problem the HSS-preconditioned matrix is diagonalizable and all its eigenvalues are real (and, of course, positive). Hence, there is a nonstandard inner product in which the preconditioned matrix is symmetric positive definite; see [6].

5.3. The effect of inexact solves. In all the results presented so far we have always used "exact" solves (in the form of sparse direct factorizations) for the linear systems that arise in the application of HSS preconditioning. As is well known, the storage and arithmetic complexity of direct solvers scales superlinearly with problem size when they are applied to linear systems arising from the discretization of elliptic PDEs; even with the best fill-reducing orderings currently available, for 3D problems their cost becomes quickly prohibitive as h is decreased. The poor scaling of sparse direct solvers is largely responsible for the superlinear scaling observed in the solution timings reported in the previous two sections. Other factors include (possibly) the nonconstant number of iterations as well as memory traffic issues.

Faster solution and better scalability may be achieved if the exact solves are replaced by inexact ones using inner iterations. We recall that for the (generalized) Stokes problem, all the inner solves correspond to shifted Poisson-type equations, which can be (approximately) solved efficiently by standard multigrid or by preconditioned conjugate gradients. For the Oseen problem, two (or three in three dimensions) of the systems are shifted Poisson-type equations, and one corresponds to a nonsymmetric but strongly elliptic problem. The latter system can be solved by

TABLE 5.11
Inexact versus exact solves, 2D unsteady Stokes problem with $\sigma = h^{-1}$, $\nu = 0.001$.

	Inexact				Exact			
Grid	set-up	Solve	Total	Its	set-up	Solve	Total	Its
32×32	0.04	0.18	0.22	13	0.05	0.10	0.15	13
64×64	0.19	0.94	1.13	16	0.20	0.51	0.71	14
128×128	1.24	5.60	6.84	18	1.07	2.74	3.81	14
256×256	10.4	29.0	39.4	19	8.10	13.5	21.6	15
512×512	106.	192.	298.	23	87.8	74.7	162.	16

TABLE 5.12
Inexact versus exact solves, 2D unsteady Oseen problem with $\sigma = h^{-1}$, $\nu = 0.001$.

	Inexact				Exact			
Grid	set-up	Solve	Total	Its	set-up	Solve	Total	Its
32×32	0.05	0.33	0.38	13	0.05	0.11	0.16	13
64×64	0.25	1.72	1.97	13	0.34	0.52	0.86	13
128×128	2.25	8.63	10.8	16	2.96	3.18	6.14	16
256×256	24.1	46.2	70.3	20	32.3	20.3	52.6	20
512×512	206.	343.	549.	22	503.	129.	632.	22

preconditioned GMRES or by a suitable multigrid method.

In Tables 5.11–5.12 we compare results for HSS-preconditioned FGMRES with inexact solves with those for HSS-preconditioned GMRES with exact solves. The test problems are unsteady 2D Stokes and Oseen problems with $\nu = 0.001$. We used drop tolerance-based incomplete Cholesky factorization preconditioning, available in MATLAB (function `cholinc`), for the two linear systems in (4.5). For (4.8) we used incomplete Cholesky in the Stokes case (where the Schur complement is symmetric and positive definite; see (4.9)) and incomplete LU (function `luinc`) in the Oseen case.

The drop tolerance τ in the incomplete factorizations was chosen so as to ensure that just 1–2 inner PCG iterations sufficed to reduce the inner residuals by at least a factor of 10. This low accuracy is already enough to ensure that the total number of (outer) FGMRES iterations is nearly the same as that of GMRES with exact solves. The use of incomplete factorizations leads to a reduction in memory requirements for the triangular factors, but this is partially offset by the memory overhead induced by the use of a flexible outer iteration. For the two shifted Laplacians in (4.5), which are very well conditioned, we used $\tau = 10^{-2}$. For the Schur complement system (4.9), in the Stokes case we used $\tau = 10^{-3}$ for all grids, except for the 512×512 case, where we used $\tau = 10^{-4}$. The 512×512 problem has a total of $N = 785,408$ unknowns. For the Schur complement in the Oseen problem we set the drop tolerance to 10^{-4} on all grids except for the 512×512 case, where we used $\tau = 10^{-5}$. Larger drop tolerances can be used at the expense of more inner iterations, and this tends to increase solution times.

The first observation to be made is that the use of inexact solves does not lead to a strong increase in the number of outer (FGMRES) iterations, even when a very loose tolerance ($tol = 10^{-1}$) is used for the inner iterations. This is especially true for the Oseen problem, for which the number of outer iterations is exactly the same as with exact solves.

Second, it appears from the results in Tables 5.11–5.12 that replacing the exact inner solves with inexact ones does not typically result in faster solution and may actually lead to higher computing times. Indeed, even for the set-up phase the

advantages of using incomplete factorizations instead of complete ones are small or nonexistent. Inexact solves are clearly counterproductive for the Stokes problem. The situation is somewhat better for the Oseen problem; however, the only case where inexact solves resulted in faster time to solution is in the case of the 512×512 grid. Note that in this case, the savings are entirely a result of reduced set-up costs. It should be kept in mind that the incomplete factorizations and inner iterations used are not very efficient. For the complete factorizations, the MATLAB code makes use of highly optimized sparse direct solvers. Also, the matrices are first reordered with an approximate minimum degree heuristic, which results in relatively low fill-in. In contrast, the implementations of the incomplete Cholesky and incomplete LU factorization functions in MATLAB are not very efficient. Furthermore, minimum degree reordering of the equations tends to have an adverse effect on the rate of convergence of the inner iterations; hence, no reordering was used when computing the incomplete factorizations. Because of this, the incomplete factors are not much cheaper to compute than the complete factors, and the additional costs induced by the (few) inner Krylov subspace iterations needed to satisfy the convergence criteria for the inexact solves lead in almost all cases to increased overall solution costs compared to the case of exact solves.

Nevertheless, the fact that HSS preconditioning does not significantly deteriorate in the presence of inexact solves is encouraging. We expect that with a more optimized implementation, or with the use of more efficient solvers for elliptic problems (i.e., multigrid schemes) it will be possible to achieve faster solutions and better scalability than with exact solves, at least for sufficiently large problems.

Finally, we note that similar conclusions hold for the case of steady problems and problems in three dimensions.

5.4. Comparison with other methods. Extensive experimental comparisons with other solvers have been carried out in [19]. We give here a brief summary of our findings.

To our knowledge, the only other solver explicitly developed for the rotation form of the Navier–Stokes equations is Olshanskii's block triangular preconditioner; see, e.g., [21] or the brief discussion in section 4.2. This preconditioner is very effective for the solution of unsteady Oseen-type problems. For steady problems, the exact variant of the preconditioner appears to be both h- and ν-independent; when inexact inner solves are used, however, a fairly high number of inner iterations may be needed to keep the number of outer iterations from growing as ν becomes smaller.

A possible drawback of this approach is the need to solve linear systems with coefficient matrix A at each iteration. In this system, all components of the velocities are coupled. Standard sparse solvers (direct or iterative) do not perform well on such systems; instead, specialized multigrid solvers have to be used; see [24]. The work involved in the inversion of the (2,2) block is comparable to that required by the Schur complement solve in HSS preconditioning; note, however, that in the case of HSS the eigenvalues of the Schur complement are shifted away from zero (see (4.8)). The diagonal shift in (4.8) helps making the problem arising in HSS preconditioning better conditioned.

Summarizing, for many problems Olshanskii's preconditioner is about as effective as or better than HSS preconditioning on Oseen-type problems in terms of iteration counts. We do not have an efficient implementation of Olshanskii's preconditioner in MATLAB, and therefore we do not have a way to compare timings. The main advantage of HSS preconditioning, in our opinion, is the greater ease of implemen-

tation. Furthermore, the linear systems that needs to be solved with HSS have a more convenient structure and better conditioning than the systems to be solved with Olshanskii's preconditioner, so the cost per iteration is generally lower.

Numerical experiments were also performed comparing HSS with Uzawa-type and SIMPLE preconditioning; see [19]. These techniques exhibit significant deterioration for small ν and were found to be generally inferior to both HSS and Olshanskii's preconditioner.

Concerning the Stokes problem, both steady and generalized, there are many optimal solvers available. In particular, the steady case is best handled by block diagonal preconditioners of the form

$$\mathcal{P} = \begin{bmatrix} \hat{A} & O \\ O & \hat{S} \end{bmatrix}, \tag{5.1}$$

where $\hat{A}$ is spectrally equivalent to A and $\hat{S}$ is spectrally equivalent to $S = BA^{-1}B^T$; see [32]. When using LBB-stable discretizations, finding spectrally equivalent approximations for the Schur complement S is easy: the pressure mass matrix (or a diagonal approximation of it) will do; for MAC, it is enough to take $S = I_m$. We performed some experiments with exact solves for the (1,1) block (i.e., $\hat{A} = A$) and we verified the expected h-independent convergence. The number of iterations for the finer grids settled around 20 in the 2D case. Since this preconditioner is cheaper than HSS, requiring only two Poisson-type solves rather than three at each step, we conclude that HSS cannot be recommended for the [illegible] Stokes problem, since its rate of convergence is not h-independent.

For generalized Stokes problems (case $\sigma > 0$), on the other hand, HSS offers an attractive alternative to existing preconditioners (see [8, 17, 25]). The rates of convergence obtained with HSS and its cost per iteration appear to be competitive with those that have been reported in the literature; furthermore, implementation of the HSS preconditioner is straightforward.

5.5. Behavior of the Picard iteration. An important question that has not been addressed so far in this paper is that of the rate of convergence of the nonlinear Picard iteration for the rotation form of the Navier–Stokes equations. For the standard (convection) form of the steady Navier–Stokes equations, conditions for the (global) convergence of the Picard iteration have been given in [16]. We are not aware of any such analysis for the rotation form of the equations.

Limited numerical experience reported by Olshanskii in [23] suggests that the rate of convergence of Picard's iteration for the rotation form tends to deteriorate for decreasing values of the viscosity ν. Although some deterioration is also observed for the convective form of the equations, the phenomenon appears to be more pronounced for the rotation form. This is probably the price one pays for the avoidance of convection in the linear solves.

Our own experience is in agreement with the results reported by Olshanskii. In particular, we found that Picard's iteration for the rotation form tends to converge fast for relatively large ν (say, between 2 and 6 iterations for $\nu > 0.05$ in a 2D confined flow problem); however, convergence slows down dramatically for smaller ν (for example, 68 iterations for $\nu = 0.03$), and the iteration fails to converge for smaller values of ν.

In contrast, Picard's iteration was found to converge when using the convective form of the equations, even for small values of the viscosity. The number of iterations for solving the same confined flow problem was found to be 4, 18, and 80 for $\nu = 0.1$,

TABLE 5.13
Picard iterations for rotation form with a regularization term $\xi h^2 I$.

Problem	$\nu = 0.1$	$\nu = 0.01$	$\nu = 0.001$
32×32, $\xi = 10$	3	4	6
error	6.21e-04	5.81e-04	5.79e-04
64×64, $\xi = 12$	3	4	6
error	1.49e-04	1.38e-05	1.36e-04

0.01, and 0.001, respectively. All these results are essentially independent of the mesh size, at least for fine enough grids. In these experiments, convergence of the Picard iteration is achieved when the 2-norm of the (nonlinear) residual is reduced below $tol = 10^{-5}$ times the initial residual (see (8.34) in [12]), starting from a zero initial guess. Exact solves were used at each Picard iteration.

Clearly, this is a serious drawback of the rotation form for steady problems. We found, however, that the convergence of Picard's iteration can be greatly improved by adding to the momentum equations (2.13) a term of the form $\xi \mathbf{u}$, where $\xi > 0$ is a moderate constant. Upon discretization, this amounts to the addition of a term of the form $\xi h^2 I$ to the (1,1) block of the system, and can be regarded as a form of [illegible]. Note that this modification changes the solution of the discrete problem; however, the extra term is small and tends to zero as $h \to 0$. The modification can also be interpreted as a kind of (velocity) stabilization. Although we do not have a rigorous justification for this form of regularization, we mention that the technique is similar to the one used in [13] for saddle point problems arising from the discretization of the curl-curl form of Maxwell's equations.

Nonlinear iteration counts for the modified problem on two grids with two different values of ξ are reported in Table 5.13. We also include the discrete L^2-norm of the difference between the velocity field obtained by Picard's iteration on the modified problem and the true (analytical) solution of our test problem. This error is very small in all cases. However, how to choose ξ remains an open question. Choosing too small a value of ξ leads to slow convergence for very small ν; for example, on the 32×32 grid, using $\xi = 7$ results in 3, 6, and 45 iterations for $\nu = 0.1$, 0.01, and 0.001, respectively. On the other hand, choosing ξ too large could in principle lead to a loss of accuracy in the computed solution. In our (admittedly somewhat limited) experience, however, somewhat overestimating ξ does not lead to a noticeable worsening of the error.

6. Conclusions. In this paper we have described a preconditioning technique for the large sparse linear systems arising from discretizations of the Navier–Stokes equations in rotation form. The basic idea is to decouple the symmetric and skew-symmetric components of the linearized Navier–Stokes system and to alternate between the two. The preconditioner contains a parameter which requires some tuning for optimal performance; we described some heuristics that can be used to find good estimates for the optimal parameter. However, in most cases the performance is not overly sensitive to the value of this parameter. The preconditioner can also be interpreted as an approximate block factorization of the (global) system matrix which becomes increasingly accurate in the vanishing viscosity limit $\nu \to 0$.

Implementation of the preconditioner is straightforward (more so than with other methods), and the numerical experiments show good performance on a wide range of cases, especially for unsteady problems and for low-viscosity problems. While the preconditioner cannot be recommended for steady Stokes problems, it performs very

well in the case of quasi-steady and unsteady problems. We remark that these appear to be precisely the kind of problems for which the rotation form is best suited; see [27].

Although our computational experience has been limited to uniform MAC discretizations and simple geometries, the preconditioner should be applicable to more complicated problems and discretizations, including unstructured grids.

Finally, although we have not discussed parallelization issues in this paper, the preconditioner can easily take advantage of existing elliptic solvers for parallel platforms, such as domain decomposition techniques.

Acknowledgments. The authors would like to thank Maxim Olshanskii and Eldad Haber for helpful suggestions.

REFERENCES

[1] P. R. Amestoy, T. A. Davis, and I. S. Duff, *An approximate minimum degree ordering algorithm*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 886–905.

[2] Z.-Z. Bai, G. H. Golub, and M. K. Ng, *Hermitian and skew-Hermitian splitting methods for non-Hermitian positive definite linear systems*, SIAM J. Matrix Anal. Appl., 24 (2003), pp. 603–626.

[3] M. Benzi, M. J. Gander, and G. H. Golub, *Optimization of the Hermitian and skew-Hermitian splitting iteration for saddle-point problems*, BIT, 43 (2003), pp. 881–900.

[4] M. Benzi and G. H. Golub, *A preconditioner for generalized saddle point problems*, SIAM J. Matrix Anal. Appl., 26 (2004), pp. 20–41.

[5] M. Benzi, G. H. Golub, and J. Liesen, *Numerical solution of saddle point problems*, Acta Numer., 14 (2005), pp. 1–137.

[6] M. Benzi and V. Simoncini, *On the eigenvalues of a class of saddle point matrices*, Numer. Math., 103 (2006), pp. 173–196.

[7] M. Benzi and D. B. Szyld, *Existence and uniqueness of splittings for stationary iterative methods with applications to alternating methods*, Numer. Math., 76 (1997), pp. 309–321.

[8] J. Cahouet and J. P. Chabard, *Some fast 3-D finite element solvers for the generalized Stokes problem*, Internat. J. Numer. Methods Fluids, 8 (1988), pp. 869–895.

[9] W. Chang, F. Giraldo, and J. B. Perot, *Analysis of an exact fractional step method*, J. Comput. Phys., 180 (2002), pp. 183–199.

[10] H. C. Elman, *Preconditioning for the steady-state Navier–Stokes equations with low viscosity*, SIAM J. Sci. Comput., 20 (1999), pp. 1299–1316.

[11] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Performance and analysis of saddle point preconditioners for the discrete steady-state Navier–Stokes equations*, Numer. Math., 90 (2002), pp. 665–688.

[12] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics*, Numer. Math. Sci. Comput., Oxford University Press, Oxford, UK, 2005.

[13] C. Greif and D. Schötzau, *Preconditioners for the discretized time-harmonic Maxwell equations in mixed form*, Numer. Linear Algebra Appl., 14 (2007), pp. 281–297.

[14] P. M. Gresho, *Incompressible fluid dynamics: Some fundamental formulation issues*, Ann. Rev. Fluid Mech., 23 (1991), pp. 413–453.

[15] F. H. Harlow and J. E. Welch, *Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface*, Phys. Fluids, 8 (1965), pp. 2182–2189.

[16] O. A. Karakashian, *On a Galerkin–Lagrange multiplier method for the stationary Navier–Stokes equations*, SIAM J. Numer. Anal., 19 (1982), pp. 909–923.

[17] G. M. Kobelkov and M. A. Olshanskii, *Effective preconditioning of Uzawa type scheme for a generalized Stokes problem*, Numer. Math., 86 (2000), pp. 443–470.

[18] C. Li and C. Vuik, *Eigenvalue analysis of the SIMPLE preconditioning for incompressible flow*, Numer. Linear Algebra Appl., 11 (2004), pp. 511–523.

[19] J. Liu, *Preconditioned Krylov Subspace Methods for Incompressible Flow Problems*, Ph.D. thesis, Department of Mathematics and Computer Science, Emory University, Atlanta, 2006; available online at http://www.uwf.edu/jliu/research.htm.

[20] G. Lube and M. A. Olshanskii, *Stable finite element calculation of incompressible flows using the rotation form of convection*, IMA J. Numer. Anal., 22 (2002), pp. 437–461.

[21] M. A. Olshanskii, *An iterative solver for the Oseen problem and numerical solution of incompressible Navier–Stokes equations*, Numer. Linear Algebra Appl., 6 (1999), pp. 353–378.
[22] M. A. Olshanskii, *A low order Galerkin finite element method for the Navier–Stokes equations of steady incompressible flow: A stabilization issue and iterative methods*, Comput. Methods Appl. Mech. Engrg., 191 (2002), pp. 5515–5536.
[23] M. A. Olshanskii, *Preconditioned iterations for the linearized Navier–Stokes system in rotation form*, in Proceedings of the Second MIT Conference on Computational Fluid and Solid Mechanics, K. J. Bathe, ed., Elsevier, New York, 2003, pp. 1074–1077.
[24] M. A. Olshanskii and A. Reusken, *Navier–Stokes equations in rotation form: A robust multigrid solver for the velocity problem*, SIAM J. Sci. Comput., 23 (2002), pp. 1683–1706.
[25] J. Peters, V. Reichelt, and A. Reusken, *Fast iterative solvers for discrete Stokes equations*, SIAM J. Sci. Comput., 27 (2005), pp. 646–666.
[26] A. Quarteroni, F. Saleri, and A. Veneziani, *Factorization methods for the numerical approximation of Navier-Stokes equations*, Comput. Methods Appl. Mech. Engrg., 188 (2000), pp. 505–526.
[27] L. G. Rebholz, *An energy- and helicity-conserving finite element scheme for the Navier–Stokes equations*, SIAM J. Numer. Anal., 45 (2007), pp. 1622–1638.
[28] Y. Saad, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM J. Sci. Comput., 14 (1993), pp. 461–469.
[29] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.
[30] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.
[31] F. Saleri and A. Veneziani, *Pressure correction algebraic splitting methods for the incompressible Navier–Stokes equations*, SIAM J. Numer. Anal., 43 (2005), pp. 174–194.
[32] D. J. Silvester and A. J. Wathen, *Fast iterative solution of stabilised Stokes systems. Part* II: *Using general block preconditioners*, SIAM J. Numer. Anal., 31 (1994), pp. 1352–1367.
[33] V. Simoncini and M. Benzi, *Spectral properties of the Hermitian and skew-Hermitian splitting preconditioner for saddle point problems*, SIAM J. Matrix Anal. Appl., 26 (2004), pp. 377–389.
[34] H. A. van der Vorst, *Iterative Krylov Methods for Large Linear Systems*, Cambridge Monogr. Appl. Comput. Math. 13, Cambridge University Press, Cambridge, UK, 2003.
[35] A. Veneziani, *Block factorized preconditioners for high-order accurate in time approximation to the Navier–Stokes equations*, Numer. Methods Partial Differential Equations, 19 (2003), pp. 487–510.
[36] P. Wesseling, *Principles of Computational Fluid Dynamics*, Springer, Berlin, 2001.
[37] T. A. Zang, *On the rotation and skew-symmetric forms for incompressible flow simulations*, Appl. Numer. Math., 7 (1991), pp. 27–40.

SIAM J. SCI. COMPUT.
Vol. 29, No. 5, pp. 1982–1991
© 2007 Society for Industrial and Applied Mathematics

AN ALGEBRAIC MULTIGRID PRECONDITIONER FOR A CLASS OF SINGULAR M-MATRICES*

ELENA VIRNIK†

Abstract. We apply algebraic multigrid (AMG) as a preconditioner for solving large singular linear systems of the type $(I - T^T)x = 0$ with GMRES. Here, T is assumed to be the transition matrix of a Markov process. Although AMG and GMRES were originally designed for the solution of regular systems, with adequate adaptation their applicability can be extended to problems as described above.

Key words. algebraic multigrid, singular systems, Markov chains

AMS subject classifications. 65C20, 65N55, 65F10

DOI. 10.1137/060659272

1. Introduction. This paper treats the application of an algebraic multigrid (AMG) method [12] as a preconditioner for solving large singular linear systems of the type

$$Ax := (I - T^T)x = 0 \tag{1}$$

using the generalized minimal residual (GMRES) method [13]. Here, T is the transition matrix of a Markov process. The applicability of GMRES to singular systems was examined, e.g., in [1, 4]. Our AMG strategy is based on the classical theory in [12] originally designed for systems involving regular, symmetric, positive definite M-matrices. The approach in [12] was generalized to symmetric positive definite and semidefinite matrices without the M-matrix property, e.g., in [3, 5, 6, 11]. A multigrid-like algorithm based on aggregation and disaggregation of Markov chains can be found in [9]. The main result of this paper is the extension of the AMG approach to a class of singular nonsymmetric M-matrices.

The theory of Markov chains [2, 14] represents an extremely important tool that has a broad variety of applications not only in the sciences, such as biology, physics, and chemistry, but also in business and economics. As an example, we consider a Markov chain model for blood circulation in the human body.

EXAMPLE 1. Let us imagine the organs within a human body as states of a blood particle. Within one time step, it can advance from one organ to another with a certain probability. It is also possible that it stays within the organ and only continues its journey a few time steps later. On the other hand, an organ can also be considered as a set of substates, since the blood does not simply pass an organ but can also recirculate within it in some chaotic way. Hence, if we define a set of (sub)organs as states of the Markov chain and the corresponding transition probabilities, we get a large sparse (row) stochastic matrix. We are interested in the stationary distribution of the Markov chain, which would, e.g., represent the concentration of a dissolved substance after injection into an organ. For a more detailed description, see [16].

*Received by the editors May 8, 2006; accepted for publication (in revised form) January 18, 2007; published electronically September 28, 2007. This work was supported by the DFG Research Center MATHEON.

http://www.siam.org/journals/sisc/29-5/65927.html

†Technische Universität Berlin, Institut für Mathematik, Straße des 17. Juni 136, 10623 Berlin, Germany (virnik@math.tu-berlin.de).

2. Preliminaries. We call a vector $v \in \mathbb{R}^n$, and we write $v > 0$ if all entries v_i are positive. A matrix $T \in \mathbb{R}^{n\times n}$, $T = (t_{ij})_{i,j=1,\dots,n}$, is called , and we write $T > 0$ ($T \geq 0$) if all entries t_{ij} are positive (nonnegative). A matrix $T \in \mathbb{R}^{n\times n}$ is called if there exists a permutation matrix $P \in \mathbb{R}^{n\times n}$, such that $PTP^T = \left[\begin{smallmatrix} T_{11} & 0 \\ T_{21} & T_{22} \end{smallmatrix}\right]$. Otherwise it is called .

A scalar $\lambda \in \mathbb{R}$ is called an of the matrix $T \in \mathbb{R}^{n\times n}$ if a vector $v \in \mathbb{R}^n$, $v \neq 0$ exists, such that $Tv = \lambda v$. The vector v is called a of T associated with λ. Accordingly, a vector $w \in \mathbb{R}^n$, $w \neq 0$, with $w^T T = \lambda w^T$ is called a of T. Let $T \in \mathbb{R}^{n\times n}$ have the eigenvalues λ_i, $i = 1, \dots, n$. Then we call $\rho(T) = \max_{1\leq i\leq n} |\lambda_i|$ the T.

A process is called a if it has n states $s_1, \dots, s_n$ and the transition probability $P[s_i \rightsquigarrow s_j] =: t_{ij}$ is time-independent. The matrix $T = [t_{ij}]_{i,j=1,\dots,n}$ satisfies $t_{ij} \geq 0$ and $\sum_{j=1}^n t_{ij} = 1$ for $i, j = 1, \dots, n$, i.e., it is and is called the of a Markov chain. We denote by $x^k = (x_i^k)$ the , where x_i^k is the probability that the system is in state s_i after k steps. We have $x_i^k \geq 0$ and $\sum_{i=1}^n x_i^k = 1$ for each k. A distribution vector x is said to be if $x^T T = x^T$.

The well-known Perron–Frobenius theorem guarantees the existence and uniqueness of a stationary distribution.

THEOREM 1 (Perron–Frobenius theorem [2, p. 27]). $T \geq 0$ $\rho(T)$ $\rho(T)$ T $\rho(T)$

COROLLARY 2 (see [2, p. 28]). x T $\rho(T)$

COROLLARY 3. T

We define by $\mathcal{Z}^{n\times n} = \{A = [a_{ij}] \in \mathbb{R}^{n\times n} : a_{ij} \leq 0, i \neq j\}$ the set of all real matrices with nonpositive off-diagonal entries. Let $B \geq 0$ with spectral radius $\rho(B)$. A matrix A of the form $A = sI - B$, with $s > 0$ and $s \geq \rho(B)$, is called an . If $s > \rho(B)$, then A is a M-matrix, and if $s = \rho(B)$, then A is a M-matrix. Hence, in our case $A = (I - T^T)$ is a singular M-matrix.

3. AMG preconditioning. The GMRES method we use was introduced in [13]. Algorithm 1 shows a version with left preconditioning.

As we will see in the following, the application of AMG in our case requires the property that the matrix A have row sums zero. Since $A = (I - T^T)$ has column sums zero instead, for the construction of the preconditioner we use the matrix A^T and then use M^T for the iteration process.

3.1. Basic framework of AMG. The AMG method that is presented here follows the concept introduced in [12], which is motivated by geometric multigrid, where a sequence of grids is constructed from the underlying geometry with corresponding transfer operators between the grids. The main idea of geometric multigrid is to remove the smooth error, which cannot be eliminated by relaxation on the fine grid, by coarse grid correction. The solution process then as usual consists of presmoothing, transfer of residuals from fine to coarse grids, interpolation of corrections from coarse to fine levels, and optional postsmoothing. In contrast to geometric multigrid, the idea of AMG is to define an sequence of systems of equations decreasing in

ALGORITHM 1: Preconditioned MGS-GMRES.

Input: $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$, starting vector $x_0 \in \mathbb{R}^n$, preconditioner M
Output: $x_m \in \mathbb{R}^n$ solution approximate in the mth step

```
1  Compute r_0 = M^{-1}(b - Ax_0), β := ||r_0||_2 and v_1 := r_0/β
2  Define the (m + 1) × m matrix H̄_m = {h_ij}_{1≤i≤m+1,1≤j≤m}. Set H̄_m = 0.
3  for j = 1, 2, ..., m do
4  |  Compute w_j := M^{-1} A v_j
5  |  for i = 1, ..., j do
6  |  |  h_ij := (w_j, v_i)
7  |  |  w_j := w_j - h_ij v_i
8  |  h_{j+1,j} = ||w_j||_2. If h_{j+1,j} = 0, set m := j and go to 12
9  |  v_{j+1} = w_j / h_{j+1,j}
10 Compute y_m the minimizer of ||βe_1 - H̄_m y||_2
11 Set x_m = x_0 + V_m y_m.
```

size,

$$A^m u^m = b^m, \tag{2}$$

where the superscript m denotes the mth iterate, directly from the underlying matrix. We have $A^1 = A$, and for $m = 1$ the system (2) is identical to (1). We call these equations "coarse grid" equations. The interpolation operator P and the restriction operator R define the transfer from finer to coarser grids and vice versa. More precisely, let $A^m \in \mathbb{R}^{n_m \times n_m}$; then

$$R^m : \mathbb{R}^{n_m} \to \mathbb{R}^{n_{m+1}},$$
$$P^m : \mathbb{R}^{n_{m+1}} \to \mathbb{R}^{n_m},$$

and the operator on the coarser grid is defined by

$$A^{m+1} = R^m A^m P^m \ \in \mathbb{R}^{n_{m+1}\times n_{m+1}}. \tag{3}$$

Thus, we do not need a geometry behind the problem.

The AMG method consists of two main parts: the setup phase and the solution phase. During the setup phase, the coarse grids and the corresponding operators are defined. We describe this in section 3.2. The solution phase consists of a multilevel iteration. Here, as an example, we illustrate a two-level iteration, which involves the combination of a smoothing process (e.g., Gauss–Seidel; see [7]), with a correction on a coarser grid:

$$x^{(k+1)} = x^{(k)} + B^{-1}(b^m - A^m x^{(k)}), \tag{4a}$$
$$x^{(k+2)} = x^{(k+1)} + P^m (A^{m+1})^{-1} R^m (b^m - A^m x^{(k+1)}). \tag{4b}$$

Here, the matrix $(A^{k+1})^{-1}$ is replaced by the recursive application of (4) to the solution of the coarse grid system. The number of recursive calls, which is the number of levels m, depends on the size and structure of the matrix. For our method, we use the V-cycle pattern (see, e.g., [8]), as shown in Figure 1. Other recursive patterns, e.g., the W-cycle that involves two recursive calls per cycle, are presented in [8].

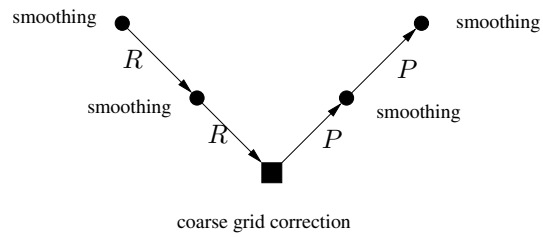


FIG. 1. *V-cycle pattern.*

3.2. Coarsening. The first main step of the coarsening process as introduced in [12] can be read as a permutation of the matrix $A \in \mathbb{R}^{n\times n}$ such that the fine grid nodes come first:

$$A \rightsquigarrow \pi^T A \pi = \left[\begin{array}{c|c} A_{FF} & A_{FC} \\ \hline A_{CF} & A_{CC} \end{array}\right].$$

We choose the fine grid nodes such that the block A_{FF} is close to a diagonal matrix. A further sparsification step is applied in which "small" off-diagonal entries in A_{FF} are added to the diagonal and the remaining off-diagonal entries in A_{FF} are redistributed onto the A_{FC} block:

$$A \rightsquigarrow \tilde{A} = \left[\begin{array}{c|c} \tilde{A}_{FF} & \tilde{A}_{FC} \\ \hline A_{CF} & A_{CC} \end{array}\right]. \tag{5}$$

It is shown in [12] that the "sparsification" process leaves the row sums of the matrix A unchanged.

To obtain the restriction and interpolation operators, we consider the incomplete block LU decomposition of the matrix $\tilde{A}$:

$$\tilde{A} = \left[\begin{array}{c|c} \tilde{A}_{FF} & \tilde{A}_{FC} \\ \hline A_{CF} & A_{CC} \end{array}\right] = \left[\begin{array}{c|c} I & 0 \\ \hline A_{CF}\tilde{A}_{FF}^{-1} & I \end{array}\right] \left[\begin{array}{c|c} \tilde{A}_{FF} & \tilde{A}_{FC} \\ \hline 0 & S \end{array}\right], \tag{6}$$

where $S = A_{CC} - A_{CF}\tilde{A}_{FF}^{-1}\tilde{A}_{FC}$ is the Schur complement with respect to $\tilde{A}_{FF}$. We take S to be the new coarse grid operator. This is the main difference from the classical approach in [12]. For the coarse grid operator we compute the exact Schur complement of an approximated matrix $\tilde{A}$, whereas in the classical approach an approximation to the Schur complement of the exact matrix A is computed. For other choices of restriction, interpolation, and coarse grid operators see, e.g., [5, 6, 12, 15]. Neither of these choices works here, since they do not preserve in our case important properties such as the singular M-matrix property and the row sums zero property; see section 3.3 for details.

The following lemma summarizes the main results for the coarsening process.

LEMMA 4. A (6) N S (6)

1.

$$R = \begin{bmatrix} -A_{CF}\tilde{A}_{FF}^{-1} & I \end{bmatrix} \in \mathbb{R}^{N\times n} \qquad P = \begin{bmatrix} -\tilde{A}_{FF}^{-1}\tilde{A}_{FC} \\ I \end{bmatrix} \in \mathbb{R}^{n\times N},$$

[illegible] S [illegible]

$$S = R\tilde{A}P \in \mathbb{R}^{N\times N}. \tag{7}$$

2. [illegible] P [illegible]
3. [illegible] S [illegible]

[illegible]

1. By straightforward calculation we get

$$\begin{aligned} R\tilde{A}P &= \begin{bmatrix} -A_{CF}\tilde{A}_{FF}^{-1} & I \end{bmatrix} \begin{bmatrix} \tilde{A}_{FF} & \tilde{A}_{FC} \\ A_{CF} & A_{CC} \end{bmatrix} \begin{bmatrix} -\tilde{A}_{FF}^{-1}\tilde{A}_{FC} \\ I \end{bmatrix} \\ &= \begin{bmatrix} -A_{CF}\tilde{A}_{FF}^{-1} & I \end{bmatrix} \begin{bmatrix} -\tilde{A}_{FC} + \tilde{A}_{FC} \\ -A_{CF}\tilde{A}_{FF}^{-1}\tilde{A}_{FC} + A_{CC} \end{bmatrix} \\ &= A_{CC} - A_{CF}\tilde{A}_{FF}^{-1}\tilde{A}_{FC} = S \in \mathbb{R}^{N\times N}. \end{aligned}$$

2. Let $\mathbb{1} = [1, \ldots, 1]^T$ represent the vector containing all entries equal to one. Hence, for all $i \in \{1, \ldots, N\}$ we have

$$0 = e_i^T \tilde{A}\mathbb{1} = \sum_{k\in\mathcal{C}_i} \tilde{a}_{ik} + \tilde{a}_{ii},$$

and thus, as all $\tilde{a}_{ik} \le 0$ for $i \ne k$, we get

$$\frac{\sum_{k\in\mathcal{C}_i} |\tilde{a}_{ik}|}{\tilde{a}_{ii}} = 1,$$

where $\mathcal{C}_i$ is the set of interpolatory connections of i, i.e., we sum up the entries in the ith row of the $\tilde{A}_{FC}$ block. Hence, all row sums of the interpolation operator P are equal to one.

3. From 2. we conclude that

$$e_i^T S\mathbb{1} = e_i^T R\tilde{A}P\mathbb{1} = e_i^T R\tilde{A}\mathbb{1} = 0 \quad \text{for all } i,$$

i.e., the zero row sums property is preserved for the new coarse grid operator S. □

3.3. Multilevel setup. For the setup of a recursive multilevel cycle, it is essential to ensure that the important properties of the finest grid operator carry over to all coarser grids. In our case it is on the one hand the row sums zero property and on the other hand the singular M-matrix property. These are important for an adequate interpolation of the smooth error that needs to be reduced by the coarse grid correction and can be written as

$$e_i \approx \sum_{j\ne i} \frac{|a_{ij}|}{a_{ii}} e_j, \quad \text{where } \sum_{j\ne i} \frac{|a_{ij}|}{a_{ii}} = 1.$$

Here, we can use only significant weights $|a_{ij}|/a_{ii}$ to interpolate e_i. Also the singular M-matrix property is important for the direct solution on the coarsest level; see section 3.5. In Lemma 4 we have shown that the coarsening process leaves the row sums zero. The following theorem states that the singular M-matrix property is preserved after the performance of one coarsening step.

THEOREM 5. [illegible] $A = \left[\begin{smallmatrix} A_{FF} & A_{FC} \\ A_{CF} & A_{CC} \end{smallmatrix}\right]$ [illegible] $\tilde{A} = \left[\begin{smallmatrix} \tilde{A}_{FF} & \tilde{A}_{FC} \\ A_{CF} & A_{CC} \end{smallmatrix}\right]$ [illegible] (5) [illegible] $S = A_{CC} - A_{CF}\tilde{A}_{FF}^{-1}\tilde{A}_{FC}$ [illegible] (7) [illegible]

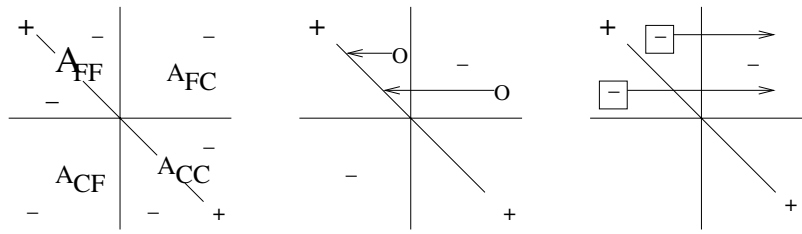


FIG. 2.

1. Show that $\tilde{A}$ is a singular M-matrix:
   A is a singular M-matrix. Therefore, it can be expressed in the form

$$A = sI - T, \qquad \text{where } T \text{ is the transition matrix of a Markov process}$$
$$\rho(T) = 1 = s.$$

   In [12] it was shown that the coarsening process leaves the row sums invariant, i.e., in our case equal to zero. Hence, we can write $\tilde{A} = I - \tilde{T}$, where $\tilde{T}$ has row sums one. Furthermore, we have $\tilde{A} \in \mathcal{Z}^{n\times n}$, i.e., the off-diagonal entries in $\tilde{A}$ are nonpositive and the diagonal is nonnegative (as otherwise the row sums could not be zero). The diagonal entries are also at most one. One can see this by considering the following sparsification steps (see [12] for a detailed description of the "sparsification" process):
   - The initial matrix A has a sign and block structure, as illustrated in Figure 2(a). The sign "−" indicates that all entries in the corresponding section are nonpositive. Accordingly, "+" signifies that the entries in the corresponding section are nonnegative.
   - In the first step, "small" off-diagonal entries in each row are added to the diagonal (see Figure 2(b)). The diagonal entries become smaller but stay nonnegative.
   - Then the remaining off-diagonal entries of the A_{FF}-block are distributed onto the A_{FC}-block (see Figure 2(c)). The entries of the A_{FC}-block become smaller, but remain greater than -1, as otherwise the row sums could not stay zero.

   Thus, $\tilde{T}$ is in addition nonnegative and all entries are at most one. From this we conclude that $\tilde{T}$ is stochastic and thus $\rho(\tilde{T}) = 1 = s$. Consequently, $\tilde{A}$ is a singular M-matrix.
2. Show that S as in (7) is a singular M-matrix:
   In Lemma 4 we have shown that $S\mathbb{1} = 0$, i.e., S has zero row sums. From this and from $A_{CC} \in \mathcal{Z}^{N\times N}$ and $A_{CF}\tilde{A}_{FF}^{-1}\tilde{A}_{FC} \geq 0$, we can conclude that $S = A_{CC} - A_{CF}\tilde{A}_{FF}^{-1}\tilde{A}_{FC} \in \mathcal{Z}^{N\times N}$. Let S now have the representation

$$S = sI - \hat{T}, \qquad \hat{T} \geq 0, \quad s > 0.$$

   Then we get

$$S\mathbb{1} = (sI - \hat{T})\mathbb{1} = s\mathbb{1} - \hat{T}\mathbb{1} = 0$$
$$\Rightarrow \hat{T}\mathbb{1} = s\mathbb{1}.$$

As $\mathbb{1} > 0$ is a positive eigenvector of $\hat{T}$, it follows from Corollary 2 that $s = \rho(\hat{T})$. Hence, S is a singular M-matrix. □

From Theorem 5 we conclude that the singular M-matrix property is preserved on all coarser levels.

3.4. Integration of the side constraint into the solution process. For a badly chosen starting vector x_0 the GMRES method in Algorithm 1 may converge to the trivial solution, instead of the stationary distribution. In fact, for some problems we cannot choose the starting vector randomly but need to employ a specifically fixed starting vector as, e.g., in the case of the blood circulation model. Here, the initial distribution might represent an injection into a certain organ. Therefore, it makes sense to set $x_0 = [0, \ldots, 1, \ldots, 0]^T$, where the only nonzero entry $[x_0]_i = 1$ is positioned to represent the injection into the organ i. In this case the naive application of the preconditioned GMRES algorithm leads to a convergence to the trivial solution.

Hence, we have the side constraint $\mathbb{1}^T x = 1$ that we have to include into the solution process. The idea now is to shift the system such that the trivial solution is excluded as a possible solution. Consider the orthogonal projection matrix $Q := I - \frac{1}{n}\mathbb{1}\mathbb{1}^T$ with respect to $\mathbb{1}$, that is, $\mathbb{1}^T Q = 0$. Any vector x can be decomposed into

$$x := Qy + z$$

such that $z = \alpha\mathbb{1}$ for some constant $\alpha \in \mathbb{R}$. In particular, for a vector $\tilde{x}$ that fulfills $\mathbb{1}^T\tilde{x} = 1$, we obtain $z := \frac{1}{n}\mathbb{1}$.

Now we transform the initial problem as follows:

$$\begin{aligned} & Ax = 0 \\ \Leftrightarrow\ & A(Qy + z) = 0 \\ \Leftrightarrow\ & AQy = -\frac{1}{n}A\mathbb{1}. \end{aligned} \tag{8}$$

By setting $\hat{A} := AQ$ and $b := -\frac{1}{n}A\mathbb{1}$, we get a formulation of the linear system where the side constraint is embedded:

$$\hat{A}y = b. \tag{9}$$

Now, we can solve this system instead of (8) using AMG preconditioned GMRES. The system is still consistent since $b \in \mathrm{im}(A)$. As Q is orthogonal with respect to z, we have $\tilde{x} = Qy + z \neq 0$ for all y.

Note that in the implementation we do not compute the product $\hat{A} := AQ$ explicitly as Q is not sparse and matrix-matrix products are expensive, but rather successively calculate the corresponding matrix-vector products.

3.5. Direct solution on the coarsest level. On the coarsest level we solve the singular system

$$A^m x^m = b^m \tag{10}$$

directly using LU decomposition. From section 3.3 we know that the operator A^m on the coarsest level is still a singular M-matrix. In theory, the coarsening process does not necessarily guarantee irreducibility of A^m, although in our practical examples this is the case. Since the treatment of this problem is beyond the scope of this

paper, in the following discussion we will assume that A^m is irreducible. Then an LU decomposition of A^m with L nonsingular exists [10] and appears as follows:

$$A^m = LU = \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix} \begin{bmatrix} & & \\ & \ddots & \\ & & 0 \end{bmatrix}.$$

To solve the system (10), we need only prove that it is consistent. That is, we need to ensure that L^{-1} applied to the right-hand side produces a zero in the last entry of the resulting vector. Since the entry in the lower right corner of U is zero, we have a consistent singular system. From the following lemma we conclude that whenever the right-hand side b^m is a zero vector the last entry of $L^{-1}b^m$ will be zero, and thus our system is consistent.

LEMMA 6. [illegible] A [illegible] $A = LU$ [illegible] LU [illegible] A [illegible] L [illegible]

$$e_n^T L^{-1} = \mathbb{1}^T.$$

[illegible] $A = LU \Leftrightarrow L^{-1}A = U$. If we apply the nth unit vector from the left, we get

$$e_n^T L^{-1} A = e_n^T U = [0, \ldots, 0].$$

Thus, it must hold that $e_n^T L^{-1} = \alpha \mathbb{1}^T$ for some scalar α. Yet, as L^{-1} has ones on the diagonal, we get $\alpha = 1$ and hence $e_n^T L^{-1} = \mathbb{1}^T$. □

By construction, the AMG ensures that on any level j, the system (10) satisfies $\mathbb{1}^T b^j = 0$ since this holds for the initial right-hand side in (9), and the new right-hand sides on coarser grids are obtained via restriction of the residual, i.e.,

$$b^{j+1} = R^j(b^j - A^j x^k).$$

Note that Lemma 4 is applied to A^T. Thus, R^j corresponds to $(P^j)^T$ in Lemma 4.

3.6. Numerical tests. For numerical tests, we used MATLAB Version 7.1 run on a PC with an Intel Pentium 4 CPU 3.20GHz processor. The relative machine precision was `eps` $= 2.2204 \times 10^{-16}$. As convergence criterion for GMRES we used the tolerance $t = \|r_0\|\sqrt{\texttt{eps}}$, where r_0 is the initial residual, i.e., GMRES has converged when the residual norm falls under the tolerance t. For the AMG iteration we used the V-cycle with one presmoothing and one postsmoothing sweep; see section 3.1. The time measurement was conducted via the commands `tic` and `toc` and represents an upper bound for the required time. It can be further reduced by a more efficient implementation of the AMG algorithm.

In section 1, we have discussed the Markov chain model of blood circulation in the human body. Table 1 presents some results of the AMG preconditioned GMRES algorithm applied to transition matrices of Markov chains that arise from the blood circulation problem; see Example 1. Here, we consider the case where the blood particles can recirculate in the neighboring organs before passing on. Hence, the considered matrices have about five nonzero entries per row.

In the first column of Table 1 we have the size n of the problem. The second column states the number nnz of nonzero elements in the matrix. The column labeled Grid shows the time required by AMG for the grid construction. In *#level* the

TABLE 1
Performance of AMG preconditioned GMRES.

		AMG+GMRES				∑AMG
n	nnz	Grid	$\#level$	Time	Iters.	Time
1125	5571	1.94	7	0.2	12	2.14
2598	12936	4.28	8	0.39	13	**4.67**
4952	24706	8.27	9	0.85	13	**9.12**
9544	47666	16.97	10	2.69	12	**19.66**

TABLE 2
Comparison with other methods.

		AMGsa		∑AMGsa	GS		GS+GMRES	
n	nnz	Time	Iters.	Time	Time	Iters.	Time	Iters.
1125	5571	0.20	23	2.14	0.97	1061	**0.75**	144
2598	12936	0.68	27	5.96	8.49	4214	5.25	297
4952	24706	1.73	27	10.0	26.25	5547	18.01	428
9544	47666	4.08	20	21.05	–	–	162.60	955

number of constructed grids is given. The following Time and Iters. columns represent the computation time of AMG preconditioned GMRES and the corresponding number of iterations. The last column summarizes the total time required by AMG preconditioned GMRES.

In Table 2 we give the results of alternative methods for the same problems as in Table 1. The first two columns as in Table 1 contain the sizes and the numbers of nonzeros of the problems. The following three columns contain the results for AMG as a stand-alone function, where the total required time is given in the third column. The next two columns labeled GS display the time and number of iterations required by Gauss–Seidel iteration. The last two columns present the results for GMRES with Gauss–Seidel as preconditioner. The top times are marked in bold.

As we can see, although for the smallest problem the Gauss–Seidel iteration and the Gauss–Seidel preconditioned GMRES outperform the AMG method, for larger problems AMG preconditioning becomes very effective. The computational effort of AMG preconditioned GMRES grows linearly with the problem size, whereas the number of iterations stays constant. The number of iterations for the Gauss–Seidel preconditioned GMRES, in contrast, grows linearly with the problem size. The effectiveness of AMG preconditioning is due to the fact that the overall computation time of the AMG preconditioned GMRES is dominated by the time required to construct the "algebraic grid." The computational effort required for grid construction, although it also grows linearly, grows much slower with the matrix size than the computational effort for Gauss–Seidel preconditioned GMRES. The AMG algorithm as a stand-alone function also works quite well, although the computational effort is slightly smaller in the AMG preconditioned GMRES due to fewer iterations. This also supports the suitability of the AMG method as a preconditioner for this type of problem.

4. Conclusions. In this paper, the application of the AMG method as a preconditioner for solving large singular linear systems of the type $(I - T^T)x = 0$ with GMRES was examined. In doing so, we concentrated on the case in which T is the transition matrix of a Markov process. The AMG method [12] was originally devel-

oped for regular, symmetric, positive definite M-matrices. In our case, although the matrix $(I - T^T)$ is singular and nonsymmetric, it is a singular M-matrix that possesses a number of useful properties. It turns out that with the adaptation discussed in sections 3.3–3.5, the method becomes applicable to singular matrices such as those in the blood circulation model. Numerical experiments illustrate that preconditioning with the AMG method leads to significant acceleration of the convergence speed.

Acknowledgments. The author thanks M. Bollhöfer for suggesting the problem and for the fruitful discussions throughout the project, as well as V. Mehrmann and D. B. Szyld for reading the manuscript with great care.

REFERENCES

[1] V. A. Barker, *Numerical solution of sparse singular systems of equations arising from ergodic Markov chains*, Comm. Statist. Stochastic Models, 5 (1989), pp. 335–381.
[2] A. Berman and R. J. Plemmons, *Nonnegative Matrices in the Mathematical Sciences*, Classics Appl. Math. 9, SIAM, Philadelphia, 1994 (revised reprint of the 1979 original).
[3] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge, *Adaptive algebraic multigrid*, SIAM J. Sci. Comput., 27 (2006), pp. 1261–1286.
[4] P. N. Brown and H. F. Walker, *GMRES on (nearly) singular systems*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 37–51.
[5] Q. Chang and Z. Huang, *Efficient algebraic multigrid algorithms and their convergence*, SIAM J. Sci. Comput., 24 (2002), pp. 597–618.
[6] Q. Chang, Y. S. Wong, and H. Fu, *On the algebraic multigrid method*, J. Comput. Phys., 125 (1996), pp. 279–292.
[7] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, 1983.
[8] W. Hackbusch, *Multigrid Methods and Applications*, Springer-Verlag, Berlin, 1985.
[9] G. Horton and S. T. Leutenegger, *A multi-level solution algorithm for steady-state Markov chains*, in Proceedings of the 1994 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, ACM Press, New York, 1994, pp. 191–200.
[10] I W. Kuo, *A note on factorizations of singular M-matrices*, Linear Algebra and Appl., 16 (1977), pp. 217–220.
[11] S. MacLachlan, T. Manteuffel, and S. McCormick, *Adaptive reduction-based AMG*, Numer. Linear Algebra Appl., 13 (2006), pp. 599–620.
[12] J. W. Ruge and K. Stüben, *Algebraic multigrid*, in Multigrid Methods, S. F. McCormick, ed., Frontiers Appl. Math. 3, SIAM, Philadelphia, 1987, pp. 73–130.
[13] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.
[14] W. J. Stewart, *Introduction to the Numerical Solution of Markov Chains*, Princeton University Press, Princeton, NJ, 1994.
[15] U. Trottenberg, C. W. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, San Diego, CA, 2001.
[16] E. Virnik, *Algebraic Multigrid Preconditioners for Computing Stationary Distributions of Markov Processes*, diplomarbeit, Technische Universität Berlin, Berlin, 2004; available online from http://www.math.tu-berlin.de/preprints/abstracts/Report-10-2005.rdf.html.

SIAM J. SCI. COMPUT.
Vol. 29, No. 5, pp. 1992–2007
© 2007 Society for Industrial and Applied Mathematics

A PRECONDITIONER FOR LINEAR SYSTEMS ARISING FROM INTERIOR POINT OPTIMIZATION METHODS*

TIM REES† AND CHEN GREIF‡

Abstract. We explore a preconditioning technique applied to the problem of solving linear systems arising from primal-dual interior point algorithms in linear and quadratic programming. The preconditioner has the attractive property of improved eigenvalue clustering with increased ill-conditioning of the (1,1) block of the saddle point matrix. It fits well into the optimization framework since the interior point iterates yield increasingly ill-conditioned linear systems as the solution is approached. We analyze the spectral characteristics of the preconditioner, utilizing projections onto the null space of the constraint matrix, and demonstrate performance on problems from the NETLIB and CUTEr test suites. The numerical experiments include results based on inexact inner iterations.

Key words. block preconditioners, saddle point systems, primal-dual interior point methods, augmentation

AMS subject classifications. 65F10, 65K05

DOI. 10.1137/060661673

1. Introduction.

Interior point methods for solving linear and quadratic programming problems have been gaining popularity in the last two decades. These methods have forged connections between previously disjoint fields and allowed for a fairly general algebraic framework to be used; see, for example, [11] for a comprehensive survey. The size of many problems of interest is very large and the matrices involved are frequently sparse and often have a special structure. As a result, there is an increasing interest in iterative solution methods for the saddle point linear systems that arise throughout the iterations.

The general optimization framework is as follows. Consider the quadratic programming (QP) problem

$$\min_x \frac{1}{2}x^T Hx + c^T x$$
$$\text{subject to}$$
$$Ax = b,\ Cx \geq d.$$

Here $x, c \in \mathbb{R}^n$ and H is an $n \times n$ Hessian, often symmetric positive semidefinite; the constraint matrix A is $m_1 \times n$ with $m_1 < n$, and we assume it has rank m_1. Inequality constraints are expressed in the $m_2 \times n$ matrix C. Often simple bounds on the variables are given, so C could be an identity matrix or its concatenation with a negative identity matrix.

When H is symmetric positive semidefinite and the constraints are linear, satisfaction of the first-order, Karush–Kuhn–Tucker (KKT) conditions is sufficient to

*Received by the editors June 1, 2006; accepted for publication (in revised form) February 8, 2007; published electronically September 28, 2007.
http://www.siam.org/journals/sisc/29-5/66167.html

†Department of Applied Mathematics, University of Waterloo, Waterloo ON, N2L 3G1, Canada (twrees@math.uwaterloo.ca). The work of this author was done while he was at the University of British Columbia, in partial fulfillment of the requirements for an M.Sc. degree.

‡Department of Computer Science, University of British Columbia, Vancouver BC, V6T 1Z4, Canada (greif@cs.ubc.ca). The work of this author was supported in part by the Natural Sciences and Engineering Research Council of Canada.

guarantee global optimality of a solution [23, Chap. 16]. If Lagrange multipliers y, z and slack variables s are introduced, the KKT conditions for this problem are

$$\begin{gathered} Hx - A^T y - C^T z = -c, \\ Ax = b, \\ Cx - s = d, \\ s \geq 0,\ z \geq 0,\ s^T z = 0. \end{gathered}$$

Typical interior point methods [23, 28] for QP problems define a function whose roots coincide with the KKT conditions and take Newton steps to progressively approach an optimal solution. One possible way of solving the problem is by performing predictor and corrector solves at each step to ensure sufficient progress towards the optimal point. After elimination of some of the unknowns we obtain the step equations, a saddle point system of the form

$$\begin{pmatrix} H + C^T S^{-1} Z C & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ -\Delta y \end{pmatrix} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}, \tag{1.1}$$

where S and Z are diagonal and keep changing throughout the QP iteration. The right-hand side vector is related to residual measures of the KKT conditions and depends on whether a predictor or corrector step is being carried out [23, 28].

While we will focus primarily on QP problems, it is also worth considering the linear programming (LP) problem, which is formulated as

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} c^T x \\ &\text{subject to} \\ &Ax = b,\ x \geq d. \end{aligned}$$

LP problems share similarities with QP problems; in fact they can be classified as simple subproblems, with a zero Hessian and further simplifications. It is convenient to present the corresponding linear system in the 2×2 block form

$$\begin{pmatrix} S^{-1} Z & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta \lambda \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}, \tag{1.2}$$

where $S^{-1}Z$ is diagonal and changes throughout the LP iteration, and the right-hand side vector is constructed in a similar manner to (1.1).

For both LP and QP problems the saddle point matrix becomes highly ill-conditioned as the solution is approached, due to increased ill-conditioning of the $(1,1)$ block. In the typical case when inequality constraints are simple bounds on the primal variables, $H + C^T S^{-1} Z C$ is a diagonal perturbation of H. The complementarity of S and Z gives rise to the simultaneous presence of extremely small and extremely large values in $S^{-1}Z$. For both LP and QP problems the matrices $S^{-1}Z$ and $H + C^T S^{-1} Z C$ will never be exactly singular but will approximate singularity as the solution is approached. This is a key property that must be taken into account.

A detailed description of the theory involved in interior point methods for LP and QP can be found, for example, in [23, 28]. The methodology of using a predictor and corrector step calculation at each iteration of the solve was presented in [21]. These references point out the importance of efficiently solving the step equations and identify the difficulties involved. Many software packages, such as IPOPT [27], LIPSOL [29], and OOQP [13], use direct solvers to solve the step equations. While

this is a sound approach for many problems, it may suffer the combined ailments of poor scaling with problem size and deterioration in conditioning and numerical stability as the QP or LP solution is approached. In particular, special care must be taken in matrix factorizations to deal with the presence of large and small pivots [22]. These factors motivate the study of iterative methods in the context of optimization. Modern solution techniques, such as Krylov subspace methods, rely on the ease of sparse matrix-vector products and converge at a rate dependent on the number of distinct eigenvalues of the preconditioned matrix [7, 25].

In this paper we study a preconditioner which has the property that the more ill-conditioned the (1,1) block of the saddle point matrix is, the faster a minimum residual solver, such as MINRES, converges (in terms of outer iterations). Therefore, the corresponding solver is particularly effective in the last few iterations of the LP or QP solver. Our approach is based on augmentation of the (1,1) block using a weight matrix. Augmentation has been used in several areas of applications and in many flavors (see, for example, [3] and references therein). Our particular methodology extends recent work done by Greif and Schötzau [17, 18] into new directions and introduces a technique that fits well into the algebraic framework of interior point methods for optimization problems.

Our preconditioner is part of a growing set of preconditioned iterative approaches for solving optimization problems. A preconditioning technique that has emerged recently as a popular choice is the class of constraint preconditioners (see Lukšan and Vlček [20] or Keller, Gould, and Wathen [19] and references therein), which rely on leaving the constraints intact and seeking to replace the $(1,1)$ block by a matrix that is much easier to invert. Recent work by Dollar and Wathen [9] introduces implicit factorizations that further facilitate the use of constraint preconditioners. Similar factorizations are applied to regularized saddle point systems by Dollar et al. in [8]. Forsgren, Gill, and Griffin [10] extend constraint-based preconditioners to regularized saddle point systems that use an approximation of the $(1,1)$ block coupled with an augmenting term. The technique is intended for interior point methods for general constrained optimization problems. In [5] Bergamaschi, Gondzio, and Zilli employ constraint preconditioners with diagonal $(1,1)$ blocks, which allow for factoring the preconditioner or its reduced normal equation form. Approximate constraint preconditioners are further explored in [4].

Other block structured preconditioning approaches are also available. For example, Oliveira and Sorensen [24] consider LP and make use of block triangular preconditioners that have the constraint matrix in their $(1,2)$ block and easy-to-invert matrices in the main diagonal. The preconditioned matrix has an eigenvalue $\lambda = 1$ with a high algebraic multiplicity, and since the $(1,1)$ block of the saddle point system is diagonal for linear programs, the preconditioner can be factored and solved for efficiently. The use of iterative methods in constrained optimization also relies on the notion of inexact interior point methods. These have been investigated in [1, 2, 12] and other papers. The findings justify the use of approximate solutions at each step of the method, and we explore this experimentally.

The remainder of this paper is organized as follows. In section 2 the preconditioner and its general form are presented, and algebraic properties are derived. In section 3 the choices of the weight matrix W and inner solves are discussed. This is followed by two schemes for reducing fill-in in the preconditioner when dense rows are present. In section 4 we present numerical results demonstrating performance of the preconditioner. In section 5 we draw some conclusions.

2. The preconditioning approach. We will adopt the general notation

$$\mathcal{A} = \begin{pmatrix} G & A^T \\ A & 0 \end{pmatrix} \tag{2.1}$$

to represent the saddle point matrices of (1.1) and (1.2). We assume that G is symmetric and positive semidefinite with nullity p, and that A is of size $n \times m$ and has full row rank. Note that the assumption that $\mathcal{A}$ is nonsingular implies that $\text{null}(A) \cap \text{null}(G) = \{0\}$, which we use in our analysis below.

2.1. A block triangular preconditioner. Consider the preconditioner

$$\mathcal{M} = \begin{pmatrix} G + A^T W^{-1} A & kA^T \\ 0 & W \end{pmatrix},$$

where k is a scalar and W is an $m \times m$ symmetric positive definite weight matrix. The eigenvalues of the preconditioned matrix $\mathcal{M}^{-1}\mathcal{A}$ satisfy the generalized eigenvalue problem

$$\begin{pmatrix} G & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \lambda \begin{pmatrix} G + A^T W^{-1} A & kA^T \\ 0 & W \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}. \tag{2.2}$$

The second block row gives $y = \frac{1}{\lambda} W^{-1} A x$, and substituting it into the first block equation gives

$$\lambda(\lambda - 1)Gx + (\lambda^2 + k\lambda - 1)A^T W^{-1} A x = 0.$$

Regardless of the choice of k, we see that $\lambda = 1$ with algebraic multiplicity $n - m$. From the nullity of G it follows that there are p linearly independent null vectors of G. For each such null vector we can find two λ values satisfying $\lambda^2 + k\lambda - 1 = 0$. Thus we have

$$\lambda_{\pm} = \frac{-k \pm \sqrt{k^2 + 4}}{2},$$

each with algebraic multiplicity p. The remaining $2(m - p)$ eigenvalues satisfy

$$-\frac{\lambda^2 - \lambda}{\lambda^2 + k\lambda - 1} Gx = A^T W^{-1} A x. \tag{2.3}$$

Since G is positive semidefinite and $A^T W^{-1} A$ is positive definite, we must have

$$-\frac{\lambda^2 - \lambda}{\lambda^2 + k\lambda - 1} > 0,$$

and thus we can write $\frac{\lambda^2 - \lambda}{\lambda^2 + k\lambda - 1} = -\mu^2$ for some $\mu \in \mathbb{R},\ \mu > 0$. We can rearrange this into

$$(1 + \mu^2)\lambda^2 + (k\mu^2 - 1)\lambda - \mu^2 = 0,$$

giving

$$\lambda_{\pm} = \frac{-(k\mu^2 - 1) \pm \sqrt{(k\mu^2 - 1)^2 + 4\mu^2(1 + \mu^2)}}{2(1 + \mu^2)}. \tag{2.4}$$

This expression gives an explicit formula in terms of the generalized eigenvalues of (2.3) and can be used to identify the intervals in which the eigenvalues lie. To illustrate this, we examine the case $k = -1$, which corresponds to setting the $(1, 2)$ block of the preconditioner to be $-A^T$. We have $\lambda = 1$ with multiplicity $n - m$, and $\lambda_\pm = \frac{1\pm\sqrt{5}}{2}$, each with multiplicity p. By (2.4) we have

$$\lambda_\pm = \frac{1 \pm \sqrt{1 + \frac{4\mu^2}{1+\mu^2}}}{2}.$$

Since λ_+ is a strictly increasing function of μ on $(0, \infty)$ (and λ_- are strictly decreasing), the intervals containing the remaining eigenvalues can be found using $\lim_{\mu\to 0,\infty} \lambda_\pm(\mu)$. From this, one finds that the remaining eigenvalues lie in the intervals $(\frac{1-\sqrt{5}}{2}, 0) \cup (1, \frac{1+\sqrt{5}}{2})$. It is worth noting that since G is typically highly singular, many of the generalized eigenvalues are large, in which case the corresponding eigenvalues $\lambda_\pm$ are bounded away from zero. For example, many of the negative eigenvalues will tend to $\frac{1-\sqrt{5}}{2}$. Since the $2p$ eigenvalues $(k \pm \sqrt{k^2 + 4})/2$ are unbounded as k goes to ∞, we conclude that k should be of moderate size.

2.2. A block diagonal preconditioner. The choice $k = 0$ yields a block diagonal symmetric positive definite preconditioner of the form

$$\mathcal{M} = \begin{pmatrix} G + A^T W^{-1} A & 0 \\ 0 & W \end{pmatrix}. \tag{2.5}$$

It is suitable for use with minimal residual methods based on short recurrences, such as MINRES. This preconditioner and its spectral properties in a general algebraic context have been recently studied in [17] and applied (in a slightly different form) to the Maxwell equations in [18]. In the context of optimization, it is useful to examine the behavior on the reduced space generated by projections onto the null space of A. We offer such results below, and prove them using orthogonal transformations, by taking steps similar to those taken in [19].

As before, we assume that G is positive semidefinite with nullity p. Suppose further that A has full rank, and choose W to be symmetric positive definite. It is straightforward to show that if $\mathcal{A}$ is nonsingular, then $G + A^T W^{-1} A$ must be symmetric positive definite. As is shown in [17], the eigenvalues of the preconditioned matrix $\mathcal{M}^{-1}\mathcal{A}$ and their multiplicities can be found directly through the generalized eigenvalue problem

$$\begin{pmatrix} G & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} = \lambda \begin{pmatrix} G + A^T W^{-1} A & 0 \\ 0 & W \end{pmatrix} \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix}. \tag{2.6}$$

Let $QR = A^T$ be an "economy size" QR factorization of A^T, where Q is $n \times m$ and R is $m \times m$. Define Z to be an $n \times (n - m)$ orthogonal basis for the null space of A. Since $Z \cup Q$ forms an orthogonal basis for $\mathbb{R}^n$, any vector $x \in \mathbb{R}^n$ can be written as $x = Zx_z + Qx_q$.

Following the spirit of the proof of [19, Thm. 2.1], we define the $(n+m) \times (n+m)$ matrix

$$P = \begin{pmatrix} Z & Q & 0 \\ 0 & 0 & I \end{pmatrix} \tag{2.7}$$

and perform a similarity transformation as follows. We express $\bar{x} = Z\bar{x}_z + Q\bar{x}_q$, and let $v = (x_z, x_q, y)^T$, where $Pv = (\bar{x}_z, \bar{x}_q, \bar{y})^T$. The generalized eigenvalue problem can then be written as $P^T\mathcal{A}Pv = \lambda P^T\mathcal{M}Pv$. This yields

(2.8)
$$\begin{pmatrix} Z^TGZ & Z^TGQ & 0 \\ Q^TGZ & Q^TGQ & R \\ 0 & R^T & 0 \end{pmatrix} \begin{pmatrix} x_z \\ x_q \\ y \end{pmatrix} = \lambda \begin{pmatrix} Z^TGZ & Z^TGQ & 0 \\ Q^TGZ & Q^TGQ + RW^{-1}R^T & 0 \\ 0 & 0 & W \end{pmatrix} \begin{pmatrix} x_z \\ x_q \\ y \end{pmatrix}.$$

By inspection, we observe that by setting $\lambda = 1$, the system reduces to

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & -RW^{-1}R^T & R \\ 0 & R^T & -W \end{pmatrix} \begin{pmatrix} x_z \\ x_q \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

Let e_i denote the ith column of the identity matrix. Evidently there are $n - m$ corresponding eigenvectors that can be written in the form

$$(x_z, x_q, y) = (e_i, 0, 0). \tag{2.9}$$

In addition, m linearly independent eigenvectors can be written in the form

$$(x_z, x_q, y) = (0, e_i, W^{-1}R^Te_i). \tag{2.10}$$

Now consider $\lambda = -1$. Equation (2.8) reduces to

$$\begin{pmatrix} 2Z^TGZ & 2Z^TGQ & 0 \\ 2Q^TGZ & 2Q^TGQ + RW^{-1}R^T & R \\ 0 & R^T & W \end{pmatrix} \begin{pmatrix} x_z \\ x_q \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

Any vector $x^* = Zx_z^* + Qx_q^*$ in the null space of G satisfies $G(Zx_z^* + Qx_q^*) = 0$. There are p such vectors, so p linearly independent eigenvectors of the form

$$(x_z, x_q, y) = (x_z^*, x_q^*, -W^{-1}R^Tx_q^*) \tag{2.11}$$

will satisfy (2.8) with $\lambda = -1$.

By [17, Thm. 2.2] the remaining eigenvalues, $\lambda \neq \pm 1$, lie in the interval $(-1, 0)$. To derive an expression for the corresponding eigenvectors we reduce (2.8) to an eigenvalue problem in x_q. From the block row in (2.8), $y = \frac{1}{\lambda}W^{-1}R^Tx_q$. The first line of the equation can be reduced to

$$x_z = -(Z^TGZ)^{-1}Z^TGQx_q.$$

Substituting this into the second line of (2.8) and simplifying yields

$$RW^{-1}R^Tx_q = \lambda\left[C^T(Z^TGZ)^{-1}C - Q^TGQx_q - RW^{-1}R^T\right]x, \tag{2.12}$$

where $C = Z^TGQ$. We have actually proved the following theorem.

THEOREM 2.1. [illegible] $\mathcal{M}^{-1}\mathcal{A}$ [illegible] $\lambda = 1$ [illegible] n [illegible] (2.9) [illegible] (2.10) [illegible] $\lambda = -1$ [illegible] p [illegible] (2.11) [illegible] [illegible] $(-1, 0)$ [illegible] (2.12)

Theorem 2.1 illustrates the strong spectral clustering when the $(1,1)$ block of $\mathcal{A}$ is singular. A well-known difficulty associated with interior point methods is the increased ill-conditioning of the $(1,1)$ block as the solution is approached. Our claim is that the preconditioner performs robustly even as the problem becomes more ill-conditioned; in fact the outer iteration count decreases. On the other hand, solving for the augmented (1,1) block may be more computationally difficult and requires an effective approach such as inexact solves.

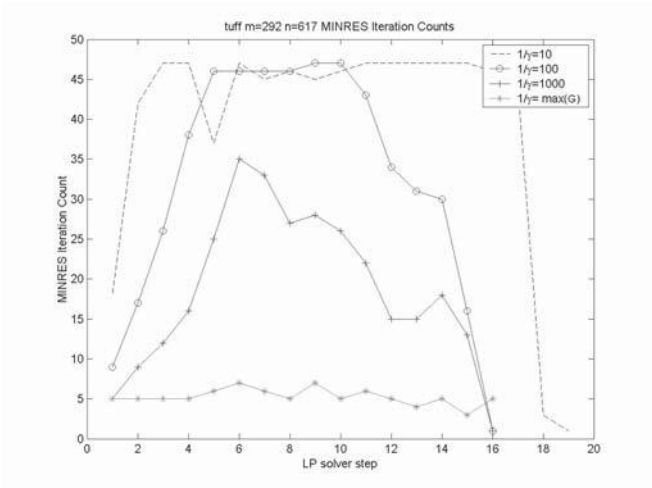


FIG. 3.1. *MINRES iteration counts with various* $W = \gamma I$. *Problem tuff has* $m = 292$, $n = 617$ *after LIPSOL preprocessing. MINRES iteration counts are plotted at each LP step for the various choices of* γ. *MINRES error tolerance was set to* 10^{-8}. *In final iterations,* $\max(G) > 10^8$.

3. Practical considerations and computational cost. In this section we discuss the choice of the weight matrix W and ways of reducing the cost of inner iterations. We also describe procedures for dealing with a dense row.

3.1. The inner iteration and choice of the weight matrix. There are two critical issues to consider in the application of the preconditioner. First, the weight matrix W must be chosen. Then, given a weight matrix, an efficient method of factoring or iteratively solving systems with the preconditioner must be sought. These considerations are motivated by the fact that each iteration of a preconditioned Krylov subspace method requires solutions to linear systems of the form $\mathcal{M}x = b$; based on the block structure of $\mathcal{M}$, this requires solving systems with $G + A^T W^{-1} A$ and W.

The simplest choice of a weight matrix is diagonal, and it clearly makes inverting W trivial. A simple, one-parameter choice is a scaled identity. Letting $W = \gamma I$, γ could be chosen so that the augmenting term $\frac{1}{\gamma} A^T A$ is of a norm comparable to G. See, for example, [14] for a general algebraic discussion. Note that since G changes at each step, γ must also be updated.

For LP problems the matrix G is diagonal, and choosing $1/\gamma$ related to an ordered statistic such as the mean, median, or maximum of the diagonal entries in G has proved to be effective in our experiments in reducing the number of MINRES iterations required at each step. This is illustrated in Figure 3.1 for the NETLIB problem "tuff," where the MINRES residual norms (in the predictor step) are plotted against each step of the LP solve. An arbitrary fixed choice of $1/\gamma$ leads to a "hump" with a large number of intermediate outer iteration counts. As predicted by our theoretical results, close to the solution the near-singularity of the (1,1) block results in fast convergence regardless of the choice of γ. But as is illustrated in the figure, it is the dynamic choice of γ^{-1} as the maximal entry in G that yields rapid convergence of MINRES throughout the LP iteration, and in fact the "hump" is flattened in this case. The choice $1/\gamma = \max(G)$ results in a set of values monotonically increasing from approximately 1 to approximately 10^{10}, and the iteration significantly outperforms

other choices in terms of the number of MINRES iterations, while the cost of each inner solve does not change.

For QP problems, similar approaches are possible. With $W = \gamma I$, a choice of γ such as $\|A\|^2/\|G\|$ (or an approximation thereof) ensures that the norm of the augmenting term is not too small in comparison with G. (See [14] for an analysis of the effect of scaling of this sort on the condition number of the matrix.) For solving $\mathcal{M}x = b$, iterative and direct methods are possible. In an iterative scheme the inner iteration can be solved using the preconditioned conjugate gradient (PCG) method. If a direct solver is preferred, the symmetric positive definiteness of the preconditioner allows for use of a sparse Cholesky factorization.

3.2. Dealing with a dense row in A. The presence of even a single dense row in A can lead to a fully dense augmenting matrix $A^T W^{-1} A$. We present two possible approaches for dealing with dense rows in the situation where it is desirable to explicitly form the $(1,1)$ block of the preconditioner.

First, we present an asymmetric preconditioner, motivated by the analysis of section 2.1. With a_i denoting the dense column i of A^T, and e_i being the ith column of an $m \times m$ identity, we define a preconditioner

$$\hat{\mathcal{M}} = \begin{pmatrix} G + A^T \bar{W} A & -a_i e_i^T \\ 0 & W \end{pmatrix}. \tag{3.1}$$

Suppose $W = \gamma I$ for some $\gamma > 0$, and let $\bar{W} = \frac{1}{\gamma} I - \frac{1}{\gamma} e_i e_i^T$. Assuming $\hat{\mathcal{M}}$ is nonsingular, the eigenvalues of the preconditioned matrix are given by the following theorem.

THEOREM 3.1. [illegible] $\hat{\mathcal{M}}^{-1}\mathcal{A}$ [illegible] $\lambda = 1$ [illegible] $n - 1$ [illegible] $\lambda = -1$ [illegible] p [illegible] A [illegible].

Exactly as in the proof of Theorem 2.1 we define Q, R, Z and transform the generalized eigenvalue problem using P as in (2.7). This yields

$$\begin{pmatrix} Z^TGZ & Z^TGQ & 0 \\ Q^TGZ & Q^TGQ & R \\ 0 & R^T & 0 \end{pmatrix} \begin{pmatrix} x_z \\ x_q \\ y \end{pmatrix} = \lambda \begin{pmatrix} Z^TGZ & Z^TGQ & 0 \\ Q^TGZ & Q^TGQ + \frac{1}{\gamma} RR^T - \frac{1}{\gamma} r_i r_i^T & -r_i e_i^T \\ 0 & 0 & \gamma I \end{pmatrix} \begin{pmatrix} x_z \\ x_q \\ y \end{pmatrix}, \tag{3.2}$$

where r_i denotes the ith column of R.

As before, by inspection we check $\lambda = 1$, which reduces the equation to

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & -\frac{1}{\gamma} RR^T + \frac{1}{\gamma} r_i r_i^T & R + r_i e_i^T \\ 0 & R^T & -\gamma I \end{pmatrix} \begin{pmatrix} x_z \\ x_q \\ y \end{pmatrix} = 0.$$

Immediately we see $n-m$ corresponding eigenvectors of the form $(x_z, x_q, y) = (u, 0, 0)$ for $(n-m)$ linearly independent vectors u. An additional $m-1$ linearly independent eigenvectors can be seen by finding consistent solutions in the free variables x_q, y to the equation

$$\begin{pmatrix} -\frac{1}{\gamma} RR^T + \frac{1}{\gamma} r_i r_i^T & R + r_i e_i^T \\ R^T & -\gamma I \end{pmatrix} \begin{pmatrix} x_q \\ y \end{pmatrix} = 0.$$

Substituting $y = \frac{1}{\gamma} R^T x_q$, this requires $2\frac{1}{\gamma} r_i r_i^T x_q = 0$. In general we can find exactly $m-1$ eigenvectors orthogonal to r_i. That is, there are $m-1$ eigenvectors of the form $(x_z, x_q, y) = (0, x_q^*, \frac{1}{\gamma} x_q^*)$, where x_q^* is orthogonal to r_i, corresponding to $\lambda = 1$.

The p eigenvectors corresponding to $\lambda = -1$ are also evident by simple inspection. Substituting $\lambda = -1$ requires finding a solution to

$$\begin{pmatrix} 2Z^TGZ & 2Z^TGQ & 0 \\ 2Q^TGZ & 2Q^TGQ + \frac{1}{\gamma}RR^T - \frac{1}{\gamma}r_ir_i^T & R - r_ie_i^T \\ 0 & R^T & \gamma I \end{pmatrix} \begin{pmatrix} x_z \\ x_q \\ y \end{pmatrix} = 0.$$

Vectors x_z, x_q, y can be found to solve this equation. Consider any $x^* = Zx_z^* + Qx_q^*$ in the null space of G. Then $GZx_z^* + GQx_q^* = 0$, and we are left with finding a y such that

$$\begin{pmatrix} \frac{1}{\gamma}RR^T - \frac{1}{\gamma}r_ir_i^T & R - r_ie_i^T \\ R^T & \gamma I \end{pmatrix} \begin{pmatrix} x_q^* \\ y \end{pmatrix} = 0$$

for the fixed x_q^*. The choice $y = -\frac{1}{\gamma}R^Tx_q^*$ correctly cancels the left-hand side, and it becomes apparent why the minus sign was chosen for the $(1,2)$ block of $\hat{\mathcal{M}}$; without it, we could not explicitly find a suitable y value. Since the p vectors in the null space of G are linearly independent, we have constructed a p-dimensional eigenbasis for $\hat{\mathcal{M}}^{-1}\mathcal{A}$, corresponding to $\lambda = -1$. □

Theorem 3.1 shows that $\hat{\mathcal{M}}$ is sparse and at the same time maintains strong spectral clustering. The preconditioner is asymmetric, however, and it is desirable to find a simpler form (that still retains the strong spectral properties).

To this end, consider replacing preconditioner $\hat{\mathcal{M}}$ from (3.1) with

$$\bar{\mathcal{M}} = \begin{pmatrix} G + A^T\bar{W}A & 0 \\ 0 & W \end{pmatrix},$$

where $\bar{W}$ is again an approximation to W^{-1}. Similarly to the asymmetric case, consider the choice

$$\bar{W} = W^{-1} - \frac{1}{\gamma}e_ie_i^T.$$

The matrix $\bar{W}$ is diagonal but singular, since each of its rows that corresponds to a dense row of A is identically zero. As a result, the matrix $A^T\bar{W}A$ no longer experiences fill-in from the product $a_ia_i^T$.

This modification does not result in significant changes in the spectral clustering of the preconditioned matrix. Since $\bar{\mathcal{M}}$ is a rank-1 perturbation of $\mathcal{M}$ by (2.5), it follows that $\bar{\mathcal{M}}^{-1}\mathcal{A}$ is just a rank-1 perturbation of $\mathcal{M}^{-1}\mathcal{A}$ and we can apply interlacing results.

If we let μ_i denote the ith largest eigenvalue of $\bar{\mathcal{M}}^{-1}\mathcal{A}$, and let λ_i be the ith largest eigenvalue of $\mathcal{M}^{-1}\mathcal{A}$, interlacing guarantees that $\lambda_{i-1} \le \mu_i \le \lambda_i$. Since the eigenvalues λ_i are known and have high algebraic multiplicities, so are the eigenvalues μ_i, and for each multiple eigenvalue λ_i the multiplicity of the corresponding μ_i reduces by at most 1, due to interlacing. Thus, if preconditioned MINRES is used, we have strong spectral clustering without introducing any fill-in. We can summarize our findings as follows.

PROPOSITION 3.2. [illegible] $\bar{\mathcal{M}}$ [illegible] $\bar{\mathcal{M}}^{-1}\mathcal{A}$ [illegible] $\lambda = 1$ [illegible] $n - 1$ [illegible] $\lambda = -1$ [illegible] $p - 1$

TABLE 4.1

Solver results using BiCG-STAB. Problems were solved to a tolerance of 1.0e−06. *BiCG-STAB tolerance was fixed at* 1.0e−02.

Problem	n	m_1	m_2	N_{QP}	N_K	N_I	$Tot(N_I)$	Time(s)
avgasa	12	10	18	5	2.80	2.00	112	0.22
avgasb	12	10	18	5	2.75	2.00	110	0.21
$blockqp1-10$	26	11	51	4	1.75	2.71	76	0.15
$blockqp2-10$	26	11	51	4	1.81	3.07	89	0.14
$blockqp3-10$	26	11	51	6	2.00	3.42	164	0.25
$blockqp4-10$	26	11	51	5	2.50	3.44	172	0.22
$blockqp5-10$	26	11	51	6	2.38	3.25	185	0.28
$cvxqp1-100$	100	50	200	34	3.66	4.67	3840	3.75
$cvxqp2-100$	100	25	200	13	3.06	3.57	567	0.81
$cvxqp3-100$	100	75	200	13	4.06	3.47	732	1.10
*dual*1	85	1	170	7	1.75	4.26	362	0.99
*dual*2	96	1	192	5	1.50	3.37	101	0.57
*dual*3	111	1	222	5	1.50	3.07	92	0.71
*dual*4	75	1	150	5	1.45	3.55	103	0.37
*gouldqp*2	699	349	1398	10	1.82	2.08	152	1.00
$gouldqp2-30$	59	29	118	7	1.64	2.59	119	0.25
*gouldqp*3	699	349	1398	11	1.52	3.03	203	1.07
$gouldqp3-30$	59	29	118	5	1.55	2.71	84	0.17
steenbra	432	108	432	11	1.86	5.28	433	2.12

4. Numerical experiments. Numerical experiments were done on problems from the CUTEr test suite [16], using MATLAB, on an Intel 2.5GHZ processor with 2GB of RAM. In our experiments we focused on QP test problems with a nondiagonal and semidefinite $(1,1)$ block, for which our preconditioner is suitable. In this section we illustrate how the number of iterations required by MINRES drops to its theoretical limit and how inexact inner iterations reduce the overall computational work. (For the latter we settle for experimental observations; see, for example, [15] or [26] for general theory and analysis.) We also include results for the row removal scheme discussed in section 3.2.

We used a variety of methods for solving the inner iteration $\mathcal{M}x = b$, but most of our experiments made use of ILUPACK [6]. This package uses multilevel incomplete LU factorizations as preconditioners for CG and GMRES, and was found to be efficient and easy to use. While we tested a range of values, the results we present here were collected with an ILUPACK backwards error setting of 1e−06.

Tables 4.1–4.4 demonstrate several measures of the work required in applying our method. In these tables, n, m_1, m_2 denote the dimensions of the problem being solved, as defined in the introduction, and N_{QP} denotes the total number of QP steps required for convergence. The average number of Krylov solver iterations per QP steps is given by N_K. The average number of iterations of PCG used by ILUPACK in the inner iteration, and the total number summed over all QP steps, are given by N_I and $Tot(N_I)$, respectively. The time (in seconds) required to solve a problem is given in the column Time(s).

Tables 4.1–4.2 show results for applying BiCG-STAB, once with a tight outer tolerance of 10^{-6} and once with a loose outer tolerance of 10^{-2}. Tables 4.3–4.4 show results, using the same tolerances, for applying MINRES. The following observations can be made. In general, loosening tolerance for the Krylov solver increases the overall number of QP iterations only modestly, and at the same time substantially reduces (in most tested examples) the overall number of solves. We mention here that loosening the convergence tolerance of the innermost iterations did not result in

TABLE 4.2
Solver results using BiCG-STAB. Problems were solved to a tolerance of $1.0e{-}06$. *BiCG-STAB tolerance was fixed at* $1.0e{-}06$.

Problem	n	m_1	m_2	N_{QP}	N_K	N_I	$Tot(N_I)$	Time(s)
$avgasa$	12	10	18	3	12.00	2.00	288	0.53
$avgasb$	12	10	18	4	10.56	2.00	338	0.48
$blockqp1-10$	26	11	51	4	2.12	2.59	88	0.15
$blockqp2-10$	26	11	51	4	2.31	3.08	114	0.16
$blockqp3-10$	26	11	51	5	7.60	3.01	457	0.53
$blockqp4-10$	26	11	51	5	5.60	3.96	444	0.39
$blockqp5-10$	26	11	51	6	6.75	3.09	500	0.56
$cvxqp1-100$	100	50	200	13	21.44	3.94	5002	4.74
$cvxqp2-100$	100	25	200	12	10.27	4.01	2008	1.91
$cvxqp3-100$	100	75	200	11	22.91	3.73	4107	4.30
$dual1$	85	1	170	7	2.07	5.42	716	1.38
$dual2$	96	1	192	5	1.50	3.37	101	0.56
$dual3$	111	1	222	5	1.50	3.07	92	0.73
$dual4$	75	1	150	5	3.50	4.40	308	0.64
$gouldqp2$	699	349	1398	9	4.28	2.05	315	1.59
$gouldqp2-30$	59	29	118	5	5.00	2.00	200	0.35
$gouldqp3$	699	349	1398	10	3.45	2.69	371	1.64
$gouldqp3-30$	59	29	118	5	3.55	2.70	192	0.29
$steenbra$	432	108	432	11	6.82	5.30	1591	5.31

TABLE 4.3
Solver results using MINRES. Problems were solved to a tolerance of $1.0e{-}06$. *MINRES tolerance was fixed at* $1.0e{-}02$.

Problem	n	m_1	m_2	N_{QP}	N_K	N_I	$Tot(N_I)$	Time(s)
$avgasa$	12	10	18	4	4.00	2.00	96	0.19
$avgasb$	12	10	18	5	3.70	2.00	114	0.23
$blockqp1-10$	26	11	51	4	1.50	2.86	80	0.16
$blockqp2-10$	26	11	51	4	1.38	3.07	86	0.15
$blockqp3-10$	26	11	51	6	1.92	3.79	178	0.24
$blockqp4-10$	26	11	51	5	2.00	3.70	148	0.20
$blockqp5-10$	26	11	51	6	2.58	3.45	190	0.26
$cvxqp1-100$	100	50	200	21	8.74	4.29	2000	2.40
$cvxqp2-100$	100	25	200	13	3.31	3.59	496	0.77
$cvxqp3-100$	100	75	200	16	5.94	3.68	946	1.47
$dual1$	85	1	170	7	1.57	5.91	384	0.88
$dual2$	96	1	192	5	1.00	3.27	98	0.58
$dual3$	111	1	222	5	1.00	3.07	92	0.74
$dual4$	75	1	150	5	2.80	5.02	266	0.57
$gouldqp2$	699	349	1398	10	1.65	2.14	156	1.09
$gouldqp2-30$	59	29	118	6	1.50	2.33	98	0.21
$gouldqp3$	699	349	1398	13	1.35	3.14	273	1.48
$gouldqp3-30$	59	29	118	6	1.58	2.86	123	0.24
$steenbra$	432	108	432	12	6.38	6.92	1390	4.14

a similar reduction of computational work. We therefore observe in our experiments that inexactness is more effective on the level of the outer Krylov iterations rather than on the level of the innermost iterations.

Comparing the performance of BiCG-STAB to the performance of MINRES is not within our stated goals, but having results using more than one Krylov solver allows us to confirm the consistency of convergence behavior for most problems. The two solvers do behave in a similar manner, and the modest running times indicate that the proposed preconditioner seems to be efficient and robust.

Next, to demonstrate the application of the row removal scheme proposed in sec-

TABLE 4.4

Solver results using MINRES. Problems were solved to a tolerance of 1.0e−06. *MINRES tolerance was fixed at* 1.0e−06.

Problem	n	m_1	m_2	N_{QP}	N_K	N_I	$Tot(N_I)$	Time(s)
avgasa	12	10	18	4	12.25	2.00	236	0.41
avgasb	12	10	18	4	11.25	2.00	220	0.38
*blockqp*1 − 10	26	11	51	4	1.75	2.73	82	0.15
*blockqp*2 − 10	26	11	51	4	1.75	3.07	92	0.16
*blockqp*3 − 10	26	11	51	5	8.10	2.85	288	0.42
*blockqp*4 − 10	26	11	51	5	7.20	3.96	404	0.42
*blockqp*5 − 10	26	11	51	6	7.25	2.99	332	0.46
*cvxqp*1 − 100	100	50	200	14	21.39	4.06	3072	3.69
*cvxqp*2 − 100	100	25	200	12	14.71	3.99	1641	1.95
*cvxqp*3 − 100	100	75	200	14	17.82	3.78	2779	3.66
*dual*1	85	1	170	7	2.64	6.66	586	1.08
*dual*2	96	1	192	5	1.00	3.27	98	0.58
*dual*3	111	1	222	5	1.00	3.07	92	0.72
*dual*4	75	1	150	5	4.60	5.65	418	0.72
*gouldqp*2	699	349	1398	9	6.22	2.03	300	2.00
*gouldqp*2 − 30	59	29	118	5	7.10	2.00	182	0.39
*gouldqp*3	699	349	1398	10	4.50	2.67	347	1.94
*gouldqp*3 − 30	59	29	118	5	4.80	2.68	182	0.32
steenbra	432	108	432	11	31.18	5.93	4524	13.12

TABLE 4.5

Results obtained using the symmetric dense row removal scheme of section 3.2. *Problems were solved using MINRES with error tolerance* 1.0e−05. *Problems were solved to accuracy* 1.0e−4. *For each blockqp problem,* $n = 2006$, $m_1 = 1001$, $m_2 = 4011$.

Problem	N_{QP}	N_K	N_I	Time(s)
*blockqp*1	3	3.00	7.67	1.78
*blockqp*2	4	3.00	8.00	2.38
*blockqp*3	8	30.69	9.44	27.28
*blockqp*4	6	18.88	8.28	17.31
*blockqp*5	8	25.38	8.21	23.32

tion 3.2 we consider the blockqp set of problems. These problems are characterized by a Hessian with two nonzero diagonals, and a constraint matrix with a single nonzero diagonal, a dense row, and a dense column. As a result, if the augmentation preconditioner is fully formed, it will be dense. To avoid this, the symmetric row removal scheme is used. This leads to a preconditioner with a nearly diagonal $(1, 1)$ block, which can be approximately factored with ILUPACK in an efficient manner. Table 4.5 presents results confirming that the row removal scheme can lead to a convergent solution while avoiding the intensive memory requirements of fully forming the augmenting $(1, 1)$ block of the preconditioner.

Theorem 2.1 guarantees increased spectral clustering of the preconditioned matrix $\mathcal{M}^{-1}\mathcal{A}$ when the $(1, 1)$ block of $\mathcal{A}$ is singular. The LP and QP saddle point matrices, however, become approximately singular only as a solution is approached. It is useful to evaluate whether the strong clustering of the preconditioned eigenvalues will be achieved under approximate conditions. To test this, we examined the eigenvalues of the preconditioned matrix at various steps in the process of solving an LP problem. Figure 4.1 depicts the sorted eigenvalues at three different steps of the LP solve for the problem share2b.

Preconditioned eigenvalues at the first, sixth, and tenth LP steps are shown from top to bottom. (The problem took 13 steps to solve.) As expected by Theorem 2.1, all

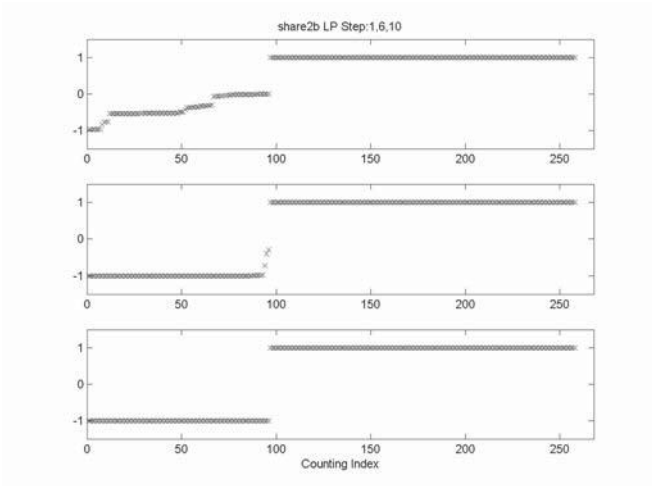


FIG. 4.1. *Eigenvalues of $\mathcal{M}^{-1}\mathcal{A}$ at different steps of the LP solution for share2b. Eigenvalues are plotted in sorted order with values along the y axis. As governed by Theorem* 2.1*, all unclustered eigenvalues lie in the interval* $(-1, 0)$.

eigenvalues lie between -1 and 1. Furthermore, right from the first iteration, $\lambda = 1$ has high multiplicity. It is interesting to note that already by the sixth step (the middle subplot), only a handful of unclustered eigenvalues remains. In the tenth LP step, all eigenvalues appear to be ± 1. These observations all confirm Theorem 2.1 and illustrate how tightly clustered the preconditioned eigenvalues can be when the saddle point system is severely ill-conditioned. This also demonstrates that even in early iterations the preconditioner can be effective.

Next, we present preliminary comparisons with constraint preconditioners. We stress that we have performed only a limited set of tests with a specific simple choice of a (1,1) block for constraint preconditioners; a comprehensive set of tests will be necessary to reach definite conclusions. Our results are mixed, as we illustrate below. Figure 4.2 shows a basic comparison of MINRES iteration counts (of the predictor computation). The plot shows the MINRES iteration counts for each QP step for the problem cvxqp1. The constraint preconditioner used for this plot was chosen to have a $(1, 1)$ block equal to the diagonal of the saddle point system. For both preconditioners, an exact inner solve was applied. For this problem the constraint preconditioner outperformed the augmentation preconditioner in most steps of the QP solve and in terms of overall computational work. In the final few steps, however, where the saddle point matrix was highly ill-conditioned, MINRES iteration counts for our preconditioner dropped significantly and convergence was almost immediate, whereas convergence of the constraint preconditioner was still within approximately 20 iterations. This again confirms Theorem 2.1 and indicates that the proposed preconditioner is most effective when the saddle point matrix is close to singular. This in fact may suggest a hybrid approach in which it may be useful to switch to an augmentation-based preconditioner when iterates approach the solution.

Figure 4.3 shows an example in which our preconditioner is superior to the constraint preconditioner throughout the iteration. For the QP problem cvxqp2, MINRES iteration counts (of the predictor computation) are plotted against the QP step. This is a large problem, but throughout the solve no more than 30 iterations are

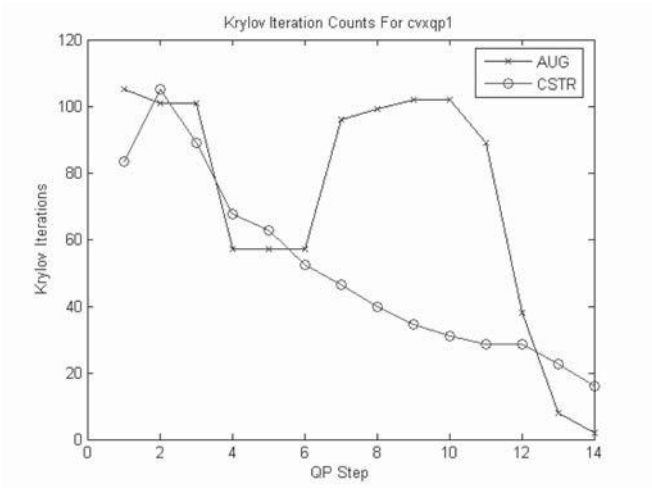


FIG. 4.2. *MINRES iteration counts for cvxqp*1, $n = 1000$, $m_1 = 250$. *Note that in the final few steps, iteration counts for the augmentation preconditioner significantly reduce, as the* $(1, 1)$ *block becomes increasingly rank deficient.*

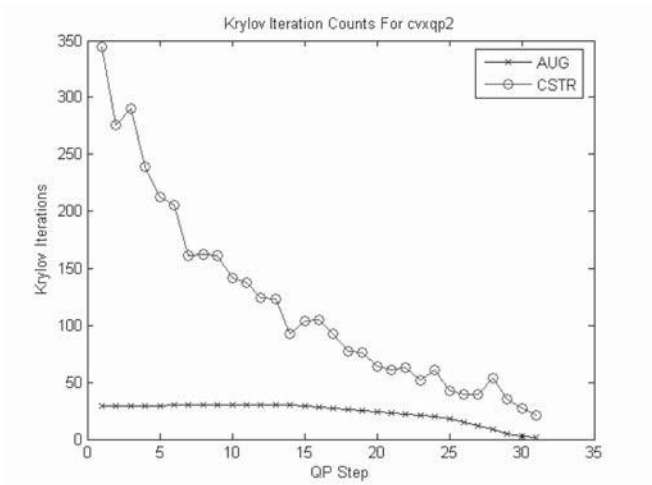


FIG. 4.3. *MINRES iteration counts for cvxqp*2, $n = 10000$, $m_1 = 2500$. *The augmentation preconditioner approaches the rate of convergence predicted by our analysis: the final two iteration counts of MINRES are* 3 *and* 2, *respectively.*

needed per step. In the final few QP steps, the MINRES iteration count approaches its theoretical limit of two. Depicted in the same plot are the corresponding predictor iteration counts for a constraint preconditioner, with the $(1, 1)$ block set to match the diagonal of the saddle point system. The constraint preconditioner consistently requires more MINRES iterations at each QP step.

5. Conclusions. We have studied preconditioners for linear and quadratic programming problems and have demonstrated their merits. The preconditioners are well suited for saddle point systems with a highly singular (1, 1) block. For interior point iterations, as convergence is approached and the $(1, 1)$ block of the saddle point ma-

trix becomes more ill-conditioned, convergence is fastest and can theoretically occur within two outer iterations (in the absence of roundoff errors). We have also provided spectral analysis on the null space of the constraint matrix.

The choice of the weight matrix W is crucial and we have tested with simple scaled identity matrices that incorporate a dynamic choice of the parameter throughout the iteration. A good choice may substantially reduce the iteration counts throughout, as we demonstrate by the "flattened hump" for an LP problem in Figure 3.1. Furthermore, we have shown that applying an inexact version of the Krylov solver throughout the iteration, with a convergence tolerance as low as 0.01, significantly reduces the overall amount of computational work.

The excellent performance of the preconditioner, in particular near the solution of the optimization problem, may suggest that, regardless of what preconditioner is used throughout the iteration, it may be worthwhile to switch to a preconditioner of the form explored here as the iterates approach the solution.

Acknowledgments. We would like to thank the two referees for their constructive comments.

REFERENCES

[1] S. Bellavia, *Inexact interior-point method*, J. Optim. Theory Appl., 96 (1998), pp. 109–121.
[2] S. Bellavia and S. Pieraccini, *Convergence analysis of an inexact infeasible interior point method for semidefinite programming*, Comput. Optim. Appl., 29 (2004), pp. 289–313.
[3] M. Benzi, G. H. Golub, and J. Liesen, *Numerical solution of saddle point problems*, Acta Numer., 14 (2005), pp. 1–137.
[4] L. Bergamaschi, J. Gondzio, M. Venturin, and G. Zilli, *Inexact constraint preconditioners for linear systems arising in interior point methods*, Comput. Optim. Appl., to appear.
[5] L. Bergamaschi, J. Gondzio, and G. Zilli, *Preconditioning indefinite systems in interior point methods for optimization*, Comput. Optim. Appl., 28 (2004), pp. 149–171.
[6] M. Bollhöfer and Y. Saad, *Multilevel preconditioners constructed from inverse-based* ILUs, SIAM J. Sci. Comput., 27 (2006), pp. 1627–1650.
[7] J. W. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, 1997.
[8] H. S. Dollar, N. I. M. Gould, W. H. A. Schilders, and A. J. Wathen, *Implicit-factorization preconditioning and iterative solvers for regularized saddle-point systems*, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 170–189.
[9] H. S. Dollar and A. J. Wathen, *Approximate factorization constraint preconditioners for saddle-point matrices*, SIAM J. Sci. Comput., 27 (2006), pp. 1555–1572.
[10] A. Forsgren, P. E. Gill, and J. Griffin, *Iterative solution of augmented systems arising in interior methods*, SIAM J. Optim., to appear.
[11] A. Forsgren, P. E. Gill, and M. H. Wright, *Interior methods for nonlinear optimization*, SIAM Rev., 44 (2002), pp. 525–597.
[12] R. W. Freund, F. Jarre, and S. Mizuno, *Convergence of a class of inexact interior-point algorithms for linear programs*, Math. Oper. Res., 24 (1999), pp. 50–71.
[13] E. M. Gertz and S. J. Wright, *Object-oriented software for quadratic programming*, ACM Trans. Math. Softw., 29 (2003), pp. 58–81.
[14] G. H. Golub and C. Greif, *On solving block-structured indefinite linear systems*, SIAM J. Sci. Comput., 24 (2003), pp. 2076–2092.
[15] G. H. Golub and M. L. Overton, *The convergence of inexact Chebyshev and Richardson iterative methods for solving linear systems*, Numer. Math., 53 (1998), pp. 571–593.
[16] N. I. M. Gould, D. Orban, and P. L. Toint, *CUTEr and SIFDEC: A constrained and unconstrained testing environment, revisited*, ACM Trans. Math. Softw., 29 (2003), pp. 373–394.
[17] C. Greif and D. Schötzau, *Preconditioners for saddle point linear systems with highly singular* (1, 1) *blocks*, Electron. Trans. Numer. Anal. 22 (2006), pp. 114–121.
[18] C. Greif and D. Schötzau, *Preconditioners for the discretized time-harmonic Maxwell equations in mixed form*, Numer. Linear Algebra Appl., 14 (2007), pp. 281–297.
[19] C. Keller, N. I. M. Gould, and A. J. Wathen, *Constraint preconditioning for indefinite linear systems*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1300–1317.

[20] L. Lukšan and J. Vlček, *Indefinitely preconditioned inexact Newton method for large sparse equality constrained nonlinear programming problems*, Numer. Linear Algebra Appl., 5 (1998), pp. 219–247.
[21] S. Mehrotra, *On the implementation of a primal-dual interior point method*, SIAM J. Optim., 2 (1992), pp. 575–601.
[22] E. Ng and B. W. Peyton, *Block sparse Cholesky algorithms on advanced uniprocessor computers*, SIAM J. Sci. Comput., 14 (1993), pp. 1034–1056.
[23] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, New York, 1999.
[24] A. Oliveira and D. Sorensen, *A New Class of Preconditioners for Large-Scale Linear Systems from Interior Point Methods for Linear Programming*, Tech. Report CRPC-TR97771, Center for Research on Parallel Computation, Rice University, Houston, TX, 1997.
[25] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.
[26] V. Simoncini and D. B. Szyld, *Theory of inexact Krylov subspace methods and applications to scientific computing*, SIAM J. Sci. Comput., 25 (2003), pp. 454–477.
[27] A. Wächter and L. T. Biegler, *On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming*, Math. Prog., 106 (2006), pp. 25–57.
[28] S. J. Wright, *Primal-Dual Interior-Point Methods*, SIAM, Philadelphia, 1997.
[29] Y. Zhang, *Solving large-scale linear programs by interior-point methods under the MATLAB environment*, Optim. Meth. Softw., 10 (1998), pp. 1–31.

SIAM J. SCI. COMPUT.
Vol. 29, No. 5, pp. 2008–2023
© 2007 Society for Industrial and Applied Mathematics

AN OCTREE METHOD FOR PARAMETRIC IMAGE REGISTRATION*

ELDAD HABER†, STEFAN HELDMANN†, AND JAN MODERSITZKI‡

Abstract. Even for reasonably sized three-dimensional (3D) images, image registration becomes a computationally intensive task. Here, we introduce and explore the concept of octrees for registration which drastically reduces the amount of processed data and thus the computational costs. We show how to map the registration problem onto an octree and present a suitable optimization technique. Furthermore, we demonstrate the performance of the new approach by academic (two-dimensional) as well as real life (3D) examples. These examples indicate that the computational time can be reduced by a factor of 3–4 compared with standard approaches.

Key words. image registration, adaptive discretization, octree, nonlinear optimization

AMS subject classifications. 74S99, 65K05, 65K10

DOI. 10.1137/060662605

1. Introduction.

Image registration is one of today's challenging image processing problems. Given a so-called reference image R and a so-called template image T, the basic idea is to find a [illegible] transformation such that a transformed version of the template image becomes [illegible] to the reference image. Image registration has to be applied whenever images resulting from different times, devices, and/or perspectives need to be compared or integrated; see, e.g., [4, 17] and references therein.

The computation of parametric image registration has two main building blocks. The first is a distance measure $\mathcal{D}$ quantifying distance or similarity of images. The second building block is a typically low dimensional parameterizable search space $\mathcal{V}$ for the desired transformation.

For example, if $\mathcal{V}$ is a linear space, any transformation can be expanded into a sum of basis functions with coefficients γ_j, i.e.,

$$\boldsymbol{\Phi}(\boldsymbol{\gamma})(\mathbf{x}) := \sum_{j=1}^{n} \gamma_j \, \boldsymbol{\psi}_j(\mathbf{x}) \quad \text{with} \quad \boldsymbol{\gamma} := (\gamma_1, \ldots, \gamma_n)^\top \in \mathbb{R}^n. \tag{1.1}$$

Therefore, any transformation in $\mathcal{V}$ is parameterized by a vector $\boldsymbol{\gamma}$.

The choice of the particular subspace can be thought of as a type of regularization, added to the problem to overcome ill-posedness; see, e.g., [17]. Typical examples include rigid, affine linear, and spline transformations [4].

In the following we model the images R and T as smooth functions mapping from $\mathbb{R}^d$ to $\mathbb{R}$ (practically, these functions are obtained by interpolation of the given discrete two-dimensional (2D) or three-dimensional (3D) data). With some abuse of notation, the transformed image is denoted by $T(\boldsymbol{\gamma})$, where

$$T(\boldsymbol{\gamma})(\mathbf{x}) := T(\boldsymbol{\Phi}(\boldsymbol{\gamma})(\mathbf{x})).$$

*Received by the editors June 9, 2006; accepted for publication (in revised form) March 13, 2007; published electronically September 28, 2007.
http://www.siam.org/journals/sisc/29-5/66260.html
†Department of Mathematics and Computer Science, Emory University, Atlanta, GA 30022 (haber@mathcs.emory.edu, heldmann@mathcs.emory.edu).
‡Institute of Mathematics, University of Lübeck, Lübeck, Germany (modersitzki@math.uni-luebeck.de).

A common treatment of parametric registration is based on the following minimization problem [23, 20]: Find a parameter $\boldsymbol{\gamma}^*$ minimizing

$$\mathcal{J}(\boldsymbol{\gamma}) := \mathcal{D}(R, T(\boldsymbol{\gamma})). \tag{1.2}$$

Generally, the optimization problem (1.2) cannot be solved analytically, and thus numerical schemes are required. Although the number of unknowns $\boldsymbol{\gamma}$ might be small, computing the transformed image $T(\boldsymbol{\gamma})$ can be computationally expensive for 3D images. For example, the computation of $T(\boldsymbol{\gamma})$ for a 512^3 voxel image results in more than 10^8 interpolation points to be evaluated. Thus, efficient algorithms are required in order to perform the registration in a reasonable amount of time.

In order to keep the computational time reasonable, it is desirable to reduce the amount of data which has to be processed. A simple reduction can be obtained through down-sampling. However, naive down-sampling results in a loss of possibly important image features. Nevertheless, down-sampling can be safely done in parts of the image with low variations. A proper use of this observation can substantially reduce the storage needed to represent the image and the amount of the computation needed for a registration.

Octrees provide a straightforward approach to such a context-driven data sparsification. The octree data presentation uses few voxels for regions with small variability and many voxels for regions with high variability.

Although generating an octree from a known image is a simple task, solving the registration problem on octrees is far less trivial. The main difficulty is that a proper registration requires the octree representation of $T(\boldsymbol{\gamma}^*)$, where γ^* is a minimizer of (1.2). Because this minimizer is unknown a priori, we cannot generate a reliable octree for $T(\boldsymbol{\gamma}^*)$. We overcome this problem by an updating of the current octree using only local refinement. Although these refinement steps require additional computations, the procedure still reduces the computational costs by one order of magnitude.

Using octrees in image processing is not new. Recent efforts have been made in the field of computer graphics [16, 15]. In particular, the work of Losasso and colleagues on octree discretization demonstrates that images of fine-detail flows and smoke can be represented efficiently and reliably with this type of data structure. Other related work on motion estimation using quadtree splines has been proposed in [21, 22, 14]. In [22, 14] the displacement field was discretized using quadtree splines, and in [21] a 2D surface was embedded in three dimensions and represented using an octree.

Our approach can be thought of as an extension of the work proposed in [21], using octrees for 3D rather than 2D volumes. Furthermore, our approach uses more advanced optimization strategy (merit function and backtracking) in order to ensure global convergence.

Our approach is different and extends the previous work to three dimensions and works with global transformation. Furthermore, we use techniques borrowed from numerical PDEs in order to efficiently solve the minimization problem obtained on the octree.

The paper is organized as follows. In section 2, we discuss the mathematical framework and the octree discretization of the problem (1.2). In section 3 we present an effective numerical optimizing procedure based on a Gauss–Newton-type scheme. In section 4, we propose a locally refined multilevel method for a solution of the nonlinear problem with self-adjusting octree refinement. Finally, in section 5, we

present numerical examples for 2D and 3D images that demonstrate the effectiveness of our algorithm.

The presented theory does not depend on a particular dimension, and our implementation supports 2D and 3D images. However, for ease of presentation, we restrict our explanations to the 2D case and comment on the 3D case whenever an extension is not straightforward.

2. Mathematical framework. In the literature we could not find a unique definition of an octree [13]. Here, we consider an octree as a nonuniform but still highly regular and sparse discretization of a domain $\Omega \subset \mathbb{R}^d$, where d denotes the spatial dimension. To this end, we define an octree as a collection of discretization points $\mathbf{x}_j$ and mesh-sizes h_j. Next, we present a formal description of octrees.

2.1. Octree definition. Let $\Omega = (a_1, b_1) \times (a_2, b_2) \times \cdots \times (a_d, b_d) \subset \mathbb{R}^d$ be a rectangular domain and $h \in \mathbb{R}$ such that $(b_j - a_j)/h \in \mathbb{N}$ for all j. We define an [illegible] $\mathcal{S}_h$ [illegible] Ω [illegible] h as

$$\mathcal{S}_h = \{(\mathbf{x}_j, h_j) \,:\, j = 1, 2, \ldots, N\} \;\subset\; \Omega \times \{h \cdot 2^\ell \,:\, \ell \in \mathbb{N}\}$$

such that the subdomains $B_j := B(\mathbf{x}_j, h_j) := \{\mathbf{y} \,:\, \|\mathbf{x}_j - \mathbf{y}\|_\infty < h_j/2\}$ essentially build a partition of Ω, i.e., $B_j \cap B_k = \emptyset$ for $j \neq k$, and $\bigcup_{j=1}^N \overline{B_j} = \overline{\Omega}$.

Furthermore, we make a few auxiliary definitions for neighborhood relations on octrees, regularity of octrees, and nested octrees that will be useful in the following.

For $(\mathbf{x}_j, h_j) \in \mathcal{S}_h$ the set of [illegible] is defined by

$$\mathrm{Adj}(\mathbf{x}_j, h_j) := \{(\mathbf{x}_k, h_k) \in \mathcal{S}_h \,:\, (\mathbf{x}_j, h_j) \neq (\mathbf{x}_k, h_k) \text{ and } \mathcal{H}^{d-1}(\overline{B}_j \cap \overline{B}_k) \neq 0\},$$

where $\mathcal{H}^{d-1}$ denotes the $(d-1)$-dimensional surface measure.

An octree discretization $\mathcal{S}_h$ is called [illegible] if for all $(\mathbf{x}_j, h_j) \in \mathcal{S}_h$ it holds

$$2^{-m} \;\leq\; h_j/h_k \;\leq\; 2^m \qquad \text{for all direct neighbors } (\mathbf{x}_k, h_k) \in \mathrm{Adj}(\mathbf{x}_j, h_j).$$

In the following we refer to an octree $\mathcal{S}_h$ with the implicit understanding of a first order regular octree discretization of finest mesh-size h.

Note that a zeroth order regular octree $\mathcal{S}_h^0$ is a cell-centered equispaced discretization of Ω. In other words, $\mathcal{S}_h^0$ is the richest possible octree of finest mesh-sizes $h_j = h$ given by

$$\mathcal{S}_h^0 := \left\{(\mathbf{x}, h) \,:\, x_\ell = (a_\ell + h(k_\ell - \tfrac{1}{2})_{\ell=1}^d,\; k_\ell = 1, \ldots, \tfrac{b_\ell - a_\ell}{h}\right\}. \tag{2.1}$$

An octree discretization $\mathcal{S}_h$ is [illegible] in an octree discretization $\mathcal{S}_h'$ if for all $(\mathbf{x}, h) \in \mathcal{S}_h$ there exists an $(\mathbf{x}', h') \in \mathcal{S}_h'$, such that

$$B(\mathbf{x}, h) \cap B(\mathbf{x}', h') = B(\mathbf{x}, h).$$

The finest octree $\mathcal{S}_h^0$ is nested in any other octree discretization of Ω.

There are many possible data structures for a given octree. Here, the tree is stored as a sparse d-dimensional array S. The relative size h_j/h of each cell is stored in the upper left corner of the cell; e.g., the octree shown in Figure 1 is represented by

$$S = \begin{pmatrix} 2 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}.$$

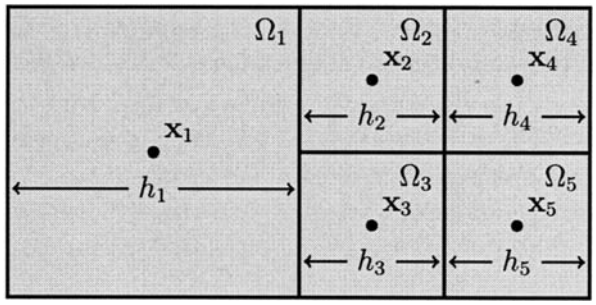


FIG. 1. *An octree discretization $\mathcal{S}_h$ in two dimensions.*

This allows us to use sparse matrix techniques in order to quickly find neighbors, which is the major operation in the discretization process. The data structure is closely related to the one suggested in [13]. As a consequence, all that is needed to address a point of the octree is the sparse array S.

2.2. Octree approximation of discrete images. Consider a discrete 2D image with $m_1 \times m_2$ pixels. Assume that this image has large regions with low variability. Then such an image can be efficiently represented by an octree. An optimal octree is the one that would have the least amount of cells for the given image. However, such an octree is difficult to find (i.e., exponentially difficult) [13]. For practical purposes, we compute a suboptimal octree by Algorithm 1. A similar algorithm is suggested in [13].

ALGORITHM 1. Generate sparse array representation S of an octree for a 2D image T where regions with gray-value differences less than tol are treated as constant.

$[S] \leftarrow$ `generateOctreeFromImage`(T, tol)

set $[m_1, m_2] \leftarrow \text{size}(T)$

Initialize $S(1\!:\!m_1, 1\!:\!m_2) \leftarrow 1$ and $T_{\min} \leftarrow T$, $T_{\max} \leftarrow T$

for $s = 2^{[0:\log_2(\min([m_1,m_2]))]}$

- find $(i_1, i_2) \in [1\!:\!s\!:\!m_1] \times [1\!:\!s\!:\!m_2]$ with $S(i_1, i_2) = s$
- get 4 subblock indices of the superblock stored at (i_1, i_2) with size $2s$: $\mathcal{I} \leftarrow [i_1, i_1 + s] \times [i_2, i_2 + s]$
- compute minimum and maximum in the superblock: $T_{\min}(i_1, i_2) \leftarrow \min(T_{\min}(\mathcal{I}))$ and $T_{\max}(i_1, i_2) \leftarrow \max(T_{\max}(\mathcal{I}))$
- **if** $T_{\max}(i_1, i_2) - T_{\min}(i_1, i_2) \leq \text{tol}$ **then** set $S(i_1, i_2) = 2s$ and $S(\mathcal{I} \setminus (i_1, i_2)) = 0$, **end if**

end for

A central point is that the overall complexity of Algorithm 1 is $\mathcal{O}(N)$, where N is the number of pixels in the image. Furthermore, we added a tolerance parameter tol. It is used as a threshold to decide whether the image is almost constant in a certain region. In our application we have set tol to the estimated noise level in the image.

Although the number of pixels in the octree is image-dependent, we have found that for typical MRI and CT images the number of pixels is reduced to less than 20% of the original size in two dimensions and to less than 10% of the size in three

dimensions, respectively. A representative 2D example, an octree discretization of a CT image, is given in Figure 2. Table 1 shows compression rates and errors obtained for several tolerances.

2.3. Octree image interpolation. For the minimization of (1.2) we must be able to evaluate R and $T(\boldsymbol{\gamma})$ where we model images as smooth functions. Typically, an image comes in digital form as discrete data, and we construct a continuous function by interpolation. Here, we consider the case of linear interpolation where the data is given on an octree. We start our considerations on the full grid $\mathcal{S}_h^0$.

We consider a pixel located at a point $\mathbf{x}$ which is not a grid point and we are interested in the value $T(\mathbf{x})$. To compute this value we use a linear interpolation scheme. In the 2D case, let nw, sw, ne, and se denote the centers of the four adjacent pixels of $\mathbf{x} = (x_1, x_2)$; cf. Figure 3. Then, defining $\xi_j = x_j/h - \lfloor x_j/h \rfloor \in [0,1]$, we have

$$T(\mathbf{x}) \approx \xi_2\Big(T(\mathrm{nw})\xi_1 + T(\mathrm{sw})(1-\xi_1)\Big) + (1-\xi_2)\Big(T(\mathrm{ne})\xi_1 + T(\mathrm{se})(1-\xi_1)\Big). \tag{2.2}$$

Now, assume that we have an octree discretization as plotted in Figure 3(b). In this case we have the octree values of the pixel in SW, SE, NW, and NE = ne, but we do not have values in nw, sw, and se. Nevertheless, since we assume that the image is almost constant over each of the larger cells we can use the prescribed value of the larger cells instead of the values of the original small cell:

$$T(\mathrm{sw}) \approx T(\mathrm{SW}), \quad T(\mathrm{se}) \approx T(\mathrm{SE}), \quad T(\mathrm{nw}) \approx T(\mathrm{NW}), \quad T(\mathrm{ne}) \approx T(\mathrm{NE}). \tag{2.3}$$

Finally, formula (2.2) is used for interpolation.

Our approach can be thought of as a two stage interpolation. First, we prolongate the image to the full grid, and each cell on the full grid obtains the value of its parent octree cell. Next, we use simple bilinear interpolation on the full grid. Note that our image model is continuous, which guarantees continuously differentiable gradients.

LEMMA 1. [illegible] $T \in C^2(\mathbb{R}^d, \mathbb{R})$ [illegible] (2.2) [illegible] (2.3) [illegible] $\mathcal{O}(h)$ [illegible]. We present the proof for $d = 1$; the extension to higher dimensions is straightforward.

Let x be an arbitrary point which lies between two grid points x^- and $x^+ = x^- + h$ such that $x^- \le x \le x^+$ and $\xi := x/h - \lfloor x/h \rfloor \in [0,1]$. From classical interpolation we have

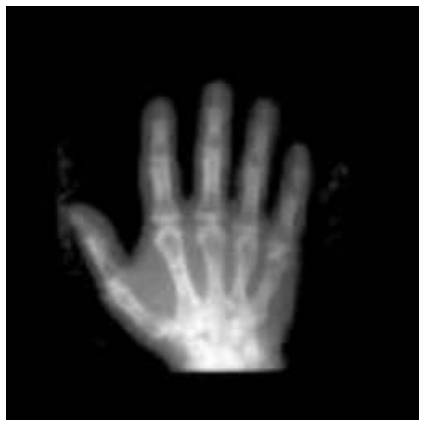
$$T(x) = T(x^- + h\xi) = T(x^-)(1-\xi) + T(x^+)\xi + \mathcal{O}(h^2).$$

Perturbing, for example, the first control point by h, the statement follows from $T(x^- - h) = T(x^-) + \mathcal{O}(h)$ and $0 \le 1 - \xi < 1$. The proof for the perturbed second control point is along the same lines. ∎

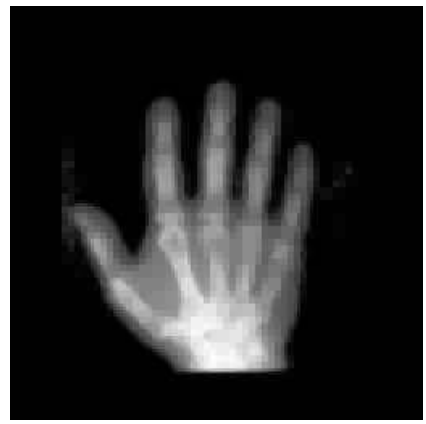
A direct consequence of this interpolation scheme is summarized in the next corollary. This corollary suggests picking a nonvanishing tolerance in Algorithm 1 even if the image data is not perturbed by noise.

COROLLARY 2. [illegible] $\mathcal{S}_h^0$ [illegible] $\mathcal{S}_h^m$ [illegible] m [illegible] $\Omega \subset \mathbb{R}^d$ [illegible] $T \in C^2(\overline{\Omega})$ [illegible] T^0 [illegible] T^m [illegible] $\mathcal{S}_h^0$ [illegible] $\mathcal{S}_h^m$ [illegible]

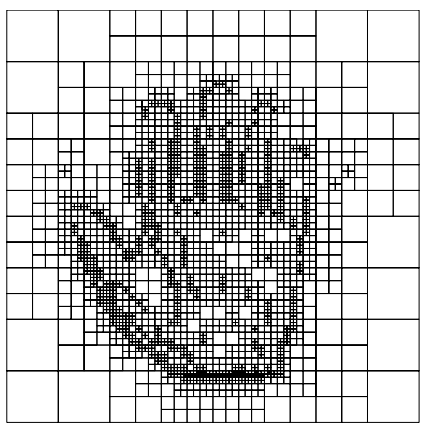
$$|T^m(\mathbf{x}) - T^0(\mathbf{x})| = \mathcal{O}(h).$$



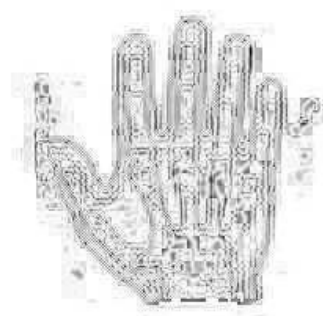
(a) original image T



(b) Octree approximation $\widehat{T}$



(c) octree



(d) absolute error $|T - \widehat{T}|$ (scaled)

FIG. 2. *Approximation of an image with an octree (cf. Table* 1, tol = 15%*).*

TABLE 1

Octree approximations $\widehat{T}$ of an image T with $N = 256^2$ pixels (cf. Figure 2(a)*), with scaled gray-values between* 0 *and* 100 *for various tolerances, $\#\mathcal{S}_h$ the number of octree points, $\#\mathcal{S}_h/\#\mathcal{S}_h^0$ the ratio between octree and full grid discretization points, average error $\frac{1}{N}\|T-\widehat{T}\|_1$, and maximum error $\|T-\widehat{T}\|_\infty$.*

tol	$\#\mathcal{S}_h$	$\#\mathcal{S}_h/\#\mathcal{S}_h^0$	Average error	Maximum error
0%	28,846	44.02%	0.00%	0.00%
1%	21,703	33.11%	0.02%	0.82%
2%	18,202	27.77%	0.06%	1.43%
5%	10,069	15.36%	0.27%	3.75%
10%	4,138	6.31%	0.59%	7.45%
15%	2,806	4.28%	0.77%	11.49%
20%	1,780	2.72%	1.02%	15.49%

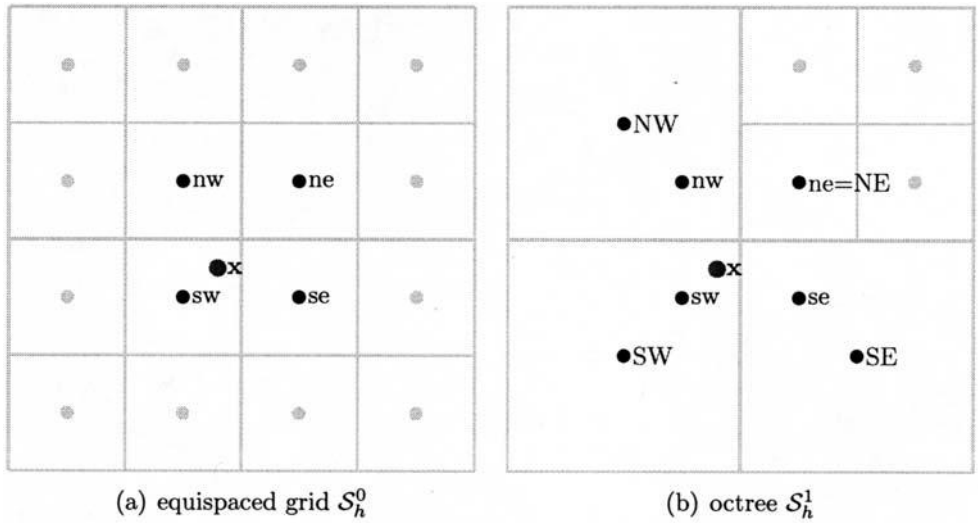


FIG. 3. *Linear interpolation on* $\mathcal{S}_h^0$ (a) *and* $\mathcal{S}_h^1$ (b).

In practice, since the image is noisy anyway, low order interpolation is sufficient. Furthermore, if the image is almost constant over the octree cell, then the coefficient in front of the h^2 term is small and we may not observe deterioration of the interpolation compared with a uniform grid interpolation.

2.4. Octree based distance measures. The octree based distance measure approximation is exemplarily discussed for the L_2-norm based sum of squared differences (SSD) distance measure [4]. Here,

$$\mathcal{D}(R, T(\boldsymbol{\gamma})) = \frac{1}{2}\|T(\boldsymbol{\gamma}) - R\|^2_{L_2(\Omega)} \tag{2.4}$$

$$= \frac{1}{2}\int_\Omega \big(T(\boldsymbol{\gamma})(\mathbf{x}) - R(\mathbf{x})\big)^2 \, d\mathbf{x}$$

$$\approx \frac{1}{2}\sum_{(\mathbf{x}_j, h_j)\in\mathcal{S}_h^0} h_j^d \big(T(\boldsymbol{\gamma})(\mathbf{x}_j) - R(\mathbf{x}_j)\big)^2 \tag{2.5}$$

$$\approx \frac{1}{2}\sum_{(\mathbf{x}_j, h_j)\in\mathcal{S}_h^1} h_j^d \big(T(\boldsymbol{\gamma})(\mathbf{x}_j) - R(\mathbf{x}_j)\big)^2 \tag{2.6}$$

$$= \frac{1}{2}\|T(\boldsymbol{\gamma}) - R\|^2_H =: D(\mathcal{S}_h, R, T(\boldsymbol{\gamma})) \tag{2.7}$$

with a weighted l_2-norm, where the diagonal weighting matrix is given by $H = \mathrm{diag}(h_j^d)$ with the cell sizes h_j^d, where (2.5) is the standard midpoint quadrature rule and (2.6) is its approximation on a first order octree. Note that $h_j = h$ in (2.5), whereas h_j can vary in (2.6). In (2.7), we make the discretized distance measure explicitly dependent on the octree. In our implementation, we use the octree generated for the transformed image $T(\boldsymbol{\gamma})$. Therefore, it is important to note that generally an octree discretization depends on $\boldsymbol{\gamma}$, i.e., $\mathcal{S}_h = \mathcal{S}_h(\boldsymbol{\gamma})$.

3. Optimization. Ideally, our aim is to minimize

$$J(\boldsymbol{\gamma}) := D(\mathcal{S}_h^0, R, T(\boldsymbol{\gamma})), \tag{3.1}$$

where $\mathcal{S}_h^0$ is the full equispaced grid (2.1). To do so efficiently, we replace this problem by a problem on a sparse and computationally less involved octree discretization. A straightforward approach is to substitute (3.1) by

$$J(\boldsymbol{\gamma}) := D(\mathcal{S}_h(\boldsymbol{\gamma}), R, T(\boldsymbol{\gamma})) \tag{3.2}$$

such that a minimizer $\boldsymbol{\gamma}^*$ leads to a transformed image $T(\boldsymbol{\gamma}^*)$ for which (2.5) can be replaced by (2.6) up to a small error.

One approach to minimizing (3.2) is to consider $\boldsymbol{\gamma} = \boldsymbol{\gamma}(t)$ as a time-dependent function and to embed the problem into a time stepping process such as the dynamical system

$$\boldsymbol{\gamma}_t = -\nabla_{\boldsymbol{\gamma}} J(\boldsymbol{\gamma}) = -\nabla_{\boldsymbol{\gamma}} D(\mathcal{S}_h(\boldsymbol{\gamma}), R, T(\boldsymbol{\gamma})).$$

Then, one can use methods that have been developed for adaptive mesh refinement for time-dependent problems when integrating to steady state. At each step one estimates both $\boldsymbol{\gamma}$ and the octree $\mathcal{S}_h(\boldsymbol{\gamma})$. However, such an approach is inefficient in the context of parametric registration since we are not interested in the dynamics of the process and evaluating the octree at each time step is not necessary.

Since the octree in (3.2) depends on $\boldsymbol{\gamma}$, the minimization is far from trivial. Therefore, we propose a two phase procedure similar to coordinate search. Starting with $k = 0$ and an initial parameter vector $\boldsymbol{\gamma}^{(k)}$, the first step is to compute the corresponding octree $\mathcal{S}_h^{(k)} := \mathcal{S}_h(\boldsymbol{\gamma}^{(k)})$. The next step is to compute a minimizer $\boldsymbol{\gamma}^{(k+1)}$ of

$$J^{(k)}(\boldsymbol{\gamma}) := D(\mathcal{S}_h^{(k)}, R, T(\boldsymbol{\gamma}))$$

for the (fixed) octree $\mathcal{S}_h^{(k)}$. These two steps are repeated to convergence. The strategy is very similar to refinement strategies used for adaptive grid problems in ordinary differential equations and partial differential equations; see, e.g., [1, 2, 8] and references therein. The algorithm is summarized in Algorithm 2.

A crucial point in our method is to ensure decrease towards a solution on the full grid. This is done by a backtracking line search.

Assume that $\mathcal{S}_h^{(k)}$ is an octree valid for $\boldsymbol{\gamma}^{(k)}$ and $\widehat{\boldsymbol{\gamma}}$ is a minimizer of $J^{(k)}$. A naive approach is to continue the iteration by setting $\boldsymbol{\gamma}^{(k+1)} := \widehat{\boldsymbol{\gamma}}$. However, this procedure may not guarantee convergence to a minimizer of 3.1. In fact, while $\widehat{\boldsymbol{\gamma}}$ minimizes $J^{(k)}$, it may increase the value of J. We therefore use a (weak) line search for computing the new transformation and the new octree. Algorithm 2 implies that we use the octree in order to compute a search direction $\delta\boldsymbol{\gamma}$ and then use this direction to decrease the objective function J, i.e., $D(\mathcal{S}_h^0, \cdot)$, on the full grid. Therefore, we refer to Algorithm 2 as outer iteration and we speak of solving for $\widehat{\boldsymbol{\gamma}}$ on a fixed octree as inner iteration.

The obvious question of whether $\delta\boldsymbol{\gamma}$ computed on a coarse octree leads to a descent direction for the full octree is positively answered in the literature [18, 11, 3]. In particular, the work of [3] gives precise conditions on the coarse grid function. Nevertheless, for our problem, it is impossible to verify that these conditions actually are fulfilled. Summarizing the above work, loosely speaking, the essential guideline for constructing a coarse grid approximation is that it must sufficiently approximate the convexity properties of the functional at finest resolution.

Finally, we have two natural stopping criteria. First, if $\mathcal{S}_h^{(k+1)} = \mathcal{S}_h^{(k)}$, we set $\boldsymbol{\gamma}^* \leftarrow \boldsymbol{\gamma}^{(k)}$ and stop. Second, if we are unable to decrease the value of the objective function in the direction $\delta\boldsymbol{\gamma}$, then we set $\boldsymbol{\gamma}^* \leftarrow \boldsymbol{\gamma}^{(k)}$ and stop.

ALGORITHM 2. Outer iteration to minimize $J(\boldsymbol{\gamma}) := D(\mathcal{S}_h^0, R, T(\boldsymbol{\gamma}))$.

$\boldsymbol{\gamma}^* \leftarrow \texttt{minimizeJ}(T, R, \boldsymbol{\gamma}^{(0)}, h)$

while not converged

- for given $\boldsymbol{\gamma}^{(k)}$ generate $\mathcal{S}_h^{(k)}$ such that $D(\mathcal{S}_h^{(k)}, \boldsymbol{\gamma}^{(k)}) \approx J(\boldsymbol{\gamma}^{(k)})$
- compute a minimizer $\widehat{\boldsymbol{\gamma}}$ of $J^{(k)} := D(\mathcal{S}_h^{(k)}, \cdot\,)$
- set $\delta\boldsymbol{\gamma} \leftarrow \widehat{\boldsymbol{\gamma}} - \boldsymbol{\gamma}^{(k)}$
- perform line search on $\mathcal{S}_h^0$: find λ such that $J(\boldsymbol{\gamma}^{(k)} + \lambda\delta\boldsymbol{\gamma}) < J(\boldsymbol{\gamma}^{(k)})$
- update $\boldsymbol{\gamma}^{(k+1)} \leftarrow \boldsymbol{\gamma}^{(k)} + \lambda\delta\boldsymbol{\gamma}$,

end while

return $\boldsymbol{\gamma}^* \leftarrow \boldsymbol{\gamma}^{(k+1)}$.

3.1. Solving for $\boldsymbol{\gamma}$ on an octree. To compute a minimizer $\widehat{\boldsymbol{\gamma}}$ of $D(\mathcal{S}_h, \cdot\,)$ on a given fixed octree $\mathcal{S}_h$, we use a Gauss–Newton-like method. Note that using linear interpolation we obtain a piecewise constant Jacobian. Nevertheless, using the Gauss–Newton method with backtracking we could obtain linear convergence.

Since the octree is constant in this step, we may rewrite the distance function (2.7) by

$$D(R, T(\boldsymbol{\gamma})) = D(\mathcal{S}_h, R, T(\boldsymbol{\gamma})).$$

To use the Gauss–Newton method, we first linearize $T(\boldsymbol{\gamma})$:

$$T(\boldsymbol{\gamma} + \delta\boldsymbol{\gamma}) \approx T(\boldsymbol{\gamma}) + T_{\boldsymbol{\gamma}}(\boldsymbol{\gamma})\ \delta\boldsymbol{\gamma},$$

where $T_{\boldsymbol{\gamma}}$ is the Jacobian matrix of T w.r.t. $\boldsymbol{\gamma}$. To compute $T_{\boldsymbol{\gamma}}$ we use the chain rule. Discretizing (1.1) the displacement everywhere, $\boldsymbol{\phi}$ can be written as

$$\boldsymbol{\phi} = P\boldsymbol{\gamma},$$

where each column of the matrix P holds the discretized grid function of the basis function $\boldsymbol{\psi}_j \quad j = 1, \ldots, n$. To differentiate w.r.t. $\boldsymbol{\phi}$ we differentiate the interpolation formula (2.2) directly. We then multiply with P to complete the evaluating of the Jacobian.

We then use the linearization with the distance function D,

$$D(R, T(\boldsymbol{\gamma} + \delta\boldsymbol{\gamma})) \approx D(R, T(\boldsymbol{\gamma}) + T_{\boldsymbol{\gamma}}(\boldsymbol{\gamma})\delta\boldsymbol{\gamma}).$$

Differentiating w.r.t. $\delta\boldsymbol{\gamma}$, we obtain the following equation which can be solved for $\delta\boldsymbol{\gamma}$:

$$T_{\boldsymbol{\gamma}}(\boldsymbol{\gamma})^\top\ D_T(R, T(\boldsymbol{\gamma}) + T_{\boldsymbol{\gamma}}(\boldsymbol{\gamma})\delta\boldsymbol{\gamma}) = 0, \tag{3.3}$$

where D_T is the gradient of the distance measure w.r.t. T.

For example, for the SSD distance measure discussed in section 2.4, $D(R, T(\boldsymbol{\gamma})) = \frac{1}{2}\|T(\boldsymbol{\gamma}) - R\|_H^2$, we end up with the usual Gauss–Newton scheme [6], where $\delta\boldsymbol{\gamma}$ is obtained by solving the system

$$T_{\boldsymbol{\gamma}}^\top H(T_{\boldsymbol{\gamma}}\ \delta\boldsymbol{\gamma} + T - R) = 0.$$

However, for more complicated distance measures (e.g., mutual information [5, 24] or normalized gradient fields [9]) an inexact solution of this system can be obtained using an inexact Newton method.

After solving for $\delta\boldsymbol{\gamma}$, we update $\boldsymbol{\gamma}$ using a weak line search which ensures sufficient reduction in the value of the objective function [19]. The algorithm for computing a single inner iteration is summarized in Algorithm 3.

ALGORITHM 3. Inner iteration to minimize $D(\mathcal{S}_h, R, T(\boldsymbol{\gamma}))$ w.r.t. $\boldsymbol{\gamma}$.

$\widehat{\boldsymbol{\gamma}} \leftarrow$ `minimizeJonFixedOctree`$(\mathcal{S}_h, R, T, \boldsymbol{\gamma})$

while not converge
- compute $T(\boldsymbol{\gamma})$ and $T_{\boldsymbol{\gamma}}(\boldsymbol{\gamma})$
- approximately solve (3.3) for $\delta\boldsymbol{\gamma}$
- perform line search:
  find λ such that $D(\mathcal{S}_h, R, T(\boldsymbol{\gamma} + \lambda\delta\boldsymbol{\gamma})) < D(\mathcal{S}_h, R, T(\boldsymbol{\gamma}))$
- update $\boldsymbol{\gamma} \leftarrow \boldsymbol{\gamma} + \lambda\delta\boldsymbol{\gamma}$

end while
return $\widehat{\boldsymbol{\gamma}} \leftarrow \boldsymbol{\gamma}$

3.2. Generating an octree for $\boldsymbol{\gamma}$. In the previous subsection we discussed how to compute a minimizer $\boldsymbol{\gamma}$ for a particular given octree. In this subsection we assume that $\boldsymbol{\gamma}$ is given and we discuss how to generate an appropriate octree.

The key for the computation of a new octree is that the transformed image must be well represented on the grid; i.e., we need to sufficiently estimate the distance function $\mathcal{D}(R, T(\boldsymbol{\gamma}))$. In order to be more specific, in this subsection we assume that $\mathcal{D}$ is the SSD distance measure; cf. section 2.4.

Since we try to approximate the integral of the difference between $T(\boldsymbol{\gamma})$ and R, the octree has to be constructed based on both $T(\boldsymbol{\gamma})$ and R. Therefore, we first evaluate $T(\boldsymbol{\gamma})$ and R on the full grid. We then use Algorithm 1 to generate octrees $\mathcal{S}_h^T$ and $\mathcal{S}_h^R$ for $T(\boldsymbol{\gamma})$ and R. Finally, we join the octrees. As a result, we obtain an octree nested in both $\mathcal{S}_h^T$ and $\mathcal{S}_h^R$; cf. section 2.1. The procedure is summarized in Algorithm 4.

ALGORITHM 4. Generate sparse octree representation S (2D).

$S \leftarrow$ `generateJointOctree`$(T, R, \boldsymbol{\gamma})$
- compute $T(\boldsymbol{\gamma})$ and R on the full grid Ω_h
- compute octree representations S_T and S_R for $T(\boldsymbol{\gamma})$ and R using Algorithm 2.2
- join S_T and S_R, i.e.,
  - find (i_1, i_2) where $S_R(i_1, i_2) \neq 0$ and $S_T(i_1, i_2) \neq 0$ and set $S(i_1, i_2) \leftarrow \min\{S_R(i_1, i_2), S_T(i_1, i_2)\}$
  - find (i_1, i_2) where $S_R(i_1, i_2) \neq 0$ and $S_T(i_1, i_2) = 0$ and set $S(i_1, i_2) \leftarrow S_R(i_1, i_2)$
  - find (i_1, i_2) where $S_R(i_1, i_2) = 0$ and $S_T(i_1, i_2) \neq 0$ and set $S(i_1, i_2) \leftarrow S_T(i_1, i_2)$

4. Multilevel continuation. To overcome nonlinearities and accelerate computations, we use a multilevel strategy. A multilevel method for image registration is not new [7, 10, 12]. However, modification of the standard algorithm is required to effectively use the octree grid. Our multilevel algorithm uses the octree structure directly in order to gradually refine the grid only in areas where a refinement is required.

We start on a coarsest octree with finest mesh-size $H = 2^L h$, which is a few multiples of the smallest mesh-size h. We then solve the optimization problem using Algorithm 2 to obtain a coarse grid solution $\boldsymbol{\gamma}_H$. We then move to the next finer

ALGORITHM 5. Multilevel solve for $\boldsymbol{\gamma}$.

$[\boldsymbol{\gamma}] \leftarrow \texttt{MLSolve}(T, R, \boldsymbol{\gamma}, h, L)$

set $H \leftarrow 2^L h$

initialize $\boldsymbol{\gamma}$ and S_H

while $H > h$

- use Algorithm 2 with initial guess $\boldsymbol{\gamma}_{2H}$ to find $\boldsymbol{\gamma}_H$ such that $D(\mathcal{S}_H^0, \boldsymbol{\gamma}_H) =$ min:
  $\boldsymbol{\gamma}_H \leftarrow \texttt{minimizeJ}(T, R, \boldsymbol{\gamma}_{2H}, H)$
- set $H \leftarrow H/2$

end while

level. We set the finest mesh-size H to $H/2$ and use the coarse grid solution $\boldsymbol{\gamma}_H$ as an initial guess to the optimization problem for the finest mesh-size $H/2$. We repeat this procedure until we arrive at the smallest grid size h. The algorithm is summarized in Algorithm 5.

It is important to note that combining our multilevel strategy with the optimization strategy naturally results in only [illegible]. This is because the optimization on every level starts with an [illegible] of the refined previous solution and this automatically refines the grid only in high variability areas.

5. Examples. In this section we demonstrate the effectiveness of our approach on a few examples. We use a 2D example which can be easily visualized and a 3D example to further demonstrate the computational saving when solving registration problems in three dimensions. In both examples we use the SSD distance measure and optimize w.r.t. affine transformations.

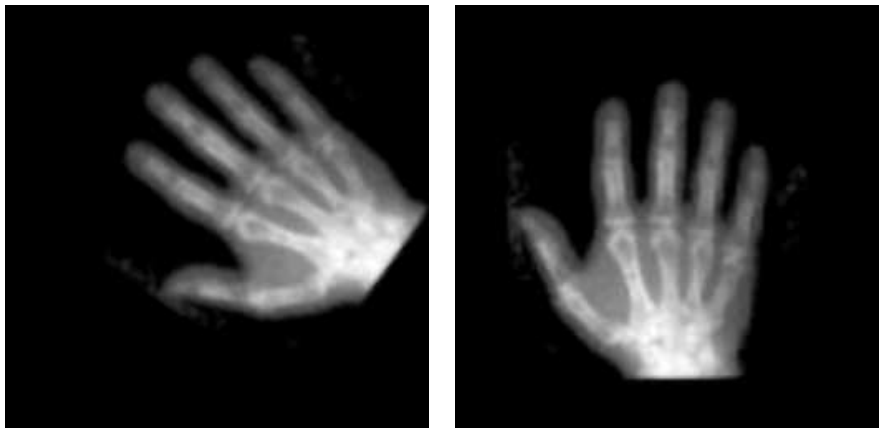
5.1. 2D registration of hand CT. Our first numerical experiment is matching 2D images of hands plotted in Figure 4. The reference image is a rotated and shifted version of the template image. Thus, the transformation parameters are explicitly known, and we should generally be able to recover them with registration. In the following we denote the true parameter by $\boldsymbol{\gamma}_{\text{true}}$ and parameters obtained from registration by $\boldsymbol{\gamma}_{\text{opt}}$. As stopping criteria for the optimization, we considered the difference between subsequent iterates where we terminated our algorithm when $\|\boldsymbol{\gamma}^{(k)} - \boldsymbol{\gamma}^{(k+1)}\|_\infty < 10^{-2}$.

The multilevel optimization process is summarized in Table 2, and a few octree grids obtained in the process of optimization are plotted in Figure 5.

For this problem, the gray-values of the image were in the range $[0, 750]$. Therefore, a tolerance of 1% in the gray-value dynamics corresponds to an absolute value of 7.5.

In Table 3 we present our results for several tolerances for the octrees. Basically, we observe that we are able to recover the true parameter. An error in the parameters of $\|\boldsymbol{\gamma}_{\text{opt}} - \boldsymbol{\gamma}_{\text{true}}\|_\infty \approx 10^{-4}$ corresponds to a misalignment of approximately 0.05 pixels which is visually indistinguishable from the reference image. Therefore, the results show that we are able to sufficiently accurately recover the true parameters with approximately only a fifth of a full discretization (cf. Table 3, tol $\leq$ 15%). Nevertheless, we did not observe a speedup in wall-clock time, which suggests that the computation overhead from the regridding and line searches coming from dealing with nonuniform grids is too large compared to the problem size.

In Table 4 we present the outcomes of our algorithm after each level in the multi-



(a) R (b) T

FIG. 4. *X-ray images used for* $2D$ *registration example.*

TABLE 2
Optimization path for the $2D$ *problem and* $\mathrm{tol} = 5\%$.

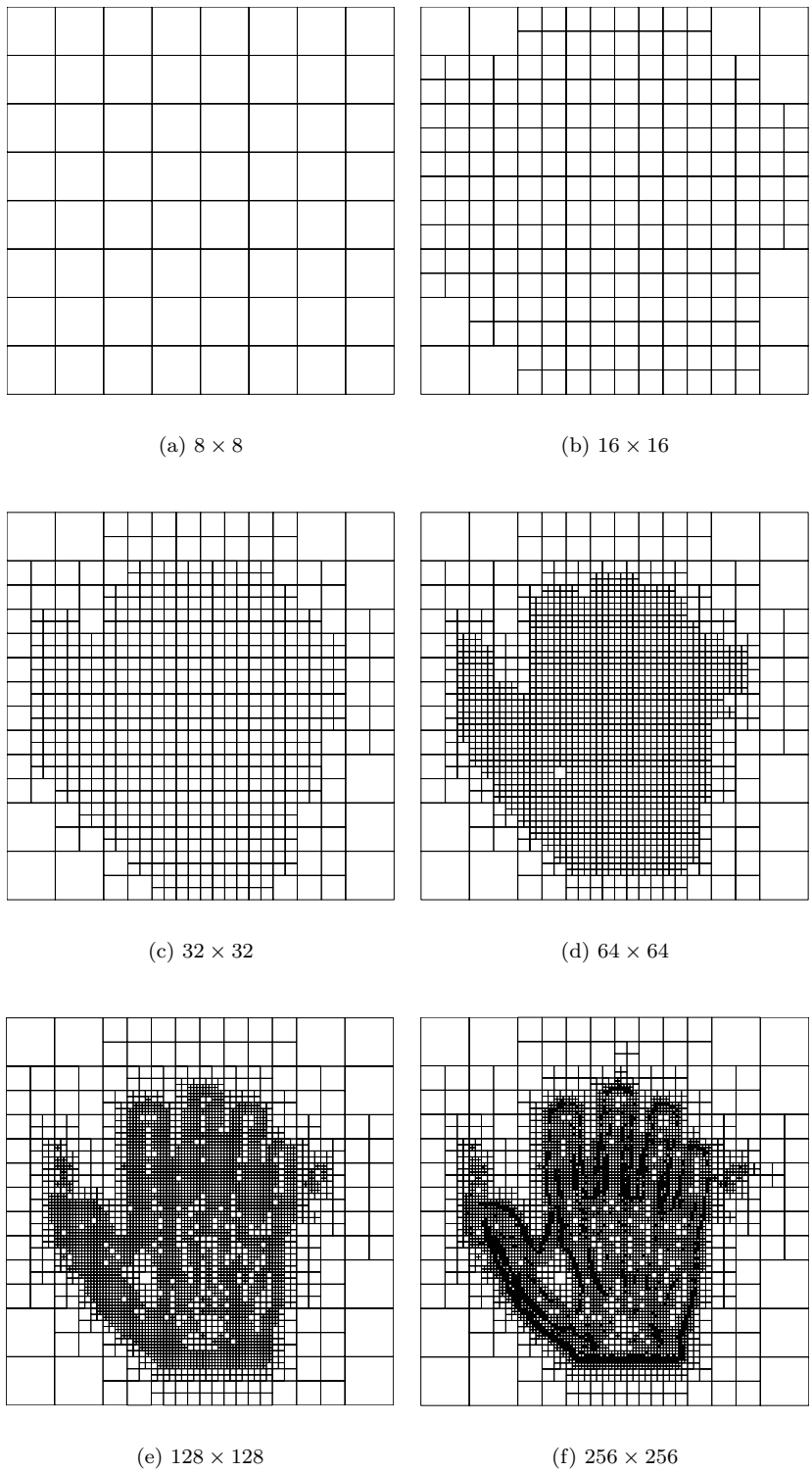
Level	$\mathcal{S}_h^0$	Outer iterations	Inner iterations
1	8×8	2	15
2	16×16	2	8
3	32×32	2	5
4	64×64	2	4
5	128×128	2	3
6	256×256	2	3

level process. As we can see, the results for the octree discretizations and the full grid are almost identical, and the finest resolution is needed to obtain the final accuracy. Furthermore, the accuracy did not improve from the 8×8 to the 32×32 level, which also suggests that we started with a too coarse grid.

5.2. 3D registration of head MRI. In our second experiment we register 3D MRI images of the head. The images were taken from the BrainWeb database and interpolated to a size of $256 \times 256 \times 256$ voxels. To create a template we rotated the reference by 20° around the center of the image in the x-, y-, and z-directions and finally made a diagonal shift by 20 pixels. The reference and template are shown in Figures 6(a) and 6(b), respectively.

The multilevel optimization path with a tolerance of 5% of the gray-value dynamics is plotted in Table 5, Table 6 lists the accuracy for the recovered parameter for various tolerances, and Table 7 shows the outcomes in the multilevel process.

Basically, we observe exactly the same behavior as in the 2D example w.r.t. the accuracy of the recovered parameters. The main difference between the two examples is the problem size. Here, we observe a speedup of a factor 3–4 when using an octree discretization versus the full grid (cf. Table 6, tol = 5–20%). Thus, the problem size is large enough such that the computational saving from the octree approach versus a full grid discretization compensates the introduced algorithmic overhead from a nonregular discretization.



(a) 8×8 (b) 16×16

(c) 32×32 (d) 64×64

(e) 128×128 (f) 256×256

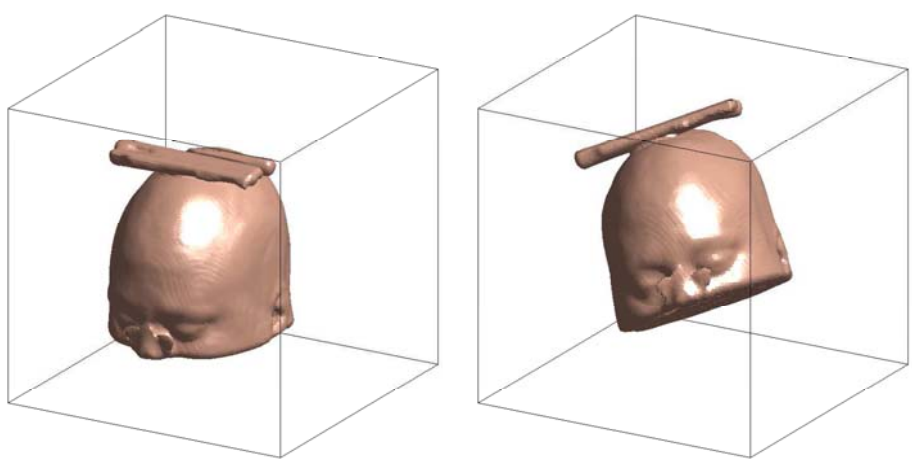
FIG. 5. *Octrees used on different levels.*

TABLE 3
Results for the 2D problem for increasing tol.

tol	$\|\gamma_{\text{opt}} - \gamma_{\text{true}}\|_\infty$	Time	$\#\mathcal{S}$	$\#\mathcal{S}/\#\mathcal{S}_0$
Full grid	6.72e-07	4.33s	65536	100.00%
0%	2.72e-13	5.00s	28309	43.20%
1%	2.28e-05	4.99s	22270	33.98%
2%	1.38e-04	4.95s	19144	29.21%
5%	8.55e-04	5.15s	11653	17.78%
10%	1.50e-03	5.31s	4918	7.50%
15%	2.72e-03	5.51s	3022	4.61%
20%	1.63e-02	5.51s	2101	3.21%
30%	4.15e-02	5.15s	922	1.41%
40%	5.52e-02	5.05s	583	0.89%
50%	2.31e+00	7.12s	283	0.43%

TABLE 4
$\|\gamma_{\text{opt}} - \gamma_{\text{true}}\|_\infty$ *in the multilevel process for the 2D problem.*

	Level					
tol	8×8	16×16	32×32	64×64	128×128	256×256
Full grid	1.38e-01	3.49e-01	1.10e-01	4.32e-02	1.42e-02	6.72e-07
0%	1.10e-01	3.37e-01	1.13e-01	4.33e-02	1.42e-02	2.72e-13
1%	1.14e-01	3.79e-01	1.08e-01	4.02e-02	1.41e-02	2.28e-05
2%	1.17e-01	3.79e-01	1.08e-01	4.10e-02	1.41e-02	1.38e-04
5%	1.13e-01	4.01e-01	1.06e-01	4.02e-02	1.60e-02	8.55e-04
10%	1.16e-01	2.89e-01	1.05e-01	4.23e-02	1.51e-02	1.50e-03
15%	1.12e-01	3.10e-01	1.21e-01	3.54e-02	1.69e-02	2.72e-03
20%	9.64e-02	3.07e-01	1.17e-01	3.61e-02	1.77e-02	1.63e-02
30%	2.91e-01	2.21e-01	1.05e-01	4.52e-02	4.38e-02	4.15e-02
40%	1.15e-01	2.29e-01	9.59e-02	6.97e-02	5.28e-02	5.52e-02
50%	2.39e+00	3.42e+00	2.36e+00	2.36e+00	2.20e+00	2.31e+00



(a) R (b) T

FIG. 6. *MRI images used in the 3D example.*

TABLE 5
Optimization path for the 3D problem with tol = 5%.

Level	$\mathcal{S}_h^0$	Outer iterations	Inner iterations
1	$8 \times 8 \times 8$	3	27
2	$16 \times 16 \times 16$	3	17
3	$32 \times 32 \times 32$	2	5
4	$64 \times 64 \times 64$	3	6
5	$128 \times 128 \times 128$	2	4
6	$256 \times 256 \times 256$	2	3

TABLE 6
Results for the 3D problem for increasing tol.

tol	$\|\gamma_{\text{opt}} - \gamma_{\text{true}}\|_\infty$	Time	$\#\mathcal{S}$	$\#\mathcal{S}/\#\mathcal{S}_0$
Full grid	2.92e-04	3484.85s	16777216	100.00 %
0%	5.11e-07	1513.24s	2314390	13.79 %
1%	1.25e-06	1474.93s	2186850	13.03 %
2%	1.77e-05	1419.62s	2034243	12.13 %
5%	4.42e-04	1017.00s	1615657	9.63 %
10%	1.22e-03	912.85s	1241696	7.40 %
15%	2.09e-03	841.06s	976053	5.82 %
20%	2.31e-03	821.60s	771191	4.60 %
30%	1.22e-02	751.91s	500865	2.99 %
40%	9.57e-03	700.83s	340110	2.03 %
50%	1.47e-02	671.41s	223700	1.33 %

TABLE 7
$\|\gamma_{\text{opt}} - \gamma_{\text{true}}\|_\infty$ *in the multilevel process for the 3D problem.*

	Level					
tol	8^3	16^3	32^3	64^3	128^3	256^3
Full grid	2.37e+00	1.05e+00	3.75e-01	1.71e-01	4.63e-02	2.92e-04
0%	2.37e+00	1.05e+00	3.74e-01	1.71e-01	4.61e-02	5.11e-07
1%	2.37e+00	1.05e+00	3.63e-01	1.66e-01	4.59e-02	1.25e-06
2%	2.37e+00	1.05e+00	3.54e-01	1.65e-01	4.59e-02	1.77e-05
5%	2.38e+00	1.05e+00	3.45e-01	1.63e-01	4.58e-02	4.42e-04
10%	2.39e+00	1.05e+00	3.41e-01	1.61e-01	4.56e-02	1.22e-03
15%	2.37e+00	1.05e+00	3.36e-01	1.58e-01	4.51e-02	2.09e-03
20%	2.33e+00	1.07e+00	3.36e-01	1.58e-01	4.46e-02	2.31e-03
30%	2.35e+00	1.04e+00	3.23e-01	1.61e-01	3.83e-02	1.22e-02
40%	2.35e+00	9.48e-01	3.16e-01	1.71e-01	4.44e-02	9.57e-03
50%	1.64e+00	9.23e-01	3.21e-01	1.77e-01	4.73e-02	1.47e-02

6. Summary and final comments. In this paper we have developed a variational octree image registration technique. We have demonstrated that by using such a technique we are able to substantially accelerate parametric image registration problems.

While we observe major computational savings for parametric registration, we expect to have much better results for nonparametric registration where large sparse linear systems are solved for the displacement.

REFERENCES

[1] U. M. ASCHER, R. M. M. MATTHEIJ, AND R. D. RUSSELL, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*, SIAM, Philadelphia, 1995.

[2] R. BECKER AND R. RANNACHER, *An optimal control approach to a posteriori error estimation in finite element methods*, Acta Numer., 10 (2001), pp. 1–102.

[3] A. Borzi, *On the convergence of the MG/OPT method*, PAMM, 5 (2005), pp. 735–736.
[4] L. G. Brown, *A survey of image registration techniques*, Surveys, 24 (1992), pp. 325–376.
[5] A. Collignon, A. Vandermeulen, P. Suetens, and G. Marchal, *Multi-modality medical image registration based on information theory*, in Information Processing in Medical Imaging, Computational Imaging and Vision 3, Kluwer Academic Publishers, Norwell, MA, 1995, pp. 263–274.
[6] J. E. Dennis, Jr., and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, SIAM, Philadelphia, 1996.
[7] M. Droske and M. Rumpf, *A variational approach to nonrigid morphological image registration*, SIAM J. Appl. Math., 64 (2004), pp. 668–687.
[8] K. Eriksson, D. Estep, P. Hansbo, and C. Johnson, *Introduction to adaptive methods for differential equations*, in Acta Numerica, 1995, Acta. Numer., Cambridge University Press, Cambridge, UK, 1995, pp. 105–158.
[9] E. Haber and J. Modersitzki, *Intensity gradient based registration and fusion of multi-modal images*, Methods of Information in Medicine, 46 (2007), pp. 292–299.
[10] E. Haber and J. Modersitzki, *A multilevel method for image registration*, SIAM J. Sci. Comput., 27 (2006), pp. 1594–1607.
[11] W. Hackbusch and A. Reusken, *Analysis of a damped nonlinear multilevel method*, Numer. Math., 55 (1989), pp. 225–246.
[12] S. Henn and K. Witsch, *Multimodal image registration using a variational approach*, SIAM J. Sci. Comput., 25 (2003), pp. 1429–1447.
[13] G. R. Hjaltason and H. Samet, *Speeding up construction of PMR quadtree-based spatial indexes*, VLDB J., 11 (2002), pp. 109–137.
[14] S. Krüger and A. Calway, *Image registration using multiresolution frequency domain correlation*, in British Machine Vision Conference, British Machine Vision Association, 1998, pp. 316–325.
[15] F. Losasso, R. Fedkiw, and S. Osher, *Spatially adaptive techniques for level set methods and incompressible flow*, Comput. & Fluids, 35 (2006), pp. 457–462.
[16] F. Losasso, F. Gibou, and R. Fedkiw, *Simulating water and smoke with an octree data structure*, in Proceedings of SIGGRAPH, ACM, 2004, pp. 457–462.
[17] J. Modersitzki, *Numerical Methods for Image Registration*, Oxford University Press, New York, 2004.
[18] S. G. Nash, *A multigrid approach to discretized optimization problems*, Optim. Methods Softw., 14 (2000), pp. 99–116.
[19] J. Nocedal and S. Wright, *Numerical Optimization*, Springer, New York, 1999.
[20] J. P. W. Pluim, J. B. A. Maintz, and M. A. Viergever, *Mutual-information-based registration of medical images: A survey*, IEEE Trans. Med. Imaging, 22 (2003), pp. 986–1004.
[21] R. Szeliski and S. Lavallée, *Matching 3-D anatomical surfaces with nonrigid deformations using octree-splines*, Internat. J. Comput. Vision, 18 (1996), pp. 171–186.
[22] R. Szeliski and H. Y. Shum, *Motion estimation with quadtree splines*, IEEE Trans. Pattern Anal. Machine Intelligence, 18 (1996), pp. 1199–1210.
[23] P. Thevenaz and M. Unser, *Optimization of mutual information for multiresolution image registration*, IEEE Trans. Image Process., 9 (2000), pp. 2083–2089.
[24] P. A. Viola, *Alignment by Maximization of Mutual Information*, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA, 1995.

SIAM J. SCI. COMPUT.
Vol. 29, No. 5, pp. 2024–2044
© 2007 Society for Industrial and Applied Mathematics

A NOTE ON GMRES PRECONDITIONED BY A PERTURBED LDL^T DECOMPOSITION WITH STATIC PIVOTING*

M. ARIOLI†, I. S. DUFF‡, S. GRATTON§, AND S. PRALET¶

Abstract. A strict adherence to threshold pivoting in the direct solution of symmetric indefinite problems can result in substantially more work and storage than forecast by a sparse analysis of the symmetric problem. One way of avoiding this is to use static pivoting where the data structures and pivoting sequence generated by the analysis are respected and pivots that would otherwise be very small are replaced by a user defined quantity. This can give a stable factorization but of a perturbed matrix. The conventional way of solving the sparse linear system is then to use iterative refinement, but there are cases where this fails to converge. In this paper, we discuss the use of more robust iterative methods, namely GMRES and flexible GMRES (FGMRES). We show both theoretically and experimentally that both approaches are more robust than iterative refinement and furthermore that FGMRES is far more robust than GMRES and that, under reasonable hypotheses, FGMRES is backward stable. We also show how restarted variants can be beneficial, although again the GMRES variant is not as robust as FGMRES.

Key words. augmented systems, sparse matrices, multifrontal methods, GMRES, flexible GMRES, static pivoting, roundoff error

AMS subject classifications. 65F05, 65F50, 65F10, 65G50

DOI. 10.1137/060661545

1. Introduction.

This paper is concerned with solving the set of linear equations

$$Ax = b, \tag{1.1}$$

where the coefficient matrix $A \in \mathbb{R}^{n\times n}$ is a symmetric indefinite sparse matrix. Our hope is to solve this system using a direct method, but sometimes the cost of doing this is too high in terms of time or memory. We have therefore looked at the possibility of using static pivoting to avoid these problems which are particularly acute if the matrix is highly indefinite as, for example, can happen for saddle-point problems.

We will use a multifrontal approach for our direct method. In this approach, we first determine an order for choosing pivots based on the sparsity structure of A and we then accommodate further pivoting for numerical stability during the subsequent numerical factorization phase. When the matrix is highly indefinite, the resulting pivot sequence used in the numerical factorization can differ substantially from that predicted by the analysis step. A simple way to avoid this problem is to force the elimination through static pivoting.

*Received by the editors May 31, 2006; accepted for publication (in revised form) March 13, 2007; published electronically September 28, 2007.
http://www.siam.org/journals/sisc/29-5/66154.html

†Rutherford Appleton Laboratory, Chilton, Didcot, Oxon, OX11 0QX, UK (m.arioli@rl.ac.uk). This author's work was supported by EPSRC grant GR/S42170/01.

‡Rutherford Appleton Laboratory, Chilton, Didcot, Oxon, OX11 0QX, UK (i.s.duff@rl.ac.uk) and CERFACS, 42 rue G. Coriolis, 31057 Toulouse, France. This author's work was supported by EPSRC grant GR/S42170/01.

§CERFACS, 42 rue G. Coriolis, 31057 Toulouse, France (Serge.Gratton@cerfacs.fr).

¶SAMTECH s.a. LIEGE Science Park, Rue des Chasseurs-Ardennais, 8, B-4031 Liège, Belgium (stephane.pralet@samcef.com). Part of the work of this author was supported by ENSEEIHT and CNRS.

We assume that the matrix A has been factorized using the HSL `MA57` package [7] with the option of using static pivoting [8]. Static pivoting will add a factor τ to a diagonal entry when it is impossible to find a suitable pivot in the fully summed blocks. It is common to choose $\tau \approx \sqrt{\varepsilon}||A||$ (ε machine precision).

Therefore, the computed factors $\hat{L}$ and $\hat{D}$ are, in exact arithmetic, the exact factorization of the perturbed problem

$$A + E = \hat{L}\hat{D}\hat{L}^T, \tag{1.2}$$

where E is a diagonal matrix of rank equal to the number of static pivots used during the factorization and $|E| \leq \tau I$. Hereafter, given the matrix B, we denote by $|B|$ the matrix having entries equal to the absolute values of the entries of B. The nonzero diagonal entries in E correspond to the positions at which static pivoting was performed and are all equal to τ. Note that if τ is chosen too small, then the factorization could be very unstable, whereas if it is chosen too large, the factorization will be stable but will not be an accurate factorization of the original matrix (that is, $|E|$ will be large). In section 2, we give a brief description of the static pivoting strategy used in `MA57`.

Equation (1.2) gives a splitting of A in terms of $M = \hat{L}\hat{D}\hat{L}^T$ and E,

$$A = M - E, \tag{1.3}$$

and the solution of (1.1) can be expressed as the solution of the equivalent systems

$$\begin{cases} (I - EM^{-1})Mx = b, \\ (I - M^{-1}E)x = M^{-1}b. \end{cases} \tag{1.4}$$

Owing to the symmetry of all the matrices in (1.4), $I - EM^{-1} = (I - M^{-1}E)^T$. Moreover, the first system corresponds to a right preconditioning of (1.1), while the second corresponds to a left preconditioning.

If the spectral radius of the matrix $I - M^{-1}E$ (or $I - EM^{-1}$) is less than one, the system (1.4) can be solved using iterative refinement. This has been used by many authors [5, 8, 13, 18] and is successful over a wide range of matrices although it is sensitive to the value of τ. If, however, the spectral radius is greater than or equal to one (or even close to one), it may be necessary to switch to a more powerful method such as GMRES [17]. Although the matrix is symmetric, we choose GMRES since it gives us much more freedom to work with a wide range of preprocessors and preconditionings.

We have found experimentally that using the factorization (1.2) as a right preconditioning for GMRES works in most cases and is, as expected, much more robust than iterative refinement. However, there are cases where we do not get convergence of a scaled residual to machine precision. Moreover, we have found that restarted GMRES performs better and that using flexible GMRES (FGMRES) [15], even though our preconditioner remains constant, does even better.

We illustrate this through numerical experiment (see section 6) and show theoretically that, under reasonable assumptions, FGMRES preconditioned by our static pivoting factorization is backward stable so that a small scaled residual can be achieved (see section 5). Our analysis also holds for the case of restarted FGMRES that we advocate for controlling the memory requirement while still achieving the desired robustness and accuracy. Indeed, we give theoretical arguments for why restarting often greatly improves the convergence.

In the following, we use the MATLAB notation $[A, B]$ to mean the matrix that consists of the columns of matrix A followed by the columns of matrix B. We denote by ε the machine precision of a finite-precision arithmetic satisfying the IEEE standard, and by $fl(\cdot)$ the result of a sequence of floating-point operations. The computed matrices, vectors, and scalars will be identified by a bar over the symbol. Finally, we will denote by $||\cdot||$ the usual Euclidean 2-norm for vectors and the corresponding induced norm for matrices.

2. LDL^T with static pivoting. In the multifrontal context, the factorization can be represented by a tree at each node of which elimination operations are performed on a partially summed frontal matrix

$$\begin{bmatrix} F_{11} & F_{12} \\ F_{12}^T & F_{22} \end{bmatrix}, \tag{2.1}$$

and pivots at that stage can be chosen only from within the fully summed block F_{11}. The problem occurs when it is impossible or numerically suicidal to eliminate all of F_{11}, resulting in more work and storage (sometimes dramatically more) than forecast.

Our mixed pivoting approach is based on two phases. In the first phase, we perform numerical pivoting in the block F_{11} of fully summed variables until no remaining variables satisfy the numerical criterion. In the second phase, we eliminate the remaining fully summed variables, adding 1×1 perturbations defined using a threshold $\mu > 0$. Let us denote by a_{ii} the generic 1×1 pivot and by

$$P = \begin{bmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{bmatrix}$$

the generic 2×2 pivot.

Then, our pivoting strategy is defined as follows. We define

$$g_1(i) = \frac{\max_{k\neq i} |a_{ki}|}{|a_{ii}|} \tag{2.2}$$

and

$$g_2(i,j) = \left|\left| |P^{-1}| \begin{pmatrix} \max_{k\neq i,j} |a_{ki}| \\ \max_{k\neq i,j} |a_{kj}| \end{pmatrix} \right|\right|_\infty . \tag{2.3}$$

During the first phase of our algorithm, we perform the usual Duff–Reid algorithm. In this approach, a 1×1 pivot a_{ii} is considered to be stable if and only if

$$g_1(i) \leq 1/u, \tag{2.4}$$

and a 2×2 pivot is considered to be stable if and only if

$$g_2(i,j) \leq 1/u. \tag{2.5}$$

During the second phase we preselect a 1×1 and a 2×2 pivot and decide which will be eliminated. The choice between a 1×1 and a 2×2 pivot is done in three stages. First, if we can eliminate a pivot and ensure a growth factor lower than $1 + 1/\mu$,

where $\mu = \sqrt{\varepsilon}$, i.e., if $g_1(i) \leq 1/\mu$ or $g_2(i,j) \leq 1/\mu$, then we select the one with the lower growth factor. Second, if $g_1(i) \geq 1/\mu$ and $g_2(i,j) \geq 1/\mu$ (we cannot ensure a growth factor lower than $1+1/\mu$), then we compare the quantities $1/|a_{ii}|$ and $||P^{-1}||_\infty$ and select the pivot associated with the smallest quantity. This second comparison is guided by the growth factor that would appear if we suppose that the largest off-diagonal entry is bounded by $\max_{ij} |a_{ij}|$. Finally, if no pivot can be chosen, we add some perturbation to make the factorization more stable. More precisely, a 1×1 pivot a_{ii} is selected and perturbed using $\delta = s(a_{ii})(\tau - |a_{ii}|)$, where s is the sign function and $\tau = \mu \max_{ij} |a_{ij}|$.

3. GMRES. The convergence of the GMRES method can be quite problematic for the general case even when we have a favorable distribution of the eigenvalues as in our case of the matrix $I - M^{-1}E$ (or $I - M^{-1}E$) [2].

However, because of the symmetry of A, the spectrum of $I-M^{-1}E$ (or $I-M^{-1}E$) has additional properties that make the behavior more regular. Let us denote by $\widetilde{I}_m$ and $\widetilde{I}_{n-m}$ the $n \times m$ and $n \times (n-m)$ matrices corresponding to the first m columns and the last $n-m$ columns of the $n \times n$ identity matrix, respectively. We assume that the matrices E, M, and A are symmetrically permuted so that the nonzero entries of E are the first m diagonal entries of the matrix E.

We observe that $\widetilde{I}_{n-m}$ is a basis for $Ker(E)$.

THEOREM 3.1. [illegible] E [illegible] $I - M^{-1}E$ [illegible] $I - EM^{-1}$ [illegible]. $M^{-1}Ex = \lambda x$ if and only if $Ex = \lambda Mx$. Since M is symmetric and nonsingular and E is positive semidefinite, λ is either zero or real, and (M, E) is an Hermitian definite pencil (see [3, p. 110]). □

We can introduce the right preconditioned version of GMRES, where we note that, although the vector z_k is computed, it is not stored.

ALGORITHM 3.1.

[illegible] $[x] =$ [illegible]
$x_0 = M^{-1}b$ $r_0 = b - Ax_0$ [illegible] $\beta = ||r_0||$
$v_1 = r_0/\beta$ [illegible] $r = r_0$
[illegible] $||r|| > \varepsilon\,(||b|| + ||A||\,||x_k||)$ [illegible] $k < maxit$
$k = k + 1$.
$z_k = M^{-1}v_k$. $w = Az_k$.
[illegible] $i = 1, \ldots, k$
$h_{i,k} = v_i^T w$.
$w = w - h_{i,k}v_i$.
[illegible]
$h_{k+1,k} = ||w||$.
$v_{k+1} = w/h_{k+1,k}$.
$V_k = [v_1, \ldots, v_k]$.
$H_k = \{h_{i,j}\}_{1 \leq i \leq j+1; 1 \leq j \leq k}$.
$y_k = \arg\min_y ||\beta e_1 - H_k y||$.
[illegible] $||\beta e_1 - H_k y_k|| \leq \varepsilon\,(||b|| + ||A||\,||x_k||)$
$x_k = x_0 + M^{-1}V_k y_k$ [illegible] $r = b - Ax_k$.
[illegible]
[illegible]
[illegible]

4. Flexible GMRES.

4.1. FGMRES versus GMRES.

First, we introduce the right-preconditioned version of FGMRES in Algorithm 4.1.

ALGORITHM 4.1.

$[x] =$

$x_0 = M_0^{-1} b \quad r_0 = b - Ax_0 \quad \beta = ||r_0||$

$v_1 = r_0/\beta \quad r = r_0$

$||r|| > \varepsilon\,(||b|| + ||A||\,||x_k||) \quad k < maxit$

$k = k + 1$

$z_k = M_k^{-1} v_k \quad w = Az_k$

$i = 1, \ldots, k$

$h_{i,k} = v_i^T w$

$w = w - h_{i,k} v_i$

$h_{k+1,k} = ||w||$

$v_{k+1} = w/h_{k+1,k}$

$Z_k = [z_1, \ldots, z_k] \quad V_k = [v_1, \ldots, v_k]$

$H_k = \{h_{i,j}\}_{1 \le i \le j+1; 1 \le j \le k}$

$y_k = \arg\min_y ||\beta e_1 - H_k y||$

$||\beta e_1 - H_k y_k|| \le \varepsilon\,(||b|| + ||A||\,||x_k||)$

$x_k = x_0 + Z_k y_k \quad r = b - Ax_k$

When $M_k = M$ for all k, Algorithms 3.1 and 4.1 differ only in the computation of x_k where in Algorithm 4.1 the vectors z_k are stored in Z_k. Since Algorithm 4.1 needs both V_k and Z_k, it requires additional storage with respect to Algorithm 3.1. In [9, 16], the convergence of FGMRES is analyzed: if each matrix H_k is full rank, the algorithm converges to the solution.

Both algorithms are based on the relations

$$C^{(k)} = [r_0, AZ_k] = V_{k+1} \begin{bmatrix} R_k \\ 0 \end{bmatrix}; \qquad V_j^T V_j = I_j \quad \forall j. \tag{4.1}$$

This corresponds to computing the orthogonal factorization of $C^{(k)}$, where each column $C_i^{(k)}$ is computed after the $(i-1)$th step of the Gram–Schmidt orthogonalization process. Furthermore, we have

$$R_k = \begin{bmatrix} \beta e_1 & H_k \end{bmatrix} \tag{4.2}$$

and

$$AZ_k = V_{k+1} H_k. \tag{4.3}$$

The vector y_k (in both Algorithms 3.1 and 4.1) is computed by a QR algorithm based on Givens rotations using the upper Hessenberg structure of H_k.

5. Roundoff error analysis. In the following, we will denote by $c_p(n, j)$ functions that depend only on the dimension n and the integer j. We will avoid a precise formulation of these dependences, but we assume that each $c_p(n, j)$ grows moderately with n and j. Finally, if $B \in \mathbb{R}^{n\times m}$, $n \geq m$, is a full rank matrix, we denote by $\kappa(B) = ||B||\,||B^+||$ its spectral condition number where $B^+ = (B^TB)^{-1}B$.

The roundoff error analysis of both FGMRES and GMRES can be made in four stages:

1. Error analysis of the Arnoldi–Krylov process. We will analyze in detail the modified Gram–Schmidt approach, but similar results can be achieved using a Householder-based approach as presented in [1, 6, 19].
2. Error analysis of the Givens process used on the upper Hessenberg matrix H_k in order to reduce it to upper triangular form.
3. Error analysis of the computation of x_k in FGMRES and GMRES.
4. Use of the static pivoting properties and of (1.3) in order to have simpler expressions.

The first two stages of the roundoff error analysis are the same for both FGMRES and GMRES, and the last two stages are specific to each algorithm.

5.1. FGMRES roundoff error analysis. In this section, we state the theorem presenting the main result that covers stages 1–3. The proofs are given in the appendices.

THEOREM 5.1. [illegible] 4.1 [illegible] (1.1) [illegible] ε [illegible]

$$\sigma_{\text{[illegible]}}(\bar{H}_k) > c_1(k,1)\varepsilon\,||\bar{H}_k|| + \mathcal{O}(\varepsilon^2) \quad \forall k, \tag{5.1}$$

$$|\bar{s}_k| < 1 - \varepsilon \quad \forall k, \tag{5.2}$$

[illegible] $\bar{s}_k$ [illegible] $\bar{H}_k$ [illegible] $\bar{y}_k$ [illegible]

$$2.12(n+1)\varepsilon < 0.01 \qquad 18.53\varepsilon\, n^{\frac{3}{2}}\kappa(C^{(k)}) < 0.1 \quad \forall k, \tag{5.3}$$

[illegible] $\hat{k}$, $\hat{k} \leq n$ [illegible] $k \geq \hat{k}$ [illegible]

$$||b - A\bar{x}_k|| \leq c_2(n,k)\varepsilon\left(||b|| + ||A||\,||\bar{x}_0|| + ||A||\,||\bar{Z}_k||\,||\bar{y}_k||\right) + \mathcal{O}(\varepsilon^2). \tag{5.4}$$

[illegible] $M_i = M$ [illegible] i [illegible]

$$\rho = 1.3\,||\hat{W}_k|| + c_3(k,1)\varepsilon\,||M||\,||\bar{Z}_k|| < 1 \quad \forall k < \hat{k}, \tag{5.5}$$

[illegible] $\hat{W}_k = [M\bar{z}_1 - \bar{v}_1, \ldots, M\bar{z}_k - \bar{v}_k]$ [illegible]

$$\begin{aligned} ||b - A\bar{x}_k|| \;\leq\; & c_4(n,k)\gamma\varepsilon\left(||b|| + ||A||\,||\bar{x}_0|| \right.\\ & \left. + ||A||\,||\bar{Z}_k||\left(||M(\bar{x}_k - \bar{x}_0)|| + \varepsilon\,||\bar{x}_0||\right)\right) + \mathcal{O}(\varepsilon^2), \\ \gamma = \frac{1.3}{1-\rho}. & \end{aligned} \tag{5.6}$$

[illegible]. See Appendix A. □

[illegible] 1. We point out that if $\bar{x}_0 = 0$ and $M = I$, FGMRES is numerically equivalent to classical GMRES. Moreover, formula (5.6) implies that GMRES is normwise backward stable, and we obtain the result in [14].

5.1.1. Stage 4: Static pivoting case. We remark that the computation of the matrices $\hat{L}$ and $\hat{D}$ by `MA57` with static pivoting in floating-point arithmetic gives the following relations:

$$\begin{cases} A + \delta A + \tau\Delta = M, \\ ||\delta A|| \leq c_5(n)\varepsilon\, ||\, |\hat{L}|\, |\hat{D}|\, |\hat{L}^T|\, ||, \\ ||\Delta|| \leq 1. \end{cases} \tag{5.7}$$

Moreover, we take into account that

$$\begin{cases} (M + \delta M)\bar{x}_0 = b, \qquad \text{where} \\ ||\delta M|| \leq c_6(n)\varepsilon\ ||\, |\hat{L}|\, |\hat{D}|\, |\hat{L}^T|\, ||. \end{cases} \tag{5.8}$$

Relations (5.7) and (5.8) follow by the use of standard techniques similar to those for the roundoff error analysis of Gaussian elimination [12, 13, 20]. We point out that, from (5.7), the perturbation δA must have a norm smaller than τ in order not to dominate the global error that would make the diagonal perturbation ineffective. Therefore, it is reasonable to assume that

$$\max\left(c_5(n), c_6(n)\right)\varepsilon\, ||\, |\hat{L}|\, |\hat{D}|\, |\hat{L}^T|\, || < \tau. \tag{5.9}$$

THEOREM 5.2. [illegible] 5.1 [illegible] (5.9) [illegible]

$$c_7(n,k)\gamma\varepsilon\, ||A||\, ||\bar{Z}_k|| < 1 \quad \forall k < \hat{k} \tag{5.10}$$

[illegible]

$$\max\{||M^{-1}||, ||\bar{Z}_k||\} \leq \frac{\tilde{c}}{\tau}, \tag{5.11}$$

[illegible] $\tilde{c}$ [illegible]

$$||b - A\bar{x}_k|| \leq 2\mu\varepsilon\left(||b|| + ||A||\left(||\bar{x}_0|| + ||\bar{x}_k||\right)\right) + \mathcal{O}(\varepsilon^{\,2}), \tag{5.12}$$

$$\mu = \frac{c_7(n,k)\gamma}{1 - c_7(n,k)\varepsilon\,\gamma||A||\, ||\bar{Z}_k||}.$$

[illegible]. See Appendix B. □

[illegible] 2. We point out that hypothesis (5.11), as we will see in section 6, is satisfied for a moderate value of $\tilde{c}$ by our static pivoting strategy on all our test problems. However, the upper bound (5.6) is more general and also gives a good estimate of the error when $\tau \ll \sqrt{\varepsilon}$. In all our experiments, we have $||\bar{Z}_k||\, ||M(\bar{x}_k - \bar{x}_0)|| < 10^4$.

5.2. GMRES roundoff error analysis. For the sake of simplicity, we make the assumption that the solution of the linear system

$$Mq = fl(\bar{V}_k\bar{y}_k)$$

is followed by a few steps of iterative refinement in order to guarantee a good local backward stability. Under this assumption, we have the following theorem.

THEOREM 5.3. [illegible] 3.1 [illegible] (1.1) [illegible] ε [illegible] (5.1), (5.2) [illegible] (5.3) [illegible] 5.1 [illegible]

$$c_8(n,1)\varepsilon\,\kappa(M) < 1, \tag{5.13}$$

[illegible] $\hat{k}$, $\hat{k} \le n$ [illegible] $k \ge \hat{k}$ [illegible]

$$\begin{aligned} ||b - A\bar{x}_k|| \le\ & c_9(n,k)\chi\varepsilon \Big\{ ||b|| + ||A||\,||\bar{x}_0|| + ||AM^{-1}||\,||M||\,||\bar{x}_k - \bar{x}_0|| \\ & + \Big[\, ||A||\,||\bar{Z}_k|| + ||AM^{-1}||\,||\,|\hat{L}|\,|\hat{D}|\,|\hat{L}^T|\,|| \,\Big] \\ & \times \Big[\, ||M(\bar{x}_k - \bar{x}_0)|| + n\varepsilon\,||M||\,(||\bar{x}_k - \bar{x}_0|| + ||\bar{x}_0||) \,\Big] \Big\} + \mathcal{O}(\varepsilon^2), \end{aligned} \tag{5.14}$$

$$\chi = \frac{1.3}{1 - 1.3k\varepsilon}.$$

[illegible]. See Appendix C. □

[illegible] 3. Both (5.6) and (5.14) do not depend on the special initial choice $\bar{x}_0 = M^{-1}b$. They will be true for any choice of $\bar{x}_0$. In both FGMRES and GMRES, a restart with a new $\bar{x}_0 = \bar{x}_k$ can improve the situation, reducing the upper bounds significantly.

[illegible] 4. Finally, we point out that when we replace the values of $c_4(n,k)\gamma$ and $c_9(n,k)\chi$ in (5.6) and (5.14) with $\hat{c} = \max(c_4(n,k)\gamma, c_9(n,k)\chi)$, (5.14) becomes an upper bound for (5.6). The numerical experiments in section 6 support this observation.

5.2.1. Stage 4: Static pivoting case. From (5.7) and (5.9), taking into account that $\tau = \varepsilon^{\sigma}||A||$ with $0 < \sigma \le 1$, we have

$$||M|| \le ||A|| + \tau + c_{10}(n)\,\varepsilon\ ||\,|\hat{L}|\,|\hat{D}|\,|\hat{L}^T|\,|| \le ||A|| + 2\,\tau \le 3\,||A||. \tag{5.15}$$

From (5.14), we have

$$\begin{aligned} ||b - A\bar{x}_k|| \le\ & c_{11}(n,k)\,\chi\,\varepsilon \Big[\, ||b|| + ||A||\,||\bar{x}_0|| + ||AM^{-1}||\,||A||\,||\bar{x}_k - \bar{x}_0|| \\ & + ||A||\,||\bar{Z}_k||\,||\,M(\bar{x}_k - \bar{x}_0)|| \\ & + ||AM^{-1}||\,||\,|\hat{L}|\,|\hat{D}|\,|\hat{L}^T|\,||\,||A||\,||\bar{x}_k - \bar{x}_0|| \Big] + \mathcal{O}(\varepsilon^2). \end{aligned} \tag{5.16}$$

Moreover, we have

$$AM^{-1} = I - \tau\Delta M^{-1} - c_5(n)\,\varepsilon\ ||\,|\hat{L}|\,|\hat{D}|\,|\hat{L}^T|\,||$$

and, from (5.11) and by assuming (5.9) is satisfied,

$$||AM^{-1}|| \le 1 + \tilde{c}.$$

Thus we have from (5.16)

$$\begin{aligned} ||b - A\bar{x}_k|| \le\ & c_{12}(n,k)\,\chi\,\varepsilon \Big[\, ||b|| + ||A||\,||\bar{x}_0|| + ||A||\,||\bar{Z}_k||\,||\,M(\bar{x}_k - \bar{x}_0)|| \\ & + \Big[\, 1 + ||\,|\hat{L}|\,|\hat{D}|\,|\hat{L}^T|\,|| \,\Big]\,||A||\,||\bar{x}_k - \bar{x}_0|| \Big] + \mathcal{O}(\varepsilon^2), \end{aligned} \tag{5.17}$$

$$\chi = \frac{1.3}{1 - 1.3k\varepsilon}.$$

. . . . 5. Formula (5.17) suggests that we might expect GMRES to have a final residual larger than the final residual in FGMRES. In particular, we could expect that a factor proportional to $\varepsilon\ ||A||\,||\,|\hat{L}|\,|\hat{D}|\,|\hat{L}^T|\,||\,||\bar{x}_k - \bar{x}_0||$ will appear.

6. Numerical experiments.

6.1. Test problems. We have run FGMRES and GMRES on several test problems. For the sake of simplicity, we present only the results obtained on particularly tough problems where iterative refinement fails for several different static pivoting strategies and values of τ [8]. We decided to omit the other results either because they are very similar to our results or because iterative refinement produced a scaled residual that is at rounding error level.

The dimension, the number of nonzero entries, and the origin of the three test problems that we use to illustrate the theory presented in section 5 are shown in Table 6.1.

TABLE 6.1
Test problems.

	n	nnz	$\|\|A\|\|_\infty$	$\|\|A\|\|$	Description
CONT201	80595	239596	8.2	8.1	KKT matrix Convex QP (M2)
CONT300	180895	562496	8.2	8.1	KKT matrix Convex QP (M2)
TUMA1	22967	76199	7.8	4.9	Mixed-hybrid finite-element

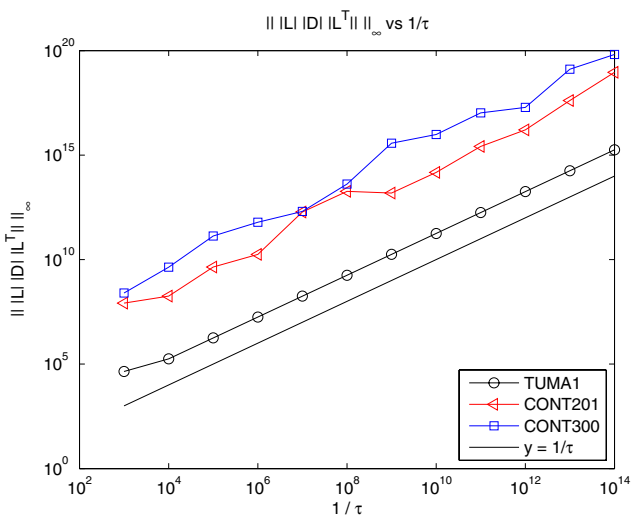


FIG. 6.1. $||\,|L|\,|D|\,|L^T|\,||$ *versus* $1/\tau$.

6.2. Numerical results. The smallest example, TUMA1, presents some interesting features and illustrates the behavior of test examples, where iterative refinement converges for the range of τ we are interested in. In Figure 6.1, we summarize the

TABLE 6.2
TUMA1 *results.*

	$\frac{\|\|b - A\bar{x}_k\|\|}{\|\|b\|\| + \|\|A\|\|\|\bar{x}_k\|\|}$			$\|\|M(\bar{x}_k - \bar{x}_0)\|\|$		
τ	GMRES (#It)	FGMRES (#It)	$\|\|Z_k\|\|$	FGMRES	GMRES	$\|\|\|L\|\|D\|\|L^T\|\|\|_\infty$
1.0e-03	1.0e-14 (26)	7.2e-17 (9)	1.2e+02	3.5e-03	3.5e-03	4.4e+04
1.0e-04	1.8e-16 (6)	3.1e-17 (6)	4.7e+01	4.4e-04	4.4e-04	1.8e+05
1.0e-05	1.3e-16 (5)	1.9e-17 (5)	4.4e+01	4.5e-05	4.5e-05	1.8e+06
1.0e-06	1.3e-16 (4)	1.9e-17 (4)	4.4e+01	4.5e-06	4.5e-06	1.8e+07
1.0e-07	1.2e-16 (3)	2.0e-17 (3)	4.3e+01	4.5e-07	4.5e-07	1.8e+08
1.0e-08	1.3e-16 (3)	1.8e-17 (3)	4.3e+01	4.5e-08	4.5e-08	1.8e+09
1.0e-09	2.8e-15 (31)	1.8e-17 (3)	2.6e+01	4.0e-08	4.0e-08	1.8e+10
1.0e-10	4.2e-13 (31)	1.8e-17 (3)	8.8e+00	4.0e-07	4.0e-07	1.8e+11
1.0e-11	1.0e-10 (31)	6.2e-17 (3)	6.8e+00	4.0e-06	4.0e-06	1.8e+12
1.0e-12	1.0e-08 (31)	2.2e-17 (4)	3.2e+01	4.3e-05	4.3e-05	1.8e+13
1.0e-13	2.4e-07 (31)	1.9e-17 (6)	1.3e+02	3.9e-04	3.9e-04	1.8e+14
1.0e-14	8.6e-06 (31)	2.1e-17 (10)	1.8e+02	4.3e-03	4.3e-03	1.8e+15

TABLE 6.3
CONT201 *results.*

	$\frac{\|\|b - A\bar{x}_k\|\|}{\|\|b\|\| + \|\|A\|\|\|\bar{x}_k\|\|}$			$\|\|M(\bar{x}_k - \bar{x}_0)\|\|$		
τ	GMRES (#It)	FGMRES (#It)	$\|\|Z_k\|\|$	FGMRES	GMRES	$\|\|\|L\|\|D\|\|L^T\|\|\|_\infty$
1.0e-03	1.8e-05 (31)	9.8e-06 (31)	*	1.5e-04	7.1e-04	8.3e+07
1.0e-04	2.0e-07 (31)	2.0e-07 (31)	*	1.9e-05	1.5e-05	1.8e+08
1.0e-05	1.8e-12 (31)	1.1e-16 (30)	4.1e+05	1.3e-05	5.9e-06	4.4e+09
1.0e-06	1.1e-11 (31)	2.1e-16 (15)	2.7e+06	7.8e-07	7.8e-07	1.8e+10
1.0e-07	4.8e-11 (31)	1.8e-16 (9)	1.4e+08	8.7e-08	8.7e-08	1.9e+12
1.0e-08	2.7e-10 (31)	5.8e-17 (6)	2.1e+07	1.3e-06	1.3e-06	1.8e+13
1.0e-09	1.8e-09 (31)	4.5e-17 (5)	1.1e+07	1.3e-06	1.3e-06	1.5e+13
1.0e-10	3.2e-09 (31)	7.2e-17 (4)	3.4e+05	9.2e-06	9.2e-06	1.5e+14
1.0e-11	2.1e-09 (31)	4.5e-17 (4)	1.9e+03	2.8e-04	2.8e-04	2.6e+15
1.0e-12	4.5e-07 (31)	3.8e-17 (5)	2.0e+02	9.5e-04	9.5e-04	1.6e+16
1.0e-13	1.3e-04 (31)	2.6e-16 (8)	1.6e+02	1.1e-02	1.1e-02	4.1e+17
1.0e-14	2.3e-01 (31)	2.5e-14 (12)	4.3e+02	1.0e-02	1.9e-02	9.2e+18

behavior of $|||L||D||L^T|||$ versus $1/\tau$ for all our test examples. We point out that $|||L||D||L^T|||$ can grow much more than $1/\tau$ because of the threshold pivoting strategy, and this growth is the reason for the poor behavior of GMRES.

In Tables 6.2, 6.3, and 6.4, we present the parameters and the scaled residual

$$\frac{||b - A\bar{x}_k||}{||b|| + ||A||_\infty \, ||\bar{x}_k||} \tag{6.1}$$

for right-preconditioned GMRES (GMRES column) and flexible GMRES (FGMRES column), for the test problems TUMA1, CONT201, and CONT300, respectively. We also show the values of $||\bar{Z}_k||$ and $||M(\bar{x}_k - \bar{x}_0)||$ at convergence, as they are crucial quantities for the residual bounds in Theorems 5.1 and 5.3.

TABLE 6.4
CONT300 *results.*

	$\frac{\|\|b - A\bar{x}_k\|\|}{\|\|b\|\| + \|\|A\|\|\|\bar{x}_k\|\|}$			$\|\|M(\bar{x}_k - \bar{x}_0)\|\|$		
τ	GMRES (#It)	FGMRES (#It)	$\|\|Z_k\|\|$	FGMRES	GMRES	$\|\|\|L\|\|D\|\|L^T\|\|\|_\infty$
1.0e-03	3.6e-05 (31)	2.5e-05 (31)	*	1.3e-04	8.7e-04	2.5e+08
1.0e-04	5.5e-07 (31)	5.5e-07 (31)	*	2.8e-05	6.5e-05	4.3e+09
1.0e-05	8.7e-09 (31)	8.7e-09 (31)	*	6.1e-06	3.7e-06	1.4e+11
1.0e-06	6.9e-11 (31)	1.4e-16 (23)	3.0e+06	9.8e-07	5.7e-07	6.2e+11
1.0e-07	2.1e-10 (31)	8.2e-17 (12)	7.6e+06	2.3e-07	2.3e-07	2.0e+12
1.0e-08	1.4e-08 (31)	1.2e-16 (8)	7.5e+07	1.8e-06	1.8e-06	4.1e+13
1.0e-09	1.6e-05 (31)	8.8e-17 (8)	3.7e+07	2.8e-04	2.8e-04	3.7e+15
1.0e-10	6.8e-07 (31)	4.1e-17 (6)	3.8e+05	3.6e-04	3.6e-04	9.6e+15
1.0e-11	1.6e-06 (31)	8.7e-17 (5)	1.4e+03	5.3e-03	5.3e-03	1.0e+17
1.0e-12	1.1e-06 (31)	2.7e-16 (5)	1.5e+02	1.0e-02	1.0e-02	1.9e+17
1.0e-13	3.4e-03 (31)	9.2e-16 (7)	1.3e+02	1.9e-01	1.9e-01	1.3e+19
1.0e-14	1.4e-01 (31)	1.8e-14 (12)	2.1e+02	4.7e-02	4.7e-02	6.6e+19

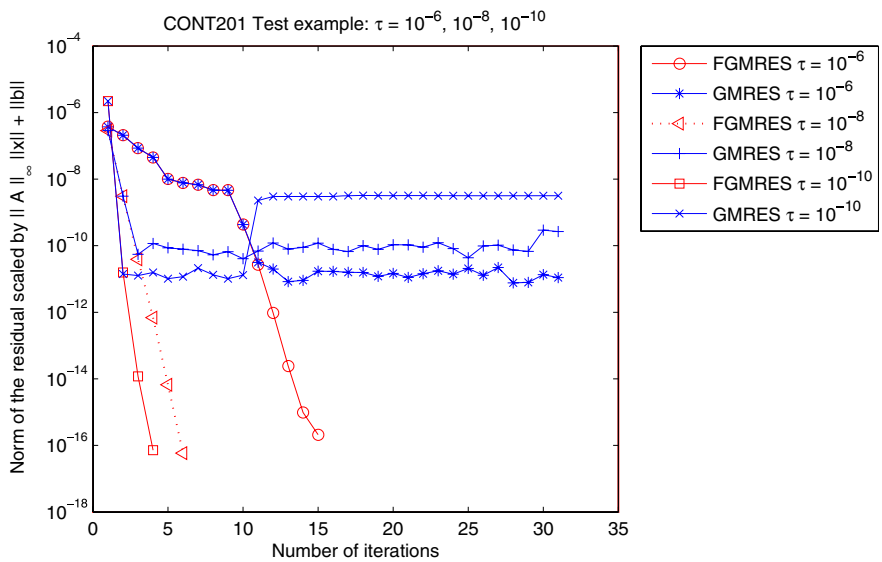


FIG. 6.2. *GMRES versus FGMRES on* CONT201 *test example:* $\tau = 10^{-6}, 10^{-8}, 10^{-10}$.

In all test examples, the values of $||\bar{Z}_k||$ and $||M(\bar{x}_k - \bar{x}_0)||$ for FGMRES show that

$$||\bar{Z}_k||\,||M(\bar{x}_k - \bar{x}_0)|| < 10^4$$

when $\tau = 10^{-9}$ (see Table 6.4). However, for the remaining values of τ the latter product is less than 2×10^2.

In Figures 6.2 and 6.3, we show the convergence histories of the scaled residual (6.1) for FGMRES and the right-preconditioned version of GMRES for the CONT201 and CONT300 test problems.

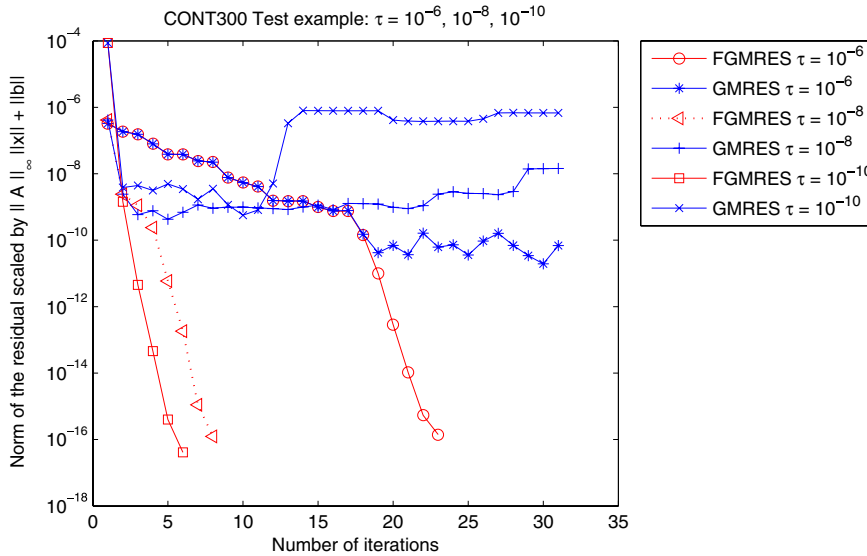


FIG. 6.3. *GMRES versus FGMRES on* CONT300 *test example:* $\tau = 10^{-6}, 10^{-8}, 10^{-10}$.

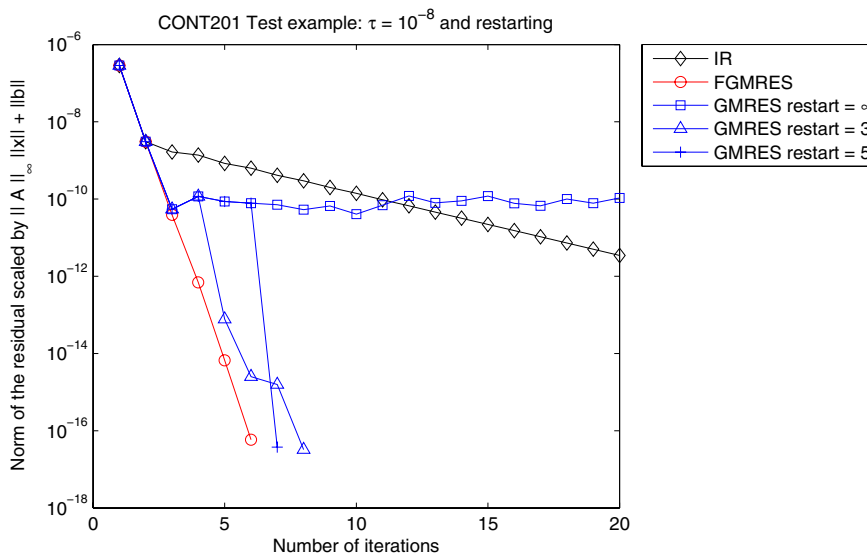


FIG. 6.4. *Restarted GMRES versus FGMRES on* CONT201 *test example:* $\tau = 10^{-8}$.

First, we point out that right-preconditioned GMRES is not backward stable. The FGMRES dependence on τ shows that the best trade-off is obtained for the recommended choice of $\tau = 10^{-8}$, where both iterative refinement and right-preconditioned GMRES are not able to obtain a scaled residual close to ε.

Both FGMRES and right-preconditioned GMRES can benefit from a restarting process. In particular, the restarted right-preconditioned GMRES converges with a scaled residual at machine precision for a small value of the restart parameter. In Figures 6.4 and 6.5, we compare the restarted right-preconditioned GMRES with

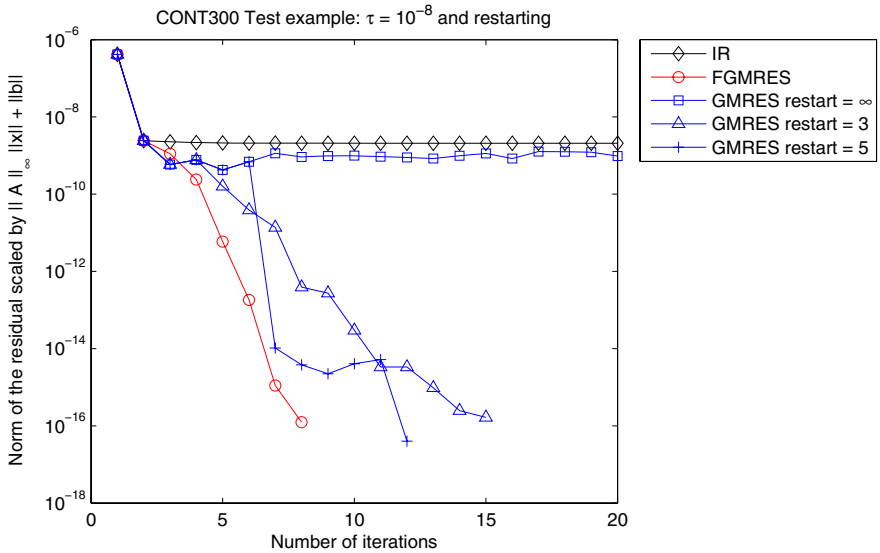


FIG. 6.5. *Restarted GMRES versus FGMRES on* CONT300 *test example:* $\tau = 10^{-8}$.

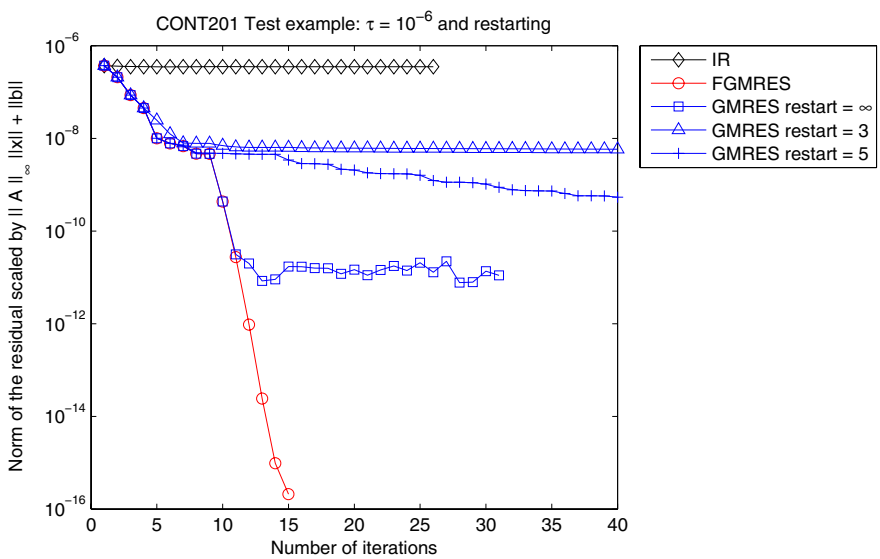


FIG. 6.6. *Restarted GMRES on* CONT201 *test example:* $\tau = 10^{-6}$.

FGMRES for $\tau = 10^{-8}$ for the CONT201 and CONT300 test examples, respectively. In the same figures, we show the behavior of classical iterative refinement which is unable to obtain a scaled residual close to rounding. Furthermore, we show in Figure 6.6 that, even with restarting, the preconditioned GMRES approach is not robust.

7. Conclusions. We have shown by experiment and analysis that FGMRES is a powerful method for obtaining the solution of sets of sparse linear equations when

a direct method using static pivoting has been used to factorize the matrix. This factorization is then used as a preconditioner for an iterative method. In particular, we show that GMRES converges when Richardson's method (iterative refinement) does not, that restarted GMRES performs better, and that FGMRES succeeds when these other methods fail. We also see that the convergence of FGMRES is much less sensitive to the choice of the static pivoting parameter.

Our analysis gives sufficient conditions for convergence, and we observe that FGMRES often converges even when our hypotheses are not satisfied. The main reason for this is that the bound $|||L||D||L^T|||$ on the norm of the solution of the preconditioned system is rather crude. Indeed, taking into account that $||M|| \approx ||A||$, we see that there is severe cancellation in the product and that small entries in D are usually balanced by large entries in L.

However, we see clearly that the analysis showing that stronger hypotheses are required for the convergence of preconditioned GMRES is reflected in the poorer convergence of this method, even when restarting is used.

A great benefit of our analysis and experiments is that the FGMRES iterative method can be used with confidence to solve systems preconditioned with a static pivoting factorization and that the method is far less sensitive to the choice of static pivoting parameter than either GMRES or iterative refinement.

Appendix A: Proof of Theorem 5.1.

1. By standard techniques [13], the computed matrix-by-vector products satisfy the relations

$$fl(r_0) = r_0 + f_1, \qquad ||f_1|| \le c_{13}(n,1)\varepsilon\,(||b|| + ||A||\,||\bar{x}_0||) + \mathcal{O}(\varepsilon^2), \tag{A.1}$$

$$fl(A\bar{Z}_k) = A\bar{Z}_k + F_k, \qquad ||F_k|| \le c_{14}(n,k)\varepsilon\,||A||\,||\bar{Z}_k|| + \mathcal{O}(\varepsilon^2), \tag{A.2}$$

where $\bar{x}_0 = fl(x_0)$.

Following [4, 10], the Gram–Schmidt orthogonalization process applied to $fl(C^{(k)})$ computes an upper triangular matrix $\overline{R}_k$ for which there exists an orthonormal matrix $\hat{V}_{k+1}$ that satisfies the relations

$$\begin{cases} [fl(r_0); fl(A\bar{Z})] + [f_2; E_k] = \hat{V}_{k+1}\overline{R}_k, \qquad \hat{V}_{k+1}^T\hat{V}_{k+1} = I_{k+1}, \\ ||f_2|| \le c_{15}(n,1)\varepsilon\,||r_0|| + \mathcal{O}(\varepsilon^2), \qquad ||E_k|| \le c_{16}(n,k)\varepsilon\,||A\bar{Z}_k|| + \mathcal{O}(\varepsilon^2) \end{cases} \tag{A.3}$$

under the hypothesis (5.3).

By combining (A.1), (A.2), and (A.3), we have

$$\begin{cases} [r_0; A\bar{Z}_k] + [f_1 + f_2; F_k + E_k] = \hat{V}_{k+1}\overline{R}_k, \\ ||f_1 + f_2|| \le c_{17}(n,1)\varepsilon\,(||b|| + ||A||\,||\bar{x}_0||) + \mathcal{O}(\varepsilon^2), \quad \text{and} \\ ||F_k + E_k|| \le c_{18}(n,k)\varepsilon\,(||A\bar{Z}_k|| + ||A||\,||\bar{Z}_k||) + \mathcal{O}(\varepsilon^2). \end{cases} \tag{A.4}$$

2. The second part of FGMRES computes the vector $\bar{y}_k = fl(y_k)$ by Givens or Householder algorithms [11, 12]. The vector $\bar{y}_k$ satisfies the relations

$$\begin{cases} \bar{y}_k = \arg\min_y ||\bar{\beta}e_1 + g^{[k]} - (\bar{H}_k + \Delta\bar{H}_k)y||, \\ ||\Delta\bar{H}_k|| \le c_{19}(k,1)\varepsilon\,||\bar{H}_k|| + \mathcal{O}(\varepsilon^2) \text{ and } ||g^{[k]}|| \le c_{20}(k,1)\varepsilon\,\bar{\beta} + \mathcal{O}(\varepsilon^2), \end{cases} \tag{A.5}$$

where $\bar{\beta} = (\overline{R}_k)_{11} = ||r_0 + f_1 + f_2||$ and the columns of $\bar{H}_k$ are columns $2, \ldots, k+1$ of

$\overline{R}_k$. Note that $\bar{\beta}$ is independent of k because at step k we compute only the kth column of $\overline{R}_k$, leaving the earlier columns unchanged. The computation of $\bar{y}_k$ is performed in two stages, and contributions to the matrix $\Delta\bar{H}_k$ come from two sources.

First, a sequence of Givens (or Householder) rotations $G^{(i)}$ is computed in order to reduce the matrix $\bar{H}$ to the upper triangular form U_k. The floating-point computation of the matrices $G^{(i)}$ gives

$$fl(G^{(i)}) = \bar{G}^{(i)} = \begin{bmatrix} I_{i-1} & & & \\ & \bar{c}_i & -\bar{s}_i & \\ & \bar{s}_i & \bar{c}_i & \\ & & & I_{n-i-1} \end{bmatrix}, \quad i = 1, \ldots, k,$$

$$\bar{c}_i = fl\left(\frac{(\bar{H}_k)_{i,i}}{\sqrt{(\bar{H}_k)_{i,i}^2 + (\bar{H}_k)_{i+1,i}^2}}\right) \qquad \text{and} \qquad \bar{s}_i = fl\left(\frac{(\bar{H}_k)_{i+1,i}}{\sqrt{(\bar{H}_k)_{i,i}^2 + (\bar{H}_k)_{i+1,i}^2}}\right).$$

The $\bar{G}^{(i)}$ matrices are also applied to the vector $\bar{\beta}e_1$, and, from the error analysis presented in [20], the floating-point arithmetic will produce an exact orthogonal matrix $G^{[k]}$ such that

$$\begin{aligned} fl(\bar{G}^{(k)} \cdots \bar{G}^{(1)} \bar{\beta} e_1) &= G^{[k]}(\bar{\beta}e_1 + g^{[k]}), \\ fl(\bar{G}^{(k)} \cdots \bar{G}^{(1)} \bar{H}_k) &= G^{[k]}(\bar{H}_k + \Delta\bar{H}_k^{(1)}), \\ ||\Delta\bar{H}_k^{(1)}|| &\leq c_{21}(k,1)\varepsilon\, ||\bar{H}_k|| + \mathcal{O}(\varepsilon^{\,2}). \end{aligned}$$

Second, the $\bar{y}_k$ vector is computed by solving the upper triangular system. The standard backward substitution algorithm will introduce an additional perturbation $\Delta\bar{H}_k^{(2)}$ of $\bar{H}_k$ but will leave the perturbation $g^{[k]}$ untouched. The perturbation $\Delta\bar{H}_k^{(2)}$ will also have the same upper Hessenberg structure of $\bar{H}_k$ and

$$||\Delta\bar{H}_k^{(2)}|| \leq c_{22}(k,1)\varepsilon\, ||\bar{H}_k|| + \mathcal{O}(\varepsilon^{\,2}).$$

This follows from the upper triangular structure of the perturbation to U_k induced by the backward substitution algorithm and from the structure and orthogonality of $G^{[k]}$. Finally, we point out that in the relations (A.5) we have (see (5.1))

$$\begin{cases} \Delta\bar{H}_k = \Delta\bar{H}_k^{(1)} + \Delta\bar{H}_k^{(2)} \qquad \text{and} \\ c_1(k,1) = c_{21}(k,1) + c_{22}(k,1), \end{cases}$$

where $c_1(k,1)$ is the constant of (5.1). Moreover, because of the special structure of $\bar{\beta}e_1$ and the orthogonality of $G^{[k]}$ we have

$$g_j^{[k]} = 0, \qquad j = k+1, \ldots, n,$$

and, denoting by $\bar{h}^{[k]} = fl(\bar{G}^{(k)} \cdots \bar{G}^{(1)} \bar{\beta} e_1)$, we have

$$\begin{aligned} \alpha_k &= ||G^{[k]}(\bar{\beta}e_1 + g^{[k]} - (\bar{H}_k + \Delta\bar{H}_k)\bar{y}_k)|| \\ &= ||(G^{[k]}(\bar{\beta}e_1 + g^{[k]}))_{k+1}|| = |\bar{h}_{k+1}^{[k]}|. \end{aligned} \tag{A.6}$$

A direct analysis of $\bar{h}^{[k]}$ shows that

$$\begin{cases} \bar{h}_1^{[1]} &= \bar{\beta}\bar{c}_1(1+\mu_1), & |\mu_1| \le \varepsilon\,, \\ \bar{h}_2^{[1]} &= \bar{\beta}\bar{s}_1(1+\rho_1), & |\rho_1| \le \varepsilon\,, \\ \bar{h}_j^{[k]} &= \bar{h}_j^{[k-1]}, & j = 1,\ldots,k-1, \quad k \ge 2, \\ \bar{h}_k^{[k]} &= \bar{h}_k^{[k-1]}\bar{c}_k(1+\mu_k), & |\mu_k| \le \varepsilon\,, \\ \bar{h}_{k+1}^{[k]} &= \bar{h}_k^{[k-1]}\bar{s}_k(1+\rho_k), & |\rho_k| \le \varepsilon\,. \end{cases}$$

Therefore, from using (A.3), (A.4), (A.5), (A.6), and the orthogonality of

$$\widetilde{V}_{k+1} = \hat{V}_{k+1}G^{[k]T},$$

we have

$$\begin{aligned} \alpha_k &= ||\widetilde{V}_{k+1}G^{[k]}(\bar{\beta}e_1 + g^{[k]} - (\bar{H}_k + \Delta\bar{H}_k)\bar{y}_k)|| \\ &= ||r_0 + f_1 + f_2 + \hat{V}_{k+1}g^{[k]} - A(\bar{Z}_k + A^{-1}(F_k + E_k + \hat{V}_{k+1}\Delta\bar{H}_k))\bar{y}_k|| \\ &= ||r_0 + \delta r_0 - A\big(\bar{Z}_k + \hat{Z}_k\big)\bar{y}_k||, \end{aligned}$$

where $\delta r_0 = f_1 + f_2 + f_3$ and $\hat{Z}_k = A^{-1}(F_k + E_k + \hat{V}_k\Delta\bar{H}_k)$. Then

$$\alpha_k = \alpha_{k-1}|\bar{s}_k|(1+\rho_k), \quad |\rho_k| \le \varepsilon\,. \tag{A.7}$$

Thus, under the hypotheses (5.2), (5.1), and (5.3) and from (A.5), the matrices $\big(\bar{Z}_k + \hat{Z}_k\big)$ have full rank for all k; i.e.,

$$\bar{H}_k + \Delta\bar{H}_k$$

is full rank for all k, and the values of α_k converge monotonically to zero for a finite value of $k = \hat{k}$. In the worst case this will happen for $\hat{k} = n$.

3. The last part of FGMRES is the computation of $\bar{x}_k = fl(x_0 + \bar{Z}\bar{y}_k)$. The value $\bar{x}_k$ satisfies the relations

$$\begin{cases} \bar{x}_k = \bar{x}_0 + \bar{Z}_k\bar{y}_k + \delta x_k, \\ ||\delta x_k|| \le c_3(k,1)\varepsilon\,||\bar{Z}_k||\,||\bar{y}_k|| + \varepsilon\,||\bar{x}_0|| + \mathcal{O}(\varepsilon^{\,2}). \end{cases} \tag{A.8}$$

Therefore, we have

$$\alpha_k = ||r_0 + \delta r_0 + A\delta x_k - A\delta x_k - A\bar{Z}_k\bar{y}_k - A\hat{Z}_k\bar{y}_k||. \tag{A.9}$$

From (A.8), we have

$$\begin{cases} \alpha_k = ||b - A\bar{x}_k + w||, \\ w = \delta r_0 + A\delta x_k - A\hat{Z}_k\bar{y}_k, \end{cases} \tag{A.10}$$

and then

$$||b - A\bar{x}_k|| \le ||w|| + \alpha_k. \tag{A.11}$$

From (A.4), (A.5), (A.8), (A.10), and (A.11), we have

$$\text{(A.12)}\qquad \begin{cases} ||\delta r_0|| \le c_{24}(n,1)\varepsilon\,(||b|| + ||A||\,||\bar{x}_0||) + \mathcal{O}(\varepsilon^{\,2}), \\ ||A\delta x_k|| \le c_{25}(n,1)\varepsilon\,||A||\,\Big[\,||\bar{Z}_k||\,||\bar{y}_k|| + ||\bar{x}_0||\Big] + \mathcal{O}(\varepsilon^{\,2}) \end{cases}$$

and, finally,

$$\text{(A.13)}\qquad \begin{aligned} ||w|| \;\le\; c_{26}(n,k)\varepsilon\,\Big(&||b|| + ||A||\,||\bar{x}_0|| + ||\bar{H}_k||\,||\bar{y}_k|| \\ &+ ||A||\,||\bar{Z}_k||\,||\bar{y}_k||\Big) + \mathcal{O}(\varepsilon^{\,2}). \end{aligned}$$

Therefore, under hypothesis (5.1) and from (A.11), we have

$$\text{(A.14)}\qquad \begin{aligned} ||b - A\bar{x}_k|| \;\le\; \alpha_k + c_2(n,k)\varepsilon\,\Big(&||b|| + ||A||\,||\bar{x}_0|| + ||\bar{H}_k||\,||\bar{y}_k|| \\ &+ ||A||\,||\bar{Z}_k||\,||\bar{y}_k||\Big) + \mathcal{O}(\varepsilon^{\,2}) \end{aligned}$$

and, then, taking into account that, from (A.4),

$$\text{(A.15)}\qquad ||\bar{H}_k|| \le ||A\bar{Z}_k|| + \mathcal{O}(\varepsilon\,) \quad \forall k < \hat{k},$$

we have (5.4).

From the analysis of [4, 10], the Gram–Schmidt orthogonalization process applied to $fl(C^{(k)})$ computes matrices $\bar{V}_k = fl(V_k)$ that lose orthogonality quickly. Nevertheless, under the mild hypothesis (5.3) on the matrix $C^{(k)}$ the spectral condition number of each $\bar{V}_k$ is close to one and

$$\begin{aligned} 1 \le ||\bar{V}_k|| \le k, \\ \kappa(\bar{V}_k) \le 1.3 \end{aligned}$$

so that

$$||\bar{V}_k^{+}|| \le 1.3/||\bar{V}_k|| \le 1.3,$$

where $\bar{V}_k^{+} = (\bar{V}^T\bar{V}_k)^{-1}\bar{V}_k^T$ (see [10, Thm. 3.1] and [14]).

Taking into account the previous relations, it is possible to express $\bar{y}_k$ as a function of $\bar{x}_k - x_0$. For each column $\bar{z}_i$ of $\bar{Z}_k$ a matrix $\mathcal{G}_i$ exists and satisfies

$$(M + \mathcal{G}_i)\bar{z}_i = \bar{v}_i, \qquad ||\mathcal{G}_i|| \le c_{27}(n,1)\,\varepsilon\,||\,|\hat{L}|\,|\hat{D}|\,|\hat{L}^T|\,|| + \mathcal{O}(\varepsilon^{\,2}),$$

where $\bar{v}_i$ is the ith column of $\bar{V}_k$. Then, we have the relation

$$\text{(A.16)}\qquad M\bar{Z}_k = \bar{V}_k + \hat{W}_k + \mathcal{O}(\varepsilon^{\,2}).$$

From the relations (A.8) and (A.16), it follows that

$$\begin{aligned} &\bar{x}_k - \bar{x}_0 - \delta x_k = \bar{Z}_k\bar{y}_k, \\ &\bar{V}_k^{+}M(\bar{x}_k - \bar{x}_0 - \delta x_k) = \bar{V}_k^{+}M\bar{Z}_k\bar{y}_k, \\ &\bar{V}_k^{+}(M(\bar{x}_k - \bar{x}_0) - M\delta x_k) = (\bar{V}_k^{+}\bar{V}_k\bar{y}_k + \bar{V}_k^{+}\hat{W}_k\bar{y}_k) + \mathcal{O}(\varepsilon^{\,2}), \\ &\bar{V}_k^{+}(M(\bar{x}_k - \bar{x}_0) - M\delta x_k) - \bar{V}_k^{+}\hat{W}_k\bar{y}_k + \mathcal{O}(\varepsilon^{\,2}) = \bar{y}_k. \end{aligned}$$

Then, we have

$$||\bar{y}_k|| \leq 1.3||M(\bar{x}_k - \bar{x}_0)|| + 1.3||\hat{W}_k||\,||\bar{y}_k|| + ||M||\,||\delta x_k|| + \mathcal{O}(\varepsilon^2)$$

and

$$\begin{aligned}||\bar{y}_k|| \leq\ & 1.3||M(\bar{x}_k - \bar{x}_0)|| + 1.3||\hat{W}_k||\,||\bar{y}_k|| \\ & + c_3(k,1)\varepsilon\,||M||\,||\bar{Z}_k||\,||\bar{y}_k|| + \varepsilon\,||\bar{x}_0|| + \mathcal{O}(\varepsilon^2).\end{aligned}$$

From hypothesis (5.5), we can state the inequality

$$||\bar{y}_k|| \leq \gamma||M(\bar{x}_k - \bar{x}_0)|| + \gamma\varepsilon\,||\bar{x}_0|| + \mathcal{O}(\varepsilon^2), \quad \text{where } \gamma = \frac{1.3}{1-\rho}. \tag{A.17}$$

Finally, by substituting (A.17) into (A.14) and taking into account (A.15), we have the following upper bound for the residual:

$$\begin{aligned}||b - A\bar{x}_k|| \ \leq\ & \alpha_k + c_4(n,k)\gamma\varepsilon\Big[||b|| + ||A||\,||\bar{x}_0|| \\ & + \Big(||A\bar{Z}_k|| + ||A||\,||\bar{Z}_k||\Big)\Big(||M(\bar{x}_k - \bar{x}_0)|| + \varepsilon\,||\bar{x}_0||\Big)\Big] + \mathcal{O}(\varepsilon^2).\end{aligned}$$

Furthermore, hypothesis (5.1) implies that there exists a value $\hat{k}$ (possibly equal to n in the worst case) for which $\alpha_k = 0$. Then, we can conclude that for $k \geq \hat{k}$ we have (5.6).

Appendix B: Proof of Theorem 5.2. We can bound $||M(\bar{x}_k - \bar{x}_0)||$ using (5.7) and (5.8). Under hypothesis (5.9), we have

$$\begin{aligned}||M(\bar{x}_k - \bar{x}_0)|| \ &\leq ||A\bar{x}_k - b|| + \tau\,||\bar{x}_k|| + ||\delta A||\,||\bar{x}_k|| + ||\delta M||\,||\bar{x}_0|| \\ &\leq ||A\bar{x}_k - b|| + 2\tau\big(\,||\bar{x}_k|| + ||\bar{x}_0||\,\big).\end{aligned} \tag{B.1}$$

From the relations (B.1) and (5.6), we have

$$\begin{aligned}||b - A\bar{x}_k|| \ \leq\ & c_4(n,k)\gamma\varepsilon\,(||b|| + ||A||\,||\bar{x}_0|| \\ & + ||A||\,||\bar{Z}_k||\,\big(||A\bar{x}_k - b|| + \tau\,(\,||\bar{x}_k|| + ||\bar{x}_0||\,)\big) + \mathcal{O}(\varepsilon^2).\end{aligned}$$

Under the hypotheses (5.9) and (5.10), we have

$$\begin{cases} ||b - A\bar{x}_k|| \ \leq\ 2\mu\varepsilon\Big(||b|| + ||A||\,||\bar{x}_0|| \\ \qquad\qquad + \tau||A||\,||\bar{Z}_k||\,(||\bar{x}_k|| + ||\bar{x}_0||)\Big) + \mathcal{O}(\varepsilon^2), \\ \mu = \dfrac{c_7(n,k)\gamma}{1 - c_7(n,k)\varepsilon\,\gamma||A||\,||\bar{Z}_k||}. \end{cases} \tag{B.2}$$

Under this final hypothesis (5.11), we obtain (5.12), i.e., the normwise backward stability for FGMRES using as preconditioner the M computed by `MA57` with static pivoting.

Appendix C: Proof of Theorem 5.3. The first two stages of GMRES roundoff error analysis are the same as the first two stages in the proof of Theorem 5.1. Under the assumption of performing a few steps of iterative refinement in solving the final linear system

$$Mq = fl(\bar{V}_k \bar{y}_k),$$

we have the following relations:

$$\text{(C.1)} \qquad \begin{cases} (M+\delta M)\big(I+\Gamma\big)^{-1}\,(\bar{x}_k - (I+\Gamma)\,\bar{x}_0) = \bar{V}_k\,\bar{y}_k + \widetilde{f}, \\ |\Gamma| \le \varepsilon\, I, \\ ||\widetilde{f}|| \le k\varepsilon\ ||\bar{V}_k||\,||\bar{y}_k|| + \mathcal{O}(\varepsilon^{\,2}) \le 1.3k\varepsilon\,||\bar{y}_k|| + \mathcal{O}(\varepsilon^{\,2}), \\ ||\delta M|| \le n\varepsilon\ ||M|| + \mathcal{O}(\varepsilon^{\,2}). \end{cases}$$

Taking into account that

$$(M+\delta M)\,(I+\Gamma)^{-1} = M + \big(\delta M - M\,\Gamma\big)\big(I+\Gamma\big)^{-1} = M + \widetilde{\delta M}$$

with

$$||\widetilde{\delta M}|| \le (n+1)\frac{\varepsilon}{1-\varepsilon}||M|| = c_{28}(n,1)\varepsilon\,||M||,$$

we have, from (C.1), that

$$\text{(C.2)} \qquad (M+\widetilde{\delta M})(\bar{x}_k - \bar{x}_0) = \bar{V}_k\,\bar{y}_k + \widetilde{f} + (M+\widetilde{\delta M})\,\Gamma\,\bar{x}_0.$$

Therefore, we have from (5.13) and (C.2) that (A.8) can be replaced by

$$\text{(C.3)} \qquad \bar{x}_k = \bar{x}_0 + (M+\widetilde{\delta M})^{-1}(\bar{V}_k\,\bar{y}_k + \widetilde{f}) + \Gamma\,\bar{x}_0$$

$$\text{(C.4)} \qquad = \bar{x}_0 + M^{-1}\,\bar{V}_k\,\bar{y}_k + \delta\bar{x}_k$$

with

$$\begin{aligned} \delta\bar{x}_k &= \Gamma\,\bar{x}_0 + M^{-1}\,\widetilde{f} - M^{-1}\,\widetilde{\delta M}\,M^{-1}\,\bar{V}_k\,\bar{y}_k + \mathcal{O}(\varepsilon^{\,2}) \\ &= \Gamma\,\bar{x}_0 + M^{-1}\,\widetilde{f} - M^{-1}\,\widetilde{\delta M}\,(\bar{x}_k - \bar{x}_0 - \delta\bar{x}_k) + \mathcal{O}(\varepsilon^{\,2}) \end{aligned}$$

and

$$\text{(C.5)} \qquad \begin{aligned} \delta\bar{x}_k &= \big(I - M^{-1}\,\widetilde{\delta M}\big)^{-1}\big(\Gamma\,\bar{x}_0 + M^{-1}\,\widetilde{f} - M^{-1}\,\widetilde{\delta M}\,(\bar{x}_k - \bar{x}_0)\big) + \mathcal{O}(\varepsilon^{\,2}) \\ &= \Gamma\,\bar{x}_0 + M^{-1}\,\widetilde{f} - M^{-1}\,\widetilde{\delta M}\,(\bar{x}_k - \bar{x}_0) + \mathcal{O}(\varepsilon^{\,2}). \end{aligned}$$

Under hypotheses (5.3) and (5.13), we obtain from (C.3), (C.4), (C.5), and (A.16)

$$\text{(C.6)} \qquad \begin{cases} \alpha_k = ||b - A\bar{x}_k + w_g||, \qquad \text{where} \\ w_g = \delta r_0 + A\delta x_k - A\hat{Z}_k\bar{y}_k + A(M^{-1}\,\bar{V}_k\,\bar{y}_k - \bar{Z}_k\,\bar{y}_k) \\ \qquad = \delta r_0 + A\delta x_k - A\hat{Z}_k\bar{y}_k + AM^{-1}\,\hat{W}_k\,\bar{y}_k + \mathcal{O}(\varepsilon^{\,2}), \end{cases}$$

and then

$$||b - A\bar{x}_k|| \leq \alpha_k + ||w_g||. \tag{C.7}$$

From (A.4), (A.5), (A.7), (C.3), (C.5), (C.6), and (C.7), we have

$$\begin{cases} ||\delta r_0|| \;\leq\; c_{29}(n,1)\varepsilon\,(||b|| + ||A||\,||\bar{x}_0||) + \mathcal{O}(\varepsilon^{\,2}) \\ ||A\delta x_k|| \;\leq\; c_{30}(n,1)\varepsilon\left[||AM^{-1}||\left(||M||\,||\bar{x}_k - \bar{x}_0|| + ||\bar{y}_k||\right)\right. \\ \qquad\qquad \left. + ||A||\,||\bar{x}_0||\right] + \mathcal{O}(\varepsilon^{\,2}) \end{cases} \tag{C.8}$$

and, finally,

$$\begin{aligned} ||w_g|| \leq c_{31}(n,k)\varepsilon\;\Big[\;&||b|| + ||A||\,||\bar{x}_0|| + ||\bar{H}_k||\,||\bar{y}_k|| + ||A||\,||\bar{Z}_k||\,||\bar{y}_k|| \\ &+ ||AM^{-1}||\left(||M||\,||\bar{x}_k - \bar{x}_0|| + ||\bar{y}_k|| + ||\,|\hat{L}|\,|\hat{D}|\,|\hat{L}^T|\,||\,||\bar{y}_k||\right)\Big] + \mathcal{O}(\varepsilon^{\,2}). \end{aligned} \tag{C.9}$$

From (C.2), we can compute an upper bound for $||\bar{y}_k||$ by multiplying the equation by $\bar{V}_k^+$:

$$\bar{y}_k = \bar{V}_k^+\,(M + \widetilde{\delta M})(\bar{x}_k - \bar{x}_0) - \bar{V}_k^+\,(\widetilde{f} + (M + \widetilde{\delta M})\,\Gamma\,\bar{x}_0) \tag{C.10}$$

$$= \bar{V}_k^+\,M(\bar{x}_k - \bar{x}_0) - \bar{V}_k^+\,(\widetilde{f} - \widetilde{\delta M}(\bar{x}_k - \bar{x}_0) + (M + \widetilde{\delta M})\,\Gamma\,\bar{x}_0), \tag{C.11}$$

and then

$$\begin{aligned} ||\bar{y}_k|| \;&\leq\; 1.3||M(\bar{x}_k - \bar{x}_0)|| + 1.3k\varepsilon\,||\bar{y}_k|| \\ &+\; 1.3n\varepsilon\,||M||\,(||\bar{x}_k - \bar{x}_0|| + ||\bar{x}_0||) + \mathcal{O}(\varepsilon^{\,2}). \end{aligned} \tag{C.12}$$

From hypothesis (5.3), we can assume that $1.3\,k\varepsilon < 1$, and thus we have

$$\begin{cases} ||\bar{y}_k|| \leq \chi||M(\bar{x}_k - \bar{x}_0)|| + \chi n\varepsilon\,||M||\,(||\bar{x}_k - \bar{x}_0|| + ||\bar{x}_0||) + \mathcal{O}(\varepsilon^{\,2}), \\ \chi = \dfrac{1.3}{1 - 1.3k\varepsilon}. \end{cases} \tag{C.13}$$

Furthermore, as in the proof of Theorem 5.1, hypothesis (5.1) implies that there exists a value $\hat{k}$ (possibly equal to n in the worst case) for which $\alpha_k = 0$. Then, under hypotheses (5.1), (5.3) and from (C.13), (C.7), and (C.9), we can conclude that for $k \geq \hat{k}$ we have (5.14).

Acknowledgment. We would like to thank the anonymous referees for their helpful comments that have hopefully made the paper easier to read.

REFERENCES

[1] M. Arioli and C. Fassino, *Roundoff error analysis of algorithms based on Krylov subspace methods*, BIT, 36 (1996), pp. 189–206.

[2] M. Arioli, V. Pták, and Z. Strakoš, *Krylov sequences of maximal length and convergence of GMRES*, BIT, 38 (1998), pp. 1–9.

[3] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, SIAM, Philadelphia, 2000.

[4] Å. Björck and C. C. Paige, *Loss and recapture of orthogonality in the modified Gram–Schmidt algorithm*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 176–190.
[5] J. Demmel, Y. Hida, W. Kahan, X. S. Li, S. Mukherjeek, and E. J. Riedy, *Error bounds from extra precise iterative refinement*, ACM Trans. Math. Software, 32 (2006), pp. 325–351.
[6] J. Drkosova, A. Geenbaum, M. Rozložník, and Z. Strakoš, *Numerical stability of GMRES*, BIT, 35 (1995), pp. 309–330.
[7] I. S. Duff, *MA57–A code for the solution of sparse symmetric indefinite systems*, ACM Trans. Math. Software, 30 (2004), pp. 118–144.
[8] I. S. Duff and S. Pralet, *Towards a stable static pivoting strategy for the sequential and parallel solution of sparse symmetric indefinite systems*, SIAM J. Matrix Anal. Appl., to appear.
[9] L. Giraud, S. Gratton, and J. Langou, *A Note on Relaxed and Flexible GMRES*, Technical report TR/PA/04/41, CERFACS, Toulouse, France, 2004.
[10] L. Giraud and J. Langou, *When modified Gram–Schmidt generates a well-conditioned set of vectors*, IMA J. Numer. Anal., 22 (2002), pp. 521–528.
[11] G. H. Golub, *Numerical methods for solving linear least squares problems*, Numer. Math., 7 (1965), pp. 206–216.
[12] G. H. Golub and C. F. V. Loan, *Matrix Computations*, 2nd ed., The Johns Hopkins University Press, Baltimore, MD, 1989.
[13] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM, Philadelphia, 2002.
[14] C. C. Paige, M. Rozložník, and Z. Strakoš, *Modified Gram–Schmidt (MGS), least squares, and backward stability of MGS-GMRES*, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 264–284.
[15] Y. Saad, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM J. Sci. Comput., 14 (1993), pp. 461–469.
[16] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.
[17] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.
[18] R. D. Skeel, *Iterative refinement implies numerical stability for Gaussian elimination*, Math. Comp., 35 (1980), pp. 817–832.
[19] H. F. Walker, *Implementation of the GMRES method using Householder transformations*, SIAM J. Sci. Stat. Comput., 9 (1988), pp. 152–163.
[20] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, New York, 1965.

SIAM J. SCI. COMPUT.
Vol. 29, No. 5, pp. 2045–2058
© 2007 Society for Industrial and Applied Mathematics

INTERPOLATION OPERATORS FOR ALGEBRAIC MULTIGRID BY LOCAL OPTIMIZATION*

CLARK R. DOHRMANN†

Abstract. An approach is presented to construct interpolation operators for algebraic multigrid (AMG) based on solutions of local optimization problems. The local problems are posed in such a way that the null space can be approximated exactly at each multigrid level. In addition, a condition ensuring nonsingularity of a local matrix highlights the need for special considerations when the null space dimension exceeds the number of unknowns for each node. Numerical examples for the Poisson equation and linear elasticity demonstrate the effectiveness of the approach and the optimal performance of an ensuing AMG preconditioner. Comparisons with a smoothed aggregation approach are also made.

Key words. algebraic multigrid, interpolation operators, energy minimization, iterative methods, smoothed aggregation

AMS subject classifications. 65F10, 65F50, 65N22, 65N55

DOI. 10.1137/06066103X

1. Introduction.

Several approaches have been proposed recently to construct multigrid interpolation operators (matrices) based on energy minimization concepts. In [12] and [16], minimizing the sum of coarse interpolation energies subject to null space constraints leads to a global optimization problem for the different coarsening strategies considered. This problem is then solved approximately using an iterative method. A global optimization problem is avoided in algebraic multigrid based on elements (AMGe) [2, 9], but local energy minimization problems require individual element matrices. Studies of element-free alternatives to AMGe not requiring access to element matrices appear in [8, 10].

Our approach can be interpreted as a new way of constructing element-free AMGe operators that explicitly incorporates available null space information. We show how the interpolation operators can be obtained conveniently from solutions of local energy minimization problems. The approach ultimately leads to effective multigrid methods, not only for problems like the Poisson equation, but for more complicated systems such as linear elasticity. Other problem types such as plate bending can also be treated using the same framework, but they are not discussed here. We note for elasticity problems that special considerations involving the introduction of nodal rotations are required at the first coarse level.

Another approach used for constructing AMG interpolation operators is smoothed aggregation [15, 14]. In smoothed aggregation, an interpolation called a tentative prolongator is constructed using information about the null space for the problem. Often, the tentative prolongator is then smoothed using damped Jacobi iterations.

*Received by the editors May 26, 2006; accepted for publication (in revised form) March 26, 2007; published electronically September 28, 2007. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94-AL85000. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights.

http://www.siam.org/journals/sisc/29-5/66103.html

†Structural Dynamics Research Department, Sandia National Laboratories, Mail Stop 0847, Albuquerque, NM 87185-0847 (crdohrm@sandia.gov).

Although these iterations can reduce the energy of the interpolations, they increase the support of the aggregates and require spectral estimates. In contrast, coarse node supports are specified beforehand for the present approach and no eigenvalue estimates are needed. Like smoothed aggregation, the present approach uses knowledge about the null space, but it does so in a fundamentally different way. Here, the null space is used to approximate nodal values in a neighborhood around a noncoarse node.

To keep the presentation simple, we assume that the coarse nodes for each level are given. The subject approach for constructing interpolation operators is presented in section 2. Numerical examples and comparisons with a smoothed aggregation approach are provided in section 3. A few concluding remarks are given in section 4.

2. Interpolation operators. Let A denote the stiffness matrix for a given multigrid level. We assume that A is symmetric and at least positive semidefinite. Each row and column of A corresponds to a particular node and local degree of freedom (dof) number of the computational mesh (grid). Let $r(i)$ denote the rows of A corresponding to node i. Two nodes i and j are said to be adjacent if $a_{kl} \neq 0$ for some $k \in r(i)$ and $l \in r(j)$. We also say that dofs k and l are adjacent if $a_{kl} \neq 0$. Each column of the interpolation matrix P corresponds to a particular node called a coarse node.

2.1. Motivation and connections. To begin, let f (fine) denote the set of dofs for a particular noncoarse node. The set of dofs potentially affecting the interpolation of f is denoted by c (coarse). Define $I = f \cup c$ as the set of [illegible] dofs. The set of [illegible] dofs B contains all dofs adjacent to at least one dof in I but not contained in I. Depending on the context, symbols such as f and c can refer to either these dof sets or their associated node sets. The null space matrix associated with the problem operator has the same number of rows as A and is denoted by N.

The vector y_I contains all the dofs in I and is partitioned such that

$$y_I = \begin{bmatrix} y_f \\ y_c \end{bmatrix}, \quad A_{II} = \begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix}, \quad A_{IB} = \begin{bmatrix} A_{fB} \\ A_{cB} \end{bmatrix}, \quad N_I = \begin{bmatrix} N_f \\ N_c \end{bmatrix}. \tag{2.1}$$

We assume that A_{II} is positive definite. A key part of our approach is constraining y_B to be of the form

$$y_B = N_B q\,. \tag{2.2}$$

The harmonic extension of y_B for y_I is then given by

$$y_{Ih} = -A_{II}^{-1} A_{IB} N_B q. \tag{2.3}$$

For a fixed y_c, we would like (motivated by [8]) to choose y_f and q such that y_I is the best approximation in energy norm to y_{Ih}. Thus, minimizing

$$E = (y_I - y_{Ih})^T A_{II} (y_I - y_{Ih})$$

with respect to y_f and q leads to the linear system

$$\underbrace{\begin{bmatrix} A_{ff} & A_{fB}N_B \\ N_B^T A_{fB}^T & N_B^T A_{IB}^T A_{II}^{-1} A_{IB} N_B \end{bmatrix}}_{\mathcal{A}} \begin{bmatrix} y_f \\ q \end{bmatrix} = - \begin{bmatrix} A_{fc} \\ N_B^T A_{cB}^T \end{bmatrix} y_c\,. \tag{2.4}$$

LEMMA 2.1.

$$A_{II}N_I + A_{IB}N_B = 0\,. \tag{2.5}$$

$\mathcal{A}$ (2.4) N_c . Consider the linear system

$$\mathcal{A}x = d, \quad x = \begin{bmatrix} y_f \\ q \end{bmatrix}, \quad d = \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}.$$

Solving the first block of equations for y_f and substituting the result back into the second block leads to

$$y_f = A_{ff}^{-1}(d_1 - A_{fB}N_B q)$$

and

$$Dq = d_2 + N_B^T A_{fB}^T A_{ff}^{-1} d_1,$$

where

$$D = N_B^T(A_{IB}^T A_{II}^{-1} A_{IB} - A_{fB}^T A_{ff}^{-1} A_{fB})N_B\,.$$

From (2.5) we have

$$A_{II}^{-1}A_{IB}N_B = -N_I \quad \text{and} \quad A_{ff}^{-1}A_{fB}N_B = -(N_f + A_{ff}^{-1}A_{fc}N_c)$$

so that

$$D = N_I^T A_{II} N_I - (N_f + A_{ff}^{-1}A_{fc}N_c)^T A_{ff}(N_f + A_{ff}^{-1}A_{fc}N_c) = N_c^T S N_c,$$

where

$$S = A_{cc} - A_{fc}^T A_{ff}^{-1} A_{fc}\,.$$

Since A_{II} is positive definite, both A_{ff} and the Schur complement S are positive definite. Thus, D is positive definite if the columns of N_c are linearly independent. Since D is the Schur complement of A_{ff} in $\mathcal{A}$, positive definiteness of A_{ff} and D implies positive definiteness of $\mathcal{A}$. Now if the columns of N_c are linearly dependent, then there exists a nonzero vector v such that $N_c v = 0$. It follows that $Dq = 0$ has the nontrivial solution $q = v$, which implies that $\mathcal{A}$ is singular and thus not positive definite. □

Returning for a moment to the proof of Lemma 2.1, by setting $d_1 = -A_{fc}y_c$ and $d_2 = -N_B^T A_{cB}^T y_c$ as in (2.4), we obtain

$$N_c^T S N_c q = N_c^T S y_c \tag{2.6}$$

and

$$y_f = -A_{ff}^{-1}(A_{fc}y_c + A_{fB}N_B q). \tag{2.7}$$

Equation (2.6) means that q gives the coordinates (in the N_c column basis) of the projection of y_c onto the range of N_c in the $(S\cdot,\cdot)$ inner product. Thus, y_B can be

viewed as a weighted projection of y_c onto the null space. This defines an extension mapping in the spirit of [8], which in the simplest case of a Poisson problem is given by a weighted averaging:

$$y_B = \frac{e^T S y_c}{e^T S e} e,$$

where $e = N_c$ is a unit vector with all entries equal to 1. It follows from (2.2) and (2.7) that y_f is a harmonic extension of y_c and y_B. Interpolation operators based on harmonic extensions appear in other approaches such as those in [4] and [8].

LEMMA 2.2. [illegible] (2.5) [illegible] N_c [illegible]

(2.8) $$y_c = N_c p\,,$$

[illegible] $y_f = N_f p$ [illegible] $q = p$ [illegible] (2.4) [illegible]. From Lemma 2.1 we have that $\mathcal{A}$ is nonsingular so that the solution of (2.4) is unique. It remains to show that the stated choices for y_f and q result in the smallest possible value of E. From (2.3) and (2.5) we have $y_{Ih} = N_I p$. We also have $y_I = N_I p$ so that $E = 0$, which is the smallest possible value of E since A_{II} is positive definite. ∎

Let $w = Np$ denote a vector in the range of the null space matrix N. Lemma 2.2 implies that if the problem satisfies a null space relation as in (2.5) and the coarse dofs are chosen such that $y_c = w_c$ as in (2.8), then y_f obtained from the solution of (2.4) equals w_f. In other words, the interpolation for y_f is consistent with the rigid body motion of the coarse nodes. This property is important in ensuring a scalable approach.

2.2. Construction. We assume that the interpolation of each noncoarse node is affected by all adjacent coarse nodes and that each noncoarse node is adjacent to at least one coarse node. Additional details on coarse nodes and their selection are provided in section 3. We now discuss how solutions from (2.4) can be used to construct interpolation operators for y_f. Combining (2.6) and (2.7) gives

$$y_f = -A_{ff}^{-1}(A_{fc} + A_{fB} N_B (N_c^T S N_c)^{-1} N_c^T S) y_c. \tag{2.9}$$

If the columns N_c are linearly independent, then (2.9) could be used as an interpolation formula without modification. Unfortunately, it may be problematic to satisfy this condition in practice for certain problem types such as elasticity.

The number of columns in N_c is one for the Poisson equation. Thus, N_c has full column rank provided that c contains at least one coarse node. For two-dimensional (2D) elasticity problems, the number of columns in N_c is three, but there are at most two dofs for each coarse node. Thus, if c consists of just a single coarse node, then the columns of N_c will be linearly dependent. The situation is potentially even more grave for three-dimensional (3D) elasticity problems for which the number of columns in N_c is six, but each coarse node has at most three dofs. Even if c consists of two coarse nodes, the columns of N_c will be linearly dependent as illustrated in Figure 2.1. We note for structural problems involving plate or shell elements that the columns of N_c will typically be linearly independent even if c consists of just one coarse node. This is true because the coarse nodes have both translational and rotational dofs.

One idea for addressing linear dependence in the columns of N_c is to augment the node set c. Doing so increases the number of rows in N_c and has the potential of

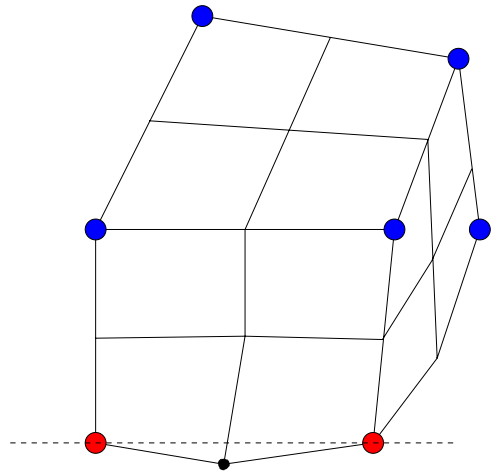


FIG. 2.1. *Simple mesh with eight elements and coarse nodes shown as large filled circles. The two coarse nodes influencing the bottom midedge node are connected by a dashed line. For an elasticity problem, these two coarse nodes do not move for a rotation about the dashed line. However, the midedge node will move unless it is on the line connecting the two coarse nodes. Thus, interpolation of the midedge node based on these two coarse nodes cannot reproduce all rigid body motions. In other words, the columns of N_c are not linearly independent. This problem can be avoided by introducing rotational dofs for the coarse nodes.*

removing linearly dependent columns. For example, if c contains at least two noncoincident coarse nodes, then N_c will have full column rank for 2D elasticity problems. The requirement for 3D elasticity is that c contains at least three noncolinear coarse nodes. Although this option may be easy to implement, it has its drawbacks. Namely, it can lead to significantly more coarse nodes or greater coupling at coarser multigrid levels.

We build on the basic idea of augmenting the set c, but with an important twist. The approach can be viewed as a two-step process in which the coarse node set c is first replaced by its augmented counterpart $\tilde{c}$. The formula in (2.9) then becomes

$$y_f = -A_{ff}^{-1}(A_{f\tilde{c}} + A_{f\tilde{B}}N_{\tilde{B}}(N_{\tilde{c}}^T\tilde{S}N_{\tilde{c}})^{-1}N_{\tilde{c}}^T\tilde{S})y_{\tilde{c}},$$

where $\tilde{}$ indicates that terms are for the augmented set $\tilde{c}$ rather than c. We assume the choice of $\tilde{c}$ ensures that the columns of $N_{\tilde{c}}$ are linearly independent. Since the dimension of $y_{\tilde{c}}$ may be significantly larger than that of y_c, we next impose constraints of the form

$$y_{\tilde{c}} = C\hat{y}_c, \tag{2.10}$$

where $\hat{y}_c$ is associated with the original coarse node set c. The net result is the modified interpolation formula

$$y_f = P_f\hat{y}_c, \tag{2.11}$$

where

$$P_f = -A_{ff}^{-1}(A_{f\tilde{c}} + A_{f\tilde{B}}N_{\tilde{B}}(N_{\tilde{c}}^T\tilde{S}N_{\tilde{c}})^{-1}N_{\tilde{c}}^T\tilde{S})C.$$

We have $\hat{y}_c = y_c$ except at the finest multigrid level for elasticity problems. For that special case the coarse vector $\hat{y}_c$ has rotational dofs whereas y_c does not. In summary,

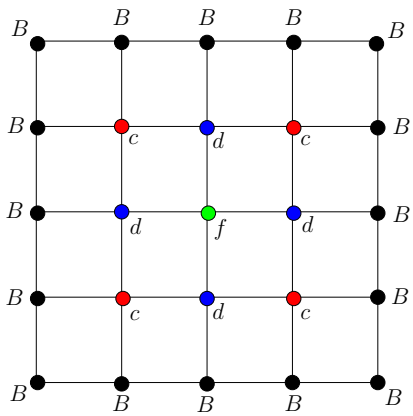


FIG. 2.2. *Example mesh showing node sets c, d, f, and B for a 2D problem.*

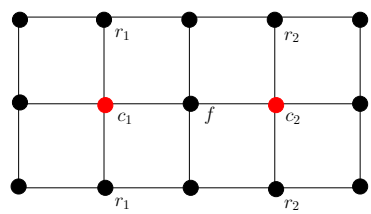


FIG. 2.3. *Example mesh showing node sets c_i, r_i, and f for 2D elasticity.*

the original coarse nodes c are still used for interpolation of y_f, but these coarse nodes now have both translational and rotational dofs for elasticity problems. We note that a different AMG approach for elasticity problems, which can be viewed as a blockwise generalization of classical AMG, is described in [7].

The remainder of this subsection discusses selection of the augmented coarse node set $\tilde{c}$ and the constraint matrix C. To begin, define

$$\tilde{c} = c \cup d \cup r,$$

where node sets d and r are subsets of the set $n(f)$ of noncoarse nodes adjacent to f. Let $c(f)$ denote the set of coarse nodes involved in the interpolation of f and observe that $c(f) = c$. The set of [illegible] nodes $d = \{d_j\}$ consists of all nodes in $n(f)$ for which $c(d_j) \subset c$ (see Figure 2.2). At the finest multigrid level for elasticity problems, we also consider a set of nodes $r = \{r_i\}$ associated with [illegible]. For each coarse node in $c = \{c_i\}$, r_i is defined as the set of all nodes in $n(f) \setminus d$ adjacent to c_i but not adjacent to any other nodes in c. A view of a 2D mesh showing nodes in the sets f, c_i, and r_i is shown in Figure 2.3. Note that $r = \emptyset$ for other multigrid levels or for problem types other than elasticity. It is also worth mentioning that r may be an empty set for certain noncoarse nodes such as the one at the center of the front face in Figure 2.1. This does no harm as long as f is adjacent to a sufficient number of coarse nodes.

Returning to (2.10), constraint equations for [illegible] dofs d_j are based on the interpolation formula (2.11). At the finest multigrid level for elasticity problems, constraint equations for r_i are based on the condition that all nodes in r_i and c_i be in the same rigid body. In order to describe the constraint matrix C in more detail,

we first rewrite (2.10) as

$$\underbrace{\begin{bmatrix} y_c \\ y_d \\ y_r \end{bmatrix}}_{y_{\tilde{c}}} = \underbrace{\begin{bmatrix} C_c \\ C_d \\ C_r \end{bmatrix}}_{C} \underbrace{\begin{bmatrix} \hat{y}_{c_1} \\ \vdots \\ \hat{y}_{c_{|c|}} \end{bmatrix}}_{\hat{y}_c}, \tag{2.12}$$

where

$$y_d = \begin{bmatrix} y_{d_1} \\ \vdots \\ y_{d_{|d|}} \end{bmatrix}, \quad y_r = \begin{bmatrix} s_1 \\ \vdots \\ s_{|c|} \end{bmatrix}, \quad s_i = \begin{bmatrix} y_{r_i^1} \\ \vdots \\ y_{r_i^{|r_i|}} \end{bmatrix}, \quad \hat{y}_{c_i} = \begin{bmatrix} y_{c_i} \\ y_{c_{i\theta}} \end{bmatrix}. \tag{2.13}$$

Except at the finest multigrid level for elasticity problems, C_c is a suitably dimensioned identity matrix I, and the matrix C_r and vectors y_r and $y_{c_{i\theta}}$ are ignored. Recall that the coarse nodes involved in the interpolation of d_j are contained in c. Thus, the interpolation formula (2.11) for y_{d_j} can be expressed as

$$y_{d_j} = P_{d_j} R_{d_j} \hat{y}_c,$$

where each row of the selection matrix R_{d_j} contains exactly one nonzero entry of 1. Note that $R_{d_j}\hat{y}_c$ is simply the vector of coarse dofs involved in the interpolation of y_{d_j}. It then follows from (2.12) and (2.13) that

$$C_d = \begin{bmatrix} P_{d_1} R_{d_1} \\ \vdots \\ P_{d_{|d|}} R_{d_{|d|}} \end{bmatrix}. \tag{2.14}$$

It remains to specify the matrices C_c and C_r for elasticity problems at the finest multigrid level. We restrict our attention to the 3D case since the 2D case is simpler. Let $\mathbf{r}_b$ and $\mathbf{r}_p$ denote position vectors of points b and p. If b and p are in the same rigid body, then for small rotations the displacement of p is given approximately by

$$\mathbf{y}_p = \mathbf{y}_b + \boldsymbol{\theta}_b \times (\mathbf{r}_p - \mathbf{r}_b), \tag{2.15}$$

where $\mathbf{y}_b$ is the displacement of b and $\boldsymbol{\theta}_b$ is a vector of rotations about b. Define $y_{pk} = \mathbf{y}_p \cdot \mathbf{e}_k$, where $\mathbf{e}_k$ is a unit vector in coordinate direction k. In matrix notation, (2.15) can then be expressed as

$$y_p = C(\mathbf{r}_p - \mathbf{r}_b) \begin{bmatrix} y_b \\ \theta_b \end{bmatrix},$$

where

$$y_p = \begin{bmatrix} y_{p1} \\ y_{p2} \\ y_{p3} \end{bmatrix}, \quad C(\mathbf{r}) = \begin{bmatrix} 1 & 0 & 0 & 0 & r_3 & -r_2 \\ 0 & 1 & 0 & -r_3 & 0 & r_1 \\ 0 & 0 & 1 & r_2 & -r_1 & 0 \end{bmatrix}.$$

We then have

$$C_c = \begin{bmatrix} C(\mathbf{0}) R_{c_1} \\ \vdots \\ C(\mathbf{0}) R_{c_{|c|}} \end{bmatrix}, \quad C_r = \begin{bmatrix} C_{s_1} R_{c_1} \\ \vdots \\ C_{s_{|c|}} R_{c_{|c|}} \end{bmatrix}, \quad C_{s_i} = \begin{bmatrix} C(\mathbf{r}_i^1 - \mathbf{c}_i) \\ \vdots \\ C(\mathbf{r}_i^{|r_i|} - \mathbf{c}_i) \end{bmatrix}, \tag{2.16}$$

where **0** denotes a vector of length zero and $\hat{y}_{c_i} = R_{c_i}\hat{y}_c$. In summary, the matrix C_d is based on the interpolation formula in (2.11), while the matrix C_r is based on rigid body constraints.

Noncoarse nodes f are processed in order of increasing $|c|$ so the interpolation matrix P_{d_j} will already be available for use in (2.14). Note in the expression for P_f below (2.11) that factorizations are required for the matrices A_{ff} and $N_{\tilde{c}}^T S N_{\tilde{c}}$. The largest possible dimension of these matrices in this study is 6×6 for 3D elasticity. The interpolation of y_f by coarse dofs is summarized in Algorithm 1.

ALGORITHM 1. Interpolation operator for y_f.

assemble constraint matrix C_d in (2.14).
if finest multigrid level for elasticity problem **then**
  determine C_c and C_r in (2.16)
else
  set $C_c = I$ and $C_r = [\ ]$, a null matrix with zero rows
end if
assemble constraint matrix C in (2.12)
determine interpolation operator from (2.11)

Recall that each column of the interpolation matrix P corresponds to a particular coarse node. At the finest multigrid level for elasticity problems, it may happen, e.g., due to essential boundary conditions, that the columns of P for such a node are linearly dependent. If this is the case, then the diagonal block of the coarse matrix $P^T A P$ corresponding to this node, call it A_n, will be singular. In order to address such a possibility, consider the eigendecomposition

$$A_n \Phi = \Phi \Lambda,$$

where Φ is a matrix of orthonormal eigenvectors and the diagonal matrix Λ contains the eigenvalues. If any of the eigenvalues are zero, then we make the correction

$$A_n \leftarrow A_n + \alpha \tilde{\Phi} \tilde{\Phi}^T,$$

where α is the trace of A_n and $\tilde{\Phi}$ consists of the columns of Φ corresponding to zero eigenvalues. We have observed for very special case 3D elasticity problems that the above correction may not be sufficient to address a rank-one deficiency of the interpolation matrix P of a more global nature involving rotational dofs. This rank deficiency has only been observed for meshes of identical parallelepipeds and regular arrangements of coarse nodes at the finest level. It has not caused numerical problems for any examples considered thus far, but safeguards could be implemented if the coarse matrix $P^T A P$ is factored in a two-level multigrid for 3D elasticity. Two possible safeguards include scaling the diagonal entries of $P^T A P$ by the factor $1 + \epsilon$ with $0 < \epsilon \ll 1$ and a Laplacian-like stabilization of rotational dofs which is orthogonal to rigid body modes.

REMARK 1. If the null space matrix N is not available, then it may be possible to use a self-correcting method similar to adaptive smoothed aggregation [3] to address this problem. We do not consider such a method here.

3. Numerical examples. Numerical examples are presented in this section for both structured and unstructured meshes. The purpose of these examples is to investigate the effects of problem size, multigrid cycle type, boundary conditions, domain

aspect ratios, material property jumps, and element polynomial degree on AMG performance. All the examples have a random right-hand side and are solved to a relative residual tolerance of 10^{-8} using the preconditioned conjugate gradient algorithm. The AMG preconditioners use nodal Gauss–Seidel relaxation at each level except for the coarsest level where the problem is solved directly. In order to ensure symmetry of the preconditioners, the ordering of nodes for Gauss–Seidel relaxation is reversed in the pre- and postsmoothing steps. Estimated condition numbers are obtained using the connection between conjugate gradients and the Lanczos method [6]. These estimates are lower bounds for the actual condition numbers and are obtained from the extremal eigenvalues of a tridiagonal Lanczos matrix of dimension equal to the number of iterations. The Poisson ratio for all the elasticity examples was set to 0.3.

Coarse nodes at each level are chosen using a maximal independent set algorithm in which nodes of lowest degree are successively added to the coarse set. Thus, none of the coarse nodes are adjacent, and all the remaining noncoarse nodes are adjacent to one or more coarse nodes. We restrict our attention in all the examples to good element aspect ratios and isotropic materials. For elements with poor aspect ratios or materials with large anisotropies, additional coarse nodes would need to be introduced to avoid degraded performance. We do not consider such cases here and only mention that semicoarsening or compatible relaxation [1, 11, 5] could be used to address such problems. We specify that the interpolation of a noncoarse node involves precisely all those coarse nodes that are adjacent to it and no others.

The first examples are for structured meshes of 8-node hexahedral elements on a rectangular parallelepiped domain with dimensions L_1 and $L_2 = L_3 = 1$. In each table refers to the numbers of levels (1 for a direct solution), is the total number of unknowns in the problem, is the number of iterations, is the condition number estimate of the preconditioned operator, and is the operator complexity (sum of number of nonzeros in stiffness matrices at all levels divided by the number of nonzeros at the finest level). Columns headed by $V(m, n)$ and $W(m, n)$ are for symmetric V- and W-cycles with m presmoothing and n postsmoothing steps. For a few examples, results are also shown for symmetric full multigrid V ( ) and W ( ) cycles.

Shown in Table 3.1 are results for a cube domain with essential boundary conditions applied to all six faces. In this case, the coarse node selection algorithm leads to a regular geometric coarsening. Very good performance of both the V and W cycles is evident. Results for the same problem with just one face constrained are shown in Table 3.2. In this case, the coarse nodes do not form a regular grid, and differences between the V- and W-cycles are more pronounced. Results for a domain with an aspect ratio of 8 are shown in Table 3.3. Again, optimal performance of the W-cycle in terms of iterations, condition numbers, and operator complexity is evident. Results for a cube domain with a centered inner cube region of dimension 1/2 are shown in Table 3.4. The elastic modulus has a value of σ in the inner cube region and 1 elsewhere. Although not perfect, the performance of the AMG preconditioner is not greatly sensitive to jumps in material properties for this example.

The next group of examples deals with both 2D and 3D unstructured meshes. Figure 3.1 shows a series of quadrilateral meshes in which the number of elements approximately quadruples with each mesh refinement. Results in Table 3.5 demonstrate the very good performance of the AMG preconditioner for unstructured meshes. The 3D examples are for the unstructured mesh of tetrahedral elements shown in Figure 3.2. Results in Tables 3.6 and 3.7 are for meshes of 4-node and 10-node tetra-

TABLE 3.1
Results for cube domain with essential boundary conditions applied to all six faces.

			$V(1,1)$		$W(1,1)$		
	nlev	ndof	iter	cond	iter	cond	oc
Poisson	2	729	6	1.13	6	1.13	1.14
	3	4913	7	1.18	7	1.18	1.15
	4	35,937	7	1.19	7	1.19	1.15
	5	274,625	7	1.21	7	1.19	1.15
Elasticity	2	2187	8	1.36	8	1.36	1.56
	3	14.739	9	1.39	8	1.38	1.61
	4	107,811	9	1.39	9	1.38	1.59
	5	823,875	9	1.39	9	1.38	1.58

TABLE 3.2
Results for cube domain with essential boundary conditions applied to one face.

			$V(1,1)$		$W(1,1)$		
	nlev	ndof	iter	cond	iter	cond	oc
Poisson	2	1210	8	1.35	8	1.35	1.13
	3	6498	10	1.53	9	1.40	1.16
	4	41,650	11	1.77	9	1.41	1.17
	5	296,274	13	2.24	9	1.41	1.17
Elasticity	2	3630	10	1.63	10	1.63	1.53
	3	19,494	12	1.81	11	1.66	1.64
	4	124,950	13	2.17	11	1.66	1.67
	5	888,822	15	2.52	11	1.66	1.68

TABLE 3.3
Results for rectangular parallelepiped domain with aspect ratio of 8 *($L_1 = 8$). Essential boundary conditions applied to one of the two smallest faces.*

			$V(1,1)$		$W(1,1)$		
	nlev	ndof	iter	cond	iter	cond	oc
Poisson	2	5184	9	1.36	9	1.36	1.15
	3	36,992	10	1.54	9	1.39	1.18
	4	278,784	11	1.68	9	1.40	1.18
Elasticity	2	15,552	12	1.64	12	1.64	1.61
	3	110,976	13	1.78	12	1.63	1.71
	4	836,352	16	2.23	12	1.60	1.70

TABLE 3.4
Results for cube problem with material property jumps. The elastic modulus equals σ in centered cube region of dimension 1/2 *and* 1 *elsewhere. The mesh consists of a $32 \times 32 \times 32$ arrangement of elements, and essential boundary conditions are applied to one face. Values of σ for problems with material property jumps appear in the second column of Table* 3.8 *results.*

σ	Poisson nlev=4, ndof=34,848				Elasticity nlev=4, ndof=104,544			
	$V(1,1)$		$W(1,1)$		$V(1,1)$		$W(1,1)$	
	iter	cond	iter	cond	iter	cond	iter	cond
10^{-4}	11	1.92	10	1.57	14	2.18	12	1.82
10^{-2}	11	1.92	10	1.53	14	2.18	11	1.69
1	11	1.91	9	1.39	13	2.10	11	1.62
10^{2}	14	2.63	12	1.87	17	3.42	14	2.22
10^{4}	15	2.90	12	2.04	23	5.26	17	2.98

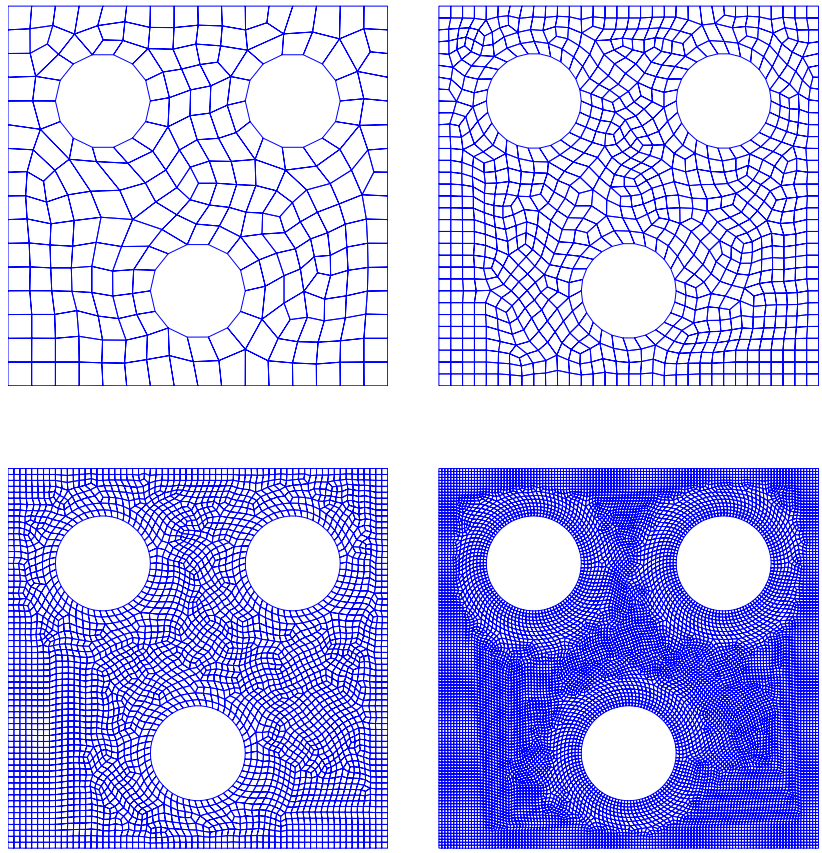


FIG. 3.1. *Unstructured quadrilateral meshes for results in Table* 3.5.

TABLE 3.5

Results for quadrilateral meshes in Figure 3.1 *with essential boundary conditions applied to the left edge.*

			$V(1,1)$		$W(1,1)$		
	nlev	ndof	iter	cond	iter	cond	oc
Poisson	2	273	9	1.43	9	1.43	1.30
	3	945	12	2.04	10	1.63	1.37
	4	3539	14	2.59	10	1.51	1.38
	5	12,671	17	3.32	11	1.55	1.38
Elasticity	2	546	11	1.68	11	1.68	1.67
plane	3	1890	15	2.56	12	1.91	1.84
stress	4	7078	19	3.56	12	1.75	1.86
	5	25,342	22	4.40	12	1.68	1.85

hedral elements, respectively. The 4-node tetrahedron is a linear element, whereas the 10-node tetrahedron is a quadratic element. Here again, the AMG preconditioners exhibit very good performance.

The final group of examples involves comparisons with the smoothed aggregation code called ML [13]. The symmetric Gauss–Seidel smoother in ML does smoothing in both forward and reverse lexicographic order. Thus, for purposes of comparison, the present approach (designated AMG-LO for AMG by local optimization) uses V-

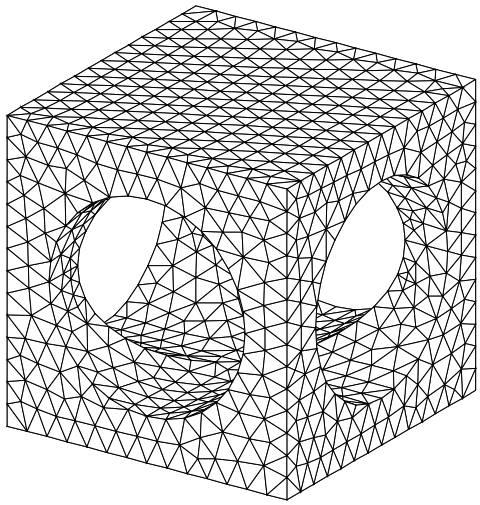


FIG. 3.2. *Unstructured tetrahedral mesh for results in Tables* 3.6 *and* 3.7. *The mesh has* 10,194 *elements.*

TABLE 3.6
Results for mesh in Figure 3.2 *with* 4*-node tetrahedral elements. Essential boundary conditions are applied to the left face.*

	nlev	$V(1,1)$		$W(1,1)$		$FMV(1,1)$		$FMW(1,1)$		oc
		iter	cond	iter	cond	iter	cond	iter	cond	
Poisson	2	12	1.92	12	1.92	10	1.54	10	1.54	1.34
ndof = 2321	3	14	2.48	13	1.98	10	1.57	10	1.55	1.38
	4	15	2.90	13	1.99	10	1.58	10	1.55	1.38
Elasticity	2	14	2.33	14	2.33	11	1.78	11	1.78	2.37
ndof = 6963	3	23	6.68	18	4.01	13	2.45	12	2.12	2.52
	4	30	9.54	19	4.41	15	2.89	13	2.24	2.53

TABLE 3.7
Results for mesh in Figure 3.2 *with* 10*-node tetrahedral elements. Essential boundary conditions are applied to the left face.*

	nlev	$V(1,1)$		$W(1,1)$		$FMV(1,1)$		$FMW(1,1)$		oc
		iter	cond	iter	cond	iter	cond	iter	cond	
Poisson	2	16	2.70	16	2.70	14	2.44	14	2.44	1.41
ndof = 16,056	3	19	3.70	16	2.72	15	2.45	15	2.44	1.42
	4	23	5.27	16	2.76	15	2.46	15	2.44	1.42
Elasticity	2	20	4.10	20	4.10	18	3.35	18	3.35	2.64
ndof = 48,168	3	26	5.95	21	4.13	19	3.49	19	3.47	2.67
	4	37	13.1	23	4.93	20	3.56	19	3.47	2.68

and W-cycles with two rather than one pre- and postsmoothing step. We note the default ML parameter value of 0.67 for [illegible] was increased to 1.0 because ML was run on a single processor and better performance was observed for this value.

Results for AMG-LO and ML applied to elasticity problems appear in Table 3.8. Coarsening for both approaches was stopped once the coarse problem size was less than 1100. For the structured meshes (Tables 3.2–3.4), the number of iterations and condition numbers for the two approaches are comparable, but the operator

TABLE 3.8
Comparison of the present approach (AMG-LO) with a smoothed aggregation approach (ML) for elasticity problems. Times are in seconds (indicates less than* 1 *second).*

		AMG-LO							
			V			W			
Table	ndof	nlev	iter	cond	time	iter	cond	time	oc
3.2	3630	2	8	1.23	*	8	1.23	*	1.53
	19,494	3	9	1.44	2	8	1.23	2	1.64
	124,950	4	11	1.81	10	8	1.22	11	1.67
	888,822	5	13	1.99	85	8	1.21	73	1.68
3.3	15,552	2	9	1.25	2	9	1.25	2	1.61
	110,976	3	11	1.46	9	9	1.24	10	1.71
	836,352	4	13	1.73	79	9	1.23	75	1.70
3.4	10^{-4}	4	12	1.93	9	9	1.40	9	1.66
σ	10^{-2}		12	1.93	9	9	1.33	9	
in	1		12	1.89	9	8	1.24	8	
next	10^2		15	2.77	10	11	1.73	10	
col	10^4		19	4.28	12	14	2.28	11	
3.5	25,342	4	15	2.40	2	10	1.40	2	1.84
3.6	6963	3	20	5.42	*	15	3.20	*	2.52
3.7	48,168	3	22	4.40	9	16	2.68	9	2.67
		ML							
3.2	3630	2	9	1.44	*	9	1.44	*	1.15
	19,494	3	12	1.92	2	10	1.58	1	1.15
	124,950	3	13	2.01	12	11	1.61	11	1.15
	888,822	4	14	2.06	97	11	1.61	88	1.15
3.3	15,552	3	16	2.66	1	13	1.98	1	1.11
	110,976	3	15	2.30	10	12	1.69	9	1.14
	836,352	4	15	2.06	86	13	1.61	82	1.14
3.4	10^{-4}	3	14	2.20	10	12	1.80	9	1.14
σ	10^{-2}		13	2.16	9	12	1.76	9	
in	1		12	1.89	9	11	1.61	9	
next	10^2		20	4.37	12	17	3.09	12	
col	10^4		29	9.08	16	23	5.62	14	
3.5	25,342	3	16	2.60	*	13	1.87	*	1.32
3.6	6963	2	14	2.30	*	14	2.30	*	1.43
3.7	48,168	3	68	54.9	14	52	30.8	12	1.14

complexity for ML is noticeably lower. This is true because smoothed aggregation typically uses a 3-1 coarsening ratio rather than the 2-1 ratio used by the present approach. The number of iterations and condition numbers for the 2D plane stress example with an unstructured mesh are also comparable. Although the performance of the two approaches is comparable for the unstructured mesh of 4-node tetrahedral elements, there is a notable difference for the mesh of 10-node quadratic elements (Table 3.7). Notice that the operator complexity for ML drops from 1.43 for linear elements to 1.14 for quadratic elements. We plan to investigate further the cause for the poorer performance of ML in this case. Although the software for AMG-LO is relatively new, it is competitive with ML in terms of run times reported in Table 3.8.

4. Conclusions. Numerical examples demonstrated that the new approach for constructing interpolation operators results in optimal performance of an ensuing AMG preconditioner. Iterations, condition numbers, and operator complexities all appeared to level off with growing problem size in the examples for W-cycles. In addition, the numerical implementation of the approach was shown to be competitive with an existing smoothed aggregation code for a suite of elasticity problems. The

new approach can be viewed as a classical AMG alternative to smoothed aggregation for elasticity problems.

In order for the new approach to be useful as a general-purpose solver or preconditioner, it must be coupled with an effective method for selecting the coarse nodes. As with other classical AMG approaches, coarse node selection is of great importance for problems with large element aspect ratios or material anisotropies. Future work will focus on developing new approaches for adaptively selecting the coarse nodes to ensure good performance for a broader class of ill-conditioned problems.

Acknowledgment. The author is grateful to the anonymous reviewers for their very helpful comments and suggestions. Their input led to both a clearer presentation and a more efficient numerical implementation of the approach.

REFERENCES

[1] A. Brandt, *General highly accurate algebraic coarsening*, Electron. Trans. Numer. Anal., 10 (2000), pp. 1–20.

[2] M. Brezina, A. J. Cleary, R. D. Falgout, V. E. Henson, J. E. Jones, T. A. Manteuffel, S. F. McCormick, and J. W. Ruge, *Algebraic multigrid based on element interpolation (AMGe)*, SIAM J. Sci. Comput., 22 (2000), pp. 1570–1592.

[3] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge, *Adaptive smoothed aggregation (αSA)*, SIAM J. Sci. Comput., 25 (2004), pp. 1896–1920.

[4] T. F. Chan and W. L. Wan, *Robust Multigrid Methods for Elliptic Linear Systems*, Technical report CAM 99-30, Department of Mathematics, UCLA, Los Angeles, CA, 1999.

[5] R. D. Falgout and P. S. Vassilevski, *On generalizing the algebraic multigrid framework*, SIAM J. Numer. Anal., 42 (2004), pp. 1669–1693.

[6] G. H. Golub and C. F. V. Loan, *Matrix Computations*, 2nd ed., The Johns Hopkins University Press, Baltimore, MD, 1989.

[7] M. Griebel, D. Oeltz, and M. A. Schweitzer, *An algebraic multigrid method for linear elasticity*, SIAM J. Sci. Comput., 25 (2003), pp. 385–407.

[8] V. E. Henson and P. S. Vassilevski, *Element-free AMGe: General algorithms for computing interpolation weights in AMG*, SIAM J. Sci. Comput., 23 (2001), pp. 629–650.

[9] J. E. Jones and P. S. Vassilevski, *AMGe based on element agglomeration*, SIAM J. Sci. Comput., 23 (2001), pp. 109–133.

[10] J. K. Kraus, *Computing interpolation weights in AMG based on multilevel Schur complements*, Computing, 74 (2005), pp. 319–335.

[11] O. E. Livne, *Coarsening by compatible relaxation*, Numer. Linear Algebra Appl., 11 (2004), pp. 205–227.

[12] J. Mandel, M. Brezina, and P. Vaněk, *Energy optimization of algebraic multigrid bases*, Computing, 62 (1999), pp. 205–228.

[13] M. Sala, M. Gee, J. Hu, and R. Tuminaro, *ML* 4.0 *Smoothed Aggregation User's Guide*, Technical report SAND 2004-4819, Sandia National Laboratories, Albuquerque, NM, 2005.

[14] P. Vaněk, M. Brezina, and J. Mandel, *Convergence of algebraic multigrid based on smoothed aggregation*, Numer. Math., 88 (2001), pp. 559–579.

[15] P. Vaněk, J. Mandel, and M. Brezina, *Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems*, Computing, 56 (1996), pp. 179–196.

[16] W. L. Wan, T. F. Chan, and B. Smith, *An energy-minimizing interpolation for robust multigrid methods*, SIAM J. Sci. Comput., 21 (2000), pp. 1632–1649.

SIAM J. SCI. COMPUT.
Vol. 29, No. 5, pp. 2059–2077
© 2007 Society for Industrial and Applied Mathematics

PARALLEL SEGREGATED SCHUR COMPLEMENT METHODS FOR FLUID DENSITY FUNCTIONAL THEORIES*

MICHAEL A. HEROUX†, ANDREW G. SALINGER†, AND LAURA J. D. FRINK†

Abstract. Numerical formulations of density functional theories for inhomogeneous fluids (fluid-DFTs) require the solution of large systems of equations with many degrees of freedom (DOF) per node on a computational grid. Historically, solvers for these problems have used simple Picard iterations across DOF or, more recently, fully coupled general algebraic techniques. In this paper we look at fluid-DFTs from a fresh perspective, retaining a fully coupled formulation but segregating variables for the purpose of introducing Schur complement formulations and specialized preconditioners. By viewing fluid-DFTs from this perspective, we develop a mathematical framework and a collection of solution algorithms that have a dramatic impact on the robustness, performance, and scalability of the implicit equations generated by fluid-DFTs.

Key words. inhomogeneous fluids, density functional theories, Schur complement, iterative methods

AMS subject classifications. 65F10, 65F50, 65M70

DOI. 10.1137/060661594

1. Introduction.

Multiple degrees of freedom (DOF) per node properties are common to many numerical applications. Segregated solvers, which attempt to view each DOF across the grid as a subproblem within the larger fully coupled problem, have been successfully used in many problem domains. Similarly Schur complement methods, which formally eliminate variables by block Gaussian elimination, can reduce the complexity and cost of solution. In this paper we present a combination of these two classes of algorithms applied to density functional theories for inhomogeneous fluids (fluid-DFTs). We will show that viewing fluid-DFTs from a segregated variable perspective yields a rich structure that can be exploited in the development of robust, scalable solution methods. Furthermore, the algorithms we develop can be applied in a similar fashion to all types of problems generated by the target application, showing that our basic approach is a useful general-purpose technique in this problem domain.

Density functional theories (DFTs) have been used to treat a variety of systems at many length scales. For interfacial systems, the fundamental problem is to predict the structure of an inhomogeneous fluid as captured by a density distribution, $\rho(\mathbf{r})$ [36]. At the smallest length scale, the most well known application of DFTs is to predict the electronic structure of materials [23]. More specifically, quantum mechanical DFTs (QM-DFTs) are used to predict the structure of an electron cloud in an external field produced by fixed nuclei. Using a similar mathematical construct, but with nonexact density functionals [13], the structure of atomic [24, 25], molecular [18, 1], and polymer fluids [2, 37, 38, 33] can be computed. Fluid inhomogeneities can result from surfaces (e.g., planar interfaces, porous materials, or large geometrically complex macromolecules [10, 35, 21, 9]) or from competing intramolecular and intermolecular

*Received by the editors May 31, 2006; accepted for publication (in revised form) March 27, 2007; published electronically September 28, 2007. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed-Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000.

http://www.siam.org/journals/sisc/29-5/66159.html

†Sandia National Laboratories, Albuquerque, NM 87185 (maherou@sandia.gov, agsalin@sandia.gov, ljfrink@sandia.gov).

interactions that can lead to self-assembly [12, 6, 20]. Mesoscale-DFTs have also been developed for colloidal fluids and biological macromolecules [4, 5]. These mesoscale-DFTs are very similar to fluid-DFTs but are based on coarsened models or potentials (e.g., the solvent averaged Yukawa potential).

Our previous efforts to develop numerical methods for fluid-DFTs began with the development of a real space approach using Newton's method that could be solved using parallel iterative solvers optimized for large distributed memory parallel computers [7, 8, 10]. This code was robust, but expensive for 3-dimensional systems. Given those limitations, we then developed a matrix-free method with fast Fourier transforms (FFTs) to compute certain convolutions in the theory [30]. This approach led to a code that could be applied to 3-dimensional problems using very modest computer resources (single processor workstations). However, the FFTs limit the application space of the DFTs to cases with periodic boundary conditions, and since no matrix is stored, matrix-based preconditioning methods cannot be applied to converge difficult nonlinear problems.

In the remainder of this paper we present a new approach to solving fluid-DFTs using a real space method based on segregated Schur complement techniques and we demonstrate the impact of this approach on our ability to solve large, complex problems. Section 2 presents the general mathematical framework of fluid-DFTs. Sections 3 discusses the complexities of the discrete formulation of DFTs with a focus on properties that can be exploited in the design of algorithms. Section 4 contains a discussion of our new block equation framework, and section 5 discusses the block framework for two particular fluid-DFTs. Section 6 presents an analysis of our new solver algorithms compared to generic methods developed for partial differential equations (PDEs) and demonstrates considerable improvement in speed and robustness.

2. Mathematical framework. In fluid-DFTs an approximate free energy functional, $\Omega[\{\rho_i(\mathbf{r})\}]$, depends on a set of density fields, $\{\rho_i(\mathbf{r})\}$. The minimization of this free energy with respect to all fields $\{\rho_i(\mathbf{r})\}$ results in a system of residual equations to be solved for the equilibrium fluid distribution. The residual equations to be solved are written as

$$\frac{\delta\Omega}{\delta\rho_i(\mathbf{r})} = 0, \tag{2.1}$$

where δ is the Fréchet (functional) derivative, leading to an Euler–Lagrange residual equation.

In a perturbation theory, the free energy Ω is typically written as a sum of terms starting with the contributions of a well-known reference fluid (often a hard-sphere fluid). Many terms in these physics-based expansions can be written as

$$F_{phys} = \int f[\{\rho_i\}]g[\{n_\gamma[\{\rho_i\}]\}]dr, \tag{2.2}$$

where f is a functional of the critical density fields, and g is a functional of some nonlocal variables, n. Taking the functional derivative of these terms to set up the residual equations in (2.1), one must apply the chain rule

$$\frac{\delta}{\delta\rho_i(\mathbf{r})}\left(\int f[\{\rho_i\}]g[\{n[\{\rho_i\}]\}]dr\right) = \frac{\partial f}{\partial\rho_i}g[\{n\}] + \int f[\{\rho_i\}]\frac{\partial g}{\partial n}\frac{\delta n(\mathbf{r}')}{\delta\rho_i(\mathbf{r})}d\mathbf{r}', \tag{2.3}$$

where for simplicity (2.3) reflects a problem with only one nonlocal variable, n.

For local terms (e.g., ideal gas contributions), $g[n] = 1$ and the second term drops out. Nonlocal terms can take on various forms. For strict mean field contributions, $g = n_i$, where n_i is a simple linear functional of the set $\{\rho_i\}$,

$$n_i(\mathbf{r}) = \sum_j \int w_{ij}(\mathbf{r}, \mathbf{r}', R_{ij})\rho_j(\mathbf{r}')d\mathbf{r}', \tag{2.4}$$

with a weight function,

$$w_{ij}(\mathbf{r}, \mathbf{r}', R_{ij}) = \theta(|\mathbf{r} - \mathbf{r}'| - R_{ij})\phi_{ij}(|\mathbf{r} - \mathbf{r}'|), \tag{2.5}$$

where θ is a Heaviside step function, ϕ_{ij} denotes a pairwise interaction (i.e., pair potential, direct correlation function, pair correlation function, etc.), and R_{ij} is the characteristic range of the pairwise interaction. On differentiation, $\partial g/\partial n_i = 1$ and $\delta n_i/\delta\rho_j = w_{ij}(\mathbf{r}, \mathbf{r}', R_{ij})$. When forming a matrix problem it is necessary to calculate second derivatives of the free energy (the Hessian). One applies the chain rule in (2.3) a second time. For these strict mean field terms, $\partial^2 g/\partial n^2 = 0$, and thus the Jacobian contribution from these terms is simply

$$J_{ij}(\mathbf{r}, \mathbf{r}') = w_{ij}(\mathbf{r}, \mathbf{r}', R_{ij}). \tag{2.6}$$

While the number of nonzeros in the matrix is determined by the characteristic length, R_{ij}, these terms do not have an integral character at the level of the Hessian.

In contrast, consider a term, F_{phys}, in the free energy expansion with the form $f[\{\rho_i\}] = 1$ and $g[\{n_\gamma\}] = \Phi[n]$, where Φ is a nonlinear functional of a set of $\{n_\gamma\}$. In this case the first term in the chain rule drops out, and the first and second derivatives of these terms will be

$$\frac{\delta F_{phys}}{\delta\rho_i(\mathbf{r})} = \int \sum_\gamma \frac{\partial\Phi}{\partial n_\gamma}\frac{\delta n_\gamma(\mathbf{r}')}{\delta\rho_i(\mathbf{r})}d\mathbf{r}' \tag{2.7}$$

and

$$\frac{\delta^2 F_{phys}}{\delta\rho_i(\mathbf{r})\delta\rho_\mathbf{j}(\mathbf{r}')} = \int \sum_\gamma \sum_\epsilon \frac{\partial^2\Phi}{\partial n_\gamma \partial n_\epsilon}\frac{\delta n_\gamma(\mathbf{r}'')}{\delta\rho_i(\mathbf{r})}\frac{\delta n_\epsilon(\mathbf{r}'')}{\delta\rho_j(\mathbf{r}')}d\mathbf{r}''. \tag{2.8}$$

Often, n_γ is again a linear functional of $\{\rho_i\}$,

$$n_\gamma(\mathbf{r}) = \sum_i \int \omega_{\gamma i}(\mathbf{r}, \mathbf{r}', R_i)\rho_i(\mathbf{r}')d\mathbf{r}', \tag{2.9}$$

and in this case the Hessian contribution may be written

$$\frac{\delta^2 F_{phys}}{\delta\rho_i(\mathbf{r})\delta\rho_\mathbf{j}(\mathbf{r}')} = \int \sum_\gamma \sum_\epsilon \frac{\partial^2\Phi}{\partial n_\gamma \partial n_\epsilon}\omega_{\gamma i}(\mathbf{r}, \mathbf{r}'', R_i)\omega_{\epsilon j}(\mathbf{r}', \mathbf{r}'', R_j)d\mathbf{r}''. \tag{2.10}$$

Note that the weight function, ω, is usually based on either a Heaviside step function, $\theta(|\mathbf{r} - \mathbf{r}'| - R_i)$, or a Dirac delta function, $\delta(|\mathbf{r} - \mathbf{r}'| - R_i)$, where R_i is often a particle radius or diameter.

Computing matrix coefficients based on (2.10) requires a second-order (N^2) operation in order to locate the intersection of the weight functions $\omega_{\gamma i}(\mathbf{r}, \mathbf{r}'', R_i)$ and $\omega_{\epsilon j}(\mathbf{r}', \mathbf{r}'', R_j)$. To reduce the complexity of the matrix fill, the nonlocal variables,

n_γ, can be included as explicit independent variables when taking the second Fréchet derivatives to form the matrix problem. We will refer to this method as a first-order formulation later in this paper, and we note that this approach is akin to the transformation of a higher-order PDE to a system of first-order PDEs by introduction of additional variables in the problem. In this case, the nonzero contributions to the Jacobian will be

$$J_{\gamma\epsilon}(\mathbf{r},\mathbf{r}') = \delta_{\gamma\epsilon}(\mathbf{r},\mathbf{r}'), \tag{2.11}$$

$$J_{\gamma i}(\mathbf{r},\mathbf{r}') = \omega_\gamma(\mathbf{r},\mathbf{r}'), \tag{2.12}$$

$$J_{i\epsilon}(\mathbf{r},\mathbf{r}') = \sum_\gamma \frac{\partial^2 \Psi}{\partial n_\gamma \partial n_\epsilon}(\mathbf{r})\omega_{\gamma i}(\mathbf{r}',\mathbf{r}), \tag{2.13}$$

where the first two contributions arise from the nonlocal variable residual equation defined by (2.9), and the third contribution comes from taking the functional derivative of the Euler–Lagrange equation in (2.7) with respect to the nonlocal variables, n_ϵ. Note that there is no contribution from the derivative of the Euler–Lagrange contribution in (2.7) with respect to the density fields in this case (i.e., $J_{ij}(\mathbf{r},\mathbf{r}') = 0$).

Integral operator terms of the form discussed above are pervasive in DFTs constructed as perturbations to the hard-sphere fluid. However, the general approach of introducing variables to reduce the complexity of the matrix problem is a strategy that can be applied to other classes of fluid-DFTs as well. The polymer-DFT fluids that we discuss later in this paper are based on a second-order density expansion using an ideal chain reference fluid. Although this DFT falls into a different class than hard-sphere perturbation DFTs, the strategy of introducing variables to reduce complexity can also be applied there.

Including nonlocal variables, n in the systems of equations can significantly increase the number of DOF that must be solved at each mesh point. As a result, no advantage was gained from this approach when coupled to a generic PDE optimized solver applied to the full matrix problem [7]. In that case the decrease in the complexity of the matrix fill operation was offset by the increase in problem size for the linear solver. In the current work we demonstrate that when generic PDE solvers are replaced with segregated modeling strategies, fluid-DFTs can be solved with significantly improved efficiency.

3. Discrete formulations. As was mentioned in the introductory section, we have previously developed a code for solving fluid-DFTs on large distributed memory computers. That code was based on a discretization using a uniform structured grid so that the numerical integration stencils for all spatial integrals could be precomputed on a reference grid and then applied anywhere on the grid [7, 8]. Linear interpolation was used for the critical fields between the mesh points, and the equations were discretized using collocation at the mesh points.

The discretized equations were previously solved using a fully coupled Newton method with an analytic Jacobian and algebraic preconditioned Krylov methods. Convergence with these methods is much improved over Picard iteration, reducing nonlinear iterations from $O(100\text{–}1000)$ to $O(10)$. However, these linear solvers were designed for PDE applications and are far from optimal for fluid-DFTs. They sometimes require very expensive preconditioners in order to converge.

The ineffectiveness of the generic methods to efficiently solve the fluid-DFT equations stems primarily from the fact that the systems of equations are very dissimilar to PDEs. Key properties of the fluid-DFTs systems of equations when compared to

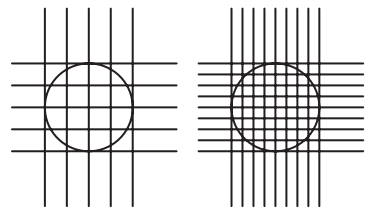


FIG. 3.1. *Stencil support for two mesh densities for integrating the area of the circle. As the mesh is refined, the stencil density increases. This corresponds to more expensive computation of the integrals and more nonzeros per row of the Jacobian matrix.*

PDEs that lead to failure of standard algorithms are as follows:

1. Interphysics versus internodal coupling: While PDEs have a strong spatial coupling, where the equation for a given variable at a given discrete mesh node involves interaction with the same variable at nearby nodes, fluid-DFTs may have remarkably little of this kind of coupling.[1]
2. Stencil support as a function of physical parameter: Unlike PDEs, whose stencils are typically independent of mesh node spacing (e.g., 9 point FEM stencil in two dimensions), the stencils in fluid-DFTs are determined by some characteristic physical parameter in the system (e.g., bead size, bond length). As the mesh is refined, more nodes fall within the range of the integral, and higher fidelity simulations result in larger problem dimensions and many more nonzeros per matrix row (see Figure 3.1).
3. Large number of DOF per node: Most PDE problems have just a handful of DOF in the global system (with the exception of reacting flow problems). First-order formulations of fluid-DFTs, on the other hand, typically have more than 10 DOF per node and may even have 50 or more. Furthermore, the stencils for each DOF can vary greatly in range and complexity.

Since standard algebraic preconditioners, including multilevel methods, incomplete factorizations, and relaxation methods, all have a bias toward internodal coupling, these methods may all miss the mark. Similarly, load balancing tools largely have the same bias and will need to be revisited for fluid-DFTs. The scaling of stencils with mesh density in fluid-DFTs suggests that standard preconditioners, sparse matrix computations, and communication patterns become increasingly inappropriate as mesh fidelity increases. Thus it is critical that a general framework be developed that is more suitable to fluid-DFTs. Because there is a wide variety of fluid-DFTs in current use in the physics community (e.g., various methods for similar fluids, and a wide range of fluids from atomistic neutral particles to polarizable polymers), the framework must be quite flexible and general. We present our current efforts to develop such a framework in section 4.

4. A new block matrix formulation and solution method. The basic framework for all our solver algorithms reflects the importance of interphysics coupling in the first-order formulation of the fluid-DFTs described in section 2. This physics coupling led us to a physics-based block matrix formulation in order to partition critical and nonlocal ancillary variables. The idea is to partition the data into

[1]Exceptions to this property occur when differential operators are part of the system of Euler–Lagrange equations, as with charged fluids [31, 10] or diffusing fluids [11].

blocks that can be optimally managed or solved. The general 2×2 block matrix is

$$\left(\begin{array}{ccc|ccc} A_{11}^{11} & \cdots & A_{11}^{1j} & A_{12}^{1,j+1} & \cdots & A_{12}^{1k} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ A_{11}^{j1} & \cdots & A_{11}^{jj} & A_{11}^{j1} & \cdots & A_{11}^{jj} \\ \hline A_{21}^{j+1,1} & \cdots & A_{21}^{j+1,j} & A_{22}^{j+1,j+1} & \cdots & A_{22}^{j+1,k} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ A_{21}^{k1} & \cdots & A_{21}^{kj} & A_{22}^{k,j+1} & \cdots & A_{22}^{kk} \end{array}\right) \left(\begin{array}{c} x_1^1 \\ \vdots \\ x_1^j \\ \hline x_2^{j+1} \\ \vdots \\ x_2^k \end{array}\right) = \left(\begin{array}{c} b_1^1 \\ \vdots \\ b_1^j \\ \hline b_2^{j+1} \\ \vdots \\ b_2^k \end{array}\right), \tag{4.1}$$

where k is the number of DOF tracked per node and j is the number of DOF associated with the first block equation. The superscript notation $A^{p,q}$ denotes the block of coefficients generated by DOF p interactions with DOF q. The subscripts and partition lines impose a coarser partitioning of the matrix into a 2×2 block system that will be used with a Schur complement approach. We denote by A_{11}, A_{12}, A_{21}, and A_{22} the upper left, upper right, lower left, and lower right submatrix of the coarse 2×2 block matrix, respectively. Similarly, x_1 and x_2 and b_1 and b_2 are the upper and lower parts of x and b, respectively.

Given this two-level structure, the basic strategy for solving each global linear system generated by Newton's method is as follows:

1. Identify and reorder DOF 1 through j such that A_{11}^{-1} (the inverse of A_{11}) is easy to apply (in parallel). The details of this step are given in section 5 for two particular fluid-DFTs.
2. Determine a preconditioner P for $S = A_{22} - A_{21}A_{11}^{-1}A_{12}$, the Schur complement of A with respect to A_{22}. (See Saad [27] for an overview of Schur complement methods.)
3. Solve $Sx_2 = (b_2 - A_{21}A_{11}^{-1}b_1)$ using a preconditioned Krylov method such as generalized minimal residual (GMRES), with preconditioner P. Note that S may or may not be explicitly formed, depending on other problem details.
4. Finally, solve for $x_1 = A_{11}^{-1}(b_1 - A_{12}x_2)$.

The attractive properties of this algorithm from a parallel computing perspective are as follows:

1. The partitioning used to distribute grid nodes can also be effective for partitioning each DOF of the linear system. Because the DOF are segregated, providing a fairly uniform nonzero pattern within each DOF block, this approach typically gives well-balanced data and work distribution for the solver.
2. A_{11}^{-1} can often be applied as a matrix-vector multiplication, or sequence of such. In other cases, A_{11}^{-1} can be explicitly computed. This means that explicitly forming S, or applying it implicitly, is very efficient in parallel.
3. The dimension of S is often a small fraction of the original matrix. Specifically, only the primitive densities, density functionals, and a few scalar DOF are part of the A_{22} block. All other variables, sometimes more than 80% of the total DOF, are part of the A_{11} block. Thus iterative methods such as GMRES will typically converge much faster because of the reduced dimension, independent of other factors such as preconditioning.
4. Assuming nodes of the mesh are equally partitioned on the parallel machine, each matrix block A^{pq} in (4.1) is distributed evenly across the parallel machine. Thus, parallel execution of this solver is well-balanced and produces identical results, independent of the number of processors, up to roundoff error.

5. The new solver applied to two fluid-DFTs. We now describe specifics of this approach for two different fluid-DFTs. The physics description for these two models is included in the appendix. Here we focus on the structure of the matrices, implications for formulating the Schur complement, and suitable preconditioners. We note that while there are other classes of fluid-DFTs, the general strategy of extending the system size to include convenient nonlocal variables and then designing efficient segregated solvers will generalize quite easily to the broader problem space.

5.1. Hard spheres. A hard-sphere fluid is the simplest type of fluid model that is of practical interest (see section A.1). In a first-order numerical formulation, these problems typically have one or more density unknowns (the critical fields) and a set of linear auxiliary variables referred to as nonlocal densities (see (2.9)). In our block matrix formulation, the equations corresponding to the density unknowns (i.e., the Euler–Lagrange equations) are collected in the lower portion of the matrix (densities are the x_2 variables). The nonlocal density equations are collected in the top portion of the matrix (nonlocal densities are the x_1 variables).

We will consider a case when there are $N_{nld} = 4 + 2 * D$ nonlocal densities per node in the problem, where D is the physical dimension (1, 2, or 3) of the problem of interest. If all the nonlocal density variables are independent, the A_{11} block (which is composed of nonlocal density interactions only) is

$$A_{11} = I. \tag{5.1}$$

In this case, the inverse of A_{11} is obviously trivial.

However, for the special case of a single component fluid, some of the nonlocal densities may be trivially dependent on others with

$$n_\gamma(\mathbf{r}) = C n_\epsilon(\mathbf{r}), \tag{5.2}$$

where C is a constant. We generally encode the linear dependency in (5.2). For the case of Rosenfeld's functionals presented in section A.1 [24], this leaves $N_{nld} = 2 + D$ variables that are described by integration stencils (nonzeros in the A_{12} block).

To address these dependencies, we further decompose the A_{11} block into a 2×2 block matrix as follows. The nonlocal density variables that have no dependence on other nonlocal densities are put in the upper left block. The lower right block contains all others, which have a dependence only on variables in the first block. The A_{11} block is then

$$A_{11} = \begin{pmatrix} -I & 0 \\ X & -I \end{pmatrix}, \tag{5.3}$$

where X contains the various proportionality constants, C, in the problem of interest. It is immediately clear from this expression of A_{11} that the inverse is simply

$$A_{11}^{-1} = \begin{pmatrix} -I & 0 \\ -X & -I \end{pmatrix}. \tag{5.4}$$

The simple form of A_{11}^{-1} allows it to be explicitly applied in a matrix-vector multiply for an efficient calculation of S. Very simple preconditioners for S (e.g., Jacobi scaling based on the diagonal of A_{22}) work well to solve hard-sphere systems. Using this approach, we have seen scalable results for 1-, 2-, and 3-dimensional problems. Section 6 provides the details.

5.2. Polymer problems. Polymer and molecular fluid-DFTs that include a deterministic treatment of chain conformations come in several varieties [2, 32, 19, 1].[2] However, they all can be characterized again as a combined system of equations with critical field variables and nonlocal ancillary variables, with the chain structure equations playing the role of the nonlocal ancillary variables. To be more specific we consider the Chandler–McCoy–Singer DFT [2] (CMS-DFT) as it is described in section A.2. In this case, the critical variables $\rho(\mathbf{r})$ and $U(\mathbf{r})$ are the densities and an unknown mean field, respectively. The polymer conformation information is described by Green's function propagator equations, $G(\mathbf{r})$ and $G^{inv}(\mathbf{r})$. The structure of these equations is quite different from the nonlocal densities of the hard-sphere problem, as these equations are significantly coupled to one another through a recursive relationship (see (A.7) and (A.8) in section A.2).

In our new method, the A_{11} block is composed of these propagator equations, and when properly ordered, A_{11} takes the block form

$$A_{11} = \begin{pmatrix} A_{11}^{11} & 0 & 0 & 0 \\ A_{11}^{21} & A_{11}^{22} & 0 & 0 \\ \vdots & \ddots & \ddots & \vdots \\ A_{11}^{j1} & \cdots & A_{11}^{j-1,j} & A_{11}^{jj} \end{pmatrix}, \tag{5.5}$$

where j is two times the length of the polymer chain in the model and each A_{11}^{ii} is diagonal. Because each A_{11}^{pq} is distributed across the parallel machine proportionally to the nodes, and because each A_{11}^{ii} is diagonal, applying A_{11}^{-1} is a sequence of j diagonal scalings and matrix multiplications. Although A_{11}^{-1} cannot be explicitly formed for polymers, we still retain sufficient parallelism to get excellent performance on distributed memory computers, and application of A_{11}^{-1} is invariant under changes in processor count up to roundoff error.

The preconditioner P for polymer problems is more challenging than for hard spheres. The A_{22} block for polymers has the following form:

$$A_{22} = \begin{pmatrix} D_{11} & F \\ D_{21} & D_{22} \end{pmatrix}. \tag{5.6}$$

The first block of equations in A_{22} is associated with the unknown fields $U(\mathbf{r})$ (see (A.6)), and the second block with the primitive densities $\rho(\mathbf{r})$ (see (A.5)). Each of the D_{pq} blocks is diagonal, whereas the F matrix describes the dependence of the unknown field, $U(\mathbf{r})$, on the primitive densities. The density of this block is dictated by the range of the direct correlation function in (A.6). The F block is by far the most dense submatrix in the global matrix. As mentioned in section 3, the density of F increases dramatically with mesh refinement. Also, for the purposes of communicating off-processor values in a distributed memory implementation of the solver, F will have a large overlap and will require the most attention to ensure good parallel communication complexity. It is worth noting that F is the ,,, submatrix block that has these unusual stencil characteristics. It is also worth noting that the values in F do not change between nonlinear iterations.

One final observation is needed to motivate our preconditioner: All values in A_{22} are $O(1)$ except the values in D_{21}, which are $O(10^{-10})$. In other words, the primitive

[2] Note that another approach takes chain conformations from a large number of samples generated with molecular simulations [37, 22].

densities have a very weak direct dependence on the unknown fields, $U(\mathbf{r})$. We take advantage of this by defining an approximation to A_{22} as

$$A_{22} \approx \tilde{A_{22}} = \begin{pmatrix} D_{11} & F \\ 0 & D_{22} \end{pmatrix}. \tag{5.7}$$

We then define a preconditioner P for S that consists of one block Gauss–Seidel (one back-solve) using $\tilde{A_{22}}$. Given that D_{22} and D_{11} are diagonal and well-distributed and that F is also distributed, applying P involves two diagonal scalings, using D_{22} and D_{11}, and a matrix-vector multiplication using F. All these steps are very efficient in parallel. Thus we once again have an effective parallel solver whose results are invariant under changes to processor count, up to roundoff error. This algorithm for CMS-DFT for polymers has attractive features which are similar to those of the hard-sphere solver, except that A_{11}^{-1} cannot easily be explicitly formed.

One additional benefit is that the solver can efficiently handle very long polymer chains, since the chain length increased only the dimension of A_{11}, which has a modest impact on the performance and robustness of the solver. We note that previous approaches to solving polymer problems typically required an incomplete factorization with a fairly high level of fill to solve the linear system of equations. Our new solver requires almost no additional memory for the preconditioner. This reduces the memory requirement for the solver by at least a factor of 2, often a factor of 4 or more.

6. Computational results. In this section, we show results for several cases when the numerical problem is either a 2-dimensional numerical problem (with a third uniform dimension) or a full 3-dimensional problem. We compare a solution via (i) a generic global solver that attempts to solve all equations simultaneously in a single matrix via standard preconditioned Krylov methods, and (ii) the new solvers described in sections 4 and 5. Each solver uses an unscaled residual tolerance of 10^{-4}. Overall the improvement due to the new algorithmic framework is dramatic (up to 10 times better), as well as making some large problems tractable. All results were generated on the Sandia system red squall, a 258-node, dual processor Opteron-based system (2.2 GHz processors with 4 GB memory per node) using a Quadrics Elan4 high-speed interconnect.

6.1. Solver approaches. All the results presented here are from the fluid-DFT application code Tramonto [7]. Tramonto is a parallel, distributed memory application that solves fluid-DFT problems using real space techniques. Early versions of Tramonto used general-purpose preconditioned Krylov solvers from the solver package Aztec [34]. Recently Tramonto was redesigned to utilize the Trilinos solver framework, a large collection of solver packages and parallel linear algebra tools [14, 17].

In this section we use the term to refer to a general-purpose preconditioned Krylov solver applied to the full linear system of equations ordered as described in (4.1). We use an overlapping Schwarz preconditioner with ILUT [26] as the local subdomain solver via the Trilinos package IFPACK [29]. We use nonrestarted GMRES [28] from the Trilinos package AztecOO [16] as the iterative method. Note that the solver is already an improvement over previous work [7] because Trilinos algorithms (an improvement over Aztec solvers) are used, and because the ordering of equations has changed from a node-first ordering to a DOF-first ordering. The DOF-first ordering results in very favorable ILUT fill and robustness properties for the general-purpose solver.

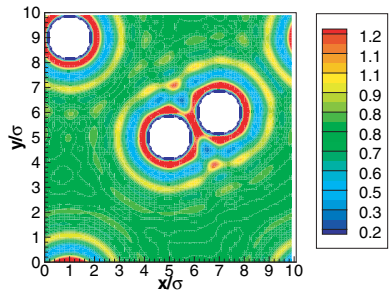


FIG. 6.1. *Fluid density ($\rho\sigma^3$) distribution in the vicinity of* 3 *nanocylinders.*

The segregated Schur solvers presented here will be denoted , , and are based on preconditioned Krylov methods that use our new segregated preconditioners and apply GMRES to the Schur complement system. All preconditioners for the new solvers are constructed and applied using the matrix and vector classes in the Trilinos package Epetra [15]. Due to the number of DOF and the sequencing requirements to apply A_{11}^{-1}, the , , solver preconditioners are composed of several (sometimes more than 100) Epetra distributed sparse matrices. The , , solvers also use GMRES from AztecOO.

6.2. Hard-sphere fluids in two dimensions. This problem considers an inhomogeneous fluid, where there are three cylindrical rod surfaces in the domain. This problem is similar to systems extensively studied previously in a study on adsorption in disordered porous media [9]. Since the solution is uniform perpendicular to the rods, the numerical problem may be solved in two dimensions (with the third dimension treated analytically). Figure 6.1 shows the density profile we compute for this case. The computational domain is $10\sigma \times 10\sigma$ in size, where σ is the diameter of the fluid particles. Periodic boundary conditions were applied in this calculation. The bulk fluid density was $\rho\sigma^3 = 0.75$. The three rods were all quite small with diameters of $2R = 0.5\sigma$ and were located at $(x/\sigma, y/\sigma) = (5,5), (1,9), (7,6)$. Volume exclusion interactions define the rod-fluid interactions, and the corresponding discontinuity in the external field is found at a distance of $R + 0.5\sigma$ from the center of the cylinders.

Table 6.1 compares the , , and , approaches. , solutions were obtained using an ILUT preconditioner with two levels of fill-in. Solves were first attempted with no preconditioning and with an ILU preconditioner. Neither were able to solve the linear problem in less than 100 iterations. As is apparent from the table, the chosen preconditioner is identical to the , , algorithm in nonlinear updates and is similar in the linear solver iteration count. The table shows both parallel scaling and scaling with mesh refinement at a fixed number of processors.

These results clearly demonstrate that the , , method is quite powerful, particularly for smaller numbers of processors. We achieve increased performance with the new algorithms, and the decrease in solve time is on the order of one to two orders of magnitude. Using the solve time data in the second part of the table (excluding data at $\Delta x = 0.2\sigma$) we find that the scaling of the , code with mesh refinement is $T \propto N^{2.7}$, while the new code has $T \propto N^{2.0}$.

6.3. 3-dimensional hard spheres. This problem considers a hard-sphere fluid in a nanotube of finite length, where the nanotube is located in a planar surface (or membrane). This particular choice is the base case for studies on ion channel pro-

TABLE 6.1

Results for a 2-dimensional hard-sphere test problem. Note that every run in the table solved with 10 nonlinear iterations. The top columns show the number of processors used for the calculation, the average number of linear iterations per nonlinear iteration, the solve time per nonlinear iteration, the speedup relative to the single processor time, and the ratio of solve times from the GenS to the SSS algorithm. The bottom columns show scaling with mesh density. The first bottom column contains the mesh spacing, and the sixth column contains timings relative to the $\Delta x = \sigma/5$ result. All data in the bottom columns were generated on 64 processors. All data in the top columns were generated with a mesh spacing of $\Delta x = \sigma/10$.

#Procs	<# Lin iters>		Time/N_{iter}		T_{1Proc}/T	T_{GenS}/T_{SSS}
	GenS	*SSS*	GenS	*SSS*	(*SSS*)	
1	8	22	495.1	14.7	1	33.8
2	10	22	148.7	8.5	1.7	17.5
4	11	22	47.4	4.2	3.5	11.3
8	12	22	16.1	2.1	6.9	7.6
16	13	22	6.3	1.1	13.3	5.7
32	15	22	2.1	0.6	24.8	3.6
64	18	22	1.1	0.3	43.1	3.2
Δx	<# Lin iters>		Time/N_{iter}		$T/T_{\Delta x=0.2}$	T_{GenS}/T_{SSS}
	GenS	*SSS*	GenS	*SSS*	(*SSS*)	
$\sigma/5$	12	17	0.1	0.07	1	1.4
$\sigma/10$	18	22	1.1	0.3	5	3.4
$\sigma/20$	18	24	32.1	4.7	69.4	6.8
$\sigma/40$	-	24	-	82.0	1206	-

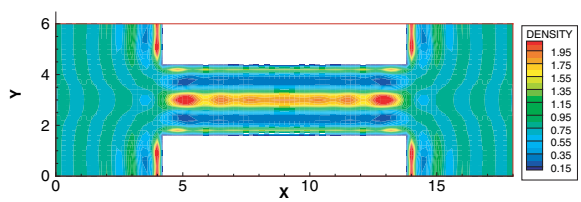


FIG. 6.2. *Fluid density ($\rho\sigma^3$) distribution for a hard-sphere fluid in a finite length nanopore.*

teins [10]. Specifically, the domain is of size $18\sigma \times 6\sigma \times 6\sigma$ with the long axis of the nanotube in the x dimension. The diameter of the nanotube is 2.5σ. The length of the nanotube is 10σ. The computational domain has bulk boundary conditions in the x dimension and continuation boundary conditions in the y and z dimensions. In the latter case, when performing integrals we assume that the density profile at the edge of the computational domain persists and is constant beyond the boundary. The bulk fluid density is again set to $\rho_b\sigma^3 = 0.75$ for studies on parallel performance, but is reduced to $\rho_b\sigma^3 = 0.6$ for the mesh refinement studies, as the system of equations becomes difficult to converge at the higher density for the most refined mesh. Figure 6.2 shows the density profile for one slice (at $z = 3\sigma$) in the computational domain.

Performance results are presented in Table 6.2. Overall the results are very similar to the 2-dimensional problem of the previous section. The , , , algorithm improves on the performance of the , , algorithm (ILUT preconditioner with two levels of fill) by up to 18 times for measurable cases. The problems are also amenable to much smaller parallel platforms and a more refined mesh with the new algorithms. These improvements can be attributed to the reduced memory requirements of these algorithms. The scaling with mesh density for this problem using the , , , algorithm is $T \propto N^{1.65}$. For the , , algorithm we find $T \propto N^{2.2}$.

TABLE 6.2

*Results for a 3-dimensional hard-sphere test problem. The second top column now contains the number of nonlinear iterations needed to solve the problem. For a description of all other columns see the Table 6.1 caption. All data in the upper part of the table were generated with a mesh spacing of $\Delta x = \sigma/5$ and with a bulk density of $\rho\sigma^3 = 0.75$. All data in the lower part of the table were generated on 128 processors with a bulk density of $\rho\sigma^3 = 0.6$. The data marked with * indicates that complete convergence was not obtained and timings are extrapolated based on the number of completed nonlinear iterations listed in parentheses.*

#Procs	N_{iter}	<# Lin iters>		Time/N_{iter}		T_{4Proc}/T	T_{GenS}/T_{SSS}
		GenS	*SSS*	*GenS*	*SSS*	(*SSS*)	
4	11	-	-	-	117.4	1	-
8	11	-	76	-	61.0	1.9	-
16	11	53*	76	544*(6)	29.8	3.9	18.*
32	11	73	76	154.5	16.7	7.0	9.3
64	11	80	76	55.4	8.3	14.1	6.7
128	11	89	76	19.9	4.7	24.8	4.2
Δx	N_{iter}	<# Lin iters>		Time/N_{iter}		$T/T_{\Delta x=0.2}$	T_{GenS}/T_{SSS}
		GenS	*SSS*	*GenS*	*SSS*	(*SSS*)	
$\sigma/5$	7	45	44	17.2	4.1	1	4.1
$\sigma/7$	8	51	49	150.5	18.9	4.6	8.0
$\sigma/10$	9	-	51	-	121.1	29.3	-

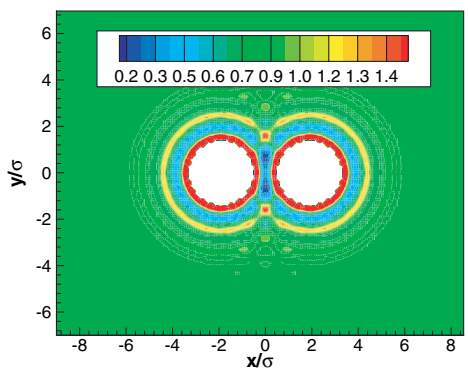


FIG. 6.3. *Density ($\rho\sigma^3$) distribution for a 10-mer homopolymer near two nanocylinders. The computational domain was 1/4 of this image with reflective boundaries at $x = 0$ and $y = 0$.*

6.4. 2-dimensional polymer. Next we consider a 2-dimensional system based on the CMS polymer DFT presented in section A.2. We consider a homopolymer in which there are 10 identical beads on the chain and the size of each segment is σ. There is a pair of infinite cylinders in the problem (diameter 2σ), and we solve for the polymer distribution around these nanocylinders. Both polymer bead interactions and polymer-surface interactions are purely repulsive, as defined by volume exclusions. A series of these calculations at different separations could be performed to compute the force between the cylinders as a function of distance. Studies were performed using a computational domain of size $7\sigma \times 9\sigma$. The initial guess was a uniform solution at the bulk site type density of $\rho\sigma^3 = 0.85$. Figure 6.3 shows the solution we obtain for this problem.

Performance results for this 2-dimensional polymer test problem are shown in Table 6.3. We find very similar behavior to the atomic fluids cases presented in the previous examples with up to a 25-fold improvement in performance. Again the

TABLE 6.3

*Results for a 2-dimensional polymer problem. The top columns show scaling with the number of processors. The middle columns show scaling with mesh density. The bottom columns show scaling with polymer chain length. The column descriptions can be found in the caption to Table 6.1. All data in the top columns were generated with a mesh spacing of $\Delta x = \sigma/10$ and for polymer chains of length $N_{seg} = 10$. All data in the middle columns were generated on 128 processors for polymer chains of length $N_{seg} = 10$. All data in the bottom columns were generated on 32 processors with $\Delta x = \sigma/10$. The data marked with * indicates that complete convergence was not obtained and timings are extrapolated based on the number of completed nonlinear iterations listed in parentheses.*

#Procs	N_{iter}	<# Lin iters>		Time/N_{iter}		T_{1Proc}/T	T_{GenS}/T_{SSS}
		GenS	*SSS*	*GenS*	*SSS*	(*SSS*)	
1	9		70	-	88	1	-
2	9		70	-	45	2.0	-
4	9	25	70	290*(8)	26.2	3.4	11.0
8	9	28	70	77	12	7.3	6.4
16	10	33	70	23	6.7	13.1	3.4
32	8	40	70	7.1	3.8	23.2	1.9
64	9	46	69	2.8	2.2	40.0	1.3
128	9	58	69	1.4	1.8	49.0	0.8
Δx	N_{iter}	<# Lin iters>		Time/N_{iter}		$T/T_{\Delta x=\sigma/10}$	T_{GenS}/T_{SSS}
		GenS	*SSS*	*GenS*	*SSS*	(*SSS*)	
$\sigma/10$	9	58	69	1.4	1.8	1	0.8
$\sigma/20$	10	55	69	28.9	13.1	7.3	2.2
$\sigma/30$	9	57	70	1580* (7)	62.3	34.6	25
N_{seg}	N_{iter}	<# Lin iters>		Time/N_{iter}		$T/T_{N_{seg}=10}$	T_{GenS}/T_{SSS}
		GenS	*SSS*	*GenS*	*SSS*	(*SSS*)	
10	8	40	70	7.1	3.7	1	1.9
20	8	48	69	21.7	8.9	2.4	2.4
40	8	62	70	58.7	15.1	4.1	3.9
80	7	93	68	257.4	30.9	8.4	8.3

method was solved using an ILUT preconditioner with two levels of fill. While both methods become more expensive as the polymer chain length increases, we find that the cost of the , , , algorithms increases approximately linearly with chain length, while a steeper performance penalty occurs for the , , method.

6.5. 3-dimensional polymer. Finally, we consider a 3-dimensional system where a self-assembled lipid bilayer is sandwiched between two planar arrays of spheres of diameter 9σ separated by 10σ in a square lattice. The particular chemical model we consider has two components. The first is a model lipid molecule with head group and tail group beads. There are two head group beads on a chain, and 16 tail group beads on a linear chain. The head groups are in the middle, so we have an 8-2-8 morphology on the chain. The head group beads are larger than the tail group beads with $\sigma_h/\sigma_t = 1.44$. The second component in the system is a single site solvent of the same size as the tail beads. The interactions between various species are chosen to favor self-assembly of a lipid bilayer, where the head groups form an interface between the tail beads and the solvent beads. Lipid bilayers formed from this coarse grained model have been shown to be in reasonable agreement with molecular dynamics simulation of the same models and to map reasonably well to fluid experimental bilayers [6].

Figure 6.4 shows one slice through the solution we obtain for this problem. Computing interactions of nanoscale surfaces and colloidal particles with lipid bilayers are needed to provide a molecular theory–based analysis of surface forces experiments (e.g., surface forces apparatus, atomic force microscope, optical tweezers, etc.) for

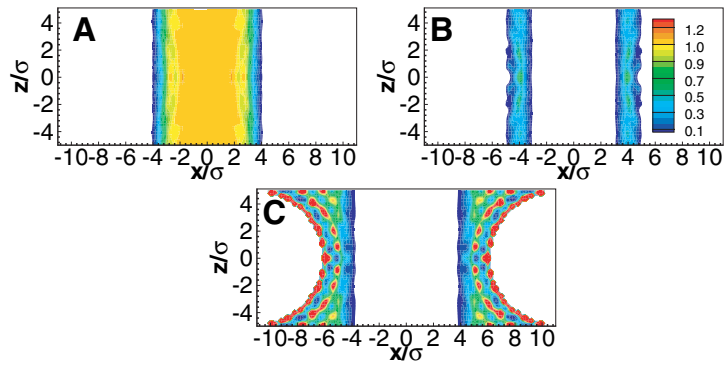


FIG. 6.4. *One slice (at $y = 0$) through a 3-dimensional computational volume. The shaded contours show density ($\rho\sigma^3$) distributions for lipid tail beads* (A)*, lipid head beads* (B)*, and solvent* (C) *for a lipid bilayer assembly sandwiched between planar arrays of large spheres. The computational domain is* 1/4 *of the domain shown in the figure and utilized reflective boundary conditions on all edges. The legend in* B *shows the contour scale for all three figures. Densities less than $\rho\sigma^3 = 0.01$ are shown in white for clarity.*

TABLE 6.4

Results for a 3-dimensional polymer test problem. Each case required 10 *nonlinear iterations for a solution. For a description of all other columns see the caption to Table* 6.1*. All data were generated with a mesh spacing of $\Delta x = \sigma/5$.*

#Procs	<# Lin iters>	Time/N_{iter}	T_{32Proc}/T
32	93	1510	1
64	93	414	3.6
128	93	190	7.9

these systems. This type of calculation is also needed for studying the interactions of lipid bilayers with proteins.

The computational domain for this problem was $11\sigma \times 5\sigma \times 5\sigma$, and the initial guess for the density distribution is a previously converged uniform bilayer result. Note that this case had 37856 nodes in the computational domain and 44 unknowns per node for a total of 1.66×10^6 unknowns in the problem. Algorithm performance is presented as a function of number of processors only in Table 6.4. In this case, the , , , algorithms provide the ability to solve this difficult problem, as we were not able to solve this problem with the , , approach despite using 128 processors and trying a variety of parameters for solver tolerances and ILUT fill factors. Finally, note that with the , , , approach we find a significant superlinear speedup for this case from 32 to 64 processors.

6.6. Results summary. The results presented in this section are quite promising for the new solvers. In particular there are several observations worth noting.

6.6.1. Memory use. The new solvers use almost no extra memory for the preconditioners. Furthermore, the dimension of the implicit problem that GMRES must solve is 10 to 100 times smaller. Since all solvers are using nonrestarted GMRES and performing 100 or more iterations, the GMRES storage cost is $O(100n)$, where n is the dimension of the GMRES problem. Letting k denote the nonzero count of the global matrix, we see that the memory use for GMRES vectors in the , , solver is comparable to the matrix: from $3k$ to $10k$ storage. The ILUT preconditioner for the

solver requires from $2k$ to $10k$ storage. Thus the solver requires minimally $6k$ storage, up to $20k$ or more. In contrast, the cost of GMRES storage for the new solvers is from $0.1k$ to $0.01k$, and the overall storage is less than $2k$ with the storage cost of the matrix being dominant. The overall reduction in memory use for the solvers over the solver is therefore minimally a factor of 5 and sometimes more than a factor of 20.

6.6.2. Tuning parameters. One difficulty common to preconditioned iterative methods is the presence of tuning parameters. This is true for the solver in that ILUT requires ad hoc parameters to determine fill, and the user must prescribe these parameter values. Furthermore, as is well known, overlapping Schwarz methods tend to lose robustness as the processor count increases, making the choice of ILUT parameters and the maximum number of GMRES iterations a function of processor count. In contrast, the solvers have tuning parameters and have identical convergence behavior independent of processor count. This behavior may be the most important feature to an application user.

6.6.3. Solver scalability in processor count. The processor scalability results for the solver are quite remarkable. In all cases, once the number of processors is large enough for the solver to work at all, the performance improvement as a function of processor count is superlinear. This behavior is often observed in practice and is a function of the following two factors: (i) Increasing processor count increases the amount of cache memory and, for a fixed size problem, reduces the working data set on a processor, and (ii) as processor count increases, the subdomains for overlapping Schwarz decrease in size, resulting in smaller ILUT factors and less work per iteration, even though the number of iterations increases. Some of the solvers scalability results are also superlinear, due to factor (i) above. However, since the solvers are already very efficient in serial cost and apply GMRES to a much smaller problem, there is much less work to distribute as processor counts increase. In effect, the solvers cannot benefit as much as the solver from large processor counts for the same global problem. However, the solvers scale quite well and, because of their low memory usage, can be effective using far fewer processors and can solver much larger problems than the solver.

6.6.4. Scalability in mesh density and chain length. The solvers are not invariant under mesh density changes, but nearly so. In fact, except for the coarsest mesh, the number of solver iterations remains nearly constant as the mesh density increases. The solver iteration count also remains fixed as mesh density increases. However, the overall cost of the solver grows much faster than the solvers.

For polymer problems, the solver has constant iteration count as the chain size increases. Furthermore, the cost of the solver grows approximately linearly with the length of the chain. Again, this is a marked improvement over the solver.

7. Conclusions. In this article we have presented a general mathematical framework for describing fluid-DFT problems and solving them using a new family of segregated Schur complement solvers. By viewing fluid-DFTs from a segregated variable perspective we obtain a rich structure that can be exploited in the development of low-cost, robust, scalable solution methods. We have shown that this approach is very effective for two major classes of fluid-DFTs and is very promising for other classes as well. The improvement in solution time, robustness, and memory use opens the door to easy solution of previously intractable problems and broadens the scope of

applicability for real space fluid-DFT methods.

Although our new solvers are faster and require much less memory than the old solver, perhaps most important is the fact that no tuning parameters are required. This fact makes the routine use of Tramonto much easier for complex applications.

Appendix. Two specific fluid-DFTs. In this section we present the particulars of the two distinctly different fluid-DFTs that we consider explicitly in this paper. The first is used to treat atomic fluids, and in particular hard-sphere systems. The second is a DFT used to treat polymer fluids. We note that many other kinds of DFTs have been developed as well. In fact this field contains a whole collection of disparate approaches that must be analyzed individually from the perspective of optimizing solution algorithms.

A.1. An atomistic fluid model. We first consider the fundamental measures theory DFT (FMT-DFT) that was first developed by Rosenfeld, and that has been modified by others [24, 25]. This theory specifically treats hard-sphere fluids as a reference system with other physical effects (e.g., attractions, Coulomb interactions, and even bond constraints) being treated as a perturbations. The grand free energy functional for a multicomponent hard-sphere fluid is

$$\begin{aligned}\Omega[\{\rho_i(\mathbf{r})\}] = &\sum_i \int \rho_i(\mathbf{r})[ln\rho_i(\mathbf{r}) - 1]d\mathbf{r} \\ &+ \int \Phi(\{n_\gamma[\{\rho_i\}]\})d\mathbf{r} + \sum_i \int \rho_i(\mathbf{r})[V_i(\mathbf{r}) - \mu_i]d\mathbf{r},\end{aligned} \tag{A.1}$$

where V is a one body external field, and μ_i is a constant chemical potential in the case of an equilibrium-DFT, and the set of nonlocal variables n_γ are linear as was defined in (2.9). The Euler–Lagrange equation to be solved is

$$ln\rho_i(\mathbf{r}) - V_i(\mathbf{r}) + \mu_i + \int \sum_\gamma \frac{\partial \Phi}{\partial n_\gamma}(\mathbf{r}')\frac{\delta n_\gamma(\mathbf{r}')}{\delta \rho_i(\mathbf{r})}d\mathbf{r}' = 0. \tag{A.2}$$

The energy density Φ depends on a set of nonlocal variables, $\{n_\gamma\}$, where there are four scalar variables, n_0, n_1, n_2, n_3 and two vector variables, n_{V1}, n_{V2}. In Rosenfeld's original theory [24], the free energy density is

$$\begin{aligned}\Phi(\{n\}) = &-n_0 ln(1-n_3) + \frac{n_1 n_2}{1-n_3} - \frac{\vec{n}_{V1}\cdot\vec{n}_{V2}}{1-n_3} \\ &+ \frac{1}{24\pi(1-n_3)^2}\left(n_2 - \frac{\vec{n}_{V2}\cdot\vec{n}_{V2}}{n_2}\right)^3.\end{aligned} \tag{A.3}$$

The block matrix formulation is built on both the Euler–Lagrange equation (residual denoted R_{EL}) and the definition of the nonlocal density variables (residual denoted R_{NL}), and may be written in the form

$$\begin{bmatrix} -I & w_\gamma(\mathbf{r},\mathbf{r}') \\ \sum_\gamma \dfrac{\partial^2\Phi}{\partial n_\gamma \partial n_\epsilon}(\mathbf{r}) & D_{22}(\mathbf{r}) \end{bmatrix} \begin{bmatrix} \Delta n_\gamma \\ \Delta\rho \end{bmatrix} = -\begin{bmatrix} R_{NL}(\mathbf{r}) \\ R_{EL}(\mathbf{r}) \end{bmatrix}, \tag{A.4}$$

where $D_{22}(\mathbf{r})$ is a diagonal block matrix, and each entry is $1/\rho(\mathbf{r})$.

A.2. A polymer fluid model. The particular DFT for polymers we will consider was developed by Chandler, McCoy, and Singer [2], and our formulation is similar to that of Donely and McCoy [3, 12]. This particular theory is developed by minimization of a free energy functional with respect to both the density and an effective field variable, U. This effective field variable constrains a fluid of ideal chains to have the same density profile as the interacting chains of interest in the real external field V^{ext}. The theory solves simultaneously for critical fields $\rho(\mathbf{r})$ and $U(\mathbf{r})$ for each type of monomer segment α in the system. The two residual equations for these critical fields are

$$\rho_\alpha(\mathbf{r}) = \frac{\rho_{b,\alpha}}{N_\alpha} \sum_{s\in\alpha} \frac{G_s(\mathbf{r})G_s^{inv}(\mathbf{r})}{\exp[-\beta U_\alpha(\mathbf{r})]}, \quad \alpha = 1,\dots,N_\alpha, \tag{A.5}$$

$$U_\alpha(\mathbf{r}) = V_\alpha(\mathbf{r}) - \sum_\beta \int c_{\alpha\beta}(\mathbf{r}-\mathbf{r}')(\rho_\beta(\mathbf{r}') - \rho_{b\beta})d\mathbf{r}', \quad \alpha = 1,\dots,N_\alpha, \tag{A.6}$$

where N_α is the number of segments of type α, $c_{\alpha\beta}$ is the direct correlation function taken from a liquid state theory for bulk fluids, the sum in (A.5) is taken over all N_s monomer segments in the fluid of interest, ρ_b denotes the homogeneous bulk density far from the surface, s is a particular segment on the polymer chain of interest, and the G and G^{inv} functions are propagator functions that describe chain connectivity. Specifically,

$$G_s(\mathbf{r}) = \exp[-\beta U_{\alpha(s)}(\mathbf{r})] \int \omega_{\alpha\beta}(\mathbf{r}-\mathbf{r}')G_{s-1}(\mathbf{r}')d\mathbf{r}', \quad s = 1,\dots,N_s, \tag{A.7}$$

$$G_s^{inv}(\mathbf{r}) = \exp[-\beta U_{\alpha(s)}(\mathbf{r})] \int \omega_{\alpha\beta}(\mathbf{r}-\mathbf{r}')G_{s+1}^{inv}(\mathbf{r}')d\mathbf{r}', \quad s = 1,\dots,N_s, \tag{A.8}$$

where ω is a delta function with a range equal to the bond length between segments s and $s-1$ of types α and β, respectively. We note that for an end bead, only the initial field term is present.

To summarize, we have $2N_\alpha$ critical variables ($\{\rho_\alpha\}$ and $\{U_\alpha\}$, and $2N_s$ nonlocal ancillary variables ($\{G\}$ and $\{G^{inv}\}$). The concrete example we discussed in this paper is for an 18 bead polymer mixed with a single site solvent, where the polymer chain has two types of beads on the chain. For this 3 component, 19 segment problem, we have 6 critical field variables and 38 nonlocal ancillary variables.

This situation is quite extreme with respect to complexity in the integral equations. If the discrete matrix problem were formed only in terms of the critical variables, the nested nonlocal variables would result in multidimensional integrals for each Jacobian entry. Specifically, for a polymer with N_s segments, an $N_s - 1$ dimensional integral would need to be performed. This is clearly out of the question. Thus we form the matrix problem with each of the nonlocal propagator functions treated as independent variables. The structure of the resulting 2×2 block matrix is described more fully in section 4.

Acknowledgments. The authors thank the MICS program under the U.S. Department of Energy for support, and thank the referees for comments that led to an improved paper.

REFERENCES

[1] D. Cao and J. Wu, *Density functional theory for semiflexible and cyclic polyatomic fluids*, J. Chem. Phys., 121 (2004), pp. 4210–4220.

[2] D. Chandler, J. D. McCoy, and S. J. Singer, *Density functional theory of nonuniform polyatomic systems.* 1. *General formulation*, J. Chem. Phys., 85 (1986), pp. 5971–5979.

[3] J. P. Donley, J. G. Curro, and J. D. McCoy, *A density-functional theory for pair correlation-functions in molecular liquids*, J. Chem. Phys., 101 (1994), pp. 3205–3215.

[4] S. A. Egorov, *Effect of repulsive and attractive interactions on depletion forces in colloidal suspensions: A density functional theory treatment*, Phys. Rev. E, 70 (2004), 031402.

[5] F. Fang and I. Szleifer, *Kinetics and thermodynamics of protein adsorption: A generalized molecular theoretical approach*, Biophys. J., 80 (2001), pp. 2568–2589.

[6] L. J. D. Frink and A. L. Frischknecht, *Density functional theory approach for coarse-grained lipid bilayers*, Phys. Rev. E, 72 (2005), 041923.

[7] L. J. D. Frink and A. G. Salinger, *Two- and three-dimensional nonlocal density functional theory for inhomogeneous fluids,* I. *Algorithms and parallelization*, J. Chem. Phys., 159 (2000), pp. 407–424.

[8] L. J. D. Frink and A. G. Salinger, *Two- and three-dimensional nonlocal density functional theory for inhomogeneous fluids,* II. *Solvated polymers as a benchmark problem*, J. Chem. Phys., 159 (2000), pp. 425–439.

[9] L. J. D. Frink and A. G. Salinger, *Rapid analysis of phase equilibria with density functional theory* II. *Capillary condensation in disordered porous media*, J. Chem. Phys., (2003), pp. 7466–7476.

[10] L. J. D. Frink, A. G. Salinger, M. P. Sears, J. D. Weinhold, and A. L. Frischknecht, *Numerical challenges in the application of density functional theory to biology and nanotechnology*, J. Phys. Cond. Matter, 14 (2002), pp. 12167–12187.

[11] L. J. D. Frink, A. Thompson, and A. G. Salinger, *Applying molecular theory to steady-state diffusing systems*, J. Chem. Phys., 112 (2000), pp. 7564–7571.

[12] A. L. Frischknecht, J. G. Curro, and L. J. D. Frink, *Density functional theory for inhomogeneous polymer systems,* II. *Application to block copolymer thin films*, J. Chem. Phys., 117 (2002), pp. 10398–10411.

[13] D. Henderson, ed., *Fundamentals of Inhomogeneous Fluids,* Marcel Dekker, New York, 1992.

[14] M. A. Heroux, *Trilinos Home Page*, http://trilinos.sandia.gov/ (2003).

[15] M. A. Heroux, *Epetra Home Page*, http://trilinos.sandia.gov/packages/epetra (2004).

[16] M. A. Heroux, *AztecOO Users Guide*, Technical report SAND2004-3796, Sandia National Laboratories, Albuquerque, NM, 2004.

[17] M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring, A. Williams, and K. S. Stanley, *An overview of the Trilinos project*, ACM Trans. Math. Softw., 31 (2005), pp. 397–423.

[18] E. Kierlik and M. L. Rosinberg, *A perturbation density-functional theory for polyatomic fluids.* 3. *Application to hard chain molecules in slitlike pores*, J. Chem. Phys., 100 (1994), pp. 1716–1730.

[19] Z. D. Li, D. P. Cao, and J. Z. Wu, *Density functional theory and Monte Carlo simulations for the surface structure and correlation functions of freely jointed Lennard–Jones fluids*, J. Chem. Phys., 122 (2005), 174708.

[20] H. Löwen, *Density functional theory of inhomogeneous classical fluids: Recent developments and new perspectives*, J. Phys. Cond. Matter, 14 (2002), pp. 11897–11905.

[21] A. P. Malanoski and F. van Swol, *Lattice density functional theory investigation of pore shape effects. Adsorption in collections of noninterconnected pores*, Phys. Rev. E, 66 (2002), 41603.

[22] S. K. Nath, A. L. Frischknecht, and J. G. Curro, *Density functional theory and molecular dynamics simulation of poly (dimethylsiloxane) melts near silica surfaces*, Macromolecules, 38 (2005), pp. 8562–8573.

[23] R. G. Parr and W. Yang, *Density-Functional Theory of Atoms and Molecules*, Oxford Science Publications, New York, 1989.

[24] Y. Rosenfeld, *Structure and effective interactions in multi-component hard-sphere liquids: The fundamental-measure density functional approach*, J. Phys. Cond. Matter, 14 (2002), pp. 9141–9152.

[25] R. Roth, R. Evans, A. Lang, and G. Kahl, *Fundamental measure theory for hard-sphere mixtures revisited: The white bear version*, J. Phys. Cond. Matter, 14 (2002), pp. 12063–12078.

[26] Y. Saad, *ILUT: A dual threshold incomplete lu preconditioner*, Numer. Linear Algebra Appl., 1 (1994), pp. 387–402.
[27] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.
[28] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.
[29] M. Sala and M. A. Heroux, *Robust Algebraic Preconditioners Using IFPACK* 3.0, Technical report SAND2005-0662, Sandia National Laboratories, Albuquerque, NM, 2005.
[30] M. P. Sears and L. J. D. Frink, *A new efficient method for density functional theory calculations of inhomogeneous fluids*, J. Comput. Phys., 190 (2003), pp. 184–200.
[31] Z. Tang, L. Mier y Teran, H. T. Davis, L. E. Scriven, and H. S. White, *Non-local free energy density functional theory applied to the electrical double layer*. I. *Symmetrical electrolytes*, Mol. Phys., 10 (1990), pp. 369–392.
[32] S. Tripathi and W. G. Chapman, *Microstructure of inhomogeneous polyatomic mixtures from a density functional formalism for atomic mixtures*, J. Chem. Phys., 122 (2005), 094506.
[33] S. Tripathi and W. G. Chapman, *Microstructure and thermodynamics of inhomogeneous polymer blends and solutions*, Phys. Rev. Lett., 94 (2005), 087801.
[34] R. S. Tuminaro, M. A. Heroux, S. A. Hutchinson, and J. N. Shadid, *Official Aztec User's Guide, Version* 2.1, Sandia National Laboratories, Albuquerque, NM, 1999.
[35] H.-J. Woo, L. Sarkisov, and P. A. Monson, *Mean field theory of adsorption in porous glasses*, Langmuir, 17 (2001), pp. 7472–7475.
[36] J. Wu, *Density functional theory for chemical engineering: From capillarity to soft materials*, AIChE J., 52 (2006), pp. 1169–1193.
[37] A. Yethiraj, *Density functional theory of polymers: A Curtin–Ashcroft type weighted density approximation*, J. Chem. Phys., 109 (1998), pp. 3269–3275.
[38] Y. Yu and J. Wu, *Density functional theory of inhomogeneous mixtures of polymeric fluids*, J. Chem. Phys., 117 (2002), p. 2368.

SIAM J. SCI. COMPUT.
Vol. 29, No. 5, pp. 2078–2095
© 2007 Society for Industrial and Applied Mathematics

A MULTIGRID METHOD FOR THE PSEUDOSTRESS FORMULATION OF STOKES PROBLEMS*

ZHIQIANG CAI† AND YANQIU WANG†

Abstract. The purpose of this paper is to develop and analyze a multigrid solver for the finite element discretization of the pseudostress system associated with the differential operator $\mathcal{A} - \gamma\,\mathbf{grad\,div}$ over 2×2 matrix-valued functions. This system is derived from the pseudostress-velocity formulation [Z. Cai, B. Lee, and P. Wang, *SIAM J. Numer. Anal.*, 42 (2004), pp. 843–859] of two-dimensional Stokes problems through the penalty method or natural time discretization for the respective steady- or unsteady-state problems. Here $\gamma > 0$ is a constant associated with either the penalty parameter or the time-step size, and $\mathcal{A}$ is a singular map. In this paper, we develop a multigrid solver for the discrete problem using both the Raviart–Thomas (RT) and the Brezzi–Douglas–Marini (BDM) finite elements. We show that the multigrid convergence rate is $O(\frac{1+\gamma^{-2}}{1+\gamma^{-2}+m})$, where m is the number of smoothings. This convergence rate is independent of the mesh size and the number of levels used in multigrid. Moreover, numerical results presented in this paper show that the multigrid convergence rate for the BDM elements does not depend on the parameter γ.

Key words. multigrid, mixed finite element, pseudostress-velocity formulation, Stokes problem

AMS subject classifications. 65N30, 65N55

DOI. 101137/060661429

1. Introduction.

The stress-velocity-pressure formulation [7, 15] for incompressible Newtonian flows comes from the original physical equations. Recently, this formulation has been gaining consistent attention because of the arising interest in non-Newtonian flows [6, 21]. For some non-Newtonian flows, the stress cannot be eliminated and, hence, a formulation containing the stress as a fundamental unknown is unavoidable. In addition, accurate direct calculation of the stress is important for computing the traction on a fluid-solid interface. Although the stress can be obtained in the velocity-pressure method by differentiating the velocity, this degrades the order of the approximation.

In this paper we are interested in mixed finite element methods for the Stokes problem. Notice that one disadvantage of using the stress-velocity-pressure formulation is that the symmetry constraint of the stress tensor poses extra difficulty in the discretization. For example, stable conforming mixed elements for symmetric matrices, e.g., [3, 5, 20], employ many degrees of freedom and, consequently, lead to large linear systems even for coarse meshes. There are different ways to avoid this difficulty. One way is to use the idea of pseudostress [11] and a pseudostress-velocity formulation. The pseudostress is nonsymmetric and, hence, any stable pairs for the Darcy flows can be adapted to the pseudostress-velocity formulation. This is documented in [13]. Furthermore, the velocity can be eliminated through the penalty method for the steady-state problem or the natural time discretization for the unsteady-state problem. So one need only solve numerically the pseudostress system associated with a differential operator $\mathcal{A} - \gamma\,\mathbf{grad\,div}$ over 2×2 matrix-valued functions. Here, $\gamma > 0$

*Received by the editors May 30, 2006; accepted for publication (in revised form) March 27, 2007; published electronically September 28, 2007. This work was supported in part by the National Science Foundation under grant DMS-0511430.
http://www.siam.org/journals/sisc/29-5/66142.html
†Department of Mathematics, Purdue University, 150 N. University Street, West Lafayette, IN 47907-2067 (zcai@math.purdue.edu, yqwang@math.purdue.edu).

is a constant depending on either the penalty parameter or the time-step size, and $\mathcal{A}$ is a projection operator whose kernel intersects the kernel of the divergence operator. The purpose of this paper is to construct and analyze a multigrid solver for this pseudostress system. It must be pointed out that this problem is more delicate to handle than a normal $\boldsymbol{H}(\mathbf{div})$ problem because of the complicated kernel structure.

We also mention other methods of circumventing the difficulty posed by the symmetric stress. One is to use a stabilization technique [7]. Another is to use the least-squares method, which does not require stable pairs of finite elements. Least-squares methods with fast multigrid solvers for the stress-velocity-pressure formulation and its variants, stress-velocity and pseudostress-velocity formulations, were studied in [11].

Multigrid methods are efficient for many problems. A vast amount of research has been done in this area, e.g., [8, 18, 22, 25]. However, the classical techniques for the multigrid method do not work for the $\boldsymbol{H}(\mathbf{div})$-type problem since standard smoothers cannot damp out the highly oscillating part of divergence-free components. Hence, special smoothers have to be developed in order to achieve an optimal multigrid convergence rate [1, 2, 19, 24]. An additional difficulty in the multigrid analysis for the pseudostress system is the handling of the kernel of $\mathcal{A}$. To overcome the above difficulties, we employ a Helmholtz-like decomposition of the discrete space, which also utilizes the operator $\mathcal{A}$, so that the analysis can be done on separate subspaces. In this paper we consider multigrid solvers for systems discretized using both the Raviart–Thomas (RT) and the Brezzi–Douglas–Marini (BDM) finite elements. We show that the multigrid convergence rate is bounded above by $\rho = \frac{C(1+\gamma^{-2})}{C(1+\gamma^{-2})+2m} < 1$, where m is the number of smoothings and C is a constant independent of the mesh size or the number of levels used in the multigrid. When $\gamma \gg 1$, which is always true for the penalty method of the stationary problems as the penalty parameter goes to zero, the multigrid convergence for the pseudostress system can be considered independent of γ and, hence, the penalty parameter. When $\gamma \ll 1$, which corresponds to the small time-step size in the time discretization of unsteady-state problems, ρ is close to one. Nevertheless, numerical results presented in this paper show that the multigrid convergence rate for the BDM elements does not depend on the parameter γ even if γ is small.

We give the outline of the paper as follows. In sections 2 and 3, we describe the pseudostress system of the Stokes problem and its finite element discretization. The key Helmholtz-type decomposition of the finite element space is given in section 4. We construct a multigrid solver for the discrete $\boldsymbol{H}(\mathbf{div})$-type problem and give a convergence rate estimate in section 5. Some supporting numerical results are presented in section 6. Finally, in the appendix, we prove an essential inequality needed by the analysis in section 4.

1.1. Notation. Let Ω be a convex polygon. Denote by $\mathbb{R}^2$ the set of two-dimensional vector-valued functions and by $\mathbb{M}_2$ the set of 2×2 matrix-valued functions. Throughout the paper, we adopt the convention that a Greek character denotes a 2×2 matrix and a bold roman character in lower case denotes a vector. Letting $\boldsymbol{\tau} = (\tau_{ij})_{2\times 2} \in \mathbb{M}_2$ and $\boldsymbol{v} = (v_1, v_2)^t \in \mathbb{R}^2$, define

$$\operatorname{div} \boldsymbol{v} = \frac{\partial v_1}{\partial x} + \frac{\partial v_2}{\partial y}, \qquad \mathbf{div}\boldsymbol{\tau} = \begin{pmatrix} \frac{\partial \tau_{11}}{\partial x} + \frac{\partial \tau_{12}}{\partial y} \\ \frac{\partial \tau_{21}}{\partial x} + \frac{\partial \tau_{22}}{\partial y} \end{pmatrix},$$

$$\nabla \boldsymbol{v} = \begin{pmatrix} \frac{\partial v_1}{\partial x} & \frac{\partial v_1}{\partial y} \\ \frac{\partial v_2}{\partial x} & \frac{\partial v_2}{\partial y} \end{pmatrix}, \qquad \mathbf{curl}\, \boldsymbol{v} = \begin{pmatrix} -\frac{\partial v_1}{\partial y} & \frac{\partial v_1}{\partial x} \\ -\frac{\partial v_2}{\partial y} & \frac{\partial v_2}{\partial x} \end{pmatrix}.$$

Define the inner product between vectors and 2×2 matrices by

$$\boldsymbol{u} \cdot \boldsymbol{v} = u_1\, v_1 + u_2\, v_2 \quad \text{and} \quad \boldsymbol{\sigma} : \boldsymbol{\tau} = \sum_{1 \le i,j \le 2} \sigma_{ij}\, \tau_{ij},$$

respectively. We use the usual notation $L^2(\Omega)$ to denote the set of square integrable functions on Ω and $H^s(\Omega)$, where $s > 0$ is a real number, to denote the Sobolev space on Ω with index s. Let $C_0^\infty(\Omega)$ be the space of infinitely differentiable functions on Ω with compact support. Denote by $H_0^s(\Omega)$ the closure of $C_0^\infty(\Omega)$ under the norm of $H^s(\Omega)$ and define $H^{-s}(\Omega) = (H_0^s(\Omega))'$, the dual space of $H_0^s(\Omega)$. It is natural to extend the above definitions to vector- and matrix-valued functions using product spaces. For example, $\boldsymbol{H}^s(\Omega, \mathbb{R}^2)$ and $\boldsymbol{H}^s(\Omega, \mathbb{M}_2)$ denote the Sobolev spaces over the set of vector- and 2×2 matrix-valued functions, respectively. Other notations, such as $\mathbf{L}^2(\Omega, \mathbb{R}^2)$, $\mathbf{L}^2(\Omega, \mathbb{M}_2)$, are defined in the same fashion. For simplicity, denote $\|\cdot\|_{s,\Omega}$ to be the Sobolev norm with index s over scalar-, vector-, or matrix-valued functions, depending on the type of the function. Similarly, denote by $|\cdot|_{s,\Omega}$ the Sobolev seminorm and by $(\cdot, \cdot)$ the $\mathbf{L}^2$ inner product. Define the space

$$\boldsymbol{H}(\mathbf{div}, \Omega, \mathbb{M}_2) = \{\boldsymbol{\sigma} \in \mathbf{L}^2(\Omega, \mathbb{M}_2) \,|\, \mathbf{div}\, \boldsymbol{\sigma} \in \mathbf{L}^2(\Omega, \mathbb{R}^2)\}$$

with the norm $\|\boldsymbol{\sigma}\|^2_{\boldsymbol{H}(\mathbf{div},\Omega,\mathbb{M}_2)} = (\boldsymbol{\sigma}, \boldsymbol{\sigma}) + (\mathbf{div}\boldsymbol{\sigma}, \mathbf{div}\boldsymbol{\sigma})$.

2. Pseudostress system. Consider the two-dimensional steady-state Stokes problem

$$\begin{cases} -\Delta \boldsymbol{u} + \nabla p = -\boldsymbol{f} & \text{in } \Omega, \\ \operatorname{div} \boldsymbol{u} = 0 & \text{in } \Omega, \\ \boldsymbol{u}|_{\partial\Omega} = \mathbf{0}, \end{cases} \tag{2.1}$$

where $\boldsymbol{u}$ is the velocity and p is the pressure which satisfies the compatibility condition $\int_\Omega p\, d\boldsymbol{x} = 0$.

Let $\mathcal{A} : \mathbb{M}_2 \to \mathbb{M}_2$ be a fourth order tensor defined by

$$\mathcal{A}\, \boldsymbol{\tau} = \boldsymbol{\tau} - \frac{1}{2} (\operatorname{tr} \boldsymbol{\tau})\, \mathbf{I},$$

where $\operatorname{tr} \boldsymbol{\tau} = \tau_{11} + \tau_{22}$ is the trace of $\boldsymbol{\tau}$ and $\mathbf{I}$ is the 2×2 identity matrix. We immediately notice that $\mathcal{A}$ is a projection onto the trace-free subspace of $\mathbb{M}_2$ and $Ker(\mathcal{A}) = \{v\, \mathbf{I} \,|\, v \text{ is a scalar function}\}$. Define the pseudostress by

$$\boldsymbol{\sigma} = -p\, \mathbf{I} + \nabla \boldsymbol{u}.$$

Then the Stokes problem (2.1) can be rewritten in the pseudostress-velocity formulation:

$$\begin{cases} \mathbf{div}\boldsymbol{\sigma} = \boldsymbol{f} & \text{in } \Omega, \\ \mathcal{A}\boldsymbol{\sigma} - \nabla \boldsymbol{u} = \mathbf{0} & \text{in } \Omega, \\ \boldsymbol{u}|_{\partial\Omega} = \mathbf{0}. \end{cases} \tag{2.2}$$

The incompressible constraint $\operatorname{div} \boldsymbol{u} = 0$ is enforced through $\operatorname{div} \boldsymbol{u} = \operatorname{tr}(\nabla \boldsymbol{u}) = 0$. Notice that $\operatorname{tr} \boldsymbol{\sigma} = -2p$ also needs to satisfy the compatibility condition $\int_\Omega \operatorname{tr} \boldsymbol{\sigma}\, d\boldsymbol{x} = 0$.

From the well-known regularity estimate of problem (2.1) on convex polygons and the definition of the pseudostress, we have the following regularity results. Let $\boldsymbol{f} \in \mathbf{L}^2(\Omega)$; then the solutions to problem (2.1) and (2.2) satisfy $\boldsymbol{u} \in \boldsymbol{H}^2(\Omega) \cap \boldsymbol{H}_0^1(\Omega)$, $p \in H^1(\Omega)/\mathbb{R}$, $\boldsymbol{\sigma} \in \boldsymbol{H}^1(\Omega, \mathbb{M}_2)$, and

$$\|\boldsymbol{u}\|_{2,\Omega} \le c\|\boldsymbol{f}\|_{0,\Omega}, \quad \|p\|_{H^1(\Omega)/\mathbb{R}} \le c\|\boldsymbol{f}\|_{0,\Omega}, \quad \|\boldsymbol{\sigma}\|_{1,\Omega} \le c\|\boldsymbol{f}\|_{0,\Omega}, \tag{2.3}$$

where c is a positive constant independent of $\boldsymbol{f}$.

REMARK 1. Notice that the symmetric physical stress $\boldsymbol{\sigma}_{phy} = -p\mathbf{I} + (\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^t)$ and the pressure p can be expressed algebraically in terms of the pseudostress $\boldsymbol{\sigma}$:

$$\boldsymbol{\sigma}_{phy} = \boldsymbol{\sigma} + (\mathcal{A}\boldsymbol{\sigma})^t, \qquad p = -\frac{1}{2}\operatorname{tr}\boldsymbol{\sigma}.$$

Therefore, they can be computed in a postprocessing procedure in the same accuracy as the approximation of $\boldsymbol{\sigma}$.

Denote $\mathbf{V} = \mathbf{L}^2(\Omega, \mathbb{R}^2)$ and

$$\boldsymbol{\Sigma} = \boldsymbol{H}(\mathbf{div}, \Omega, \mathbb{M}_2)/span\{\mathbf{I}\} = \left\{ \boldsymbol{\tau} \in \boldsymbol{H}(\mathbf{div}, \Omega, \mathbb{M}_2) \mid \int_\Omega \operatorname{tr}\boldsymbol{\tau}\, d\boldsymbol{x} = 0 \right\}.$$

Then we have the following variational form for (2.2). [illegible] $\boldsymbol{f} \in \mathbf{V}$ [illegible] $\boldsymbol{\sigma} \in \boldsymbol{\Sigma}$ [illegible] $\boldsymbol{u} \in \mathbf{V}$ [illegible]

$$\begin{cases} (\mathcal{A}\boldsymbol{\sigma}, \boldsymbol{\tau}) + (\mathbf{div}\boldsymbol{\tau}, \boldsymbol{u}) = 0 & \text{for all } \boldsymbol{\tau} \in \boldsymbol{\Sigma}, \\ (\mathbf{div}\boldsymbol{\sigma}, \boldsymbol{v}) = (\boldsymbol{f}, \boldsymbol{v}) & \text{for all } \boldsymbol{v} \in \mathbf{V}. \end{cases} \tag{2.4}$$

Notice that the velocity boundary condition becomes the nature boundary condition in the mixed formulation. According to [10], the existence and uniqueness of problem (2.4) follows from the well-known inf-sup condition $\sup_{\boldsymbol{\tau} \in \boldsymbol{H}^1(\Omega, \mathbb{M}_2)} \frac{(\mathbf{div}\boldsymbol{\tau}, \boldsymbol{v})}{\|\boldsymbol{\tau}\|_{1,\Omega}} \ge \kappa \|\boldsymbol{v}\|_{0,\Omega}$ for all $\boldsymbol{v} \in \mathbf{V}$ and the following lemma [12].

LEMMA 2.1. [illegible] $\boldsymbol{\tau} \in \boldsymbol{\Sigma}$ [illegible]

$$\|\boldsymbol{\tau}\|_{0,\Omega}^2 \le C_A(\|\mathcal{A}^{1/2}\boldsymbol{\tau}\|_{0,\Omega}^2 + \|\mathbf{div}\boldsymbol{\tau}\|_{-1,\Omega}^2),$$

[illegible] $C_A \ge 1$ [illegible] $\boldsymbol{\tau}$

Using the penalty method [10] on problem (2.4) gives a new system

$$\begin{cases} (\mathcal{A}\boldsymbol{\sigma}_\gamma, \boldsymbol{\tau}) + (\mathbf{div}\boldsymbol{\tau}, \boldsymbol{u}_\gamma) = 0 & \text{for all } \boldsymbol{\tau} \in \boldsymbol{\Sigma}, \\ (\mathbf{div}\boldsymbol{\sigma}_\gamma, \boldsymbol{v}) - \frac{1}{\gamma}(\boldsymbol{u}_\gamma, \boldsymbol{v}) = (\boldsymbol{f}, \boldsymbol{v}) & \text{for all } \boldsymbol{v} \in \mathbf{V}, \end{cases} \tag{2.5}$$

where γ is a positive constant which is usually very large. One advantage of using the penalty method is that the mixed system (2.5) can be easily decoupled. Define a bilinear form over $\boldsymbol{H}(\mathbf{div}, \Omega, \mathbb{M}_2)$ by

$$\boldsymbol{\Lambda}(\boldsymbol{\sigma}, \boldsymbol{\tau}) = (\mathcal{A}\boldsymbol{\sigma}, \boldsymbol{\tau}) + \gamma(\mathbf{div}\boldsymbol{\sigma}, \mathbf{div}\boldsymbol{\tau}) \qquad \text{for all } \boldsymbol{\sigma}, \boldsymbol{\tau} \in \boldsymbol{\Sigma}.$$

By eliminating $\boldsymbol{u}_\gamma$, system (2.5) can be reduced to the following $\boldsymbol{H}(\mathbf{div})$ problem:

$$\boldsymbol{\Lambda}(\boldsymbol{\sigma}_\gamma, \boldsymbol{\tau}) = \gamma(\boldsymbol{f}, \mathbf{div}\boldsymbol{\tau}) \quad \text{for all } \boldsymbol{\tau} \in \boldsymbol{\Sigma}. \tag{2.6}$$

By Lemma 2.1, it is easy to see that $\boldsymbol{\Lambda}(\cdot, \cdot)$ is coercive over space $\boldsymbol{\Sigma}$. Hence problem (2.6) is well-posed. Then $\boldsymbol{u}_\gamma$ can be easily calculated through $\boldsymbol{u}_\gamma = \gamma(\mathbf{div}\boldsymbol{\sigma}_\gamma - \boldsymbol{f})$.

Finally, we mention the case of time-dependent Stokes problems. It is not hard to see that the implicit time discretization usually leads to the following pseudostress-velocity form:

$$\begin{cases} \mathbf{div}\boldsymbol{\sigma} - \frac{1}{\gamma}\boldsymbol{u} = \boldsymbol{f} & \text{in } \Omega, \\ \mathcal{A}\boldsymbol{\sigma} - \nabla \boldsymbol{u} = \mathbf{0} & \text{in } \Omega, \\ \boldsymbol{u}|_{\partial\Omega} = \mathbf{0}, \end{cases} \tag{2.7}$$

where $\gamma \ll 1$ is related to the time-step size. Again, problem (2.7) can easily be decoupled and will lead to the pseudostress problem (2.6). The only difference is that γ is small in this case.

3. Finite element approximation. In the remainder of this paper, we will focus on solving problem (2.6) using the finite element method and a multigrid solver. To discretize the problem, we need a good finite element approximation to the space $\boldsymbol{\Sigma}$. There are several well-known conforming elements for the $H(\mathrm{div})$ space over the vector field $\mathbb{R}^2$. We will consider the lowest order RT element [23] and the lowest order BDM element [9]. It should be easy to extend the results to higher order elements.

Let $\mathcal{T}_h$ be a quasi-uniform triangulation of Ω with characteristic mesh size h. On each triangle $T \in \mathcal{T}_h$, denote $P_n(T)$ to be the set of all polynomials with degrees less than or equal to n. Define

$$RT_T = span\left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} x \\ y \end{pmatrix} \right\}$$

and

$$BDM_T = (P_1(T))^2.$$

The degrees of freedom for the RT element are the zeroth order moments of the normal components on each edge of T. The degrees of freedom for the BDM element are the zeroth order and the first order moments of the normal components on each edge of T.

We use two copies of the RT/BDM element to approximate the space $\boldsymbol{H}(\mathbf{div}, \Omega, \mathbb{M}_2)$. Let

$$\boldsymbol{\Sigma}_T^{RT} = \{\boldsymbol{\tau} \in \mathbb{M}_2 \,|\, (\tau_{i1},\, \tau_{i2}) \in RT_T \text{ for } i = 1,\, 2\},$$

$$\boldsymbol{\Sigma}_T^{BDM} = \{\boldsymbol{\tau} \in \mathbb{M}_2 \,|\, (\tau_{i1},\, \tau_{i2}) \in BDM_T \text{ for } i = 1,\, 2\}.$$

Define the finite element spaces

$$\boldsymbol{\Sigma}_h^{RT} = \{\boldsymbol{\tau} \in \boldsymbol{\Sigma} \,|\; \boldsymbol{\tau}|_T \in \boldsymbol{\Sigma}_T^{RT}\} \quad \text{and} \quad \boldsymbol{\Sigma}_h^{BDM} = \{\boldsymbol{\tau} \in \boldsymbol{\Sigma} \,|\; \boldsymbol{\tau}|_T \in \boldsymbol{\Sigma}_T^{BDM}\}.$$

In later analysis, we use more common properties of the RT and the BDM elements than their differences. Hence we will use $\boldsymbol{\Sigma}_h$ to denote either $\boldsymbol{\Sigma}_h^{RT}$ or $\boldsymbol{\Sigma}_h^{BDM}$. It is well known that the necessary continuity requirement for $\boldsymbol{\Sigma}_h$ to be in $\boldsymbol{\Sigma}$ is that any function in the discrete space should have continuous normal components across each edge of the mesh. Notice that $\boldsymbol{\Sigma}_h$ also inherits the constraint $\int_\Omega \operatorname{tr} \boldsymbol{\sigma}\, d\boldsymbol{x} = 0$ from $\boldsymbol{\Sigma}$.

Consider the discrete problem of (2.6): find $\boldsymbol{\sigma}_h \in \boldsymbol{\Sigma}_h$ such that

$$\boldsymbol{\Lambda}(\boldsymbol{\sigma}_h,\, \boldsymbol{\tau}_h) = \gamma\,(\boldsymbol{f},\, \mathbf{div}\, \boldsymbol{\tau}_h) \quad \text{for all } \boldsymbol{\tau}_h \in \boldsymbol{\Sigma}_h. \tag{3.1}$$

Let $\boldsymbol{\sigma}$ and $\boldsymbol{\sigma}_h$ be solutions to problems (2.4) and (3.1), respectively. The following error bound is established in [13] for the RT element with $\gamma = O(h^{-1})$:

$$\|\boldsymbol{\sigma} - \boldsymbol{\sigma}_h\|_{0,\Omega} + \|\mathbf{div}\,(\boldsymbol{\sigma} - \boldsymbol{\sigma}_h)\|_{0,\Omega} \lesssim h\,(\|\boldsymbol{\sigma}\|_{1,\Omega} + \|\boldsymbol{f}\|_{1,\Omega})\,.$$

Here and thereafter, we use $\lesssim$ to denote "less than or equal to" with a factor c independent of the mesh size h or other parameters appearing in the inequality. The above error bound for the BDM element can be obtained in a similar fashion.

Next we give the discretization of problem (2.4) and its error estimate, which will play an important role in later analysis. Define the space $\mathbf{V}_h \subset \mathbf{L}^2(\Omega, \mathbb{R}^2)$ as follows:

$$\mathbf{V}_h = \{(v_1,\, v_2)^t \mid\ v_i|_T \in P_0(T) \text{ for } i = 1,\, 2\}.$$

Since both the RT element and the BDM element satisfy the discrete inf-sup condition [10], we have $\mathbf{div}\boldsymbol{\Sigma}_h = \mathbf{V}_h$. Consider the discrete problem for system (2.4): given $\boldsymbol{f} \in \mathbf{L}^2(\Omega, \mathbb{R}^2)$, find $\boldsymbol{\sigma}_h \in \boldsymbol{\Sigma}_h$ and $\boldsymbol{u}_h \in \mathbf{V}_h$ such that

$$\begin{cases} (\mathcal{A}\boldsymbol{\sigma}_h, \boldsymbol{\tau}_h) + (\mathbf{div}\boldsymbol{\tau}_h, \boldsymbol{u}_h) = 0 & \text{for all } \boldsymbol{\tau}_h \in \boldsymbol{\Sigma}_h, \\ (\mathbf{div}\boldsymbol{\sigma}_h, \boldsymbol{v}_h) = (\boldsymbol{f}, \boldsymbol{v}_h) & \text{for all } \boldsymbol{v}_h \in \mathbf{V}_h. \end{cases} \tag{3.2}$$

By Lemma 2.1 and the discrete inf-sup condition of the RT/BDM element space, Problem (3.2) is well-posed and admits a unique solution.

Next, we will give the mixed finite element error estimate for problem (3.2). To do this, we need to show that $\boldsymbol{\Sigma}_h$, under the constraint $\int_\Omega \operatorname{tr}\boldsymbol{\sigma}\, d\boldsymbol{x} = 0$, is a good approximation for $\boldsymbol{\Sigma}$. Denote by $\tilde{\Pi}_h$ the natural interpolation associated with the degrees of freedom onto $\boldsymbol{\Sigma}_h + span\{\mathbf{I}\}$. Let $\mathbf{P}_{\mathbf{V}_h}$ be the $\mathbf{L}^2$ projection onto $\mathbf{V}_h$. Define $\Pi_h : \boldsymbol{\Sigma} \to \boldsymbol{\Sigma}_h$ by

$$\Pi_h \boldsymbol{\tau} = \tilde{\Pi}_h \boldsymbol{\tau} - \frac{\int_\Omega (\operatorname{tr} \tilde{\Pi}_h \boldsymbol{\tau})\, d\boldsymbol{x}}{2\,|\Omega|}\,\mathbf{I} \quad \text{for all } \boldsymbol{\tau} \in \boldsymbol{\Sigma}, \tag{3.3}$$

where $|\Omega| = \int_\Omega d\boldsymbol{x}$. Clearly, Π_h is a linear operator and $\Pi_h \boldsymbol{\tau} \in \boldsymbol{\Sigma}_h$. By the properties of the natural interpolation $\tilde{\Pi}_h$ [10] and the definition of Π_h, it is easy to prove [13] the following lemma.

LEMMA 3.1. [illegible] $\mathbf{div}\Pi_h = \mathbf{P}_{\mathbf{V}_h}\mathbf{div}$ [illegible] $\boldsymbol{\tau} \in \boldsymbol{\Sigma} \cap \boldsymbol{H}^1(\Omega, \mathbb{M}_2)$ [illegible]

$$\|\boldsymbol{\tau} - \Pi_h\boldsymbol{\tau}\|_{0,\Omega} \lesssim h|\boldsymbol{\tau}|_{1,\Omega}, \tag{3.4}$$

$$\|\mathbf{div}\boldsymbol{\tau} - \mathbf{div}(\Pi_h\boldsymbol{\tau})\|_{0,\Omega} \lesssim \|\mathbf{div}\boldsymbol{\tau}\|_{0,\Omega}. \tag{3.5}$$

The following lemma gives error estimates for the mixed finite element approximation (3.2).

LEMMA 3.2. [illegible] $\boldsymbol{f} \in \mathbf{L}^2(\Omega, \mathbb{R}^2)$ [illegible] $(\boldsymbol{\sigma}, \boldsymbol{u})$ [illegible] (2.4) [illegible] $(\boldsymbol{\sigma}_h, \boldsymbol{u}_h)$ [illegible] (3.2) [illegible]

$$\|\boldsymbol{\sigma} - \boldsymbol{\sigma}_h\|_{0,\Omega} \lesssim h|\boldsymbol{\sigma}|_{1,\Omega},$$
$$\|\boldsymbol{u} - \boldsymbol{u}_h\|_{0,\Omega} \lesssim h|\boldsymbol{u}|_{1,\Omega} + h|\boldsymbol{\sigma}|_{1,\Omega}.$$

[illegible] By using the mixed finite element theory and the properties of operator Π_h, it is standard to prove that [10, 5]

$$\|\mathcal{A}(\boldsymbol{\sigma} - \boldsymbol{\sigma}_h)\|_{0,\Omega} \lesssim h|\boldsymbol{\sigma}|_{1,\Omega},$$
$$\|\boldsymbol{u} - \boldsymbol{u}_h\|_{0,\Omega} \lesssim h|\boldsymbol{u}|_{1,\Omega} + h|\boldsymbol{\sigma}|_{1,\Omega}.$$

Next, notice that $\mathbf{div}(\boldsymbol{\sigma}-\boldsymbol{\sigma}_h) = \boldsymbol{f}-\mathbf{P}_{\mathbf{V}_h}\boldsymbol{f}$. By Lemma 2.1 and the approximation property of the $\mathbf{L}^2$ projection, we have

$$\begin{aligned}
\|\boldsymbol{\sigma}-\boldsymbol{\sigma}_h\|_{0,\Omega} &\lesssim \|\mathcal{A}(\boldsymbol{\sigma}-\boldsymbol{\sigma}_h)\|_{0,\Omega} + \|\mathbf{div}(\boldsymbol{\sigma}-\boldsymbol{\sigma}_h)\|_{-1,\Omega} \\
&\lesssim h|\boldsymbol{\sigma}|_{1,\Omega} + \sup_{\boldsymbol{v}\in \boldsymbol{H}_0^1(\Omega,\mathbb{R}^2)} \frac{(\boldsymbol{f}-\mathbf{P}_{\mathbf{V}_h}\boldsymbol{f},\boldsymbol{v})}{\|\boldsymbol{v}\|_{1,\Omega}} \\
&= h|\boldsymbol{\sigma}|_{1,\Omega} + \sup_{\boldsymbol{v}\in \boldsymbol{H}_0^1(\Omega,\mathbb{R}^2)} \frac{(\boldsymbol{f},\boldsymbol{v}-\mathbf{P}_{\mathbf{V}_h}\boldsymbol{v})}{\|\boldsymbol{v}\|_{1,\Omega}} \\
&\lesssim h|\boldsymbol{\sigma}|_{1,\Omega} + h\|\boldsymbol{f}\|_{0,\Omega} = h|\boldsymbol{\sigma}|_{1,\Omega} + h\|\mathbf{div}\boldsymbol{\sigma}\|_{0,\Omega} \\
&\lesssim h|\boldsymbol{\sigma}|_{1,\Omega}.
\end{aligned}$$

This completes the proof of the lemma. ∎

4. A decomposition in $\boldsymbol{\Sigma_h}$. In this section we introduce a Helmholtz-type decomposition of the discrete space $\boldsymbol{\Sigma}_h$. This decomposition and its properties borrowed many ideas from the decomposition presented in the work of Arnold, Falk, and Winther on $H(\mathrm{div})$ problems [1, 2]. The difference is that our decomposition needs to use the projection operator $\mathcal{A}$, which makes the proof more complicated. For the convenience of readers, we give all details of the proof even if some parts are similar to the proof given in [1, 2].

Consider the mixed finite element problem (3.2). Let $(\boldsymbol{\sigma}_h, \boldsymbol{u}_h)$ be the solution to (3.2) with the right-hand side function $\boldsymbol{f} \in \mathbf{L}^2(\Omega,\mathbb{R}^2)$. Denote $\mathbf{div}_h^{-1}\boldsymbol{f} = \boldsymbol{\sigma}_h$. The operator $\mathbf{div}_h^{-1}$ maps $\mathbf{L}^2(\Omega,\mathbb{R}^2)$ to $\boldsymbol{\Sigma}_h$ and can be considered as a pseudoinverse of the $\mathbf{div}$ operator. It is not hard to see that for all $\boldsymbol{\tau}_h \in \boldsymbol{\Sigma}_h$ such that $\mathbf{div}\boldsymbol{\tau}_h = \mathbf{0}$,

$$(\mathcal{A}\mathbf{div}_h^{-1}\boldsymbol{f},\boldsymbol{\tau}_h) = -(\mathbf{div}\boldsymbol{\tau}_h,\boldsymbol{u}_h) = 0.$$

Define

$$\boldsymbol{\Sigma}_h^0 = \{\boldsymbol{\tau}_h \in \boldsymbol{\Sigma}_h \text{ such that } \mathbf{div}\boldsymbol{\tau}_h = \mathbf{0}\}, \tag{4.1}$$

$$\boldsymbol{\Sigma}_h^1 = \{\boldsymbol{\tau}_h \in \mathbf{div}_h^{-1}\mathbf{V}_h\}. \tag{4.2}$$

Then $\boldsymbol{\Sigma}_h^0$ and $\boldsymbol{\Sigma}_h^1$ are orthogonal under $\boldsymbol{\Lambda}(\cdot,\cdot)$. It is easy to see that $\boldsymbol{\Sigma}_h = \boldsymbol{\Sigma}_h^0 \oplus \boldsymbol{\Sigma}_h^1$.

Define the space $\tilde{\mathbf{U}}_h \subset \boldsymbol{H}^1(\Omega,\mathbb{R}^2)$ as follows: for the RT element, let

$$\tilde{\mathbf{U}}_h = \{(q_1,q_2)^t \in \boldsymbol{H}^1(\Omega,\mathbb{R}^2) \mid q_1,\, q_2 \text{ are } P_1 \text{ polynomials on each } T \in \mathcal{T}_h\}$$

and for the BDM element, let

$$\tilde{\mathbf{U}}_h = \{(q_1,q_2)^t \in \boldsymbol{H}^1(\Omega,\mathbb{R}^2) \mid q_1,\, q_2 \text{ are } P_2 \text{ polynomials on each } T \in \mathcal{T}_h\}.$$

Define

$$\mathbf{U}_h = \tilde{\mathbf{U}}_h / span\left\{\begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}-y\\x\end{pmatrix}\right\}.$$

It is well known that every $\boldsymbol{\tau}_h \in \boldsymbol{\Sigma}_h^0$ has a potential $\boldsymbol{q}_h \in \tilde{\mathbf{U}}_h$ such that $\boldsymbol{\tau}_h = \mathbf{curl}\,\boldsymbol{q}_h$ [10]. The constraint $\int_\Omega \operatorname{tr}\boldsymbol{\tau}_h\, d\boldsymbol{x} = 0$ further implies that $\boldsymbol{q}_h$ can be chosen from $\mathbf{U}_h$. In other words, $\boldsymbol{\Sigma}_h^0 = \mathbf{curl}\,(\mathbf{U}_h)$. Hence we have the following exact sequence:

$$0 \longrightarrow \mathbf{U}_h \xrightarrow{\mathbf{curl}} \boldsymbol{\Sigma}_h \xrightarrow{\mathbf{div}} \mathbf{V}_h \longrightarrow 0.$$

For each $\boldsymbol{\tau}_h \in \boldsymbol{\Sigma}_h$, we have the decomposition $\boldsymbol{\sigma}_h = \mathbf{curl}\, q_h + \mathbf{div}_h^{-1} v_h$, where $q_h \in \mathbf{U}_h$, $v_h = \mathbf{div}\boldsymbol{\sigma}_h \in \mathbf{V}_h$ and the decomposition is orthogonal under both $(\mathcal{A}\cdot,\cdot)$ and $(\mathbf{div}\cdot,\mathbf{div}\cdot)$. This decomposition is usually referred to as the discrete Helmholtz decomposition.

Next, we study some properties related to the discrete Helmholtz decomposition. Define an operator $\mathbf{grad}_h : \mathbf{V}_h \to \boldsymbol{\Sigma}_h$ as follows:

$$(\mathbf{grad}_h v_h, \boldsymbol{\tau}_h) = -(\mathbf{div}\boldsymbol{\tau}_h, v_h) \quad \text{for all } \boldsymbol{\tau}_h \in \boldsymbol{\Sigma}_h.$$

The operator $\mathbf{grad}_h$ is actually the discrete dual operator of $\mathbf{div}$. Notice that for an arbitrary $\boldsymbol{\sigma}_h \in \boldsymbol{\Sigma}_h$, $\mathcal{A}\boldsymbol{\sigma}_h$ is not necessarily in $\boldsymbol{\Sigma}_h$ since it may not satisfy the $\boldsymbol{H}(\mathbf{div}, \Omega, \mathbb{M}_2)$ continuity requirement on normal components. Therefore we need to define discrete operators $A_h : \boldsymbol{\Sigma}_h \to \boldsymbol{\Sigma}_h$ and $\boldsymbol{\Lambda}_h : \boldsymbol{\Sigma}_h \to \boldsymbol{\Sigma}_h$ by

$$\begin{aligned}(A_h\boldsymbol{\sigma}_h, \boldsymbol{\tau}_h) &= (\mathcal{A}\boldsymbol{\sigma}_h, \boldsymbol{\tau}_h) \quad \text{for all } \boldsymbol{\tau}_h \in \boldsymbol{\Sigma}_h,\\ (\boldsymbol{\Lambda}_h\boldsymbol{\sigma}_h, \boldsymbol{\tau}_h) &= \boldsymbol{\Lambda}(\boldsymbol{\sigma}_h, \boldsymbol{\tau}_h) \quad \text{for all } \boldsymbol{\tau}_h \in \boldsymbol{\Sigma}_h.\end{aligned}$$

It is equivalent to say that $A_h = \mathbf{P}_{\boldsymbol{\Sigma}_h}\mathcal{A}$, where $\mathbf{P}_{\boldsymbol{\Sigma}_h}$ is the $\mathbf{L}^2$ projection onto the finite dimensional space $\boldsymbol{\Sigma}_h$ and $\boldsymbol{\Lambda}_h\boldsymbol{\sigma}_h = A_h\boldsymbol{\sigma}_h - \gamma\mathbf{grad}_h\mathbf{div}\boldsymbol{\sigma}_h$. By the property of the projection operator $\mathbf{P}_{\boldsymbol{\Sigma}_h}$, we have

$$\|A_h\boldsymbol{\tau}_h\|_{0,\Omega} \lesssim \|\mathcal{A}\boldsymbol{\tau}_h\|_{0,\Omega} \qquad \text{for all } \boldsymbol{\tau}_h \in \boldsymbol{\Sigma}_h. \tag{4.3}$$

Let $\mathcal{T}_H$ be a quasi-uniform triangulation of Ω with characteristic mesh size H and let $\mathcal{T}_h$ be a refinement of $\mathcal{T}_H$ with characteristic mesh size h. We follow the notation previously introduced with the understanding that spaces and operators with subscript H are defined over the mesh $\mathcal{T}_H$. For example, $\boldsymbol{\Sigma}_H$ is defined on $\mathcal{T}_H$ in the same way that $\boldsymbol{\Sigma}_h$ is defined on $\mathcal{T}_h$. Besides, the RT/BDM finite element spaces are known to be nested; that is, $\boldsymbol{\Sigma}_H \subset \boldsymbol{\Sigma}_h$.

LEMMA 4.1. [illegible] $v_h \in \mathbf{V}_h$ [illegible] $\boldsymbol{\sigma}_h = \mathbf{div}_h^{-1} v_h$ [illegible] $\boldsymbol{\sigma}_H = \mathbf{div}_H^{-1} v_h$ [illegible]

$$\begin{aligned}\|\boldsymbol{\sigma}_h - \boldsymbol{\sigma}_H\|_{0,\Omega} &\lesssim H\|v_h\|_{0,\Omega},\\ \|\mathbf{div}(\boldsymbol{\sigma}_h - \boldsymbol{\sigma}_H)\|_{0,\Omega} &\lesssim H\|\mathbf{grad}_h v_h\|_{0,\Omega}.\end{aligned}$$

[illegible] Let $\boldsymbol{\sigma} \in \boldsymbol{\Sigma}$ be the solution to

$$\begin{cases}(\mathcal{A}\boldsymbol{\sigma}, \boldsymbol{\tau}) + (\mathbf{div}\boldsymbol{\tau}, u) = 0 & \text{for all } \boldsymbol{\tau} \in \boldsymbol{\Sigma},\\ (\mathbf{div}\boldsymbol{\sigma}, w) = (v_h, w) & \text{for all } w \in \mathbf{L}^2(\Omega, \mathbb{R}^2).\end{cases}$$

By the regularity property (2.3), Lemma 3.2, and the fact that $h < H$, we have

$$\|\boldsymbol{\sigma}_h - \boldsymbol{\sigma}_H\|_{0,\Omega} \le \|\boldsymbol{\sigma} - \boldsymbol{\sigma}_h\|_{0,\Omega} + \|\boldsymbol{\sigma} - \boldsymbol{\sigma}_H\|_{0,\Omega} \lesssim H|\boldsymbol{\sigma}|_{1,\Omega} \lesssim H\|v_h\|_{0,\Omega}.$$

Notice that $\mathbf{div}(\boldsymbol{\sigma}_h - \boldsymbol{\sigma}_H) = v_h - \mathbf{P}_{\mathbf{V}_H} v_h$, where $\mathbf{P}_{\mathbf{V}_H}$ is the $\mathbf{L}^2$ projection onto $\mathbf{V}_H$. Define $\tilde{\boldsymbol{\sigma}} \in \boldsymbol{\Sigma}$ by

$$\begin{cases}(\mathcal{A}\tilde{\boldsymbol{\sigma}}, \boldsymbol{\tau}) + (\mathbf{div}\boldsymbol{\tau}, u) = 0 & \text{for all } \boldsymbol{\tau} \in \boldsymbol{\Sigma},\\ (\mathbf{div}\tilde{\boldsymbol{\sigma}}, w) = (v_h - \mathbf{P}_{\mathbf{V}_H} v_h, w) & \text{for all } w \in \mathbf{L}^2(\Omega, \mathbb{R}^2).\end{cases}$$

Recall that Π_h and Π_H satisfy $\mathbf{div}\Pi_h = \mathbf{P}_{\mathbf{V}_h}\mathbf{div}$ and $\mathbf{div}\Pi_H = \mathbf{P}_{\mathbf{V}_H}\mathbf{div}$. By using Lemma 3.1, the Schwarz inequality, and the regularity result (2.3), we have

$$\begin{aligned}
\|\boldsymbol{v}_h - \mathbf{P}_{\mathbf{V}_H}\boldsymbol{v}_h\|_{0,\Omega}^2 &= (\mathbf{div}\tilde{\boldsymbol{\sigma}}, \boldsymbol{v}_h - \mathbf{P}_{\mathbf{V}_H}\boldsymbol{v}_h) \\
&= (\mathbf{div}(\Pi_h\tilde{\boldsymbol{\sigma}} - \Pi_H\tilde{\boldsymbol{\sigma}}), \boldsymbol{v}_h) = -((\Pi_h\tilde{\boldsymbol{\sigma}} - \Pi_H\tilde{\boldsymbol{\sigma}}), \mathbf{grad}_h\boldsymbol{v}_h) \\
&\le \|\Pi_h\tilde{\boldsymbol{\sigma}} - \Pi_H\tilde{\boldsymbol{\sigma}}\|_{0,\Omega}\|\mathbf{grad}_h\boldsymbol{v}_h\|_{0,\Omega} \lesssim H|\tilde{\boldsymbol{\sigma}}|_{1,\Omega}\|\mathbf{grad}_h\boldsymbol{v}_h\|_{0,\Omega} \\
&\lesssim H\|\boldsymbol{v}_h - \mathbf{P}_{\mathbf{V}_H}\boldsymbol{v}_h\|_{0,\Omega}\|\mathbf{grad}_h\boldsymbol{v}_h\|_{0,\Omega}.
\end{aligned}$$

This completes the proof of the lemma. □

LEMMA 4.2. For $\boldsymbol{\sigma}_h \in \boldsymbol{\Sigma}_h$ satisfying $\boldsymbol{\Lambda}(\boldsymbol{\sigma}_h, \boldsymbol{\tau}_H) = 0$ for all $\boldsymbol{\tau}_H \in \boldsymbol{\Sigma}_H$, $\boldsymbol{\sigma}_h = \mathbf{curl}\,\boldsymbol{q}_h + \mathbf{div}_h^{-1}\boldsymbol{v}_h$ for some $\boldsymbol{q}_h \in \mathbf{U}_h$ and $\boldsymbol{v}_h = \mathbf{div}\boldsymbol{\sigma}_h \in \mathbf{V}_h$. Let $\boldsymbol{\Lambda}$ be elliptic, and let $\boldsymbol{\sigma}_h^1 = \mathbf{div}_h^{-1}\boldsymbol{v}_h$. Then

$$(\mathcal{A}\boldsymbol{\sigma}_h^1, \boldsymbol{\sigma}_h^1) \lesssim \gamma^{-1}H^2\boldsymbol{\Lambda}(\boldsymbol{\sigma}_h, \boldsymbol{\sigma}_h), \tag{4.4}$$

$$(1 + \gamma H^{-2})\|\boldsymbol{\sigma}_h^1\|_{0,\Omega}^2 \lesssim (1 + \gamma^{-2})\boldsymbol{\Lambda}(\boldsymbol{\sigma}_h, \boldsymbol{\sigma}_h), \tag{4.5}$$

$$\|\boldsymbol{q}_h\|_{0,\Omega}^2 \lesssim H^2(\mathcal{A}\mathbf{curl}\,\boldsymbol{q}_h, \mathbf{curl}\,\boldsymbol{q}_h). \tag{4.6}$$

The proof of inequality (4.6) is a little long and hence will be put in the appendix. Here we give the proof of inequalities (4.4) and (4.5). Define $\tilde{\boldsymbol{\sigma}}_h \in \boldsymbol{\Sigma}_h$ by

$$\boldsymbol{\Lambda}(\tilde{\boldsymbol{\sigma}}_h, \boldsymbol{\tau}_h) = (\mathcal{A}\boldsymbol{\sigma}_h^1, \boldsymbol{\tau}_h) \quad \text{for all } \boldsymbol{\tau}_h \in \boldsymbol{\Sigma}_h. \tag{4.7}$$

It is clear that

$$\boldsymbol{\Lambda}(\tilde{\boldsymbol{\sigma}}_h, \mathbf{curl}\,\boldsymbol{p}_h) = (\mathcal{A}\boldsymbol{\sigma}_h^1, \mathbf{curl}\,\boldsymbol{p}_h) = 0 \quad \text{for all } \boldsymbol{p}_h \in \mathbf{U}_h.$$

This implies $\tilde{\boldsymbol{\sigma}}_h \in \boldsymbol{\Sigma}_h^1$. In other words, if we denote $\mathbf{div}\tilde{\boldsymbol{\sigma}}_h = \tilde{\boldsymbol{v}}_h$, then $\tilde{\boldsymbol{\sigma}}_h = \mathbf{div}_h^{-1}\tilde{\boldsymbol{v}}_h$. Define $\tilde{\boldsymbol{\sigma}}_H = \mathbf{div}_H^{-1}\tilde{\boldsymbol{v}}_h$. By the Schwarz inequality,

$$\begin{aligned}
(\mathcal{A}\boldsymbol{\sigma}_h^1, \boldsymbol{\sigma}_h^1) &= \boldsymbol{\Lambda}(\tilde{\boldsymbol{\sigma}}_h, \boldsymbol{\sigma}_h^1) = \boldsymbol{\Lambda}(\tilde{\boldsymbol{\sigma}}_h, \boldsymbol{\sigma}_h) \\
&= \boldsymbol{\Lambda}(\tilde{\boldsymbol{\sigma}}_h - \tilde{\boldsymbol{\sigma}}_H, \boldsymbol{\sigma}_h) \le \boldsymbol{\Lambda}(\tilde{\boldsymbol{\sigma}}_h - \tilde{\boldsymbol{\sigma}}_H, \tilde{\boldsymbol{\sigma}}_h - \tilde{\boldsymbol{\sigma}}_H)^{1/2}\boldsymbol{\Lambda}(\boldsymbol{\sigma}_h, \boldsymbol{\sigma}_h)^{1/2}.
\end{aligned} \tag{4.8}$$

We also notice that (4.7) implies that $\boldsymbol{\Lambda}_h\tilde{\boldsymbol{\sigma}}_h = A_h\boldsymbol{\sigma}_h^1$. Furthermore, by setting $\boldsymbol{\tau}_h = \tilde{\boldsymbol{\sigma}}_h$ in (4.7) and using the Schwarz inequality, we can easily see that $\boldsymbol{\Lambda}(\tilde{\boldsymbol{\sigma}}_h, \tilde{\boldsymbol{\sigma}}_h) \le (\mathcal{A}\boldsymbol{\sigma}_h^1, \boldsymbol{\sigma}_h^1)$. Hence, by the fact that $\mathcal{A}$ is a projection, Lemma 4.1, the triangular inequality, and inequality (4.3), we have

$$\begin{aligned}
\boldsymbol{\Lambda}(\tilde{\boldsymbol{\sigma}}_h - \tilde{\boldsymbol{\sigma}}_H, \tilde{\boldsymbol{\sigma}}_h - \tilde{\boldsymbol{\sigma}}_H) &\le \|\tilde{\boldsymbol{\sigma}}_h - \tilde{\boldsymbol{\sigma}}_H\|_{0,\Omega}^2 + \gamma\|\mathbf{div}(\tilde{\boldsymbol{\sigma}}_h - \tilde{\boldsymbol{\sigma}}_H)\|_{0,\Omega}^2 \\
&\lesssim H^2\gamma^{-1}(\gamma\|\mathbf{div}\tilde{\boldsymbol{\sigma}}_h\|_{0,\Omega}^2 + \gamma^2\|\mathbf{grad}_h\mathbf{div}\tilde{\boldsymbol{\sigma}}_h\|_{0,\Omega}^2) \\
&\lesssim H^2\gamma^{-1}(\gamma\|\mathbf{div}\tilde{\boldsymbol{\sigma}}_h\|_{0,\Omega}^2 + \|A_h\tilde{\boldsymbol{\sigma}}_h\|_{0,\Omega}^2 + \|(A_h - \gamma\mathbf{grad}_h\mathbf{div})\tilde{\boldsymbol{\sigma}}_h\|_{0,\Omega}^2) \\
&\lesssim H^2\gamma^{-1}(\boldsymbol{\Lambda}(\tilde{\boldsymbol{\sigma}}_h, \tilde{\boldsymbol{\sigma}}_h) + \|\boldsymbol{\Lambda}_h\tilde{\boldsymbol{\sigma}}_h\|_{0,\Omega}^2) \\
&\lesssim H^2\gamma^{-1}((\mathcal{A}\boldsymbol{\sigma}_h^1, \boldsymbol{\sigma}_h^1) + \|A_h\boldsymbol{\sigma}_h^1\|_{0,\Omega}^2) \\
&\lesssim H^2\gamma^{-1}(\mathcal{A}\boldsymbol{\sigma}_h^1, \boldsymbol{\sigma}_h^1).
\end{aligned} \tag{4.9}$$

Combining (4.8) and (4.9) gives $(\mathcal{A}\boldsymbol{\sigma}_h^1, \boldsymbol{\sigma}_h^1) \lesssim H\gamma^{-1/2}(\mathcal{A}\boldsymbol{\sigma}_h^1, \boldsymbol{\sigma}_h^1)^{1/2}\boldsymbol{\Lambda}(\boldsymbol{\sigma}_h, \boldsymbol{\sigma}_h)^{1/2}$. Thus (4.4) holds.

Next, we prove (4.5). Notice that $\boldsymbol{\Lambda}(\boldsymbol{\sigma}_h, \boldsymbol{\tau}_H) = 0$ for all $\boldsymbol{\tau}_H \in \boldsymbol{\Sigma}_H$ implies

$$(\mathbf{div}\boldsymbol{\sigma}_h^1, \mathbf{div}\boldsymbol{\tau}_H) = -\gamma^{-1}(\mathcal{A}\boldsymbol{\sigma}_h^1, \boldsymbol{\tau}_H) - \gamma^{-1}(\mathcal{A}\mathbf{curl}\, \boldsymbol{q}_h, \boldsymbol{\tau}_H). \tag{4.10}$$

By definition,

$$\|\mathbf{div}\boldsymbol{\sigma}_h^1\|_{-1,\Omega} = \sup_{\boldsymbol{v}\in \boldsymbol{H}_0^1(\Omega,\mathbb{R}^2)} \frac{(\mathbf{div}\boldsymbol{\sigma}_h^1, \boldsymbol{v})}{\|\boldsymbol{v}\|_{1,\Omega}}.$$

For $\boldsymbol{v} \in \boldsymbol{H}_0^1(\Omega, \mathbb{R}^2)$, let $\boldsymbol{v}_H$ be the $\mathbf{L}^2$ projection of $\boldsymbol{v}$ onto $\mathbf{V}_H$. Then the following estimates are well known:

$$\|\boldsymbol{v} - \boldsymbol{v}_H\|_{0,\Omega} \lesssim H\|\boldsymbol{v}\|_{1,\Omega},$$
$$\|\boldsymbol{v}_H\|_{0,\Omega} \le \|\boldsymbol{v}\|_{0,\Omega} \le \|\boldsymbol{v}\|_{1,\Omega}.$$

Define $\boldsymbol{\tau}_H = \mathbf{div}_H^{-1}\boldsymbol{v}_H$ and $\boldsymbol{\tau}_h = \mathbf{div}_h^{-1}\boldsymbol{v}_H$. We have $(\mathcal{A}\mathbf{curl}\, \boldsymbol{q}_h, \boldsymbol{\tau}_h) = 0$. Besides, by Lemma 3.2 and the regularity result (2.3), it is not hard to see that $\|\boldsymbol{\tau}_H\|_{0,\Omega} \lesssim \|\boldsymbol{v}_H\|_{0,\Omega}$. Combining the above and using Lemma 4.1 and (4.10), we have

$$\begin{aligned}
\sup_{\boldsymbol{v}\in \boldsymbol{H}_0^1(\Omega,\mathbb{R}^2)} \frac{(\mathbf{div}\boldsymbol{\sigma}_h^1, \boldsymbol{v})}{\|\boldsymbol{v}\|_{1,\Omega}} &\le \sup_{\boldsymbol{v}\in \boldsymbol{H}_0^1(\Omega,\mathbb{R}^2)} \frac{(\mathbf{div}\boldsymbol{\sigma}_h^1, \boldsymbol{v} - \boldsymbol{v}_H)}{\|\boldsymbol{v}\|_{1,\Omega}} + \sup_{\boldsymbol{v}\in \boldsymbol{H}_0^1(\Omega,\mathbb{R}^2)} \frac{(\mathbf{div}\boldsymbol{\sigma}_h^1, \boldsymbol{v}_H)}{\|\boldsymbol{v}\|_{1,\Omega}} \\
&\lesssim H\|\mathbf{div}\boldsymbol{\sigma}_h^1\|_{0,\Omega} + \sup_{\boldsymbol{v}\in \boldsymbol{H}_0^1(\Omega,\mathbb{R}^2)} \frac{(\mathbf{div}\boldsymbol{\sigma}_h^1, \mathbf{div}\boldsymbol{\tau}_H)}{\|\boldsymbol{v}\|_{1,\Omega}} \\
&= H\|\mathbf{div}\boldsymbol{\sigma}_h^1\|_{0,\Omega} + \sup_{\boldsymbol{v}\in \boldsymbol{H}_0^1(\Omega,\mathbb{R}^2)} \frac{-\gamma^{-1}(\mathcal{A}\boldsymbol{\sigma}_h^1, \boldsymbol{\tau}_H) - \gamma^{-1}(\mathcal{A}\mathbf{curl}\, \boldsymbol{q}_h, \boldsymbol{\tau}_H - \boldsymbol{\tau}_h)}{\|\boldsymbol{v}\|_{1,\Omega}} \\
&\lesssim H\|\mathbf{div}\boldsymbol{\sigma}_h^1\|_{0,\Omega} + \gamma^{-1}(\mathcal{A}\boldsymbol{\sigma}_h^1, \boldsymbol{\sigma}_h^1)^{1/2} + \gamma^{-1}H(\mathcal{A}\mathbf{curl}\, \boldsymbol{q}_h, \mathbf{curl}\, \boldsymbol{q}_h)^{1/2}.
\end{aligned}$$

Using the above result, Lemma 2.1, inequality (4.4), the inequality $2\gamma^{-1} \le 1 + \gamma^{-2}$, and the fact that $H < O(1)$, we have

$$\begin{aligned}
\|\boldsymbol{\sigma}_h^1\|_{0,\Omega}^2 &\lesssim (\mathcal{A}\boldsymbol{\sigma}_h^1, \boldsymbol{\sigma}_h^1) + \|\mathbf{div}\boldsymbol{\sigma}_h^1\|_{-1,\Omega}^2 \\
&\lesssim (1+\gamma^{-2})\boldsymbol{\Lambda}(\boldsymbol{\sigma}_h^1, \boldsymbol{\sigma}_h^1) + \|\mathbf{div}\boldsymbol{\sigma}_h^1\|_{0,\Omega}^2 + \gamma^{-2}(\mathcal{A}\mathbf{curl}\, \boldsymbol{q}_h, \mathbf{curl}\, \boldsymbol{q}_h) \\
&\lesssim (1+\gamma^{-2})\boldsymbol{\Lambda}(\boldsymbol{\sigma}_h, \boldsymbol{\sigma}_h)
\end{aligned}$$

and

$$\begin{aligned}
\gamma H^{-2}\|\boldsymbol{\sigma}_h^1\|_{0,\Omega}^2 &\lesssim (1+\gamma^{-2})\gamma H^{-2}(\mathcal{A}\boldsymbol{\sigma}_h^1, \boldsymbol{\sigma}_h^1) + \gamma\|\mathbf{div}\boldsymbol{\sigma}_h^1\|_{0,\Omega}^2 + \gamma^{-1}(\mathcal{A}\mathbf{curl}\, \boldsymbol{q}_h, \mathbf{curl}\, \boldsymbol{q}_h) \\
&\lesssim (1+\gamma^{-2})\boldsymbol{\Lambda}(\boldsymbol{\sigma}_h, \boldsymbol{\sigma}_h).
\end{aligned}$$

Adding up the above inequalities gives the estimate (4.5). This completes the proof of the lemma. ∎

5. A multigrid solver. For simplicity, we rewrite problem (3.1) in the following form: find $\boldsymbol{\sigma}_h \in \boldsymbol{\Sigma}_h$ such that

$$\boldsymbol{\Lambda}(\boldsymbol{\sigma}_h, \boldsymbol{\tau}_h) = (\boldsymbol{F}, \boldsymbol{\tau}_h) \quad \text{for all } \boldsymbol{\tau}_h \in \boldsymbol{\Sigma}_h, \tag{5.1}$$

where $\boldsymbol{F} \in \boldsymbol{\Sigma}_h$ is defined by $(\boldsymbol{F}, \boldsymbol{\tau}_h) = \gamma(\boldsymbol{f}, \mathbf{div}\boldsymbol{\tau}_h)$.

In this section, we construct and analyze a multigrid solver for the discrete problem (5.1). Let $\mathcal{T}_k$, $k = 1, \ldots, K$, be a series of nested meshes under uniform refinements and assume that $\mathcal{T}_1$ has characteristic mesh size $O(1)$. Denote the characteristic mesh size of $\mathcal{T}_k$ to be h_k. Define $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}_{h_k}$, the RT/BDM finite element space based on $\mathcal{T}_k$. Notice that $\boldsymbol{\Sigma}_{k-1} \subset \boldsymbol{\Sigma}_k$. Similarly, we define spaces $\mathbf{U}_k$ and $\mathbf{V}_k$. Denote $\mathbf{div}_k^{-1} = \mathbf{div}_{h_k}^{-1}$. Denote $\mathbf{Q}_{k-1} : \boldsymbol{\Sigma}_k \to \boldsymbol{\Sigma}_{k-1}$ and $\mathbf{P}_{k-1} : \boldsymbol{\Sigma}_k \to \boldsymbol{\Sigma}_{k-1}$ to be the $\mathbf{L}^2$ projection and the $\boldsymbol{\Lambda}$ projection, respectively. Let $\mathbf{R}_k : \boldsymbol{\Sigma}_k \to \boldsymbol{\Sigma}_k$ be a symmetric positive definite operator, which will be defined later. Let m be the smoothing times on each level. Finally, denote A_k to be A_{h_k}.

We define a V-cycle multigrid iteration operator $Mg_k(\cdot,\cdot) : \boldsymbol{\Sigma}_k \times \boldsymbol{\Sigma}_k \to \boldsymbol{\Sigma}_k$ inductively as follows.

ALGORITHM 1. Set $Mg_1(\boldsymbol{\tau}, \boldsymbol{F}) = \boldsymbol{\Lambda}_1^{-1}\boldsymbol{F}$. Assuming that $Mg_{k-1}(\cdot,\cdot)$ has been defined, define $Mg_k(\boldsymbol{\tau}, \boldsymbol{F})$ as follows: set $\boldsymbol{\tau}^0 = \boldsymbol{\tau}$ and define

(1) $\boldsymbol{\tau}^l = \boldsymbol{\tau}^{l-1} + \mathbf{R}_k(\boldsymbol{F} - \boldsymbol{\Lambda}_k\boldsymbol{\tau}^{l-1})$ for $l = 1, \ldots, m$;

(2) $\boldsymbol{\sigma}^m = \boldsymbol{\tau}^m + Mg_{k-1}(\mathbf{0}, \mathbf{Q}_{k-1}(\boldsymbol{F} - \boldsymbol{\Lambda}_k\boldsymbol{\tau}^m))$;

(3) $\boldsymbol{\sigma}^l = \boldsymbol{\sigma}^{l-1} + \mathbf{R}_k(\boldsymbol{F} - \boldsymbol{\Lambda}_k\boldsymbol{\sigma}^{l-1})$ for $l = m+1, \ldots, 2m$.

Set $Mg_k(\boldsymbol{\tau}, \boldsymbol{F}) = \boldsymbol{\sigma}^{2m}$.

The V-cycle multigrid iteration for solving problem (5.1) over $\boldsymbol{\Sigma}_k$ is defined below. Given an initial guess $\boldsymbol{\sigma}^0 \in \boldsymbol{\Sigma}_k$, we defined a sequence approximating the solution $\boldsymbol{\sigma}_h$ by

$$\boldsymbol{\sigma}^{i+1} = Mg_k(\boldsymbol{\sigma}^i, \boldsymbol{F}), \qquad i = 0, 1, \ldots.$$

The error propagation operator $\boldsymbol{\varepsilon} : \boldsymbol{\Sigma}_k \to \boldsymbol{\Sigma}_k$ for the above iteration is [8]

$$\boldsymbol{\sigma}_h - \boldsymbol{\sigma}^{i+1} = \boldsymbol{\varepsilon}(\boldsymbol{\sigma}_h - \boldsymbol{\sigma}^i) = Mg_k(\boldsymbol{\sigma}_h - \boldsymbol{\sigma}^i, 0).$$

Similarly to the proof of Theorem 7.4 in [8] and Theorem 5.1 in [1], one can show that the multigrid solver with $\mathbf{R}_k$ as the smoother and m smoothing times on each level will have the convergence rate

$$\boldsymbol{\Lambda}(\boldsymbol{\varepsilon}\boldsymbol{\tau}, \boldsymbol{\tau}) \le \frac{C_p}{C_p + 2m}\boldsymbol{\Lambda}(\boldsymbol{\tau}, \boldsymbol{\tau}) \quad \text{for all } \boldsymbol{\tau} \in \boldsymbol{\Sigma}_k \tag{5.2}$$

if the following two assumptions are true:

(M.1) The spectrum of $I - \mathbf{R}_k\boldsymbol{\Lambda}_k$ is in $[0, 1)$. That is,

$$0 \le \boldsymbol{\Lambda}((I - \mathbf{R}_k\boldsymbol{\Lambda}_k)\boldsymbol{\tau}, \boldsymbol{\tau}) < \boldsymbol{\Lambda}(\boldsymbol{\tau}, \boldsymbol{\tau}) \qquad \text{for all } \boldsymbol{\tau} \in \boldsymbol{\Sigma}_k.$$

(M.2) There exists a positive constant C_p such that

$$\boldsymbol{\Lambda}((I - \mathbf{P}_{k-1})\boldsymbol{\tau}, \boldsymbol{\tau}) \le C_p\mathbf{R}_k(\boldsymbol{\Lambda}_k\boldsymbol{\tau}, \boldsymbol{\Lambda}_k\boldsymbol{\tau}) \quad \text{for all } \boldsymbol{\tau} \in \boldsymbol{\Sigma}_k.$$

In the remainder of this section, we will construct a smoother $\mathbf{R}_k$ which satisfies assumptions (M.1) and (M.2).

Let $\mathcal{V}_k$ be the set of all vertices in $\mathcal{T}_k$, including the boundary vertices. For each $\boldsymbol{x} \in \mathcal{V}_k$, define $\Omega_{k,\boldsymbol{x}}$ to be the interior of the union of all triangles in $\mathcal{T}_k$ which have $\boldsymbol{x}$ as a vertex. Let $\boldsymbol{\Sigma}_{k,\boldsymbol{x}}$ be the RT/BDM space defined on $\mathcal{T}_k$ such that each function in $\boldsymbol{\Sigma}_{k,\boldsymbol{x}}$ vanishes outside $\Omega_{k,\boldsymbol{x}}$. Notice that $\boldsymbol{\Sigma}_{k,\boldsymbol{x}} \subset (\boldsymbol{\Sigma}_k + span\{\mathbf{I}\})$, but $\boldsymbol{\Sigma}_{k,\boldsymbol{x}} \nsubseteq \boldsymbol{\Sigma}_k$ in general, since $\int_\Omega \operatorname{tr}\boldsymbol{\tau}\, d\boldsymbol{x}$ may not vanish for $\boldsymbol{\tau} \in \boldsymbol{\Sigma}_{k,\boldsymbol{x}}$.

Define operator $\boldsymbol{\Lambda}_{k,\boldsymbol{x}} : \boldsymbol{\Sigma}_{k,\boldsymbol{x}} \to \boldsymbol{\Sigma}_{k,\boldsymbol{x}}$ by

$$(\boldsymbol{\Lambda}_{k,\boldsymbol{x}}\boldsymbol{\sigma}, \boldsymbol{\tau}) = \boldsymbol{\Lambda}(\boldsymbol{\sigma}, \boldsymbol{\tau}) \quad \text{for all } \boldsymbol{\sigma},\ \boldsymbol{\tau} \in \boldsymbol{\Sigma}_{k,\boldsymbol{x}}.$$

Notice that $\mathbf{\Lambda}_{k,\boldsymbol{x}}$ is symmetric and $(\mathbf{\Lambda}_{k,\boldsymbol{x}}\boldsymbol{\sigma},\boldsymbol{\sigma}) = 0$ if and only if $\boldsymbol{\sigma} = c\mathbf{I}$, where c is a constant. Since $c\mathbf{I} \notin \mathbf{\Sigma}_{k,\boldsymbol{x}}$ and $\mathbf{\Sigma}_{k,\boldsymbol{x}}$ is a finite dimensional space, it is easy to see that $\mathbf{\Lambda}_{k,\boldsymbol{x}}$ is symmetric positive definite over $\mathbf{\Sigma}_{k,\boldsymbol{x}}$, although the spectrum of $\mathbf{\Lambda}_{k,\boldsymbol{x}}$ may not have a uniform lower bound independent of all k and $\boldsymbol{x}$.

Let $\mathbf{Q}_{k,\boldsymbol{x}} : (\mathbf{\Sigma}_k + span\{\mathbf{I}\}) \to \mathbf{\Sigma}_{k,\boldsymbol{x}}$ and $\mathbf{P}_{k,\boldsymbol{x}} : (\mathbf{\Sigma}_k + span\{\mathbf{I}\}) \to \mathbf{\Sigma}_{k,\boldsymbol{x}}$ be the $\mathbf{L}^2$ projection and the $\mathbf{\Lambda}$ projection, respectively. Here $\mathbf{P}_{k,\boldsymbol{x}}$ is well defined since the operator $\mathbf{\Lambda}_{k,\boldsymbol{x}}$ is symmetric positive definite. Define operator $\tilde{\mathbf{R}}_k : \mathbf{\Sigma}_k \to (\mathbf{\Sigma}_k + span\{\mathbf{I}\})$ using an additive Schwarz scheme:

$$\tilde{\mathbf{R}}_k = \frac{1}{3}\sum_{\boldsymbol{x}\in\mathcal{V}_k} \mathbf{P}_{k,\boldsymbol{x}}\mathbf{\Lambda}_k^{-1} = \frac{1}{3}\sum_{\boldsymbol{x}\in\mathcal{V}_k} \mathbf{\Lambda}_{k,\boldsymbol{x}}^{-1}\mathbf{Q}_{k,\boldsymbol{x}}. \tag{5.3}$$

Define $\mathbf{R}_k : \mathbf{\Sigma}_k \to \mathbf{\Sigma}_k$ by

$$\mathbf{R}_k = \mathbf{Q}_{\mathbf{\Sigma}_k}\tilde{\mathbf{R}}_k, \tag{5.4}$$

where $\mathbf{Q}_{\mathbf{\Sigma}_k} : (\mathbf{\Sigma}_k + span\{\mathbf{I}\}) \to \mathbf{\Sigma}_k$ is the $\mathbf{L}^2$ projection. It is clear that $\mathbf{R}_k$ is symmetric positive definite over $\mathbf{\Sigma}_k$ under the $\mathbf{L}^2$ inner product.

REMARK 2. For simplicity, in this paper we will analyze only the smoother $\mathbf{R}_k$ as defined in (5.4). The analysis can easily be extended to the Hiptmair smoother [19], which is usually more time-efficient in terms of a single sweep of smoothing process, and the same multigrid convergence rate estimate holds.

Next, we show that $\mathbf{R}_k$ satisfies assumptions (M.1) and (M.2). The proof of assumption (M.1) follows directly from the Schwarz inequality and the fact that each triangle in $\mathcal{T}_k$ is covered by exactly three subdomains in $\{\Omega_{k,\boldsymbol{x}}\}_{\boldsymbol{x}\in\mathcal{V}_k}$. In the following, we concentrate on the proof of assumption (M.2).

First, we state the following result.

LEMMA 5.1.
(R.1) $\boldsymbol{\tau}_k \in \mathbf{\Sigma}_k$

$$(\mathbf{I} - \mathbf{P}_{k-1})\boldsymbol{\tau}_k = \sum_{\boldsymbol{x}\in\mathcal{V}_k} \boldsymbol{\tau}_{k,\boldsymbol{x}}, \qquad \boldsymbol{\tau}_{k,\boldsymbol{x}} \in \mathbf{\Sigma}_{k,\boldsymbol{x}},$$

C_0

$$\sum_{\boldsymbol{x}\in\mathcal{V}_k} \mathbf{\Lambda}(\boldsymbol{\tau}_{k,\boldsymbol{x}},\boldsymbol{\tau}_{k,\boldsymbol{x}}) \le C_0\mathbf{\Lambda}((\mathbf{I} - \mathbf{P}_{k-1})\boldsymbol{\tau}_k,\boldsymbol{\tau}_k).$$

(M.2) $C_p = 3C_0$

Assume (R.1) holds. By using the Schwarz inequality, we have

$$\begin{aligned}
\mathbf{\Lambda}((\mathbf{I} - \mathbf{P}_{k-1})\boldsymbol{\tau}_k,\boldsymbol{\tau}_k) &= \sum_{\boldsymbol{x}\in\mathcal{V}_k} \mathbf{\Lambda}(\boldsymbol{\tau}_{k,\boldsymbol{x}},\boldsymbol{\tau}_k) = \sum_{\boldsymbol{x}\in\mathcal{V}_k} \mathbf{\Lambda}(\boldsymbol{\tau}_{k,\boldsymbol{x}},\mathbf{P}_{k,\boldsymbol{x}}\boldsymbol{\tau}_k) \\
&\le \left(\sum_{\boldsymbol{x}\in\mathcal{V}_k} \mathbf{\Lambda}(\boldsymbol{\tau}_{k,\boldsymbol{x}},\boldsymbol{\tau}_{k,\boldsymbol{x}})\right)^{1/2} \mathbf{\Lambda}^{1/2}\left(\sum_{\boldsymbol{x}\in\mathcal{V}_k} \mathbf{P}_{k,\boldsymbol{x}}\boldsymbol{\tau}_k,\boldsymbol{\tau}_k\right) \\
&\le \sqrt{3}C_0^{1/2}\mathbf{\Lambda}^{1/2}((\mathbf{I} - \mathbf{P}_{k-1})\boldsymbol{\tau}_k,\boldsymbol{\tau}_k)\mathbf{\Lambda}^{1/2}(\tilde{\mathbf{R}}_k\mathbf{\Lambda}_k\boldsymbol{\tau}_k,\boldsymbol{\tau}_k) \\
&= \sqrt{3}C_0^{1/2}\mathbf{\Lambda}^{1/2}((\mathbf{I} - \mathbf{P}_{k-1})\boldsymbol{\tau}_k,\boldsymbol{\tau}_k)\mathbf{\Lambda}^{1/2}(\mathbf{R}_k\mathbf{\Lambda}_k\boldsymbol{\tau}_k,\boldsymbol{\tau}_k).
\end{aligned}$$

The last step comes from the fact that $span\{\mathbf{I}\}$ is in the kernel of $\mathbf{\Lambda}_k$. This completes the proof of the lemma. ∎

Finally, we need to prove that condition (R.1) holds.

LEMMA 5.2. [illegible] (R.1) [illegible] $C_0 = c(1+\gamma^{-2})$ [illegible] c [illegible] k [illegible] γ

[illegible] Let $\boldsymbol{\sigma}_k = (\mathbf{I} - \mathbf{P}_{k-1})\boldsymbol{\tau}_k$, where $\boldsymbol{\tau}_k \in \boldsymbol{\Sigma}_k$. We decompose $\boldsymbol{\sigma}_k$ into $\boldsymbol{\sigma}_k = \mathbf{curl}\, \boldsymbol{q}_k + \mathbf{div}_k^{-1}\boldsymbol{v}_k$, where $\boldsymbol{q}_k \in \mathbf{U}_k$ and $\boldsymbol{v}_k = \mathbf{div}\boldsymbol{\sigma}_k$. Denote $\boldsymbol{\sigma}_k^1 = \mathbf{div}_k^{-1}\boldsymbol{v}_k$. Clearly,

$$\boldsymbol{\Lambda}(\mathbf{curl}\, \boldsymbol{q}_k, \mathbf{curl}\, \boldsymbol{q}_k) + \boldsymbol{\Lambda}(\boldsymbol{\sigma}_k^1, \boldsymbol{\sigma}_k^1) = \boldsymbol{\Lambda}(\boldsymbol{\sigma}_k, \boldsymbol{\sigma}_k).$$

Therefore we can consider the decomposition of $\mathbf{curl}\, \boldsymbol{q}_k$ and $\boldsymbol{\sigma}_k^1$ separately.

Next we need a partition of unity. Let $\theta_{\boldsymbol{x}}$ be a continuous piecewise linear function defined on $\mathcal{T}_k$ such that the support of each $\theta_{\boldsymbol{x}}$ is in $\Omega_{k,\boldsymbol{x}}$ and

$$|\theta_{\boldsymbol{x}}|_{W^{j,\infty}(\Omega)} \lesssim h_k^{-j}, \quad j = 0, 1,$$

where $W^{j,\infty}(\Omega)$ is the Sobolev space with index (j, ∞) [14]. This kind of partition of unity exists and has been widely used in the analysis of overlapping Schwarz domain decomposition methods.

Clearly we have the decomposition $\mathbf{curl}\, \boldsymbol{q}_k = \sum_{\boldsymbol{x}\in\mathcal{V}_k} \mathbf{curl}\, (\Pi_{\mathbf{U}_k}(\theta_{\boldsymbol{x}}\boldsymbol{q}_k))$, where $\Pi_{\mathbf{U}_k}$ is the natural interpolation onto $\mathbf{U}_k$ associated with the degrees of freedom. By the standard scaling argument, it is not hard to see that

$$\|\mathbf{curl}\, (\Pi_{\mathbf{U}_k}(\theta_{\boldsymbol{x}}\boldsymbol{q}_k))\|_{0,\Omega} \lesssim \|\mathbf{curl}\, (\theta_{\boldsymbol{x}}\boldsymbol{q}_k)\|_{0,\Omega}.$$

Combining the above, Lemma 4.2, and the fact that $h_k = h_{k-1}/2$, we have

$$\begin{aligned}
&\sum_{\boldsymbol{x}\in\mathcal{V}_k} \boldsymbol{\Lambda}(\mathbf{curl}\, (\Pi_{\mathbf{U}_k}(\theta_{\boldsymbol{x}}\boldsymbol{q}_k)), \mathbf{curl}\, (\Pi_{\mathbf{U}_k}(\theta_{\boldsymbol{x}}\boldsymbol{q}_k))) \\
&\qquad = \sum_{\boldsymbol{x}\in\mathcal{V}_k} (\mathcal{A}\mathbf{curl}\, (\Pi_{\mathbf{U}_k}(\theta_{\boldsymbol{x}}\boldsymbol{q}_k)), \mathbf{curl}\, (\Pi_{\mathbf{U}_k}(\theta_{\boldsymbol{x}}\boldsymbol{q}_k))) \\
&\qquad \le \sum_{\boldsymbol{x}\in\mathcal{V}_k} \|\mathbf{curl}\, (\Pi_{\mathbf{U}_k}(\theta_{\boldsymbol{x}}\boldsymbol{q}_k))\|_{0,\Omega}^2 \le \sum_{\boldsymbol{x}\in\mathcal{V}_k} \|\mathbf{curl}\, (\theta_{\boldsymbol{x}}\boldsymbol{q}_k)\|_{0,\Omega} \\
&\qquad \lesssim \|\mathbf{curl}\, \boldsymbol{q}_k\|_{0,\Omega}^2 + h_k^{-2}\|\boldsymbol{q}_k\|_{0,\Omega}^2 \lesssim (\mathcal{A}\mathbf{curl}\, \boldsymbol{q}_k, \mathbf{curl}\, \boldsymbol{q}_k) \\
&\qquad \le \boldsymbol{\Lambda}(\boldsymbol{\sigma}_k, \boldsymbol{\sigma}_k).
\end{aligned} \tag{5.5}$$

Next, for $\boldsymbol{\sigma}_k^1$, we consider the decomposition $\boldsymbol{\sigma}_k^1 = \sum_{\boldsymbol{x}\in\mathcal{V}_k} \Pi_{\boldsymbol{\Sigma}_k}(\theta_{\boldsymbol{x}}\boldsymbol{\sigma}_k^1)$, where $\Pi_{\boldsymbol{\Sigma}_k}$ is the interpolation operator as defined in (3.3) on the mesh $\mathcal{T}_k$. This decomposition is well defined since $\Pi_{\boldsymbol{\Sigma}_k}$ is a linear operator. By the standard scaling argument, we have

$$\|\Pi_{\boldsymbol{\Sigma}_k}(\theta_{\boldsymbol{x}}\boldsymbol{\sigma}_k^1)\|_{0,\Omega} \lesssim \|\theta_{\boldsymbol{x}}\boldsymbol{\sigma}_k^1\|_{0,\Omega}.$$

Therefore, by Lemma 4.2 and the fact that $2h_k = h_{k-1}$,

$$\begin{aligned}
&\sum_{\boldsymbol{x}\in\mathcal{V}_k} \boldsymbol{\Lambda}(\Pi_{\boldsymbol{\Sigma}_k}(\theta_{\boldsymbol{x}}\boldsymbol{\sigma}_k^1), \Pi_{\boldsymbol{\Sigma}_k}(\theta_{\boldsymbol{x}}\boldsymbol{\sigma}_k^1)) \\
&\qquad \lesssim \sum_{\boldsymbol{x}\in\mathcal{V}_k} (\|\Pi_{\boldsymbol{\Sigma}_k}(\theta_{\boldsymbol{x}}\boldsymbol{\sigma}_k^1)\|_{0,\Omega}^2 + \gamma\|\mathbf{div}(\Pi_{\boldsymbol{\Sigma}_k}(\theta_{\boldsymbol{x}}\boldsymbol{\sigma}_k^1))\|_{0,\Omega}^2) \\
&\qquad \lesssim \|\boldsymbol{\sigma}_k^1\|_{0,\Omega}^2 + \gamma(\|\mathbf{div}\boldsymbol{\sigma}_k^1\|_{0,\Omega}^2 + h_k^{-2}\|\boldsymbol{\sigma}_k^1\|_{0,\Omega}^2) \\
&\qquad \lesssim (1+\gamma^{-2})\boldsymbol{\Lambda}(\boldsymbol{\sigma}_k, \boldsymbol{\sigma}_k).
\end{aligned} \tag{5.6}$$

Finally, we define the decomposition

$$\boldsymbol{\sigma}_k = \sum_{\boldsymbol{x}\in\mathcal{V}_k} (\mathbf{curl}\,(\Pi_{\mathbf{U}_k}(\theta_{\boldsymbol{x}}\boldsymbol{q}_k)) + \Pi_{\boldsymbol{\Sigma}_k}(\theta_{\boldsymbol{x}}\boldsymbol{\sigma}_k^1)).$$

The stability of this decomposition comes from (5.5) and (5.6). This completes the proof of the lemma. □

By inequality (5.2) and Lemmas 5.1 and 5.2, the convergence rate of the multigrid solver is $O(\frac{1+\gamma^{-2}}{1+\gamma^{-2}+m})$.

6. Numerical results. We report some numerical results for the multigrid solver for the $\boldsymbol{H}(\mathbf{div})$-type problem (5.1). Let Ω be the unit square $(0,1)\times(0,1)$. The multigrid solver with a smoother as defined in section 5 is used. The coarsest mesh is as shown in Figure 6.1. We solve problem (5.1) on the 2nd–5th multigrid levels. The right-hand side $\boldsymbol{F} \in \boldsymbol{\Sigma}_h$ is selected randomly.

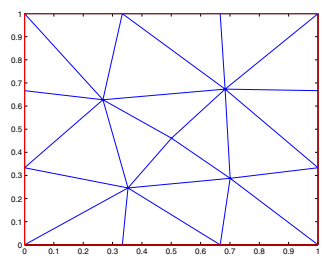


FIG. 6.1. *The coarsest mesh.*

Both the lowest order RT element and the lowest order BDM element are used in the discretization. The experiments are done for $\gamma = 10^{-4}$, 10^{-2}, 1, 10^2, 10^4. We tested the iteration numbers needed for reaching a relative residual 10^{-6}. The results are listed in Tables 6.1 and 6.2.

TABLE 6.1
Iteration numbers using the RT element.

Level	$\gamma = 10^{-4}$	$\gamma = 10^{-2}$	$\gamma = 1$	$\gamma = 10^2$	$\gamma = 10^4$
2	22	13	11	11	11
3	62	18	14	14	14
4	132	19	16	15	16
5	200	19	16	16	16

TABLE 6.2
Iteration numbers using the BDM element.

Level	$\gamma = 10^{-4}$	$\gamma = 10^{-2}$	$\gamma = 1$	$\gamma = 10^2$	$\gamma = 10^4$
2	6	6	6	6	6
3	7	7	7	7	7
4	7	7	8	7	7
5	8	7	7	7	7

From the tables we can see that when γ is large, the iteration numbers seem to be independent of γ and the number of levels, which agrees with the theory. When γ is

small, the theory suggests that the convergence rate deteriorates. From the numerical results we can see that this is true for the RT element. However, for the BDM element, the convergence rate still seems to be independent of γ.

Appendix. Proof of (4.6).

As stated before, we assume that Ω is a convex polygon. Let $\boldsymbol{\sigma}_h$ and $\boldsymbol{q}_h$ be defined as in Lemma 4.2. Then $\boldsymbol{q}_h$ is in $\boldsymbol{H}^1(\Omega,\mathbb{R}^2)/\mathbb{R}^2$ and for all $\boldsymbol{p}_H \in \tilde{\mathbf{U}}_H$, we have

$$(\mathcal{A}\mathbf{curl}\,\boldsymbol{q}_h, \mathbf{curl}\,\boldsymbol{p}_H) = \boldsymbol{\Lambda}(\mathbf{curl}\,\boldsymbol{q}_h, \mathbf{curl}\,\boldsymbol{p}_H) = \boldsymbol{\Lambda}(\boldsymbol{\sigma}_h, \mathbf{curl}\,\boldsymbol{p}_H) = 0. \tag{A.1}$$

Denote $\boldsymbol{n}$ and $\boldsymbol{t}$ to be the unit outer normal vector and the unit tangential vector along $\partial\Omega$. Define the curl operator for all $\boldsymbol{\tau} \in \boldsymbol{H}^1(\Omega,\mathbb{M}_2)$ by

$$\operatorname{curl}\boldsymbol{\tau} = \begin{pmatrix} -\frac{\partial \tau_{11}}{\partial y} + \frac{\partial \tau_{12}}{\partial x} \\ -\frac{\partial \tau_{21}}{\partial y} + \frac{\partial \tau_{22}}{\partial x} \end{pmatrix}.$$

First we notice that the following lemma holds.

LEMMA A.1. [illegible] $\boldsymbol{\sigma} \in \boldsymbol{H}^1(\Omega,\mathbb{M}_2)$ [illegible] $\boldsymbol{q}_h = \operatorname{curl}\boldsymbol{\sigma}$ [illegible]

(S1) $\mathcal{A}\boldsymbol{\sigma} = \boldsymbol{\sigma}$ [illegible] $\boldsymbol{\sigma}$ [illegible]

(S2) $(\boldsymbol{\sigma}\boldsymbol{t})|_{\partial\Omega} = \mathbf{0}$;

(S3) $\|\boldsymbol{\sigma}\|_{1,\Omega} \lesssim \|\boldsymbol{q}_h\|_{0,\Omega}$

[illegible] (4.6) [illegible]

Define $\boldsymbol{\sigma}^1$ by

$$\begin{cases} (\mathcal{A}\boldsymbol{\sigma}^1, \boldsymbol{\tau}) + (\mathbf{div}\boldsymbol{\tau}, \boldsymbol{u}) = 0 & \text{for all } \boldsymbol{\tau} \in \boldsymbol{\Sigma}, \\ (\mathbf{div}\boldsymbol{\sigma}^1, \boldsymbol{v}) = (\mathbf{div}\boldsymbol{\sigma}, \boldsymbol{v}) & \text{for all } \boldsymbol{v} \in \mathbf{V}. \end{cases}$$

The regularity result (2.3) implies $\|\boldsymbol{\sigma}^1\|_{1,\Omega} \lesssim \|\mathbf{div}\boldsymbol{\sigma}\|_{0,\Omega}$. Since $\boldsymbol{\sigma} - \boldsymbol{\sigma}^1 \in \boldsymbol{H}^1(\Omega,\mathbb{M}_2)$ and $\mathbf{div}(\boldsymbol{\sigma} - \boldsymbol{\sigma}^1) = \mathbf{0}$, there must exist a potential function $\boldsymbol{p} \in \boldsymbol{H}^2(\Omega,\mathbb{R}^2)$ such that $\mathbf{curl}\,\boldsymbol{p} = \boldsymbol{\sigma} - \boldsymbol{\sigma}^1$ and

$$|\boldsymbol{p}|_{2,\Omega} \le \|\boldsymbol{\sigma}\|_{1,\Omega} + \|\boldsymbol{\sigma}^1\|_{1,\Omega} \lesssim \|\boldsymbol{\sigma}\|_{1,\Omega} \lesssim \|\boldsymbol{q}_h\|_{0,\Omega}.$$

Let $\boldsymbol{p}_H$ be the natural interpolation of $\boldsymbol{p}$ onto $\tilde{\mathbf{U}}_H$. Then by the assumptions on $\boldsymbol{\sigma}$, (A.1), and integration by parts,

$$\begin{aligned} \|\boldsymbol{q}_h\|_{0,\Omega}^2 &= (\operatorname{curl}\boldsymbol{\sigma}, \boldsymbol{q}_h) = -(\boldsymbol{\sigma}, \mathbf{curl}\,\boldsymbol{q}_h) = -(\mathcal{A}\boldsymbol{\sigma}, \mathbf{curl}\,\boldsymbol{q}_h) \\ &= -(\mathcal{A}\mathbf{curl}\,\boldsymbol{p}, \mathbf{curl}\,\boldsymbol{q}_h) = -(\mathcal{A}\mathbf{curl}\,(\boldsymbol{p} - \boldsymbol{p}_H), \mathbf{curl}\,\boldsymbol{q}_h) \\ &\lesssim H|\boldsymbol{p}|_{2,\Omega}\|\mathcal{A}\mathbf{curl}\,\boldsymbol{q}_h\|_{0,\Omega} \lesssim H\|\boldsymbol{q}_h\|_{0,\Omega}\|\mathcal{A}\mathbf{curl}\,\boldsymbol{q}_h\|_{0,\Omega}. \end{aligned}$$

This completes the proof of (4.6). ∎

In the remainder of this section, we will construct a $\boldsymbol{\sigma}$ that satisfies the assumptions in Lemma A.1.

For any scalar function f, define $\mathbf{curl}\,f = (-\frac{\partial f}{\partial y}, \frac{\partial f}{\partial x})^t$. Then there exists an $\mathbf{L}^2$-orthogonal Helmholtz decomposition (see [16]) $\boldsymbol{q}_h = \mathbf{curl}\,\phi + \nabla\psi$, where $\phi \in H_0^1(\Omega)$ and $\psi \in H^1(\Omega)/\mathbb{R}$ satisfy $\|\phi\|_{1,\Omega} \le \|\boldsymbol{q}_h\|_{0,\Omega}$ and $\|\psi\|_{1,\Omega} \le \|\boldsymbol{q}_h\|_{0,\Omega}$. Then

$$\boldsymbol{q}_h = \operatorname{curl}\begin{pmatrix} \phi & \psi \\ -\psi & \phi \end{pmatrix} = \operatorname{curl}\begin{pmatrix} \phi & (\psi + C) \\ -(\psi + C) & \phi \end{pmatrix} + \operatorname{curl}\nabla\boldsymbol{u}$$

for all $C \in \mathbb{R}$ and $\boldsymbol{u} \in \boldsymbol{H}^2(\Omega, \mathbb{R}^2)$. Define

$$\boldsymbol{\sigma} = \begin{pmatrix} \phi & (\psi + C) \\ -(\psi + C) & \phi \end{pmatrix} + \nabla \boldsymbol{u}. \tag{A.2}$$

Clearly $\boldsymbol{\sigma} \in \boldsymbol{H}^1(\Omega, \mathbb{M}_2)$, and we need only find C and $\boldsymbol{u}$ such that $\boldsymbol{\sigma}$ satisfies (S1), (S2), and (S3) in Lemma A.1, which can be translated to

$$\begin{aligned} \operatorname{div} \boldsymbol{u} &= -2\phi, & \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{t}}|_{\partial\Omega} &= (\psi + C)\boldsymbol{n}|_{\partial\Omega}, \\ |C| &\lesssim \|\boldsymbol{q}_h\|_{0,\Omega}, & \|\boldsymbol{u}\|_{2,\Omega} &\lesssim \|\boldsymbol{q}_h\|_{0,\Omega}. \end{aligned} \tag{A.3}$$

Since $\boldsymbol{q}_h \in \boldsymbol{H}^1(\Omega, \mathbb{R}^2)/\mathbb{R}^2$ and $\phi \in H_0^1(\Omega)$, we have

$$\begin{aligned} \int_{\partial\Omega} \psi \boldsymbol{n}\, ds &= \int_\Omega \nabla\psi\, d\boldsymbol{x} = \int_\Omega (\boldsymbol{q}_h - \mathbf{curl}\,\phi)\, d\boldsymbol{x} \\ &= -\int_\Omega \mathbf{curl}\,\phi\, d\boldsymbol{x} = \int_{\partial\Omega} \phi \boldsymbol{t}\, ds = \boldsymbol{0}. \end{aligned} \tag{A.4}$$

Recall that Ω is a convex polygon. Denote $\boldsymbol{z}_i$, $i = 1, \ldots, N$, to be the corners of Ω. To close the polygon, we set $\boldsymbol{z}_0 = \boldsymbol{z}_N$ or $\boldsymbol{z}_{N+1} = \boldsymbol{z}_1$. Denote Γ_i to be the edge of Ω connecting $\boldsymbol{z}_i$ and $\boldsymbol{z}_{i+1}$. Define $\boldsymbol{n}_i = (n_{1,i}, n_{2,i})^t$ and $\boldsymbol{t}_i = (n_{2,i}, -n_{1,i})^t$ to be the unit outer normal vector and the unit tangential vector on Γ_i, respectively. According to the trace theorem on polygons (see Theorem 1.4.6 in [17]), we have

$$\int_0^\varepsilon s^{-1} \left| -\psi(\boldsymbol{z}_i + s\boldsymbol{t}_i) + \psi(\boldsymbol{z}_i - s\boldsymbol{t}_{i-1}) \right|^2 ds \le \infty \tag{A.5}$$

for all $\varepsilon < min_{i=1,\ldots,N} |z_i - z_{i-1}|$.

Let s_0, s be points on $\partial\Omega$ and $\int_{s_0}^s dt$ be the line integral from s_0 to s along $\partial\Omega$ in counterclockwise direction. Define

$$C = \frac{-\int_\Omega 2\phi\, d\boldsymbol{x} - \int_{\partial\Omega} (\int_{s_0}^s \psi \boldsymbol{n}\, dt) \cdot \boldsymbol{n}\, ds}{\int_{\partial\Omega} (\int_{s_0}^s \boldsymbol{n}\, dt) \cdot \boldsymbol{n}\, ds} \tag{A.6}$$

and

$$\boldsymbol{g}(s) = \int_{s_0}^s (\psi + C)\boldsymbol{n}\, dt. \tag{A.7}$$

By (A.4) and Green's formula, we immediately see that $\boldsymbol{g}$ is continuous on $\partial\Omega$. Furthermore, we clearly have $\boldsymbol{g}|_{\Gamma_i} \in \boldsymbol{H}^{3/2}(\Gamma_i, \mathbb{R}^2)$ and

$$\begin{aligned} |C| &\lesssim \|\phi\|_{0,\Omega} + \|\psi\|_{0,\partial\Omega} \lesssim \|\phi\|_{0,\Omega} + \|\psi\|_{1,\Omega} \lesssim \|\boldsymbol{q}_h\|_{0,\Omega}, \\ \|\boldsymbol{g}\|_{3/2,\Gamma_i} &\lesssim \|\psi\|_{1/2,\partial\Omega} + |C| \lesssim \|\phi\|_{0,\Omega} + \|\psi\|_{1,\Omega} \lesssim \|\boldsymbol{q}_h\|_{0,\Omega}. \end{aligned}$$

By (A.5), (A.6), and (A.7), it is not hard to see that $\int_\Omega -2\phi\, d\boldsymbol{x} = \int_{\partial\Omega} \boldsymbol{g} \cdot \boldsymbol{n}\, ds$ and

$$\begin{aligned} &\int_0^\varepsilon s^{-1} \left| \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{t}}(z_i + s\boldsymbol{t}_i) \cdot \boldsymbol{n}_{i-1} - \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{t}}(z_i - s\boldsymbol{t}_{i-1}) \cdot \boldsymbol{n}_i \right|^2 ds \\ &= \int_0^\varepsilon s^{-1} \left| (\psi + C)(z_i + s\boldsymbol{t}_i) - (\psi + C)(z_i - s\boldsymbol{t}_{i-1}) \right|^2 (\boldsymbol{n}_{i-1} \cdot \boldsymbol{n}_i)^2\, ds \\ &< \infty \end{aligned}$$

for all $\varepsilon < \min_{i=1,\ldots,N}|z_i - z_{i-1}|$. Recall that $\phi \in H_0^1(\Omega)$. Combining all the above and according to Theorem 7.1 in [4], there exists $\boldsymbol{u} \in \boldsymbol{H}^2(\Omega, \mathbb{R}^2)$ such that $\operatorname{div} \boldsymbol{u} = -2\phi$, $\boldsymbol{u}|_{\partial\Omega} = \boldsymbol{g}$, and

$$\|\boldsymbol{u}\|_{2,\Omega} \lesssim \|\phi\|_{1,\Omega} + \sum_{i=1,\ldots,N} \|\boldsymbol{g}\|_{3/2,\Gamma_i} \lesssim \|\phi\|_{1,\Omega} + \|\psi\|_{1,\Omega} \lesssim \|\boldsymbol{q}_h\|_{0,\Omega}.$$

Clearly $\boldsymbol{u}$ defined above and C defined in (A.6) satisfy (A.3), and hence $\boldsymbol{\sigma}$ defined as in (A.2) satisfies (S1), (S2), and (S3). This completes the proof for (4.6). ∎

REFERENCES

[1] D. N. Arnold, R. S. Falk, and R. Winther, *Preconditioning in H*(div) *and applications*, Math. Comp., 66 (1997), pp. 957–984.

[2] D. N. Arnold, R. S. Falk, and R. Winther, *Multigrid in H*(div) *and H*(curl), Numer. Math., 85 (2000), pp. 197–217.

[3] D. N. Arnold, Jr., J. Douglas, and C. P. Gupta, *A family of higher order mixed finite element methods for plane elasticity*, Numer. Math., 45 (1984), pp. 1–22.

[4] D. N. Arnold, L. R. Scott, and M. Vogelius, *Regular inversion of the divergence operator with Dirichlet boundary conditions on a polygon*, Ann. Scuola Norm. Sup. Pisa Cl. Sci. (4), 15 (1988), pp. 169–192.

[5] D. N. Arnold and R. Winther, *Mixed finite elements for elasticity*, Numer. Math., 92 (2002), pp. 401–419.

[6] F. P. T. Baaijens, *Mixed finite element methods for viscoelastic flow analysis: A review*, J. Non-Newtonian Fluid Mech., 79 (1998), pp. 361–385.

[7] M. Behr, L. P. France, and T. E. Tezduyar, *Stabilized finite element methods for the velocity-pressure-stress formulation of incompressible flows*, Comput. Methods Appl. Mech. Engrg., 104 (1993), pp. 31–48.

[8] J. H. Bramble, *Multigrid Methods*, Pitman Res. Notes Math. Ser. 294, Longman Scientific & Technical, Harlow, UK, 1993.

[9] F. Brezzi, J. Douglas, and L. D. Marini, *Two families of mixed finite elements for second order elliptic problems*, Numer. Math., 47 (1985), pp. 217–235.

[10] F. Brezzi and M. Fortin, *Mixed and Hybrid Finite Element Methods*, Springer-Verlag, New York, 1991.

[11] Z. Cai, B. Lee, and P. Wang, *Least-squares methods for incompressible Newtonian fluid flow: Linear stationary problems*, SIAM J. Numer. Anal., 42 (2004), pp. 843–859.

[12] Z. Cai and G. Starke, *First-order system least squares for the stress-displacement formulation: Linear elasticity*, SIAM J. Numer. Anal., 41 (2003), pp. 715–730.

[13] Z. Cai, C. Tong, P. Vassilevski, and C. Wang, *Mixed Finite Element Methods for Incompressible Flow: Stationary Stokes Equations*, manuscript, 2005.

[14] P. G. Ciarlet, *The Finite Element Method for Elliptic Problems*, North–Holland, Amsterdam, 1978.

[15] M. I. Gerritsma and T. N. Phillips, *Compatible spectral approximations for the velocity-pressure-stress formulation of the Stokes problem*, SIAM J. Sci. Comput., 20 (1999), pp. 1530–1550.

[16] V. Girault and P. A. Raviart, *Finite Element Methods for Navier–Stokes Equations*, Springer Ser. Comput. Math. 5, Springer-Verlag, New York, 1986.

[17] P. Grisvard, *Singularities in Boundary Value Problems*, Research in Applied Mathematics 22, Springer-Verlag, Berlin, 1992.

[18] W. Hackbusch, *Multigrid Methods and Applications*, Springer-Verlag, Berlin, 1985.

[19] R. Hiptmair, *Multigrid method for H*(div) *in three dimensions*, Electron. Trans. Numer. Anal., 6 (1997), pp. 133–152.

[20] C. Johnson and B. Mercier, *Some equilibrium finite element methods for two-dimensional elasticity problems*, Numer. Math., 30 (1978), pp. 103–116.

[21] R. Keunings, *Advances in the computer modeling of the flow of polymeric liquids*, Comp. Fluid Dyn. J., 9 (2001), pp. 449–458.

[22] J. Mandel, S. McCormick, and R. Bank, *Variational multigrid theory*, in Multigrid Methods, S. F. McCormick, ed., SIAM, Philadelphia, 1987, pp. 131–177.

[23] P. A. Raviart and J. M. Thomas, *A mixed finite element method for* 2*nd order elliptic problems*, in Mathematical Aspects of Finite Element Methods, Lecture Notes in Math. 606, I. Galligani and E. Magenes, eds., Springer-Verlag, 1977, pp. 292–315.

[24] P. S. Vassilevski and J. Wang, *Multilevel iterative methods for mixed finite element discretizations of elliptic problems*, Numer. Math., 63 (1992), pp. 503–520.

[25] J. Xu, *Iterative methods by space decomposition and subspace correction*, SIAM Rev., 34 (1992), pp. 581–613.

SIAM J. SCI. COMPUT.
Vol. 29, No. 5, pp. 2096–2114
© 2007 Society for Industrial and Applied Mathematics

PARALLEL DOMAIN DECOMPOSITION METHODS FOR STOCHASTIC ELLIPTIC EQUATIONS*

CHAO JIN†, XIAO-CHUAN CAI‡, AND CONGMING LI†

Abstract. We present parallel Schwarz-type domain decomposition preconditioned recycling Krylov subspace methods for the numerical solution of stochastic elliptic problems, whose coefficients are assumed to be a random field with finite variance. Karhunen–Loève (KL) expansion and double orthogonal polynomials are used to reformulate the stochastic elliptic problem into a large number of related but uncoupled deterministic equations. The key to an efficient algorithm lies in "recycling computed subspaces." Based on a careful analysis of the KL expansion, we propose and test a grouping algorithm that tells us when to recycle and when to recompute some components of the expensive computation. We show theoretically and experimentally that the Schwarz preconditioned recycling GMRES method is optimal for the entire family of linear systems. A fully parallel implementation is provided, and scalability results are reported in the paper.

Key words. stochastic elliptic equations, domain decomposition, recycling Krylov subspace method, parallel scalability

AMS subject classifications. 35R60, 60H15, 60H35, 65C30, 47B80, 65N55, 65M55, 65M60, 65M70

DOI. 10.1137/060662381

1. Introduction.

Many physical phenomena are modeled by partial differential equations whose coefficients are measured through experiments which often contain some levels of uncertainty. To understand the model accurately, the randomness must be considered in the differential equations. In the last decade, many researchers have studied the so-called stochastic partial differential equation (SPDE). One of the approaches for solving SPDEs numerically is known as the stochastic Galerkin method; see, for example, [1, 2, 11, 12, 17, 22, 25, 30] and references therein. The method expands the random field in the equation. Several such expansions are available; for example, [1, 2, 9, 13, 20] use the Karhunen–Loève (KL) expansion introduced in [21], [17, 29, 30] use the Wiener chaos expansion [31], and [29] uses the generalized chaos expansion. With the expansion, the SPDE is reduced to a high-dimensional deterministic equation, which can be solved by different numerical methods. Based on the solution, the statistics of the physical solution of the original SPDE, such as the mean and the variance, can be derived.

In this paper, we focus on elliptic problems with stochastic diffusion coefficients. For this type of SPDEs, we use the double orthogonal basis [3, 16] to decouple the high-dimensional equation in the probability space and produce a sequence of independent equations. After the discretization, we obtain a large number of linear systems:

$$A_i x_i = b_i, \quad i = 1, 2, \ldots, \tag{1.1}$$

*Received by the editors June 7, 2006; accepted for publication (in revised form) April 2, 2007; published electronically September 28, 2007. This research is supported in part by the National Science Foundation, CCR-0219190, ACI-0305666, DMS-0401174, and DMS-0530901, and in part by the Department of Energy, DE-FC02-01ER25479.

http://www.siam.org/journals/sisc/29-5/66238.html

†Department of Applied Mathematics, University of Colorado at Boulder, Boulder, CO 80309 (chao.jin@colorado.edu, cli@colorado.edu).

‡Department of Computer Science, University of Colorado at Boulder, Boulder, CO 80309 (cai@cs.colorado.edu).

where the matrices A_i and right-hand sides b_i are closely related but independent from each other. There are a number of techniques for solving a sequence of related linear systems of equations. In this paper, we focus on the recently introduced recycling Krylov subspace method [24], which retains a Krylov subspace generated while solving a previous system with GMRES and uses it to reduce the cost of solving the current system. In our implementation, we extend the original algorithm in [24] to the flexible version of GMRES, which allows the change of preconditioner during the iteration.

Mathematically we prove, under certain assumptions, that all these linear systems from the KL expansion are spectrally equivalent; however, we also show numerically that it is generally not a good idea to recycle or reuse the same Krylov subspace and/or preconditioner for the entire sequence of linear systems, since some of the spectrally equivalent systems are quite different. Based on some analysis of the KL expansion and the lower bounds of the elliptic coefficients, we propose a grouping algorithm which partitions the large number of systems into several groups, and our tests show that within each group, recycling Krylov subspaces and recycling preconditioners can be safely and efficiently applied.

For preconditioning, we use an overlapping additive Schwarz domain decomposition method [27]. Our parallel implementation is based on the portable extensible toolkit for scientific computation (PETSc) package from Argonne National Laboratory [4]. The scalability and parallel performance of the algorithm are studied theoretically and experimentally.

The rest of the paper is organized as follows. In section 2, we describe the stochastic Galerkin method including the stochastic weak formulation, the KL expansion, the double orthogonal basis, and the discretization. Section 3 presents the additive Schwarz preconditioned recycling Krylov subspace method. A grouping algorithm for efficiently recycling the Krylov subspace and the preconditioner is also introduced. Some experimental results are reported in section 4, and some concluding remarks are given in section 5.

2. Stochastic Galerkin method. In this section, we briefly describe how to transform a stochastic elliptic partial differential equation to a sequence of independent deterministic partial differential equations. We use an elliptic equation with a stochastic diffusion coefficient as a model problem. See [16] for a complete description of the methodology.

2.1. The stochastic weak formulation. We begin with a brief review of notation. Given a probability space $(\Omega, \mathcal{A}, \mathcal{P})$ with sample space Ω, σ-algebra $\mathcal{A}$, and probability measure $\mathcal{P}$, a real-valued random variable is a function $\xi(\omega)$: $\Omega \to \mathbf{R}$. The probability distribution measure of ξ is defined on the Borel set $\mathcal{B}$ as $\mu(\mathcal{B}) = \mathcal{P}(\xi^{-\infty}(\mathcal{B}))$. The mean, or expected value of $\xi(\omega)$, is

$$\langle \xi \rangle = \int_\Omega \xi(\omega) d\mathcal{P}(\omega) = \int_{\mathbf{R}} z d\mu(z) = \int_{\mathbf{R}} z\rho(z) dz, \tag{2.1}$$

where ρ is the probability density function of ξ. We also define the space

$$L^2(\Omega) = \left\{ \xi(\omega) \,\middle|\, \int_\Omega |\xi|^2 d\mathcal{P}(\omega) < \infty \right\}.$$

Let $D \subset R^2$ be the domain of the variable x. A random field $a(x, \omega)$: $D \times \Omega \to \mathbf{R}$ is a real-valued function jointly measurable with respect to the Lebesgue measure on

D and the probability measure $\mathcal{P}$ on Ω. Define the space

$$L^2(D \times \Omega) = \{u(x,\omega) |\ \langle \|u(x,\omega)\|_{L^2(D)} \rangle < \infty\}.$$

The stochastic Sobolev space $H_0^1(D \times \Omega)$ is defined analogously. Now recall that for the classical deterministic two-dimensional elliptic equation

$$\begin{cases} -\nabla \cdot (a(x)\nabla u(x)) = f(x) & \text{in } D, \\ u(x) = 0 & \text{on } \partial D, \end{cases} \tag{2.2}$$

the weak form of the problem is to find $u(x) \in H_0^1(D)$ such that

$$B[u,v] = (f,v) \quad \forall v \in H_0^1(D), \tag{2.3}$$

where

$$B[u,v] = \int_D a(x)\nabla u(x) \cdot \nabla v(x) dx, \qquad (f,v) = \int_D f(x)v(x)dx.$$

If the diffusion coefficient is a random field $a(x,\omega) \in L^2(D \times \Omega)$, so is the solution u. Consequently, we have the stochastic elliptic equation

$$\begin{cases} -\nabla \cdot (a(x,\omega)\nabla u(x,\omega)) = f(x), & x \in D,\ \ \omega \in \Omega, \\ u(x,\omega) = 0, & x \in \partial D,\ \ \omega \in \Omega. \end{cases} \tag{2.4}$$

The weak form of (2.4) is to find $u(x,\omega) \in H_0^1(D \times \Omega)$ such that

$$\langle B[u,v] \rangle = \langle (f,v) \rangle \quad \forall v \in H_0^1(D \times \Omega). \tag{2.5}$$

We assume that $a(x,\omega) \in L^\infty(D \times \Omega)$ is strictly positive, with lower and upper bounds α and β, respectively,

$$0 < \alpha \le a(x,\omega) \le \beta. \tag{2.6}$$

Under this assumption, the existence and uniqueness of a solution u to (2.4) follow from the Lax–Milgram lemma. Note that we assume the source term f is deterministic. This condition can be relaxed to include randomness.

2.2. The KL expansion. Here, we use the KL expansion [21] of the random field $a(x,\omega)$ to separate the deterministic and stochastic components. We assume that the mean and the covariance of $a(x,\omega)$ are known, respectively, as

$$\begin{aligned} a_0(x) &= \int_\Omega a(x,\omega) dP(\omega) \quad \text{and} \\ C_a(x,x') &= \int_\Omega (a(x,\omega) - a_0(x))(a(x',\omega) - a_0(x')) dP(\omega). \end{aligned} \tag{2.7}$$

By the KL expansion, $a(x,\omega)$ can be represented in the form of a series as follows:

$$a(x,\omega) = a_0(x) + \sum_{j=1}^{\infty} \sqrt{\lambda_j}\, k_j(x)\, y_j(\omega), \tag{2.8}$$

where λ_j and $k_j(x)$ are the eigenvalues and orthogonal eigenfunctions of $C_a(x,x')$; i.e.,

$$\int_D C_a(x,x') k_j(x') d(x') = \lambda_j k_j(x). \tag{2.9}$$

This series converges in the mean-square sense.

By definition, $C_a(x,x')$ is symmetric and positive semidefinite. This implies that there exists a countable sequence of eigenpairs $\{(\lambda_j,k_j)\}$, where the eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq \cdots \to 0$ are nonnegative and the eigenfunctions $\{k_j(x)\}$ are orthogonal in $L^2(D)$. Moreover, $\{y_j\}$ is a set of uncorrelated random variables with mean value zero. If the eigenfunctions are normalized, y_j all have unit variance; i.e., $\langle y_j(\omega)\rangle = 0$, $\langle y_i(\omega)y_j(\omega)\rangle = \delta_{ij}$. For the computation, we approximate $a(x,\omega)$ by a truncation of (2.8),

$$a_M(x,\omega) = a_0(x) + \sum_{j=1}^{M} \sqrt{\lambda_j}\; k_j(x)\; y_j(\omega),$$

where M denotes the number of terms in the truncation.

In this paper, we also assume that the $\{y_j\}_{j=1}^M$ are independent. The probability density function of y_j is denoted as ρ_j. The joint probability density function of $y = (y_1,\ldots,y_M)$ is $\rho = \rho_1 \times \cdots \times \rho_M$. Let Γ_j denote the image of y_j and $\Gamma = \Gamma_1 \times \cdots \times \Gamma_M$. We treat a_M: $D \times \Gamma \to R$ as

$$a_M(x,y) = a_0(x) + \sum_{j=1}^{M} \sqrt{\lambda_j}\; k_j(x)\; y_j(\omega). \tag{2.10}$$

Now, we arrive at the following deterministic elliptic problem:

$$-\nabla(a_M(x,y)\nabla u_M(x,y)) = f(x), \qquad u_M(x,y) \text{ in } H_0^1(D) \times L^2(\Gamma,\rho). \tag{2.11}$$

2.3. Double orthogonal basis. For $y \in \Gamma$, we use the double orthogonal polynomial function space [3, 16] to approximate $L^2(\Gamma,\rho)$, which decouples the equation in the y-space, yielding a sequence of uncoupled equations. The double orthogonal basis is constructed as follows. For any $r \in \mathbf{N}$, the space of single-variable polynomials of degree at most r is

$$P_r := \operatorname{span}\{1, t, t^2, \ldots, t^r\}. \tag{2.12}$$

For $\mathbf{r} = (r_1, r_2, \ldots, r_M) \in \mathbf{N}^M$ we construct the multivariable polynomial space

$$P_{\mathbf{r}} := P_{r_1} \otimes P_{r_2} \otimes \cdots \otimes P_{r_M} \in L^2(\Gamma,\rho). \tag{2.13}$$

For the space $P_{r_j}, j = 1,2,\ldots,M$, we use the double orthogonal functions, denoted as $\{\psi_{k,j}(t)\}_{k=0}^{r_j}$, as the basis instead of $\{1,t,t^2,\ldots,t^{r_j}\}$. We require that $\psi_{k,j}(t)$, $k = 0,\ldots,r_j$, satisfy two orthogonality conditions:

$$\begin{cases} \displaystyle\int_{\Gamma_j} \psi_{p,j}(t)\psi_{q,j}(t)\rho_j(t)dt = \delta_{p,q}, \quad p,q = 0,\ldots,r_j, \\ \displaystyle\int_{\Gamma_j} t\psi_{p,j}(t)\psi_{q,j}(t)\rho_j(t)dt = C_{p,j}\delta_{p,q}, \quad p,q = 0,\ldots,r_j, \end{cases} \tag{2.14}$$

where $\{C_{p,j}\}_{p=0}^{r_j}$ are nonzero constants. Next we construct a basis function of $P_{\mathbf{r}}$ by selecting one polynomial basis function from each $P_{r_j}, j = 1,\ldots,M$, and then multiply these selected M basis functions together. So given an $\mathbf{r} = (r_1, r_2, \ldots, r_m) \in \mathbf{N}^M$, there are in total $N_y = \prod_{j=1}^{M}(r_j+1)$ basis functions for $P_{\mathbf{r}}(y_1, y_2, \ldots, y_M)$.

Let $\mathbf{i} = \{i_1, i_2, \ldots, i_M\}$. If $0 \le i_j \le r_j \ \forall 1 \le j \le M$, we say that $\mathbf{i} \le \mathbf{r}$. It is obvious that there are N_y multi-indices $\mathbf{i}$, which are less than or equal to $\mathbf{r}$. Each $\mathbf{i}$ corresponds to one basis function of $P_{\mathbf{r}}$. We denote all the basis functions of $P_{\mathbf{r}}$ as the set

$$\left\{ \psi_{\mathbf{i}}(y) \,\middle|\, \psi_{\mathbf{i}}(y) = \prod_{j=1}^{m} \psi_{i_j,j}(y_j), \ \ i_j \in \{0, 1, \ldots, r_j\} \right\}_{\mathbf{i} \le \mathbf{r}}. \tag{2.15}$$

Finding $\{\psi_{k,\,j}(y_j)\}_{k=0}^{r_j}$ for spaces P_{r_j}, $j = 1, \ldots, M$, results in an eigenproblem (cf. section 8.7.2 in [18]). For the probability space of y, generally we do not need high order polynomials. So the computational work for these eigenproblems is negligible compared with the cost required to solve the coupled equations.

2.4. Discretization. We use a Galerkin method with a double orthogonal basis for the discretization of (2.11) in the y-space. Let

$$u_M(x, y) = \sum_{\mathbf{i} \le \mathbf{r}} u_{\mathbf{i}}(x) \psi_{\mathbf{i}}(y)$$

such that for any $v(x, y) = h(x)\psi_{\mathbf{j}}(y) \in H_0^1(D) \times P_{\mathbf{r}}(\Gamma)$ we have

$$\left\langle \int_D a_M(x, y) \nabla u_M(x, y) \cdot \nabla v(x, y) dx \right\rangle = \left\langle \int_D f(x) v(x, y) dx \right\rangle. \tag{2.16}$$

Writing in an explicit form,

$$\begin{aligned} &\left\langle \int_D a_M(x, y) \nabla u_M(x, y) \cdot \nabla v(x, y) dx \right\rangle \\ &= \sum_{j=1}^{M} \sum_{\mathbf{i} \le \mathbf{r}} \sqrt{\lambda_j} \left(k_j(x) \nabla u_{\mathbf{i}}(x), \nabla h(x) \right) \int_\Gamma y_j \psi_{\mathbf{i}}(y) \psi_{\mathbf{j}}(y) \rho(y) dy. \end{aligned}$$

The two orthogonality conditions (2.14) imply that

$$\int_\Gamma y_j \psi_{\mathbf{i}}(y) \psi_{\mathbf{j}}(y) \rho(y) dy = C_{i_j,j} \delta_{\mathbf{i},\,\mathbf{j}}.$$

Consequently,

$$\begin{aligned} &\left\langle \int_D a_M(x, y) \nabla u_M(x, y) \cdot \nabla v(x, y) dx \right\rangle \\ &= \sum_{j=1}^{M} \sum_{\mathbf{i} \le \mathbf{r}} \sqrt{\lambda_j} \left(k_j(x) \nabla u_{\mathbf{i}}(x), \ \ \nabla h(x) \right) C_{i_j,j} \, \delta_{\mathbf{i},\mathbf{j}}. \end{aligned}$$

On the other hand,

$$\left\langle \int_D f(x) v(x, y) dx \right\rangle = \int_D f(x) h(x) dx \langle \psi_{\mathbf{j}}(y) \rangle.$$

Now it is easy to see that the variational form of (2.11),

$$\langle B[u_M, v] \rangle = \langle (f, v) \rangle,$$

implies that

$$-\nabla\left(a_{M,\mathbf{i}}(x)\nabla u_{M,\mathbf{i}}(x)\right) = f_{\mathbf{i}}(x) \qquad \text{in } H_0^1(D), \text{ with} \tag{2.17}$$

$$\begin{cases} a_{M,\mathbf{i}}(x) := a_0(x) + \displaystyle\sum_{j=1}^{M} \sqrt{\lambda_j}\, k_j(x) C_{i_j,j}, \\ f_{\mathbf{i}}(x) := f(x) \cdot \displaystyle\prod_{j=1}^{M} \int_{\Gamma_j} \psi_{i_j,j}(t)\rho_j(t)dt. \end{cases} \tag{2.18}$$

Thus, we have decoupled (2.11) into N_y deterministic diffusion problems (2.17) and (2.18) in D. The stochastic solution is

$$u_M(x,y) = \sum_{\mathbf{i}\le\mathbf{r}} u_{M,\mathbf{i}}(x)\psi_{\mathbf{i}}(y). \tag{2.19}$$

We discretize the left-hand side of (2.17) and obtain a sequence of matrices $A_i, i = 1,\ldots,N_y$. It can be shown that these systems are positive definite if M is large enough. The following result is useful.

LEMMA 2.1. [illegible] $D \times D$ [illegible] $y_j, j = 1,\ldots,M$ [illegible] $\mathbf{i}$

$$\frac{\alpha}{2} \le a_{M,\mathbf{i}}(x) \le 2\beta$$

[illegible] M [illegible] α [illegible] β [illegible] (2.6) [illegible]. By Proposition 4.3 in [16], $\alpha/2 \le a_M(x,y) \le 2\beta$ for any x, y. The constant $C_{i_j,j}$ in (2.18) and (2.14) is bounded by $|y_j|$ in (2.10):

$$|C_{i_j,j}| \le \int_{\Gamma_j} |y_j|\psi_{i_j j}(y_j)^2 dy_j \le \max|y_j| \int_{\Gamma_j} \psi_{i_j,j}(y_j)^2 dy_j = \max|y_j|.$$

This implies that $a_{M,\mathbf{i}}$ are also bounded in $[\alpha/2, 2\beta]$. ∎

The statistics of the solution can be found from the approximate solutions. For example, the mean of $u(x,y)$ can be approximated by

$$\langle u_M\rangle = \sum_{\mathbf{i}\le\mathbf{r}} u_{M,\mathbf{i}}(x) \int_\Gamma \psi_{\mathbf{i}}(y)\rho(y)dy = \sum_{\mathbf{i}\le\mathbf{r}} u_{M,\mathbf{i}}(x) \prod_{j=1}^{M} \int_{\Gamma_j} \psi_{i_j,j}(t)\rho_j(t)dt.$$

3. Numerical method. After the discretization of (2.18) we obtain a sequence of independent linear systems of the form (1.1). In this section we propose an additive Schwarz preconditioned recycling Krylov subspace method for solving these systems. The recycling Krylov subspace method, called generalized conjugate residual with implicit inner orthogonalization and deflated restarting (GCRO-DR) [24], reduces some of the work in constructing the Krylov subspace for a new system, and the additive Schwarz algorithm [27] provides a preconditioner and a data structure for a parallel implementation. Note that we also reuse the preconditioner to further save the computational time.

3.1. Recycling Krylov subspace method. There are several recycling Krylov subspace methods; see, for example, [8, 14, 15, 19, 23, 26] and references therein. In our case, which has a sequence of linear systems with changing left-hand side matrices and right-hand side vectors, we use GCRO-DR. Here, we present only the main steps of this method. For a detailed description and derivation of the algorithm, we refer the reader to [24].

The idea of GCRO-DR is to retain a Krylov subspace for subsequent restarted GMRES cycles, or for solving other linear systems. For generality, we extend the GMRES framework introduced in [24] to include the flexible version of GMRES (FGMRES). Suppose we have solved the ith system with FGMRES. We retain k vectors

$$\tilde{Y}_k = [\tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_k]. \tag{3.1}$$

Let $C_k = Q$, $U_k = \tilde{Y}_k R^{-1}$, where Q and R are from the reduced QR decomposition of $A_{i+1}\tilde{Y}_k$. U_k, $C_k \in R^{n\times k}$ satisfy

$$A_{i+1}U_k = C_k, \quad C_k^H C_k = I_k. \tag{3.2}$$

Let x_0 and r_0 be the initial guess and initial residual of $A_{i+1}x_{i+1} = b_{i+1}$. We update the solution as $x = x_0 + U_k C_k^H r_0$, and set $r = r_0 - C_k C_k^H r_0$. In our numerical experiment, we find that for some systems this updated solution x already satisfies the error tolerance, so that we do not need any more iterations after the initial step.

However, if the x is still not good enough, we continue to generate a Krylov subspace of dimension $m - k + 1$ with $(I - C_k C_k^H)A_{i+1}$, where m is the maximum number of iterations before restarting. In the case of recycling FGMRES, it produces the Arnoldi relation

$$(I - C_k C_k^H)A_{i+1}Z_{m-k} = V_{m-k+1}\bar{H}_{m-k}, \tag{3.3}$$

where the columns of Z_{m-k} are the preconditioned orthogonal vectors V_{m-k}. In the case of GMRES, Z_{m-k} should be replaced by V_{m-k} everywhere in the algorithm.

Each of the Arnoldi vectors $V_{m-k+1} = [v_1, v_2, \ldots, v_{m-k+1}]$ is orthogonal to the range of C_k. We can rewrite (3.3) as

$$A_{i+1}[U_k \;\; Z_{m-k}] = [C_k \;\; V_{m-k+1}] \begin{bmatrix} I_k & B_{m-k} \\ 0 & \bar{H}_{m-k} \end{bmatrix}, \tag{3.4}$$

where $B_{m-k} = C_k^H A_{i+1} Z_{m-k}$. For numerical reasons, we normalize the column vectors of U_k and replace the identity matrix I_k above with a diagonal matrix D_k such that $U_k D_k$ has unit columns denoted as $\tilde{U}_k$. We define

$$\hat{V}_m = [\tilde{U}_k \;\; Z_{m-k}], \qquad \hat{W}_{m+1} = [C_k \;\; V_{m-k+1}], \qquad \bar{G}_m = \begin{bmatrix} D_k & B_{m-k} \\ 0 & \bar{H}_{m-k} \end{bmatrix} \tag{3.5}$$

and write (3.4) more compactly as

$$A_{i+1}\hat{V}_m = \hat{W}_{m+1}\bar{G}_m.$$

Note that $\bar{G}_m = \hat{W}_{m+1}^H A_{i+1}\hat{V}_m$ is upper Hessenberg with diagonal D_k. The columns of $\hat{W}_{m+1}$ are orthogonal, but this is not true for the columns of $\hat{V}_m$.

In each cycle, we need to find y to minimize

$$\begin{aligned}\|r - A_{i+1}\hat{V}_m y\|_2 &= \|r - \hat{W}_{m+1}\bar{G}_m y\|_2 \\ &= \|\hat{W}_{m+1}^H r - \bar{G}_m y\|_2 \\ &= \|e_{k+1}\|r\|_2 - \bar{G}_m y\|_2.\end{aligned}$$

Once we have y, the residual and solution are updated by

$$r := r - A_{i+1}\hat{V}_m y = r - \hat{W}_{m+1}\bar{G}_m y \quad \text{and} \quad x := x + \hat{V}_m y. \tag{3.6}$$

Next, we solve the generalized eigenvalue problem

$$\bar{G}_m^H \bar{G}_m z_i = \theta_i \bar{G}_m^H \hat{W}_{m+1}\hat{V}_m z_i. \tag{3.7}$$

The k vectors $\tilde{Y}_k$ are chosen from $\tilde{y}_i = \hat{V}_m z_i, i = 1, 2, \ldots, m$. Theoretically, any k eigenvectors z_i can be used to compute $\tilde{Y}_k$. In our implementation, we choose the eigenvectors corresponding to the eigenvalues of smallest magnitude [24]. For more details in computing the recycling Krylov subspace between cycles and systems, we refer to [24].

3.2. Additive Schwarz preconditioning. In this section, we show that all matrices in the sequence of linear systems are spectrally equivalent, and therefore an optimal preconditioner for any one of the matrices is also optimal for the whole sequence. Here "optimal" means that the number of iterations is independent of the mesh size and the number of subdomains. Note that optimal does not mean that the number of iterations is small or the computing time is small, which is of course more important. In the next section, we will introduce a method that partitions the sequence of linear systems into groups, and within each group we apply the preconditioning techniques to be discussed briefly below.

We start with a coarse triangulation $\mathcal{T}^H = \{\tau_i^H\}$ and a fine triangulation $\mathcal{T}^h = \{\tau_i\}$ of the domain D. Let $\mathcal{N}(\mathcal{T}^H)$ be the set of nodal points of $\mathcal{T}^H$. Let $h_i =$ diameter of τ_i, and $h = \max_i\{h_i\}$. We assume that the coarse and fine triangulation are shape regular in the sense common to finite element theory. We then partition D into nonoverlapping subdomains $\{D_i\}$, $i = 1, \ldots, N$. Let $H_i =$ diameter of D_i, $H = \max_i\{H_i\}$. We extend each D_i to a larger region $D_i^{'} \subset D$. We assume that the distance between the boundaries ∂D_i and $\partial D_i^{'}$ inside D is bounded from below by a fixed fraction of H_i, and that $\partial D_i^{'}$ does not cut through any elements.

Let V^h and V_0 be the finite element spaces of continuous piecewise linear functions associated with the triangulations $\mathcal{T}^h$ and $\mathcal{T}^H$, respectively. Let $V_i = \{v \in V^h : v(x) = 0,\ x \notin D_i^{'}\}$, $i \geq 1$. Our finite element space V^h can be represented as

$$V^h = I_H^h V_0 + V_1 + \cdots + V_N,$$

where I_H^h is a coarse to fine mesh interpolation operator [7]. We approximate $H_0^1(D)$ by V^h and define the finite element problem of (2.17) as: Find $u_h \in V^h$ such that

$$\int_D a_{M,\mathbf{i}}(x)\nabla u_h \nabla v_h dx = \int_D f_{\mathbf{i}} v_h dx \quad \forall\ v \in V^h. \tag{3.8}$$

For simplicity, we denote $a_{M,\mathbf{i}}(x)$ and $f_{\mathbf{i}}$ for system i, where $i = 1, 2, \ldots, N_y$, as $a_i(x)$ and $f_i(x)$, respectively. $a_0(x)$ is still the mean of $a(x)$. We define the bilinear form, for any $u, v \in H_0^1(D)$,

$$A(u, v) = \int_D a_1(x)\nabla u \nabla v dx,$$

$$B_i(u,v) = \int_D a_i(x)\nabla u \nabla v dx, \quad i = 1, 2, \ldots, N_y.$$

We now define the projection operators, which are the main building blocks of the additive Schwarz methods.

DEFINITION 3.1. P_0: $V^h \longrightarrow V_0$

$$B_i(P_0 u, v) = B_i(u, v) \quad \forall\, v \in V_0.$$

$i = 1, 2, \ldots, N$ P_i: $V^h \longrightarrow V_i$

$$A(P_i u, v) = B_i(u, v) \quad \forall\, v \in V_i.$$

Note that for subdomain problems we define the preconditioners using a bilinear form defined by a system usually different from the one being solved. But for the coarse problem, the preconditioner is defined by using the bilinear form of the same linear system. If we view the additive Schwarz method (ASM) as a preconditioner, the matrix form of P_i is

$$P_i = R^T K_i^{-1} R K, \tag{3.9}$$

where K_i is the stiffness matrix from system 1 associated with subdomain $D_i^{'}$ and K is the stiffness matrix from system i, $i = 1, 2, \ldots, N_y$, associated with D. R is a restriction matrix. The additive Schwarz operator is defined by

$$P = \sum_{i=0}^{N} P_i. \tag{3.10}$$

Using the standard techniques [27, 28], we easily show the following result.

THEOREM 3.2. c_p C_p h H $u \in V^h$

$$c_p A(u,u) \le A(u, Pu), \quad A(Pu, Pu) \le C_p A(u,u). \tag{3.11}$$

According to [5, 6, 7, 10], the convergence rate of the additive Schwarz preconditioned GMRES is bounded by the ratio C_p/c_p, and the theorem implies that the convergence rate is independent of the mesh size h and the number of subdomains N.

3.3. Some analysis of the sequence of linear systems and a grouping algorithm. In this section we provide some analysis of the linear systems. Although mathematically we can prove that all of the matrices are spectrally equivalent, in practice some of the systems are quite different from the others, and the same preconditioner may not work well for all the systems. Similarly, simple recycling of the Krylov subspace from the previous system may not work well for these "bad" systems. Therefore it is very important to identify these bad systems and exclude them from the rest of the systems that can be solved with the recycling of Krylov subspaces and preconditioners.

It is easy to see that the matrices depend on the diffusion coefficients $a_{M,\mathbf{i}}(x), \mathbf{i} \le \mathbf{r}$. From (2.18), it is obvious that the differences among these coefficients $a_{M,\mathbf{i}}(x)$ are determined by the eigenvalues, the eigenfunctions, and the constants $C_{i_j,j}$. If any of them change drastically, the diffusion coefficient changes drastically.

We note that the eigenfunctions are normalized in (2.18) and the eigenvalues $\lambda_j, j = 1, 2, \ldots, M$, are not increasing and converge to zero as j goes to infinity.

Therefore, the changes of the constants $C_{i_j,j}, j = 1, 2, \ldots, M$, play the main role in the perturbation of $a_{M,\mathbf{i}}$ for the terms corresponding to larger values of λ_j, i.e., when j is small. Based on the eigenvalues and the constants $C_{i_j,j}$, we provide an estimate of when the diffusion coefficients $a_{M,\mathbf{i}}$ have a relatively large change. If the constants $C_{i_j,j}$, corresponding to large eigenvalues, change drastically, the corresponding systems may become very different from the previous systems. In this case the recycled Krylov subspace and the preconditioner should not be used.

Following the index of the diffusion coefficient $a_{M,\mathbf{i}}(x)$, the sequence of systems is currently labeled by the M-dimensional index $\mathbf{i}$, which is not too convenient for the purpose of putting the systems into different groups. We next turn the M-dimensional index $\mathbf{i}$ into a one-dimensional array. Given $\mathbf{r} = \{r_1, r_2, \ldots, r_M\}$, the index set $\{\mathbf{i} = \{i_1, i_2, \ldots, i_M\} \le \mathbf{r}\}$ is ordered as follows.

Reordering scheme for the multi-index i.

```
k = 1;
for i_1 = 0, 1, ..., r_1
   for i_2 = 0, 1, ..., r_2
      ...
         for i_M = 0, 1, ..., r_M
            "system k" := "system i," with i = {i_1, i_2, ..., i_M};
            k = k + 1;
end
```

An important characteristic of this ordering is that i_k changes only after all possible ordering of $\{i_{k+1}, \ldots, i_M\}$. Following this one-dimensional reindex of $\mathbf{i}$, the set of basis functions $\psi_{\mathbf{i}}$ has a new index.

As it is known that there is a decay of the eigenvalues, consequently the first few terms in the KL expansion carry a lot more weight in perturbing the diffusion coefficient. Using the eigenvalue information and the values of $C_{i_j,j}$, we derive a heuristic algorithm that puts the matrices into several groups.

We first use the mean value of the diffusion coefficient to determine which eigenvalues should be considered to carry more weight. For this purpose, we choose a parameter

$$\theta = \frac{1}{3} \max_{x \in \mathcal{N}(\mathcal{T}^H)} \{a_0(x)\}, \tag{3.12}$$

where $\mathcal{N}(\mathcal{T}^H)$ is the set of coarse mesh nodes of the domain D. If an eigenvalue is greater than θ, we consider that the change in the term associated with this eigenvalue causes large change in the diffusion coefficient. Assume, by this criterion, that we have decided that the first p terms may cause large perturbation. Then the sequence of linear systems is divided into $(r_1+1) \times (r_2+1) \times \cdots \times (r_p+1)$ groups. In each group, there are $(r_{p+1}+1) \times \cdots \times (r_M+1)$ systems. The systems in different groups have different constants $C_{i_j,j}, j = 1, 2, \ldots, p$, in the first p terms of the KL expansion. The systems in the same group have the same first p terms of the KL expansion.

In most situations, the systems in the same group have small differences such that we can use one of the systems to construct a Krylov subspace and a preconditioner that can be reused for solving systems in the whole group. However, in some cases, a subset of systems in the group may be close to being singular, and this subgroup of systems needs to be treated separately. More precisely, for certain systems in a given group of systems, we note that when the minimum value of the diffusion coefficient is close to zero, these systems are more sensitive to small perturbations.

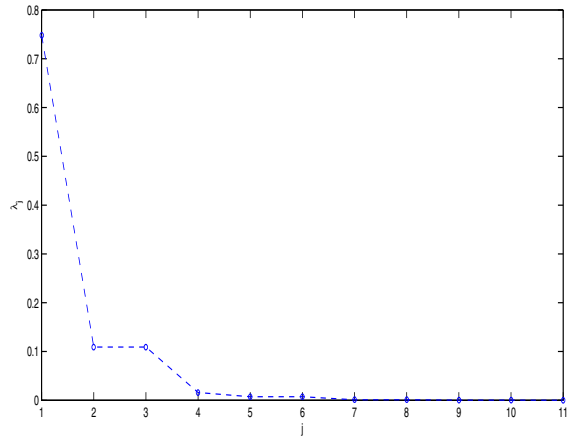


FIG. 3.1. *The decay of eigenvalues when* $C_a(x, x') = e^{-|x-x'|^2}, x \in [0,1]^2$.

The iterative solver may not converge well even when the preconditioner is constructed from another system in the same group. To single out these bad systems, we choose a cutoff parameter $\delta > 0$, and if

$$\min_{x \in \mathcal{N}(\mathcal{T}^H)} \{a_i(x)\} \leq \delta, \tag{3.13}$$

we consider the system as nearly singular. In our numerical experiments, $\delta = 0.03$. In the heuristic estimate, there is no need to compute the minimum value too accurately, and we use a coarse mesh to find the minimums of the diffusion coefficients. If the minimum is less than δ, we consider this system too sensitive to be preconditioned by any other systems that are not close to being singular. We collect these sensitive systems together into a subgroup. Experiences show that systems in the subgroup have similar lower bounds, and therefore we can solve them by recycling the Krylov subspace, recycling the symbolic factorizations of the subdomain matrices, but not the preconditioner, within the subgroup. We note that these sparse matrices have the same nonzero pattern, and by using this fact, certain data structures and communication information can be recycled.

Furthermore, based on the eigenfunctions and the values of $C_{i_j,j}, j = 1, 2, \ldots, M$, we may narrow down the interval to which the sensitive systems belong without searching through every group of systems.

To help the discussion, we consider an example with the mean and covariance functions

$$a_0(x) = 3 + \sin(\pi x_1), \qquad C_a(x, x') = e^{-|x-x'|^2}, \quad x \in [0,1]^2. \tag{3.14}$$

$M = 11$ and $\mathbf{r} = (3, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1)$. This implies that there are 9216 systems of form (2.18). We assume that y_j in the KL expansion is uniformly distributed in $\Gamma_j = [-\sqrt{3}, \sqrt{3}]$.

In Figure 3.1 we show the decay of the eigenvalues. With the above ordering, for the example (3.14), Figure 3.2 illustrates how the upper and lower bounds of the diffusion coefficient $a_{M,\mathbf{i}}$ change with $\mathbf{i}$. Note that some of the lower bounds are quite close to zero.

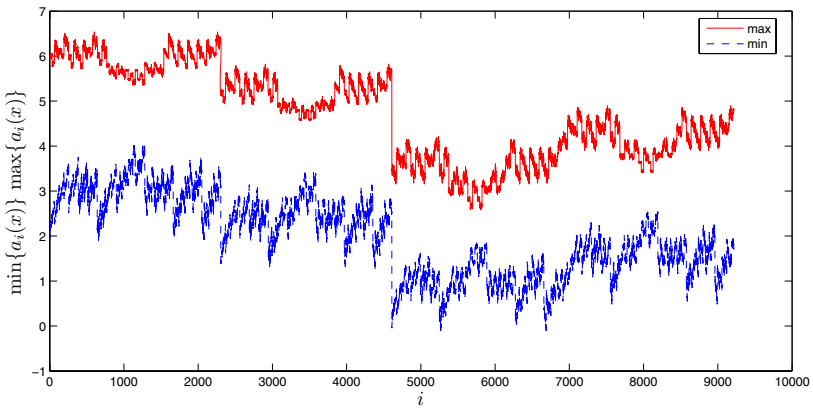


FIG. 3.2. *Top curve is the maximum of the diffusion coefficient* $\max_{x\in\mathcal{N}(\mathcal{T}^H)}\{a_{M,\mathbf{i}}(x)\}$, *and the bottom curve is the minimum of the diffusion coefficient* $\min_{x\in\mathcal{N}(\mathcal{T}^H)}\{a_{M,\mathbf{i}}(x)\}$, $M = 11$, $\mathbf{r} = \{3, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1\}$, $a_0(x) = 3 + \sin(\pi x)$.

Since λ_1 is dominantly large compared with other eigenvalues, we consider that the first term has more weight in perturbing the diffusion coefficient. $r_1 = 3$ implies there are four possible corresponding constants: $C_{i_1,1}, i_1 = 0, 1, 2, 3$. The four constants $C_{i_1,1}, i = 0, 1, 2, 3$, evenly divide all the systems into four groups. According to the reordering scheme for the multi-index $\mathbf{i}$, the first group includes systems 1 to 2304, in which all the systems' diffusion coefficients share the constant $C_{i_1,1}, i_1 = 0$, in the first term of the KL expansion. Similarly, the second group from 2305 to 4608 corresponds to $C_{i_1,1}, i_1 = 1$; the third group corresponds to $C_{i_1,1}, i_1 = 2$; and the last group corresponds to $C_{i_1,1}, i_1 = 3$. We can see clearly that the bounds of the diffusion coefficient have a dramatic change for systems 2305, 4609, 6913. This matches with the change of $C_{i_1,1}, i_1 = 0, 1, 2, 3$, which are $1.49, 0.59, -1.49, -0.59$, respectively, by numerical computation. $C_{i_j,j}$ decreases from 1.49 to 0.59 and then to -1.49. In the end, it increases to -0.59. Correspondingly, the bounds of the last group in Figure 3.2 increase too compared with the third group. In particular, since the third constant -1.49 is the smallest value among the first four constants, the minimum values of some diffusion coefficients, which are close to zero, are all in the third group 4609–6912. These systems are the bad cases, for which the iterative solver does not converge well if using a preconditioner not in the same subgroup.

Figure 3.3 shows the surface plots of the diffusion coefficients of systems 4608 to 4613. We can see clearly that the plot for system 4608 is significantly different from those of the other systems. The plots for systems 4609 to 4613 have very small differences. In particular, the minimum values of the diffusion coefficients of systems 4609 to 4613 are close to zero.

Generally speaking, we would like to recycle both the Krylov subspace and the preconditioner as much as possible. For the preconditioner, there are two ways to recycle, namely, (1) recycle the symbolic factorization of the subdomain matrices to save the computational work in the construction of the preconditioners for the next system; and (2) recycle the preconditioner for the next system. Obviously, the second choice saves more work than the first. However, when the matrices have significant differences from each other, the number of iterations can be large if we reuse the old

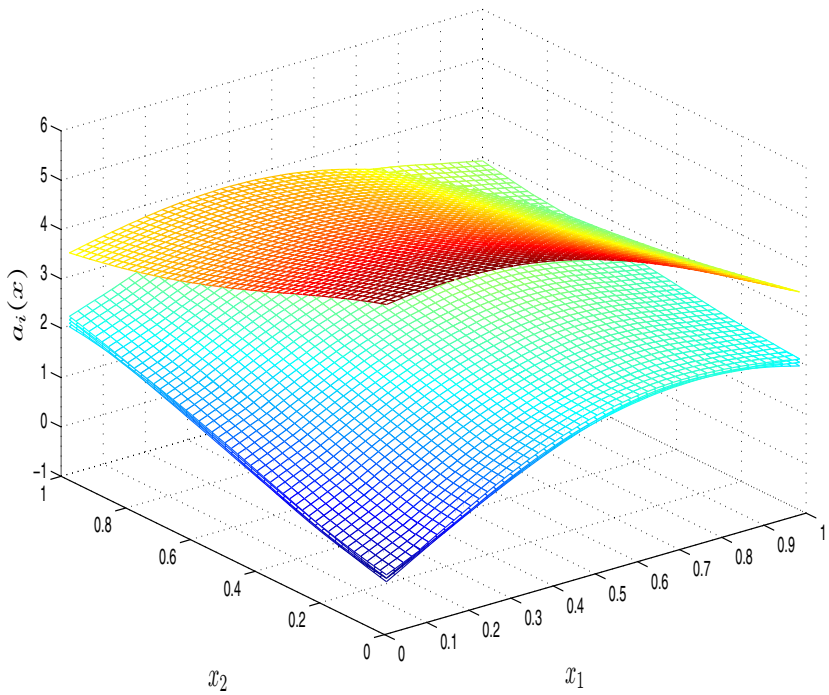


FIG. 3.3. *Surface plots of diffusion coefficients of systems* 4608–4613. *The top surface is for system* 4608, *and the bottom cluster of surfaces is for systems* 4609–4613.

preconditioner, and therefore a new preconditioner is preferred.

We summarize our analysis in term of a grouping algorithm which provides an estimate of when the systems have relatively large change and when the systems are close to being singular.

A GROUPING ALGORITHM.

- Choose M and $\mathbf{r} = \{r_1, r_2, \ldots, r_M\}$. Generate a new index for $\mathbf{i} \leq \mathbf{r}$ according to the reordering scheme. Compute the double orthogonal basis and obtain the corresponding constants $C_{i_j,j}, j = 1, 2, \ldots, M$. Compute the eigenvalues λ_j and eigenfunctions k_j, $j = 1, 2, \ldots, M$. Choose the parameters $\delta > 0$ and $\theta > 0$. (In our example we choose $\theta = 1/3 \max_{x \in \mathcal{N}(\mathcal{T}^H)}\{a_0(x)\}$ and $\delta = 0.03$.)
- Compute $p = \max\{i \mid |\lambda_i| > \theta,\ i = 1, 2, \ldots, M\}$. Divide the systems into $(r_1 + 1) \times \cdots \times (r_p + 1)$ groups. In each group, there are $q = (r_{p+1} + 1) \times \cdots \times (r_M + 1)$ systems.
- Compute the approximate minimum value of each diffusion coefficient on the coarse mesh, $\min_{x \in \mathcal{N}(\mathcal{T}^H)}\{a_i(x)\}$. If the minimum is less than or equal to δ, the corresponding system is labeled as nearly singular, or a "bad case."
- Put these bad cases into a subgroup. Construct a Krylov subspace and a preconditioner from one system in the subgroup and then recycle the Krylov subspace and recycle the symbolic factorization of the subdomain matrices for other systems in the subgroup.

- For the other systems in each group, construct a Krylov subspace and a preconditioner from one system and then recycle the Krylov subspace and recycle the preconditioner for other systems in this group.

4. Numerical experiments. We test the performance of the additive Schwarz preconditioned recycling FGMRES algorithm for solving the model problem (2.4) on a two-dimensional domain $D = [0,1]^2$. The source term $f(x) = 1$. The mean and covariance functions are explicitly given as

$$a_0(x) = 3 + \sin(\pi x_1), \qquad C_a(x,x') = e^{-|x-x'|^2}.$$

For the KL expansion of $a(x,\omega)$, we choose $M = 11$ truncation terms as the approximation. This implies that the dimension for the y-space is 11. By the selecting algorithm provided in [16], we choose $\mathbf{r} = (3,2,2,1,1,1,1,1,1,1,1)$. Consequently, we have 9216 systems of form (2.18). At the same time, we assume that y_j, $j = 1,2,\ldots,11$, are uniformly distributed in $\Gamma_j = [-\sqrt{3},\sqrt{3}]$. Thus the probability density function $\rho_j = 1/(2\sqrt{3})$. Note that this implies that the variance of y_j is unity. We compute the eigenpairs $(\lambda_j, k_j(x))$ using MATLAB. $k_j(x)$ is saved as a step function together with eigenvalues into a file to be used in the main program. The computation of the orthogonal polynomials (2.14) involves the generalized eigenvalue problem $Ax = \lambda Bx$ with symmetric and positive semidefinite matrices A, B of size $r_j, j = 1,2,\ldots,11$. These are solved with MATLAB and saved in a file to be read by the main program.

The main program implements the additive Schwarz preconditioned recycling FGMRES. PETSc [4] is employed for the parallel computation. Both the relative tolerance and absolute tolerance for all the tests are set to 10^{-6}.

4.1. Comparison of different recycling schemes. In this section, we study the following four schemes:

1. No recycling of Krylov subspace and no recycling of preconditioner.
2. Recycle Krylov subspace and recycle symbolic factorizations of subdomain matrices for the entire sequence of systems.
3. Recycle Krylov subspace and recycle the preconditioner for the entire sequence of systems.
4. An algorithm based on the grouping algorithm introduced in the previous section.

Scheme 1 is used as a basis of comparison for the other schemes. To take advantage of the fact that all the matrices have the same nonzero pattern, we recycle the symbolic factorization of subdomain matrices and call this Scheme 2. In PETSc this ensures that certain data structures and communication information are reused (instead of being regenerated) during successive steps, thereby increasing the efficiency of the software. For example, for a parallel preconditioner such as incomplete factorization, matrix colorings and communication patterns are determined only once and then reused throughout the solution process. Scheme 3 uses the same preconditioner for all systems. It saves almost all the preconditioner construction work for all the matrices. However, this causes a steep increase in iteration numbers for some of the systems. We can see in Figures 4.1, 4.2, 4.3, and 4.4 that, for Scheme 2, all the iteration numbers of the following systems are about half of the iteration of the first system. But for Scheme 3, there are sudden increases of iteration numbers for some systems. The reason was discussed in the previous section. Scheme 4 combines the advantages of Schemes 2 and 3. It adaptively recycles the Krylov subspace and the preconditioner according to the grouping algorithm.

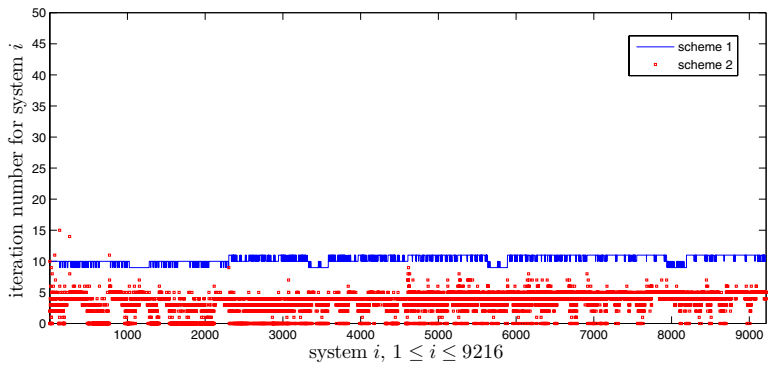


FIG. 4.1. *One-level ASM, iteration number comparison of schemes* 1 *and* 2.

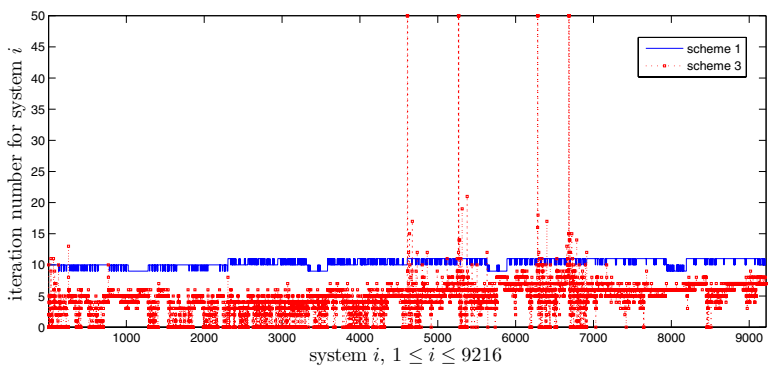


FIG. 4.2. *One-level ASM, iteration number comparison of schemes* 1 *and* 3.

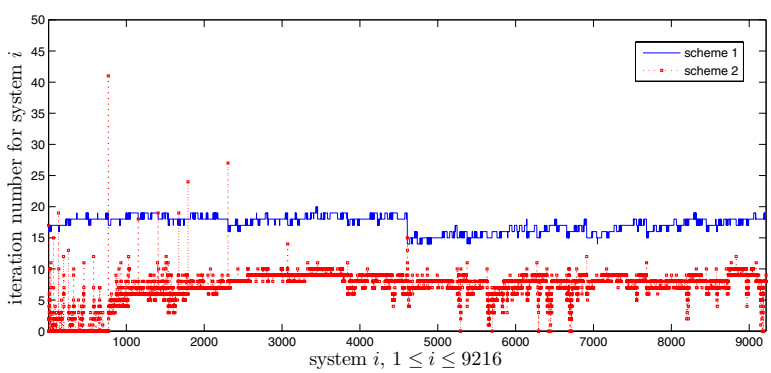


FIG. 4.3. *Two-level ASM, iteration number comparison of schemes* 1 *and* 2.

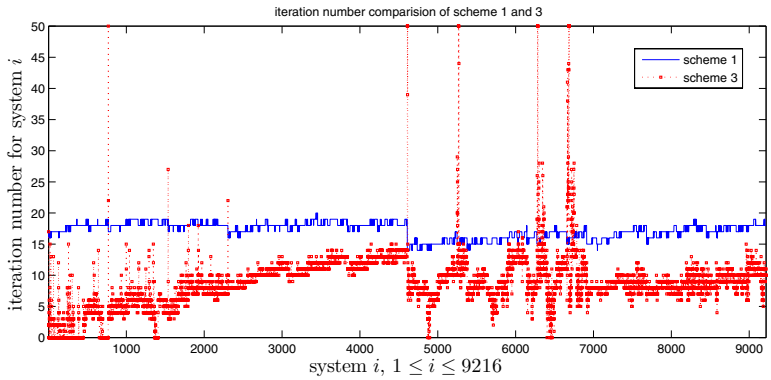


FIG. 4.4. *Two-level ASM, iteration number comparison of schemes* 1 *and* 3.

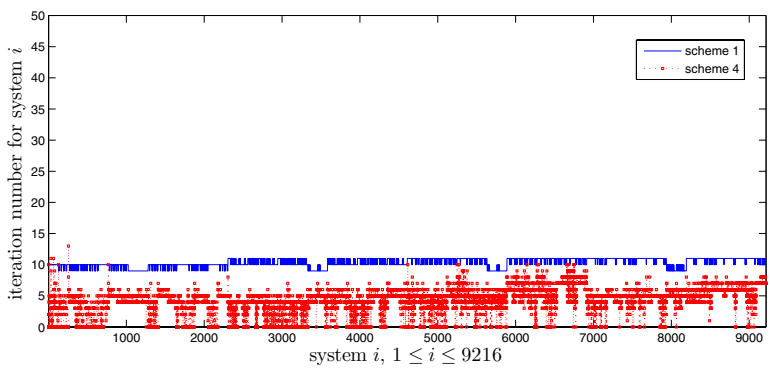


FIG. 4.5. *One-level ASM, iteration number comparison of schemes* 1 *and* 4.

TABLE 4.1
Running time for different schemes and preconditioning (seconds).

Preconditioner	Scheme			
	1	2	3	4
One-level ASM	12030	7693	4205	3882
Two-level ASM	20980	14130	10740	8476

We first study the one-level additive Schwarz method, i.e., without the coarse level preconditioning. The mesh size is 256×256. The number of processors is 16. Figures 4.1, 4.2, and 4.5 show the comparisons. The running times for both one-level and two-level ASM are in Table 4.1.

From Figure 4.1, we see that most of the iteration numbers of Scheme 2 are less than half of those obtained using Scheme 1. Many systems even have zero iterations, which means that the solution generated from the recycled Krylov subspace is already good enough for the current system. This leads to big savings in the running time. In Figure 4.2, we use the same preconditioner for all systems. The iteration numbers increase a little for most systems, but still less than those of Scheme 1. In a few cases,

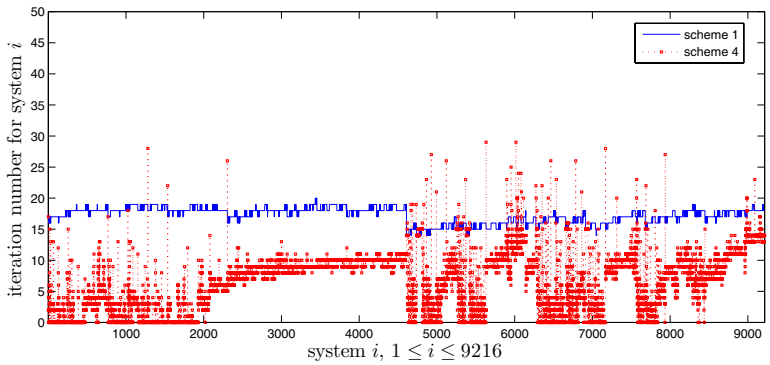


FIG. 4.6. *Two-level ASM, iteration number comparison of schemes* 1 *and* 4.

TABLE 4.2
Average iteration numbers, one-level additive Schwarz preconditioner.

Mesh	Overlap	Number of subdomains			
		1	4	16	64
128×128	4	5.89	5.81	5.83	6.83
256×256	8	3.79	5.52	4.28	7.76
512×512	16	2.08	5.46	4.10	5.78

the number of iterations even reaches 50, which is the maximum iteration number we set in the implementation. These are the bad cases we discussed in the previous section. Notice that the systems with high iteration numbers in Figure 4.2 are exactly the systems of which the minima of the coefficients are close to zero. Unsurprisingly, the running time of Scheme 3 is much less than that of Scheme 2 and only about 1/3 that of Scheme 1. In Figure 4.5, we recycle the Krylov subspace and the preconditioner by the grouping algorithm. The running time is about the same as for Scheme 3. The iteration numbers required for solving those few systems are also controlled.

Figures 4.3, 4.4, and 4.6 are for the comparison of the schemes when we use two-level additive Schwarz preconditioner in the implementation. The coarse mesh is 16×16. It shows the same effects as in the one-level case. In particular, we can see that in Scheme 4 there are more systems requiring zero iteration numbers.

4.2. Scalability of the additive Schwarz method. In this section, we study the scalability of the additive Schwarz methods. We use recycling Scheme 4 for these experiments. Table 4.2 shows the results of the one-level additive Schwarz method. Table 4.3 shows the results of the two-level additive Schwarz method. The iteration numbers in the tables are the average iteration numbers for solving all systems.

From Table 4.2, we can see that, for the one-level ASM, as the mesh size increases, the average iteration numbers actually decrease a little. As the number of processors increases, the average iteration numbers increase except for the 16 processor case. To conclude, with the one-level ASM, the method is scalable with respect to mesh size but not to the number of processors.

TABLE 4.3
Average iteration numbers, two-level additive Schwarz preconditioner, coarse mesh 16×16.

Mesh	Overlap	Number of subdomains			
		1	4	16	64
136×136	4	7.66	8.53	8.82	8.99
256×256	8	3.93	9.04	6.39	4.14
496×496	16	2.52	2.59	4.59	3.44

From Table 4.3, we can see that the two-level ASM is scalable with respect to both the mesh size and the number of processors. Actually, the average iteration numbers decrease a little as the mesh size increases as in the one-level case. When the number of processors increases, the average iteration numbers decrease too, except for the mesh 136×136. The number of iterations is higher than expected; more research is still needed to fully understand the situation.

5. Conclusion. In this paper, we introduced a parallel domain decomposition preconditioned recycling Krylov subspace method for stochastic elliptic partial differential equations. By recycling the Krylov subspace and the preconditioner, the overall computational cost and time can be saved by as much as 50%. We carefully analyzed the KL expansion and derived a useful algorithm that tells us when to recompute and when to recycle some of the expensive components of the computation. We showed theoretically and experimentally that the multilevel additive Schwarz preconditioned recycling GMRES method is optimal for the entire family of linear systems arising from the discretization of the stochastic elliptic partial differential equation. A fully parallel implementation was provided and tested with up to 64 processors, and scalability results were also reported.

Acknowledgments. The authors wish to thank Professor Dongbin Xiu for many helpful discussions, and the referees for useful suggestions.

REFERENCES

[1] I. BABUSKA AND P. CHATZIPANTELIDIS, *On solving linear elliptic stochastic partial differential equations*, Comput. Methods Appl. Mech. Engrg., 191 (2002), pp. 4093–4122.
[2] I. BABUŠKA, R. TEMPONE, AND G. E. ZOURARIS, *Galerkin finite element approximations of stochastic elliptic partial differential equations*, SIAM J. Numer. Anal., 42 (2004), pp. 800–825.
[3] I. BABUSKA, R. TEMPONE, AND G. ZOURARIS, *Solving elliptic boundary value problems with uncertain coefficients by the finite element method: The stochastic formulation*, Comput. Methods Appl. Mech. Engrg., 194 (2005), pp. 1251–1795.
[4] S. BALAY, K. BUSCHELMAN, W. D. GROPP, D. KAUSHIK, M. KNEPLEY, L. C. MCINNES, B. F. SMITH, AND H. ZHANG, *PETSc Users Manual*, ANL-95/11 - Revision 2.3.0, Argonne National Laboratory, Argonne, IL, 2006.
[5] X.-C. CAI, *An additive Schwarz algorithm for nonselfadjoint elliptic equations*, in Proceedings of the Third International Symposium on Domain Decomposition Methods for Partial Differential Equations, T. F. Chan, R. Glowinski, J. Periaux, and O. B. Widlund, eds., SIAM, Philadelphia, 1990, pp. 232–244.
[6] X.-C. CAI, *An optimal two-level overlapping domain decomposition method for elliptic problems in two and three dimensions*, SIAM J. Sci. Comput., 14 (1993), pp. 239–247.
[7] X.-C. CAI AND O. B. WILBUND, *Domain decomposition algorithms for indefinite elliptic problems*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 243–258.
[8] A. CHAPMAN AND Y. SAAD, *Deflated and argumented Krylov subspace techniques*, Numer. Linear Algebra Appl., 4 (1997), pp. 43–66.

[9] P. Dostert, Y. Efendiev, T. Y. Hou, and W. Luo, *Coarse-gradient Langevin algorithms for dynamic data integration and uncertainty quantification*, J. Comput. Phys., 217 (2006), pp. 123–142.

[10] M. Dryja and O. B. Widlund, *Some domain decomposition algorithms for elliptic problems*, in Iterative Methods for Large Linear Systems, L. Hayes and D. Kincaid, eds., Academic Press, San Diego, CA, 1989.

[11] M. Eiermann, O. Ernst, and E. Ullmann, *Computational aspects of the stochastic finite element method*, in Proceedings of ALGORITMY, Podbanske, Slovakia, 2005, pp. 1–10.

[12] H. Elman, O. Ernst, D. O'Leary, and M. Stewart, *Efficient iterative algorithms for the stochastic finite element method with application to acoustic scattering*, Comput. Methods Appl. Mech. Engrg., 194 (2005), pp. 1037–1055.

[13] H. Elman and D. Furnival, *Solving the stochastic steady-state diffusion problem using multigrid*, Tech. report UMCP-CSD:CS-TR-4786, Department of Computer Science, University of Maryland, College Park, MD, 2006.

[14] C. Farhat, L. Crivelli, and F. X. Roux, *Extending substructure based on iterative solvers to multiple load and repeated analyses*, Comput. Methods Appl. Mech. Engrg., 117 (1994), pp. 195–209.

[15] P. F. Fischer, *Projection techniques for iterative solution of $Ax = b$ with successive right-hand sides*, Comput. Methods Appl. Mech. Engrg., 163 (1998), pp. 193–204.

[16] P. Frauenfelder, C. Schwab, and R. A. Todor, *Finite elements for elliptic problems with stochastic coefficients*, Comput. Methods Appl. Mech. Engrg., 194 (2005), pp. 205–228.

[17] R. G. Ghanem and P. D. Spanos, *Stochastic Finite Elements: A Spectral Approach*, 2nd ed., Springer-Verlag, New York, 1991.

[18] G. H. Golub and C. F. Van Loan, *Matrix Computation*, 3rd ed., John Hopkins University Press, Baltimore, MD, 1996.

[19] K. Guruprasad, D. E. Keyes, and J. H. Kane, *GMRES for Sequentially Multiple Nearby Systems*, Technical report, Old Dominion University, Norfolk, VA, 1995.

[20] H. Klie, M. F. Wheeler, G. Liu, and D. Zhang, *Stochastic subspace projection methods for efficient multiphase flow uncertainty assessment*, in Proceedings of the Tenth Annual European Conference on the Mathematics of Oil Recovery, Amsterdam, The Netherlands, 2006.

[21] M. Loève, *Probability Theory*, Vol. I, II, Springer, New York, 1978.

[22] H. Matthies and A. Keese, *Galerkin methods for linear and nonlinear elliptic stochastic partial differential equations*, Comput. Methods Appl. Mech. Engrg., 194 (2005), pp. 1295–1331.

[23] R. B. Morgan, *A restarted GMRES method augmented with eigenvectors*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 1154–1171.

[24] M. L. Parks, E. D. Sturler, G. Mackey, D. D. Johnson, and S. Maiti, *Recycling Krylov subspaces for sequences of linear systems*, SIAM J. Sci. Comput., 28 (2006), pp. 1651–1674.

[25] L. Roman and M. Sarkis, *Stochastic Galerkin method for elliptic SPDEs: A white noise approach*, Discrete Contin. Dynam. Systems, 6 (2006), pp. 941–955.

[26] V. Simoncini and E. Gallopoulos, *An iterative method for nonsymmetric systems with multiple right-hand sides*, SIAM J. Sci. Comput., 16 (1994), pp. 917–933.

[27] B. Smith, P. Bjørstad, and W. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, Cambridge, MA, 1996.

[28] A. Toselli and O. Widlund, *Domain Decomposition Methods—Algorithms and Theory*, Springer-Verlag, New York, Berlin, 2005.

[29] D. Xiu and G. E. Karniadakis, *The Wiener–Askey polynomial chaos for stochastic differential equations*, SIAM J. Sci. Comput., 24 (2002), pp. 619–644.

[30] D. Xiu and G. E. Karniadakis, *Modeling uncertainty in steady state diffusion problems via generalized polynomial chaos*, Comput. Methods Appl. Mech. Engrg., 191 (2002), pp. 4927–4948.

[31] N. Wiener, *The homogeneous chaos*, Amer. J. Math., 60 (1930), pp. 897–936.

SIAM J. SCI. COMPUT.
Vol. 29, No. 5, pp. 2115–2143
© 2007 Society for Industrial and Applied Mathematics

GREEDY COARSENING STRATEGIES FOR NONSYMMETRIC PROBLEMS*

SCOTT MACLACHLAN† AND YOUSEF SAAD†

Abstract. The solution of large-scale linear systems in computational science and engineering requires efficient solvers and preconditioners. Often, the most effective such techniques are those based on multilevel splittings of the problem. In this paper, we consider the problem of partitioning both symmetric and nonsymmetric matrices based solely on algebraic criteria. A new algorithm is proposed that combines attractive features of two previous techniques proposed by the authors. It offers rigorous guarantees of certain properties of the partitioning, yet is naturally compatible with the threshold based dropping known to be effective for incomplete factorizations. Numerical results show that the new partitioning scheme leads to improved results for a variety of problems. The effects of further matrix reordering within the fine-scale block are also considered.

Key words. preconditioning, algebraic multigrid (AMG), multilevel ILU, matrix partitioning, nonsymmetric matrices

AMS subject classifications. 65F10, 65N55, 65Y20

DOI. 10.1137/060660928

1. Introduction.

Recent advances in the solution of linear systems of equations have been driven by the ever-increasing problem sizes required in computational science and engineering applications. Matrices of interest arise both from discretizations of differential equations that govern physical systems, where accuracy considerations demand fine meshes, and from naturally discrete problems with many degrees of freedom. These systems are typically not only large but also ill conditioned, requiring advanced techniques for efficient solution.

For many such large-scale linear systems, the most efficient solution techniques utilize multilevel frameworks. For elliptic PDE based problems, multigrid [12] and algebraic multigrid (AMG) [8, 25] methods have been demonstrated to have optimal efficiency. For more general problems, however, classical multigrid approaches do not perform as well without more expensive approaches to the multigrid setup phase [9, 10]. While purely algebraic approaches, such as the algebraic recursive multilevel solver (ARMS) [30] and other multilevel ILU techniques [4, 7], do not typically match the performance of multigrid for discretizations of elliptic PDEs, their robustness across many problems makes them an attractive option when complete knowledge of a problem's origin cannot be guaranteed.

When the system matrix is symmetric and positive definite, theoretical analysis gives insight into the requirements on the partitioning into fine-scale and coarse-scale degrees of freedom. Analysis of multilevel block factorization preconditioners, such as ARMS, shows that it is crucial that the fine-scale submatrix be well approximated, in a spectral sense, by the fine-scale part of the preconditioner in order to achieve effective results [24]. Motivated by this analysis and corresponding theory for multigrid for symmetric and positive-definite matrices [16, 21], we have previously developed a

*Received by the editors May 25, 2006; accepted for publication (in revised form) April 9, 2007; published electronically September 28, 2007. This work was supported by the NSF under grant ACI-0305120, and by the Minnesota Supercomputing Institute.
http://www.siam.org/journals/sisc/29-5/66092.html
†Department of Computer Science and Engineering, University of Minnesota, 200 Union Street S.E., Minneapolis, MN 55455 (maclach@cs.umn.edu, saad@cs.umn.edu).

partitioning algorithm for the symmetric case [22]. In this approach, the optimal partition is defined in terms of a diagonal-dominance property of the fine-scale block. Exactly achieving the optimal coarsening was shown to be an NP-complete problem, and so an $O(N)$ approach to approximately achieve this objective, based on a greedy strategy, was used instead.

In this paper, we consider the extension of the approach from [22] to nonsymmetric problems. A direct generalization using symmetric permutations is quite natural although may not be effective when the nonsymmetry in the matrix dominates its properties. In particular, when defining a multilevel preconditioner, the partitioning algorithm must be applied recursively; the symmetric approach of [22], however, requires stricter constraints on the coarse-grid problem than are natural to ensure that recursion is possible. While the extension to nonsymmetric matrices is simple in principle, it is quite powerful. Multilevel ILU factorizations are most effective when combined with threshold based dropping criteria. The symmetric partitioning approach [22] requires symmetric operators on all levels; however, threshold based dropping in IKJ-ordered factorizations is naturally nonsymmetric. In [22], the upper-triangular factor was discarded, and the incomplete factor, L^T, was used instead to ensure symmetry. Here, no information must be lost.

A natural approach for nonsymmetric problems is to consider a nonsymmetric permutation strategy, as was done in [28]. Here, we extend our earlier, greedy approach to such nonsymmetric permutations, seeking a permutation where the fine-scale block is row diagonally dominant. The algorithms proposed in [28], while compatible with threshold based dropping, do not necessarily guarantee the diagonal-dominance of the fine-grid subblock and, thus, may result in less accurate factors. By carefully extending the symmetric partitioning algorithm of [22], we seek to gain the best of both worlds: the robustness of nonsymmetric permutations coupled with the computational efficiency gained by large, diagonally dominant fine-grid blocks. The partitioning approaches are compared on a variety of nonsymmetric matrices. Additionally, we consider the effectiveness of the nonsymmetric approaches discussed here on the symmetric test problems of [22], without looking to exploit symmetry. This allows the ILU factors to be used as given and compares the utility of enforcing symmetry versus exploiting the full power of the incomplete factorization approach.

Another important question in the ARMS technique is the ordering of the fine-scale block, A_{ff}, so that it may be efficiently approximated by its ILUT factors. Here, we consider various approaches to reordering the fine-scale block based on standard techniques such as those of Sparspak [17] and METIS [20]. The effects of reordering on the performance of incomplete factorization preconditioners has been considered previously [5, 11, 15]. Here, we extend the work of [29] and examine the effects of reordering on multilevel ILU preconditioners, such as ARMS.

This paper is organized as follows. In section 2, we present the ARMS algorithm and the greedy partitioning strategy. Section 3 details the extension of these approaches to nonsymmetric problems, with particular focus on nonsymmetric permutations. Numerical results, for both symmetric and nonsymmetric problems, are presented in section 4. A discussion of reordering of the fine-scale block is given in section 5, followed by conclusions in section 6.

2. Background.

2.1. The algebraic recursive multilevel solver (ARMS) algorithm. The ARMS algorithm [30] arises from considering the block factorization of a given matrix, A. Partitioning the degrees of freedom of A into two sets, F and C, referred to as

the fine-grid and coarse-grid degrees of freedom (respectively), the (reordered) matrix can then be written as a block 2×2 system,

$$A = \begin{bmatrix} A_{ff} & -A_{fc} \\ -A_{cf} & A_{cc} \end{bmatrix}.$$

This block form can be directly factored as

$$A = \begin{bmatrix} I & 0 \\ -A_{cf}A_{ff}^{-1} & I \end{bmatrix} \begin{bmatrix} A_{ff} & 0 \\ 0 & \hat{A}_{cc} \end{bmatrix} \begin{bmatrix} I & -A_{ff}^{-1}A_{fc} \\ 0 & I \end{bmatrix}, \tag{1}$$

where $\hat{A}_{cc} = A_{cc} - A_{cf}A_{ff}^{-1}A_{fc}$ denotes the Schur complement of A. From this factored form, an algorithm for the direct solution of $A\mathbf{x} = \mathbf{b}$ is apparent (partitioning the vectors, $\mathbf{x} = \left(\begin{smallmatrix} \mathbf{x}_f \\ \mathbf{x}_c \end{smallmatrix}\right)$ and $\mathbf{b} = \left(\begin{smallmatrix} \mathbf{b}_f \\ \mathbf{b}_c \end{smallmatrix}\right)$).

ALGORITHM 1 (block factorization solve of $A\mathbf{x} = \mathbf{b}$).

1. $\mathbf{y}_f = A_{ff}^{-1}\mathbf{b}_f$
2. $\mathbf{y}_c = \mathbf{b}_c + A_{cf}\mathbf{y}_f$
3. Solve $\hat{A}_{cc}\mathbf{x}_c = \mathbf{y}_c$
4. $\mathbf{x}_f = \mathbf{y}_f + A_{ff}^{-1}A_{fc}\mathbf{x}_c$

For a general F/C partition, there is no advantage in using Algorithm 1 over directly factoring A. The advantage of the block factorization approach occurs when considering preconditioners, however, if the partition is chosen so that the diagonal-block matrices, A_{ff} and $\hat{A}_{cc}$, are easily (approximately) inverted. In this case, solution with an approximate block factorization can be a good preconditioner for a Krylov subspace method. Many variations on this approach have been considered; see, for example, [2, 3, 7].

In the ARMS methodology, the inversion of A_{ff} is approximated through its ILUT factors. That is, we write $A_{ff} \approx LU$, where the L and U factors are truncated based on size and number of nonzeros per row [26], and approximate A as in (1) by

$$B = \begin{bmatrix} I & 0 \\ -A_{cf}U^{-1}L^{-1} & I \end{bmatrix} \begin{bmatrix} LU & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} I & -U^{-1}L^{-1}A_{fc} \\ 0 & I \end{bmatrix}.$$

The application of the preconditioner, B, to a residual, $\mathbf{r}$, is then given by Algorithm 2.

ALGORITHM 2 (action of ARMS preconditioner on residual, $B^{-1}\mathbf{r}$).

1. $\mathbf{y}_f = L^{-1}\mathbf{r}_f$
2. $\mathbf{y}_c = \mathbf{r}_c + A_{cf}U^{-1}\mathbf{y}_f$
3. Solve $S\mathbf{x}_c = \mathbf{y}_c$
4. $\mathbf{z}_f = \mathbf{y}_f + L^{-1}A_{fc}\mathbf{x}_c$
5. $\mathbf{x}_f = U^{-1}\mathbf{z}_f$

Here, S is an approximation to $\hat{A}_{cc} = A_{cc} - A_{cf}A_{ff}^{-1}A_{fc}$, computed using $A_{ff}^{-1} \approx U^{-1}L^{-1}$ and similar truncation strategies to those of ILUT. While the solution of $S\mathbf{x}_c = \mathbf{y}_c$ in step 3 could be considered directly, the set, C, is generally still large enough that this is quite costly. Instead, the ARMS methodology is applied recursively to solve the system with S, stopping only when the dimension of the coarsest-scale problem is small enough that direct solution is practical, or when S^{-1} is seen to be easily approximated (for example, when S itself is strongly diagonally dominant).

It is then apparent that the success of Algorithm 2 as a preconditioner is dependent on the accuracy of the factorization, $A_{ff} \approx LU$. The accuracy of this factorization, in turn, depends on the partition chosen to define F. If the partition allows

direct factorization of A_{ff} with no fill (if the graph of A_{ff} is, for example, a tree), then the factorization may be done very accurately at low cost. Unfortunately, this is usually not possible without choosing F to be prohibitively small. In contrast, however, choosing C to be small often leads to an A_{ff} block whose inverse is difficult to approximate in a sparse manner. The contrasting goals of sparse factorization and effective coarsening lead to partitioning algorithms that aim at a compromise.

2.2. Original ARMS partitioning. The original ARMS algorithm [30] is a natural extension of the ILUM, BILUM, and BILUTM algorithms. The ILUM (multi-elimination ILU) algorithm [27] is a block-factorization preconditioner, where the partitioning is chosen so that the fine-scale block, A_{ff}, is diagonal (that is, F is an independent set of the degrees of freedom). As A_{ff} is diagonal, it is easily inverted, and the block-factorization solve may be easily implemented (although it is often useful to allow some dropping in the computation of the Schur complement to limit fill). The BILUM (block ILUM) algorithm [31] extends this idea using small block-independent sets to form F, yielding a block-diagonal matrix, A_{ff}. In BILUTM (block multilevel ILUT) [32], the block-independent sets used to form F were allowed to be much larger, leading to inefficiency in computing the exact inverse of the diagonal blocks of A_{ff}. Instead, the ILUT algorithm was used to compute approximate inverses of the large diagonal blocks in A_{ff}, where the block-independent sets are formed using a domain decomposition approach.

In [30], the idea of diagonal dominance was first introduced into the partitioning stage. Row-wise diagonal-dominance coefficients of A are initially computed as $\hat{w}(i) = \frac{|a_{ii}|}{\sum_{j=1}^{n} |a_{ij}|}$, and then scaled by the maximum dominance ratio, giving $w(i) = \frac{\hat{w}(i)}{\max_j \hat{w}(j)}$. An initial fine-scale block is then chosen using a similar approach as before, either by choosing an independent set of the degrees of freedom, or by choosing block-independent sets (using, for example, nested dissection [17]). Points from this set are, however, rejected based on thresholding of the diagonal-dominance ratios, $w(i)$. If a row, i, chosen to be in F is not sufficiently diagonally dominant (if $w(i) < \theta$ for some predetermined θ), then row i is rejected from the fine set and switched to a coarse-grid point. Note that the use of relative diagonal-dominance measures here implies that the fine-grid block, A_{ff}, can no longer be said to be (block) diagonally dominant—only that it tends to be, as its rows were chosen from the most diagonally dominant rows of A.

The introduction of a measure based on diagonal dominance is motivated by considering the ILUT employed in approximating the inverse of A_{ff} in the ARMS algorithm (Algorithm 2). The accuracy of the overall preconditioning depends on how accurately A_{ff} is represented by its ILUT factors. The efficiency, however, depends on how sparse these factors are. To best balance these competing concerns, we seek to choose F so that A_{ff} is well approximated by sparse ILUT factors. To achieve this balance, we reject rows from F that are poorly diagonally dominant, as these are rows in which we expect many significant off-diagonal entries in the ILUT factors. Put another way, we can most easily compute accurate, sparse ILUT factors of A_{ff} when it is strongly diagonally dominant; thus, we reject rows from F that are more likely to cause fill.

2.3. Greedy partitioning. While the use of a diagonal-dominance measure as in [30] is an important improvement in the ARMS algorithm, the approach used there is static; that is, the evolving fine/coarse partition plays no role in the computation or use of these measures. A dynamic approach to diagonal dominance was recently

introduced in [22], based on the observation that it is the dominance strictly within A_{ff} that determines how well A_{ff} may be approximated by its ILUT factors, and not just the dominance of the F rows of A (as is measured by the ratios in [30]).

In [22], the static measures, $w(i)$, are replaced by dynamic measures, $\hat{\theta}_i = \frac{|a_{ii}|}{\sum_{j \in F \cup U} |a_{ij}|}$, where the degrees of freedom of A are assumed either to have been already partitioned into F and C, or to be in U, the set of undecided points. Thus, $\hat{\theta}_i$ is a measure of the dominance of row i over all points that either are in F or have the potential to be in F. If row $i \in U$ is dominant over this set ($\hat{\theta}_i \geq \theta$ for some predetermined θ), then i is added to set F. If there are no such points, then at least one column from U must be added to C in order to determine if the dominance of the rows in U can be improved to the point where they make good F points, or if they should also be discarded into C. For the case of symmetric and positive-definite matrices, as considered in [22], the partition of rows and columns into F and C was also symmetric (that is, if row $i \in C$, then so is column i); so, the row/column pair associated with the least diagonally dominant row is chosen to be the new C point. Measures for all neighboring U rows are then updated, with any rows whose measures are now large enough added to F.

This greedy approach (so named because at each step, all sufficiently dominant rows are added to F and the least dominant row is discarded into C) is considered as an approximation to the "ideal" partitioning of the degrees of freedom of A. If diagonal dominance of A_{ff} is all that we need to ensure a good approximation of A_{ff} by its ILUT factors (in terms of both accuracy and sparsity), then the ideal partitioning is the one which maximizes the size of F, given the constraint that for all rows, $i \in F$, $\theta_i = \frac{|a_{ii}|}{\sum_{j \in F} |a_{ij}|} \geq \theta$, for some predetermined θ. In [22], we showed that finding the ideal partitioning for arbitrary A is an NP-complete problem. This greedy approach approximates the ideal partitioning in such a way that we are guaranteed that the A_{ff} block is diagonally dominant and led to effective ARMS performance for a number of symmetric problems.

It is important, however, to realize that the approach proposed in [22] is strictly limited in its applicability to symmetric problems. In particular, the decision as to which elements of A to partition into F and C is made using a solely row based criterion. A symmetric partition may be effective for a given nonsymmetric matrix; however, in general, an appropriate measure for making such a decision should take the nonsymmetry into account. This is particularly important in the multilevel setting considered here, where a naturally nonsymmetric factorization (such as the ILUT) may be used to approximate A_{ff}^{-1} within the calculation of the approximate Schur complement. If this nonsymmetry is not appropriately taken into account, the partitioning of the coarse-grid operator is no longer guaranteed to satisfy the desired diagonal-dominance property and may fail altogether.

3. Nonsymmetric partitioning approaches. While the symmetric partitioning approaches described above work well for many symmetric (and some nonsymmetric) problems, they do not always effectively partition the rows and columns of A. Consider, for example, taking a diagonally dominant tridiagonal matrix and cyclically permuting all entries two columns to the right. The resulting matrix is no longer diagonally dominant (indeed, the diagonal entries are all zero), and either of the above algorithms would fail to find any F points. A more robust approach is to define separate left and right permutation matrices, allowing large off-diagonal entries to be moved to the diagonal of the reordered system and used as pivots in the ILUT factorization.

3.1. Two-stage algorithm. In [28], a two-stage approach is used to find a nonsymmetric permutation that results in an A_{ff} block that is expected to have its largest entries on the diagonal. A preselection stage is used to first identify rows that are good potential rows for A_{ff} and then sort these rows based on a measure of their likelihood to produce sparse ILUT factors. In the second stage, the nonsymmetric permutations are defined by traversing this list (in order), permuting the dominant element for the row to the diagonal, if possible, and discarding rows for which a suitable permutation cannot be defined. In this way, a nonsymmetric permutation pair is constructed that yields an A_{ff} block that is expected to yield sparse ILUT factors based on a simple greedy strategy.

The preselection stage itself first discards rows with no significant dominant element, and then orders the remaining rows based on evaluating their potential to yield sparse ILUT factors. For each row, i, the dominant element in row i, $k_i = \text{argmax}_k |a_{ik}|$, is first calculated, along with the row's ℓ_1 norm, $t_i = \sum_j |a_{ij}|$. If t_i is small enough, relative to $|a_{ik_i}|$, measured by a preselected tolerance, then row i is admitted as a candidate row for A_{ff}. Diagonal dominance alone is not enough, however, to guarantee sparse ILUT factors. Indeed, a row with only two equal-sized nonzero entries may be more attractive than a row with one large entry and many small entries (as it tends to introduce less fill into the subsequent rows of the factors). To address this issue, the dominance ratios, $\frac{|a_{ik_i}|}{\sum_j |a_{ij}|}$, are then multiplied by $\frac{1}{|\text{Adj}(i)|}$, where $\text{Adj}(i) = \{j \neq i : a_{ij} \neq 0\}$, and $|S|$ denotes the number of elements in set S. These weighted dominance ratios are then sorted in decreasing order to establish a ranking of the attractiveness of row i to be included in A_{ff}.

Many possible strategies exist for selecting which of the rows passed by the preselection algorithm described above are then included in A_{ff} [28]. A simple greedy approach is to scan these rows in order, building a nonsymmetric permutation by accepting row i and column k_i into the F block if neither has already been accepted. As rows appear at most once in the preselection list, this requires only checking if column k_i has already been used as a diagonal by another row already selected for F. If so, row i is discarded to C; otherwise it is accepted, and node (i, k_i) is permuted to the diagonal.

While this approach is successful for many problems, it does not always yield an effective preconditioner. Adding constraints on A_{ff} such that it is diagonally dominant may yield a smaller block but a better overall preconditioner (as the ILUT factors of A_{ff} are both sparser and more accurate). Note, however, that this approach relies on static measures (i.e., measures that are not influenced by the evolving partition) and does not guarantee the diagonal dominance of the resulting A_{ff} block. A second strategy proposed in [28] is based on a dynamic measure and, thus, guarantees row diagonal dominance within the A_{ff} block of the reordered matrix. In this approach, for each row i accepted by the preselection algorithm, if column k_i has not already been reordered and element $|a_{ik_i}|$ dominates row i over those columns already added to F, row i and column k_i are added to A_{ff}. Additionally, for each remaining column $j \in \text{Adj}(i)$ that has not already been added to F or C, if a_{ij} is large enough such that row i would fail to be diagonally dominant in A_{ff} if $|\text{Adj}(i) \setminus (F \cup C)|$ entries of size a_{ij} are added to it, column j is rejected from possibly becoming an F column, and is added to C. In this way, the A_{ff} block of the reordered system is guaranteed to be row diagonally dominant, but columns that may have been acceptable choices for the A_{ff} block are rejected based on an averaged expectation and not their unique contribution. Notice also that the performance is strongly dependent on the ordering

produced by the preselection algorithm. If row i is strongly dominated by column k_i, but there are many large entries in that column, the resulting partition may suffer from accepting row i too early.

3.2. Single-stage greedy nonsymmetric partitioning. Here, we propose a technique that combines the nonsymmetric permutation approach of [28] (described in section 3.1) with the single-stage greedy algorithm approach of [22] (described in section 2.3). The resulting algorithm once again guarantees that the A_{ff} block of the reordered system is diagonally dominant but is much more reluctant to move columns into C. As such, we expect the resulting algorithm to form similar partitions to those chosen by [28] when that algorithm performs well (i.e., when the chosen F blocks are not too small) but to show improved performance when the average dominance criterion described above is too aggressive at removing columns. The practical benefit of such an approach is an expected improvement in the complexity of the resulting preconditioners; increasing the size of the A_{ff} blocks while retaining similar diagonal dominance properties should not degrade convergence of the preconditioners but should result in a sparser approximate factorization that is less expensive to apply.

Extending the diagonal-dominance measure from [22], row i of matrix A is defined to be θ-dominated by column k if $\frac{|a_{ik}|}{\sum_j |a_{ij}|} \geq \theta$. As before, we seek to find a partitioning of the rows and columns of A so that each row in A_{ff} is θ-dominated by its diagonal. To do this, consider the three sets of undecided points, U, fine points, F, and coarse points, C, but now we will maintain different sets for both rows and columns. A point in U_{row} is made into an F-row if, at any point,

$$\frac{|a_{ik_i}|}{\sum_{j \in F_{\text{col}} \cup U_{\text{col}}} |a_{ij}|} \geq \theta,$$

where $k_i = \text{argmax}_{k \in U_{\text{col}}} |a_{ik}|$. If row i is put into F_{row}, then column k_i must also be put into F_{col}, so that the entry, a_{ik_i}, may be permuted into a diagonal position in A_{ff}. If, at any point, row i is zero over all columns in U_{col}, then it is placed in C_{row}, as it is no longer possible to θ-dominate this row with any valid pivot element.

Rows and columns are always moved into the F sets in pairs, defining a nonsymmetric permutation with a square A_{ff} block. Eliminating rows and columns from the U sets into the C sets, however, should occur independently, as if row i is no longer sufficiently dominated by an eligible pivot to be put in F_{row}; it does not necessarily mean that any of the columns in $\text{Adj}(i) \cap U_{\text{col}}$ would not make a good pivot for another row in U_{row}. Thus, the elimination of rows and columns from U_{row} and U_{col} into the C sets occurs independently, as follows. If, at some point in the algorithm, no row in U_{row} is sufficiently dominated by a column in U_{col} such that another row/column pair can be added to F, a single column, $j^\star \in U_{\text{col}}$, that is deemed to be least attractive as a column in A_{ff} is selected to be added to C_{col}. For each $i \in U_{\text{row}}$ such that $a_{ij^\star} \neq 0$, the dominance ratio of row i is updated by decrementing the ℓ_1 norm by $|a_{ij^\star}|$ and, possibly, recomputing the dominant column, $k_i \in U_{\text{col}}$. If $a_{ij^\star}$ is the last nonzero entry in row i's restriction to U_{col}, then row i is added to C_{row}. Otherwise, if the updated $\hat{\theta}_i = \frac{|a_{ik_i}|}{\sum_{j \in F_{\text{col}} \cup U_{\text{col}}} |a_{ij}|} \geq \theta$, then the pair (i, k_i) is added to the F sets. If, at any point when a column is moved into F_{col} or C_{col}, a row, i, is no longer sufficiently dominated over F_{col} by some $k \in U_{\text{col}}$, that is, $\frac{|a_{ik}|}{\sum_{j \in F_{\text{col}}} |a_{ij}|} < \theta$ for all $k \in \text{Adj}(i) \cap U_{\text{col}}$, then there is no possible pivot column for row i that would result in a diagonally dominant

A_{ff}, and row i is discarded into C. This pattern continues until either U_{row} or U_{col} is empty.

There are many possible measures of which the column, $j^\star$, is chosen as the least attractive potential F column. In the symmetric permutation case considered in [22], $j^\star$ was chosen based on the attractiveness of row $j^\star$ as an F-row. Here, we consider the row diagonal-dominance measures,

$$\hat{\theta}_i = \frac{|a_{ik_i}|}{\sum_{j \in F_{\text{col}} \cup U_{\text{col}}} |a_{ij}|},$$

from which the decrease of $\hat{\theta}_i^{-1}$that occurs when column $j^\star$ is moved from U_{col} to C_{col} is easily quantified as $\frac{|a_{ij^\star}|}{|a_{ik_i}|}$. The column that is most attractive as a C column, then, is the one that brings about the greatest cumulative change in the measures, $\hat{\theta}_i$, for $i \in U_{\text{row}}$ measured, for example, as $\sum_{i \in U_{\text{row}}} \frac{|a_{ij}|}{|a_{ik_i}|}$ for each $j \in U_{\text{col}}$.

The resulting algorithm is then summarized as follows, where we ignore the important question of computational complexity and suppress the updating of k_i for $i \in U_{\text{row}}$, which is assumed to always reference the largest entry, a_{ik}, over $k \in U_{\text{col}}$. In Algorithm 3, all second-order updates are also suppressed, particularly the elimination of rows to C when new dominating columns, k_i, are computed. For full details of the implementation, see the discussion in section 3.3.

ALGORITHM 3 (single-stage greedy nonsymmetric partitioning algorithm).

1. Initialize $U_{\text{row}} = U_{\text{col}} = \{1, 2, \ldots, n\}$
2. Initialize $C_{\text{row}} = C_{\text{col}} = \emptyset$ and $F_{\text{row}} = F_{\text{col}} = \emptyset$
3. For all $i \in U_{\text{row}}$,
   (a) Compute $k_i = \text{argmax}_{k \in U_{\text{col}}} |a_{ik}|$
   (b) Compute $l_i = \sum_{j \in F_{\text{col}} \cup U_{\text{col}}} |a_{ij}|$
   (c) Compute $r_i = \sum_{j \in F_{\text{col}}} |a_{ij}|$
   (d) If $a_{ik_i} = 0$, then $C_{\text{row}} = C_{\text{row}} \cup \{i\}$, $U_{\text{row}} = U_{\text{row}} \setminus \{i\}$.
   (e) If $\frac{|a_{ik_i}|}{l_i} \geq \theta$, then make (i, k_i) a diagonal element of A_{ff}:
      - $F_{\text{row}} = F_{\text{row}} \cup \{i\}$, $U_{\text{row}} = U_{\text{row}} \setminus \{i\}$
      - $F_{\text{col}} = F_{\text{col}} \cup \{k_i\}$, $U_{\text{col}} = U_{\text{col}} \setminus \{k_i\}$
   (f) If $\frac{|a_{ik_i}|}{r_i} < \theta$, then $C_{\text{row}} = C_{\text{row}} \cup \{i\}$, $U_{\text{row}} = U_{\text{row}} \setminus \{i\}$.
4. While $U_{\text{col}} \neq \emptyset$ and $U_{\text{row}} \neq \emptyset$,
   (a) For each $j \in U_{\text{col}}$, compute $w_j = \sum_{i \in U_{\text{row}}} \frac{|a_{ij}|}{|a_{ik_i}|}$
   (b) Let $j^\star = \text{argmax}_{j \in U_{\text{col}}} \{w_j\}$ and make $j^\star$ a C-column: $C_{\text{col}} = C_{\text{col}} \cup \{j^\star\}$, $U_{\text{col}} = U_{\text{col}} \setminus \{j^\star\}$
   (c) For $i \in U_{\text{row}} \cap \{i : a_{ij^\star} \neq 0\}$,
      i. Update $l_i = l_i - |a_{ij^\star}|$
      ii. If $a_{ik_i} = 0$, then $C_{\text{row}} = C_{\text{row}} \cup \{i\}$, $U_{\text{row}} = U_{\text{row}} \setminus \{i\}$.
      iii. If $\frac{|a_{ik_i}|}{l_i} \geq \theta$, then
         - $F_{\text{row}} = F_{\text{row}} \cup \{i\}$, $U_{\text{row}} = U_{\text{row}} \setminus \{i\}$
         - $F_{\text{col}} = F_{\text{col}} \cup \{k_i\}$, $U_{\text{col}} = U_{\text{col}} \setminus \{k_i\}$
5. $C_{\text{row}} = C_{\text{row}} \cup U_{\text{row}}$
6. $C_{\text{col}} = C_{\text{col}} \cup U_{\text{col}}$

Steps 1 and 2 in Algorithm 3 simply initialize the sets U, F, and C for both columns and rows. Step 3 is an initial pass over the rows, which finds the largest entry in each row and computes the total and fine-grid row sums of each row. If the

row is zero over U_{col}, it is immediately discarded into C_{row}. If the row is already θ-dominated by its largest entry, then that entry is permuted to the diagonal and its row and column are moved into the F sets. If no θ-dominance by a column in U_{col} is possible for the row, it is also immediately discarded into C_{row}.

The main elimination loop occurs in step 4. For each undecided column, j, a measure, w_j, of column j's fitness as an F-column is computed. The larger the measure, the more large entries there are in column j, relative to the largest in the eligible part of each row. Thus, we select the column, $j^\star$, with the largest measure to be eliminated from U_{col} (and added to C_{col}) in step 4(b). For rows i such that $j^\star \in \text{Adj}(i)$, the removal of column $j^\star$ increases the dominance of a_{ik_i} over the portion of row i that is in $F_{\text{col}} \cup U_{\text{col}}$; thus, we check each adjacent row to see if it is now sufficiently dominated to be included in F.

Finally, note that the termination condition for the main loop (in step 4) allows the algorithm to stop without a full partitioning of both rows and columns. This is reasonable, as step 4(a) could be trivial in two ways: either U_{col} is empty, in which case there are no weights to be calculated, or U_{row} is empty, in which case all the weights are zero. This is possible because columns and rows are moved individually into their respective C blocks, so that while $|F_{\text{col}}| = |F_{\text{row}}|$ must always hold, there is no such constraint on the growth of the C blocks. Thus, iteration of the main loop stops whenever either the rows or the columns are completely partitioned into C and F. Steps 5 and 6 ensure that whatever rows or columns are left are placed into C, so that both the A_{ff} and A_{cc} blocks are square.

The column measures computed in step 4(a) are chosen to reflect how much of a roadblock column j poses to finding a diagonally dominant A_{ff} block. If column j has many entries that are large relative to the largest in their row, then it does, indeed, make dominance hard to achieve. One possible downside to this particular measure, however, is the inability to distinguish between good potential pivots (entries in column j that are the largest in their rows) and entries that effectively block dominance (entries that are very close to, but slightly smaller than, the largest in their rows). It may be possible to neglect entries identified as potential pivots in this algorithm, leading to a more effective measure, but we have not experimented with this option. Another option apparent from the discussion above is that this choice of measure is simply one of many possible averages of the changes in the $\hat{\theta}_i$. Other measures (including other ℓ_p norms, or other averages, such as the harmonic average) are also possible but have not been experimented with.

3.3. Implementation details. While Algorithm 3 describes the important features of the partitioning algorithm, it is intended only to convey the general outline of the scheme. Here, we provide full details of the scheme, as implemented, and as tested in section 4. In particular, we consider the choices made in order to achieve low computational complexity, as well as the details of updating the row-wise quantities k_i, l_i, and r_i. By tracking when these updates are made, the first opportunities to move a row from U_{row} to F_{row} or C_{row} are more easily identified when the relevant quantities are updated. This detailed implementation described here is presented as Algorithm 4 in the appendix.

For ease of computation, two preprocessing stages are added. First, the transpose of A is computed (step 3 of Algorithm 4). While not explicitly necessary, this is convenient because of our use of a compressed sparse row (CSR) storage scheme for the matrix A (and A^T). Thus, for loops such as in step 4(c) of Algorithm 3, we have easy access to the set, $\{i : a_{ij^\star} \neq 0\}$, as the adjacency list of row $j^\star$ in A^T. For this

reason, we use the notation $\text{Adj}^T(j) = \{i : a_{ij} \neq 0\}$ in Algorithm 4. Additionally, the adjacency lists for each row i in A are sorted in decreasing order by $|a_{ij}|$ (step 4(a)). This is done for convenience in updating k_i; instead of needing to search over all entries in $\text{Adj}(i)$ for the next smallest entry that is eligible to dominate row i when the previous k_i is removed from U_{col}, the sorted adjacency lists allow us to simply scan the list from the entry for the old k_i onward, until we find the next column in $\text{Adj}(i) \cap U_{\text{col}}$. In this manner, the adjacency list for row i is scanned from start to finish at most once over all of the updates to k_i before row i is removed from U_{row}.

The extra operations needed to update k_i, l_i, and r_i first arise in step 3(e) of Algorithm 3 (step 4(f) of Algorithm 4). Now, instead of just removing row i and column k_i to their respective F sets, the unsorted rows, m, in $\text{Adj}^T(k_i)$ are also examined. If column k_i dominated row m, then a new dominating column for row m is found from those still in U_{col}. If no such column exists, then row m is easily discarded to C_{row}, as there is no pivot that can pair with it to give a diagonally dominant row in A_{ff}. The A_{ff} row-norm for row m, r_m, is incremented by the value in the added column, and if row m is no longer sufficiently dominant relative to its F columns, it is also moved into C_{row}.

The initial computation of the column measures w_j for $j \in U_{\text{col}}$ is now broken into a separate loop (step 5), and these measures are then updated in the main loop whenever a row is sorted. To manage these measures efficiently and enable easy approximation of the column with maximal measure, a bucket-sort style data structure is used. Within the first loop (step 4 of Algorithm 4), an upper bound on the measures is calculated as the maximum length of an adjacency list in U_{row}. The interval from 0 to this upper bound is then divided equally into a fixed number of buckets, where each bucket corresponds to a doubly linked list of elements representing columns from U_{col} whose measures lie in the appropriate subinterval. A separate list of pointers is maintained, indexed by column, pointing to the list element corresponding to each column. In this way, columns are easily removed from the data structure when they are moved into C_{col} or F_{col}, by accessing the element through the column-indexed pointer list and updating the bucket from which it came using the doubly linked list properties. Columns are also easily moved when their measure is updated by removing the corresponding element from its previous list and adding it to the head of its new list. An approximately maximal column is also easily found as the first item in the nonempty bucket with largest index.

The most significant changes to the algorithm occur in the main loop, step 4 (step 6 and, in particular, step 6(b) of Algorithm 4). Now, when column $j^\star$ is removed, we first check if it dominated any row $i \in U_{\text{row}} \cap \text{Adj}^T(j^\star)$. If so, we update k_i as before and check if the new k_i is large enough to continue considering row i as a potential F row. If not, row i is moved into C_{row}. If, however, the intersection of $\text{Adj}(i)$ and U_{col} is nonempty, removing row i from U_{row} necessitates an update to the column measures for each $j \in \text{Adj}(i) \cap U_{\text{col}}$. Even if row i is not to be removed, these column weights change whenever the dominating column changes, and so, in step 6(b)ii.D of Algorithm 4, the measures for updating any columns are adjusted. The final major step, step 6(b)iii, makes row i and column k_i a pivot in A_{ff} if row i is sufficiently dominated by column k_i. Here, we again update the measure of any column in U_{col} that is affected by the removal of row i from U_{row}. Additionally, for any row that is dominated by column k_i, a new dominating column is found, the intersecting column weights are adjusted, and the row is tested for suitability as a potential F row.

4. Numerical results. We consider several sets of test problems for the proposed partitioning scheme, building on problems tested in previous papers. For each set of tests, we compare the performance of ARMS preconditioners based on the new nonsymmetric partitioning scheme to other effective preconditioners, including ARMS with other choices of partitioning schemes. All tests were run on a dual-processor 3.0GHz Intel Xeon workstation with 2GB of RAM.

In all of these examples, parameters are chosen (whenever possible) to allow easy comparison with previous work [22, 28]. It is important to realize, however, that there is some sensitivity in the performance to these choices of parameters. In general, we find that the choice of θ has a strong influence over the complexity of the algorithm but a milder influence on iteration counts. As complexity is generally lowest when the fine-grid block is largest, but still retains some diagonal dominance, θ should be taken close to 0.5, as it is in the results that follow. As in classical dual-threshold ILUT factorizations [26], the best performance is seen when the maximum fill-per-row parameter plays only a small role in keeping the factors sparse. Thus, we aim to adjust the drop tolerance in such a way as to control both accuracy and sparsity of the factors, while using the maximum fill parameter only to ensure against catastrophic fill-in. Finally, our goal is to coarsen until the coarsest-grid problem is quite small; thus, we demand high accuracy in its incomplete factorization for all test problems. While these guidelines also apply to the symmetric partitioning algorithm [22], the parameter choices for the two-stage partitioning algorithm are quite different. We refer the reader to [28] for a discussion of these choices in that context.

4.1. Symmetric PDE based problems. To compare the nonsymmetric partitioning schemes with their symmetric counterpart (as described in section 2.3), we first consider some test problems from [22]. These matrices, bilinear finite-element discretizations of the second-order elliptic PDE $-\nabla \cdot K(x,y)\nabla p(x,y)$ on regular rectangular meshes of $[0,1]^2$, are sparse, symmetric, and positive-definite. Aside from confirming that the nonsymmetric partitioning scheme preserves the desired properties of the symmetric and definite case, these tests allow another observation. The classical approach to designing solvers for symmetric problems has been to take advantage of symmetry as much as possible. The threshold based incomplete factorizations considered here are, however, naturally nonsymmetric (as thresholding is performed row-wise). Thus, comparing ARMS performance using the symmetric partitioning approach (where symmetrized ILUT factorizations are necessary) and using the nonsymmetric partitioning approach (with full (nonsymmetric) ILUT factorizations) may be done to compare the benefits of retaining symmetry versus taking full advantage of a naturally nonsymmetric factorization.

We consider four choices for the diffusion coefficient, $K(x,y)$. Constant $K(x,y) = 1$, the Poisson equation, is a standard test problem for sparse matrix solvers, particularly for multigrid and its algebraic variants. A smoothly varying, but nonconstant, isotropic coefficient, $K(x,y) = 10^{-8} + 10(x^2+y^2)$, is also considered. A more difficult problem, where $K(x,y) = 10^{-8}$ on 20% of the elements of the grid, chosen randomly, and $K(x,y) = 1$ on the remainder, results in degradation of performance of standard geometric multigrid approaches, but is typically solved efficiently by AMG and multilevel approaches. Finally, we consider the case of a constant, but anisotropic, diffusion coefficient, $K(x,y) = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 0.01 \end{smallmatrix}\right]$, which causes difficulty for many algebraic solvers and preconditioners, including the two-level ARMS preconditioner considered in [22].

Results for the symmetric ARMS preconditioner based on the symmetric greedy coarsening strategy of [22] are shown in Table 1 (Table 7 of [22]). For each choice of

TABLE 1

Performance of ARMS with symmetric partitioning and symmetrized ILUT on test matrices from discretizations of $-\nabla \cdot K(x,y)\nabla p(x,y)$ *for the given* $K(x,y)$ *and uniform grids on* $[0,1]^2$. *Results marked with a * indicate that the residual reduction criterion was a relative factor of* 10^4 *instead of* 10^6.

Coefficient	Grid	α	Lev.	c_B	t_{setup}	t_{solve}	Iter.
$K(x,y)=1$	128^2	0.01	2	2.59	0.3	0.3	28
	256^2	0.01	2	2.65	1.5	2.5	44
	512^2	0.01	2	2.68	12.7	24.5	82
	1024^2	0.1	2	1.03	159.1	34.2*	46*
$K(x,y)=10^{-8}+10(x^2+y^2)$	128^2	0.01	2	2.60	0.3	0.4	31
	256^2	0.01	2	2.65	1.5	3.4	56
	512^2	0.01	2	2.68	12.7	31.7	97
	1024^2	0.1	2	1.03	159.6	40.6*	52*
random $K(x,y)$	128^2	0.01	3	1.40	0.2	0.4	32
	256^2	0.01	3	1.41	0.7	2.5	45
	512^2	0.01	3	1.42	3.1	25.1	83
	1024^2	0.01	3	1.42	13.5	12.4*	17*
anisotropic $K(x,y)$	128^2	0.01	5	1.61	0.2	0.3	26
	256^2	0.01	5	1.62	0.8	2.3	42
	512^2	0.01	5	1.63	3.3	17.3	65
	1024^2	0.01	5	1.63	14.9	6.9*	10*

the coefficients, $K(x,y)$, we consider several levels of uniform mesh refinement. For these results, θ is chosen to be 0.55, a symmetrized ILUT algorithm is used where the L factors on all grids but the coarsest are computed with drop tolerance α, as reported in Table 1, and a maximum of twice the average number of nonzeros-per-row of A_{ff} are allowed in each row of its L factor. Coarsening is continued until either the coarsest-grid operator is θ-dominated by its diagonal or has fewer than 10 degrees of freedom. The coarsest-grid operator is then factored using ILUTP, with a drop tolerance of 10^{-5} and a maximum of twenty times the average number of nonzeros per row of this coarsest Schur complement allowed per row of its L factor. For these examples, a more efficient solver results from not rescaling the A_{ff} matrices prior to computing their ILUT factors, as discussed in [28]. In Table 1, we report the number of levels used by the symmetric ARMS preconditioner, the complexity, c_B, of the preconditioner (defined as the total number of nonzeros stored on all levels of the ARMS preconditioner divided by the number of nonzeros in the original fine-scale operator, A), the setup and solve times, and the number of iterations for preconditioned GMRES to reduce the residual by a relative factor of 10^6 (or, for the problems marked by a *, 10^4).

Table 2 shows results for the new nonsymmetric partitioning based ARMS preconditioners on these problems. For the first three problems, the same parameters are used for the nonsymmetric partitioning as are used for the symmetric partitioning. For the anisotropic $K(x,y)$, using the same parameters as the symmetric partitioning resulted in very slow convergence of ARMS with nonsymmetric partitioning. This is likely related to the fact that the average size of the nonzero offdiagonal entries for the discrete anisotropic operator is much closer to the size of its diagonal than for the isotropic problems, and only a very particular choice of coarse degrees of freedom leads to the semicoarsened grids that are known to be most effective for this problem. So, in Table 2, we use $\theta = 0.5$ instead of $\theta = 0.55$ for the anisotropic problem, which results in much better performance. Table 2 also includes two extra data points, detailing the performance of the preconditioners for the two cases from Table 1 where

TABLE 2

*Performance of ARMS-preconditioned GMRES based on the nonsymmetric partitioning strategy of section 3.2 on test matrices from discretizations of $-\nabla \cdot K(x,y)\nabla p(x,y)$ for given $K(x,y)$ and uniform grids on $[0,1]^2$. Results marked with a * indicate that the residual reduction criterion was a relative factor of 10^4 instead of 10^6.*

Coefficient	Grid	α	Lev.	c_B	t_{setup}	t_{solve}	Iter.
$K(x,y) = 1$	128^2	0.01	3	2.08	0.3	0.3	24
	256^2	0.01	3	2.17	1.2	2.4	38
	512^2	0.01	3	2.22	5.3	20.3	67
	1024^2	0.01	3	2.24	25.2	11.9*	14*
	1024^2	0.1	2	2.31	264.9	22.0*	28*
$K(x,y) = 10^{-8} + 10(x^2 + y^2)$	128^2	0.01	3	1.96	0.3	0.4	28
	256^2	0.01	3	1.96	1.1	3.5	50
	512^2	0.01	3	2.12	5.0	27.9	83
	1024^2	0.01	3	1.99	26.1	15.8*	18*
	1024^2	0.1	2	1.76	81.8	25.5*	32*
random $K(x,y)$	128^2	0.01	3	1.72	0.3	0.4	28
	256^2	0.01	4	1.52	1.2	2.8	41
	512^2	0.01	4	1.52	5.3	26.1	72
	1024^2	0.01	4	1.53	24.1	15.7*	14*
anisotropic $K(x,y)$	128^2	0.01	7	2.19	0.5	0.4	28
	256^2	0.01	6	2.24	2.0	3.6	47
	512^2	0.01	7	2.23	8.2	28.2	76
	1024^2	0.01	6	2.28	34.6	11.3*	11*

memory limitations required the choice of $\alpha = 0.1$. For these problems, we include results for both $\alpha = 0.01$ and $\alpha = 0.1$.

For the first two problems (with smooth isotropic coefficients), using ARMS based on nonsymmetric partitioning is slightly more efficient than the symmetric partitioning scheme. This can be seen, in particular, on the larger meshes, where the solve times for the 512×512 grid examples are slightly better than those of the symmetric approach, but the setup times scale much better. As a result, the overall performance of the nonsymmetric partitioning based approach with $\alpha = 0.01$ is significantly better than the performance of either partitioning scheme with $\alpha = 0.1$. For these two problems with $\alpha = 0.01$, the preconditioner complexity is better across all grids using the nonsymmetric partitioning scheme. For the second two problems, however, the performance of ARMS using the symmetric partitioning scheme is slightly better than that of ARMS using the nonsymmetric partitioning approach. In particular, the lower operator complexities for these two problems lead to notably faster solve times even though the iteration counts are comparable (in fact, the nonsymmetric partitioning algorithm results in fewer iterations for all grid sizes for the randomly selected coefficient problem). In any case, the slight difference between these results indicates that there is little benefit in abandoning the inherent advantages of a symmetric preconditioner.

An additional comparison is made with the two-stage nonsymmetric permutation algorithm of [28], as described in section 3.1. For this method, the parameters are chosen as in [28, section 6.1]; namely, a dominance threshold of 0.1 (now relative to the maximum dominance ratio), and drop tolerances of 0.001 in the computation of the ILUT factors, L and U, of A_{ff}, 0.01 in the computation of $L^{-1}A_{fc}$ and $A_{cf}U^{-1}$, 0.001 in the Schur complement computation, and 0.01 on the coarsest level are used. Fill-in is restricted to ten times the average number of nonzeros per row of A_{ff} in all thresholded computations on the fine scales, and five times the average number of

TABLE 3
*Performance of ARMS-preconditioned GMRES based on the two-stage nonsymmetric partitioning approach of [28] on test matrices from discretizations of $-\nabla \cdot K(x,y)\nabla p(x,y)$ for given $K(x,y)$ and uniform grids on $[0,1]^2$. Results marked with a * indicate that the residual reduction criterion was a relative factor of 10^4 instead of 10^6.*

Coefficient	Grid	Lev.	c_B	t_{setup}	t_{solve}	Iter.
$K(x,y) = 1$	128^2	6	3.26	0.7	0.4	25
	256^2	7	3.37	9.5	3.8	46
	512^2	7	3.44	155.7	29.9	76
	1024^2	7	3.48	2546.7	16.3*	14*
$K(x,y) = 10^{-8} + 10(x^2+y^2)$	128^2	5	2.85	0.5	0.4	25
	256^2	6	2.92	5.6	3.6	44
	512^2	6	2.94	80.2	31.9	76
	1024^2	7	2.95	1279.1	28.4*	20*
random $K(x,y)$	128^2	7	2.44	0.4	0.4	26
	256^2	7	2.48	4.2	3.3	42
	512^2	7	2.50	55.9	28.5	69
	1024^2	7	2.51	886.7	22.5*	15*
anisotropic $K(x,y)$	128^2	8	2.84	0.8	0.4	24
	256^2	8	2.82	10.4	3.2	41
	512^2	8	2.82	172.6	24.5	66
	1024^2	8	2.80	2836.2	8.6*	8*

nonzeros per row of the Schur complement on the coarsest scale. Results for the same four model problems and grid sizes are shown in Table 3.

The performance of the two-stage algorithm, shown in Table 3, is interesting because of its contrast to the other results. For each problem, the two-stage algorithm creates hierarchies with more levels, yet results in preconditioner complexities that are, in all cases, bigger than those of either of the other two methods. The setup times clearly do not scale linearly with the increases in problem sizes, yet the iteration counts are similar (and, in some cases, better) than those of the other approaches. Because the LU solves require more operations, however, iteration times with this approach are generally larger than those of the others.

4.1.1. Performance profiles. Comparing the raw results, such as those shown in Tables 1, 2, and 3, gives a good picture of the performance of these preconditioners on individual problems; however, a better sense of the overall performance may be obtained by considering performance profiles, as in [14], instead of tables to present the results in a more compact manner.

Given a measurable solver performance characteristic, such as total time to solution, number of iterations, or preconditioner complexity, performance profiles are a useful tool for comparing different solvers in terms of their measured characteristics for a given set of problems. Let S be the set of solvers and P be the set of problems. For a solver, $i \in S$, and a problem, $j \in P$, take s_{ij} to be the performance characteristic measured for solver i on problem j (e.g., the total time taken by solver i to reduce the residual of problem j by a relative factor of 10^6). For a particular problem, j, the best observed performance, in terms of this chosen characteristic, is

$$\hat{s}_j = \min_{i \in S}\{s_{ij}\}.$$

For each solver, i, we can then define

$$p_i(\alpha) = \frac{|\{j : s_{ij} \leq \alpha \hat{s}_j\}|}{|P|}$$

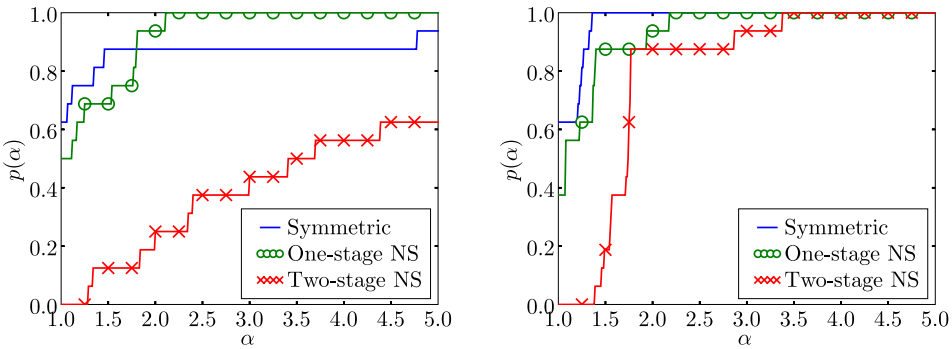


FIG. 1. *Performance profiles for total time to solution (left) and preconditioner complexity (right) for the ARMS preconditioners based on the three partitioning algorithms on the symmetric PDE based problems. Solution is taken to mean a reduction of the residual by a relative factor of* 10^6 *on all grids but* 1024×1024*, where a reduction by a relative factor of* 10^4 *is used.*

as the fraction of the problems in P for which the measured characteristic of solver i is within a relative factor of α of the optimal. If, for any reason, solver i fails on problem j, we take $s_{ij} = \infty$ (and, further, if $s_{ij} = \infty$ for all i, we take $\hat{s}_j = 0$, as no solver can be said to be "optimal"). Thus, $p_i(1)$ is the fraction of problems for which solver i is the best, and $\lim_{\alpha \to \infty} p_i(\alpha)$ is the fraction of problems for which solver i succeeded. A very good solver (relative to the chosen characteristic) is then one for which $p_i(\alpha)$ is largest (closest to one) for small values of α.

Performance profiles for the preconditioners based on the three partitioning algorithms, whose results are given in Tables 1, 2, and 3, are shown in Figure 1. At left is the profile for total time to solution, including setup time plus time required to reduce the residual by a relative factor of 10^6 (10^4 for 1024×1024 grid problems). While ARMS based on symmetric partitioning is the fastest overall in 10 of 16 cases, the new single-stage nonsymmetric partitioning algorithm is seen to be competitive, solving 8 of 16 problems in the fastest time (2 ties in overall time occur). The preconditioner based on the nonsymmetric partitioning solves all problems within a factor of just over 2 of the individual optimal times. The slow setup times of the two-stage nonsymmetric partitioning approach are reflected in the slow growth of its profile. On the right of Figure 1 is the profile for the preconditioner complexities, c_B. While the best overall complexities are seen for the symmetric partitioning approach, the single-stage nonsymmetric partitioning algorithm offers the best complexity for six problems and is seen to be quite competitive with the symmetric approach. The two-stage nonsymmetric partitioning algorithm is seen to yield complexities that are, in all cases, at least 40% larger than the best complexity.

4.2. General sparse symmetric systems. The second set of test problems considered in [22] was drawn from the collection of general, sparse, symmetric, and positive-definite matrices considered in [18, 19]. This set of matrices is the subset of the positive definite problems considered in [18] for which full data is available (discounting problems for which only a nonzero pattern is available) with fewer than 3 million nonzero entries (plus the problem, OILPAN, which has 3.5 million nonzeros, but for which a low preconditioner complexity made convergence possible). Here, we consider the same test set; matrix names, dimensions, and numbers of nonzero entries are listed in Table 4.

TABLE 4
Names, dimensions, and numbers of nonzero entries for the general sparse symmetric matrix test set selected from the RAL collection [18, 19].

Name	n	nnz
bodyy4	17546	121938
bodyy5	18589	129281
bodyy6	19366	134748
bcsstk18	11948	149090
t2dah_a	11445	176117
bcsstk25	15439	252241
OBSTACLE	40000	277600
JNLBRNG1	40000	279200
MINSURFO	40806	285230
bcsstk17	10974	428650
CVXBQP1	50000	449968

Name	n	nnz
t3dl_a	20360	509866
finan512	74752	596992
GRIDGENA	48962	610008
gyro_k	17361	1021159
bcsstk36	23052	1143140
msc23052	23052	1154814
msc10848	10848	1229778
cfd1	70656	1828364
vanbody	47072	2336898
nasasrb	54870	2677324
OILPAN	73752	3597188

Solver parameters are chosen consistently with previous work. For the symmetric permutation scheme, $\theta = 0.55$, drop tolerances are 0.01 with the number of nonzeros per row of the symmetrized ILUT factor limited to five times the average number of nonzeros per row in A_{ff} on all but the coarsest scale, where the drop tolerance is 0.0001 and twenty times the average number of nonzeros per row of the coarsest Schur complement are allowed in each row of its symmetrized ILUT factor. A maximum of 10 levels (plus the coarsest) are allowed, and the A_{ff} blocks are not rescaled before the ILUT. For the single-stage nonsymmetric partitioning approach, θ is chosen as 0.51, and a maximum of 50 levels are allowed; this partitioning scheme tends to choose smaller coarse grids than the symmetric partitioning, and so these parameters are chosen to allow a better comparison. Otherwise, parameters for the single-stage nonsymmetric partitioning approach are the same as for the symmetric partitioning scheme. For the two-stage nonsymmetric partitioning scheme, parameters are chosen to be the same as in the previous section and in [28, section 6.1]. A maximum of 100 levels are allowed, with a dominance threshold of 0.1 (relative to the most dominant row). Diagonal rescaling before the ILUT of A_{ff} is used, and drop tolerances of 0.01 and 0.001 with maximum fill factors of 10 are used on fine scales. On the coarsest scale, the drop tolerance is 0.01, with five times the average number of nonzeros per row of the coarsest Schur complement matrix allowed in each row of its symmetrized ILUT factor.

Performance profiles for total time to solution (at left) and complexity (at right) for the preconditioners resulting from using the symmetric and one- and two-stage nonsymmetric partitioning algorithms are shown in Figure 2. For these profiles, a problem is deemed to have been solved if the ℓ_2 norm of the residual is reduced by a relative factor of 10^6 within 1000 iterations. For problems that failed to meet this criteria, the solution time is taken to be infinite. Notice that each method is fastest for several problems and that, in general, the symmetric partitioning based approach is most successful. In particular, while the one-stage nonsymmetric partitioning approach is close to the fastest for about 20% of the total problems (noting that none of the solvers is successful on 4 of the problems), the symmetric and two-stage nonsymmetric partitioning schemes showed about equal overall performance. In terms of complexity (as seen on the right of Figure 2), there is little differentiation between the methods. The symmetric partitioning is generally the best, followed by the one-stage algorithm, but the difference is small.

Performance of classical AMG [8, 25] for these problems was considered in [22].

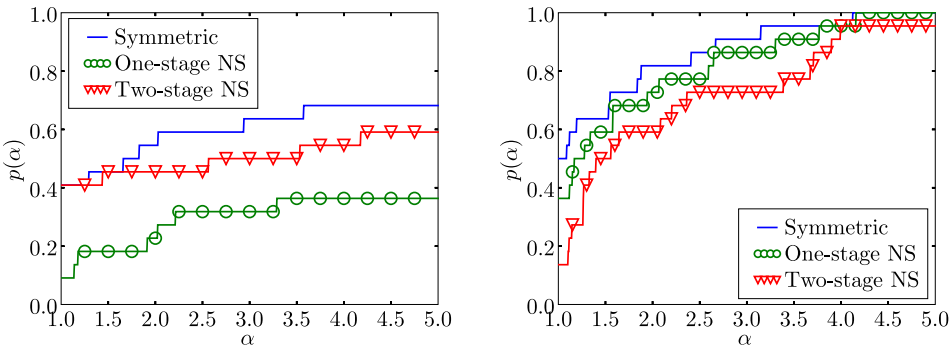


FIG. 2. *Performance profiles for total time to solution (left) and complexity (right) for the preconditioners resulting from using the three partitioning algorithms on the general spare symmetric problems. Solution is taken to mean a reduction of the residual by a relative factor of* 10^6*, within* 1000 *iterations.*

There, it was shown that a naive application of AMG performs quite poorly for this general selection of problems (only 4 of 22 problems converged in fewer than 150 iterations). This behavior emphasizes the benefits of the ARMS approach over AMG; while AMG is typically faster when it works well (including for problems such as those considered in section 4.1), ARMS is more robust as a black-box solver. For this reason, we do not include a detailed comparison of our approach with AMG for general sparse matrices.

4.3. Convection-diffusion problems. As a first set of nonsymmetric test problems, we consider simple finite difference discretizations of the convection-diffusion operator in two dimensions,

$$-\Delta u + a u_x = f,$$

for a variable wind-speed a. Using the standard second-order five-point stencil for the Laplacian term, performance should be close to that seen in section 4.1 for small values of a, no matter how the lower-order derivative has been discretized. For large values of a, however, it is important to discretize the u_x term using an upwind differencing (as opposed to a central-difference approximation) both for accuracy of the solution and to maintain nice (i.e., M-matrix) properties of the linear system.

Table 5 shows the results for both the single-stage partitioning algorithm presented here and the two-stage partitioning algorithm of [28] applied to upwind finite difference discretizations of this problem for values of a ranging from 1.0 to 10^4 on grids of 129×129, 257×257, and 513×513 degrees of freedom. For the single-stage partitioning algorithm, we see low computational complexities for all winds, on all grids. There is some growth in iteration count as grid sizes increase, but performance appears to improve as a increases. Setup times for the two-stage partitioning are much higher—sufficiently so that even though the solve times are, in general, shorter, the overall time to solution is longer for every problem than using the new partitioning algorithm. Furthermore, for all but 3 problems, the preconditioner complexities, c_B, for the two-stage partitioning algorithm are more than twice those for the single-stage partitioning.

Similar results are seen for the unstable discretization, as shown in Table 6. Once again, setup times and preconditioner complexities for the two-stage approach are

TABLE 5
Complexities, timings, and iteration counts for single-stage and two-stage ARMS partitioning approaches for the convection-diffusion problem discretized with upwind differences.

		One-stage NS				Two-stage NS			
a	grid	c_B	t_{setup}	t_{solve}	# iters.	c_B	t_{setup}	t_{solve}	# iters.
1.0	129^2	2.14	0.2	0.3	33	4.42	0.6	0.3	21
	257^2	2.21	0.7	3.0	60	4.60	8.3	2.0	34
	513^2	2.23	3.4	33.9	111	4.68	138.8	16.4	57
10.0	129^2	2.16	0.2	0.3	33	4.42	0.6	0.2	20
	257^2	2.20	0.7	3.3	64	4.60	8.2	2.0	34
	513^2	2.22	3.4	36.9	117	4.84	138.7	18.5	62
10^2	129^2	2.24	0.2	0.2	19	2.77	0.6	0.2	17
	257^2	2.28	0.7	1.8	40	4.80	8.3	1.5	26
	513^2	2.26	3.5	23.6	87	4.78	138.3	15.7	55
10^3	129^2	1.95	0.2	0.1	7	5.68	0.6	0.1	10
	257^2	2.15	0.8	0.7	18	2.73	8.6	0.7	14
	513^2	2.02	2.8	4.1	23	6.11	141.9	8.5	29
10^4	129^2	1.52	0.2	0.1	5	4.18	0.6	0.1	8
	257^2	1.57	0.8	0.3	9	4.83	8.5	0.4	8
	513^2	1.69	2.5	2.9	17	2.73	144.0	3.2	16

TABLE 6
Complexities, timings, and iteration counts for single-stage and two-stage ARMS partitioning approaches for the convection-diffusion problem discretized with central differences.

		One-stage NS				Two-stage NS			
a	grid	c_B	t_{setup}	t_{solve}	# iters.	c_B	t_{setup}	t_{solve}	# iters.
1.0	129^2	2.15	0.2	0.3	33	4.22	0.6	0.3	21
	257^2	2.21	0.7	3.0	60	4.49	8.3	2.0	34
	513^2	2.22	3.4	34.0	111	4.62	138.2	16.3	57
10.0	129^2	2.15	0.2	0.3	33	4.21	0.6	0.3	21
	257^2	2.21	0.7	3.3	64	4.61	8.3	2.2	36
	513^2	2.22	3.4	36.9	117	4.79	138.7	18.9	63
10^2	129^2	2.06	0.1	0.2	20	4.67	0.6	0.2	16
	257^2	2.32	0.7	1.8	40	4.80	8.3	2.2	35
	513^2	2.24	3.5	24.0	88	4.84	138.6	15.6	54
10^3	129^2	2.03	0.3	0.2	23	4.48	0.7	0.1	8
	257^2	2.08	1.0	1.5	32	3.63	8.5	0.6	12
	513^2	1.79	2.4	12.1	54	6.53	141.6	7.4	25
10^4	129^2	2.08	0.2	0.1	7	2.65	0.6	0.1	7
	257^2	2.19	0.9	0.6	17	3.57	8.7	0.5	10
	513^2	2.12	3.8	5.7	31	4.86	144.0	6.8	24

much worse than those for the new, single-stage partitioning algorithm of section 3.2. Interestingly, neither solver seems overly affected by the poor-quality discretization, which leads to non-M-matrices for $ah > 2.0$.

Comparison with a single-level ILUT preconditioner shows the efficiency of the ARMS approach. For the central-difference discretization, with $a = 10^4$, an ILUT factorization with drop tolerance of 10^{-3} and a maximum fill-in factor of 20 leads to preconditioner complexities of 14.85 on a 129×129 grid, 18.26 on a 257×257 grid, and 18.99 on a 513×513 grid. While such a preconditioner may be effective on a small grid (only 5 iterations are required to reduce the residual by a factor of 10^6 on the 129×129 grid), its effectiveness quickly diminishes. On the 257×257 grid, 182 iterations are necessary for the same reduction, while a reduction in residual of

TABLE 7
Complexities, timings, and iteration counts for AMG applied to the convection-diffusion problem discretized with upwind and central differences. The notation DIV denotes cases where the solver diverges.

		Upwind				Central			
a	grid	c_B	t_{setup}	t_{solve}	# iters.	c_B	t_{setup}	t_{solve}	# iters.
1.0	129^2	2.30	0.1	0.03	4	2.31	0.1	0.03	4
	257^2	2.31	0.3	0.1	4	2.31	0.3	0.1	4
	513^2	2.32	1.4	0.6	4	2.32	1.4	0.6	4
10.0	129^2	2.28	0.1	0.03	4	2.43	0.1	0.03	4
	257^2	2.31	0.3	0.1	4	2.30	0.3	0.1	4
	513^2	2.36	1.5	0.6	4	2.36	1.4	0.6	4
10^2	129^2	2.84	0.1	0.04	5	2.76	0.1	0.04	5
	257^2	2.65	0.3	0.1	4	2.60	0.3	0.2	4
	513^2	2.54	1.4	0.6	4	2.59	1.4	0.6	4
10^3	129^2	2.39	0.04	0.1	9	2.69	0.1	DIV	DIV
	257^2	3.07	0.3	0.3	9	2.86	0.3	0.4	10
	513^2	2.85	1.3	1.0	7	4.83	2.0	1.5	7
10^4	129^2	1.99	0.04	0.04	6	3.68	0.1	DIV	DIV
	257^2	2.08	0.2	0.2	7	3.80	0.3	DIV	DIV
	513^2	2.27	0.7	1.1	8	3.89	1.4	DIV	DIV

only 0.64% was reached in 500 iterations on the 513×513 mesh.

Because of the PDE nature of these systems, a comparison with the AMG algorithm is also appropriate. Results for AMG applied to both the upwind and central-difference discretizations are given in Table 7. Comparing this performance with that in Tables 5 and 6, we see that this performance is consistent with common perception. When AMG works, its performance far exceeds that of ARMS. For these problems, the discrete operators remain M-matrices for all a in the upwind discretization, and so AMG generally outperforms ARMS with either partitioning. For the central-difference discretization, however, AMG performance suffers significantly when ah is large, leading to divergence in the worst cases, whereas ARMS performance was insensitive to this change.

4.4. General sparse nonsymmetric systems. Another interesting nonsymmetric test set is from [28, section 6.1]. This collection of 58 matrices is taken from the Harwell–Boeing collection; those selected are all square matrices from the RUA (real, unsymmetric, assembled) collection that have a dimension of 500 or higher. Matrix names, dimensions, and numbers of nonzero entries are listed in Table 8.

As a comparison for these problems, we consider the performance of a single-level ILUTP preconditioner within GMRES, in addition to the two nonsymmetric partitioning based ARMS preconditioners. For the ILUTP solver, a drop tolerance of 0.0001 is used, with the number of nonzeros per row of L and U^T limited to twenty times the average number of nonzeros per row of A. For the ARMS codes, we use the same parameters as in the previous section, with the exception that A_{ff} is now always diagonally rescaled before it is factored by ILUT. An attempt was made to compare the performance of AMG with that of ARMS for these problems; however, applying a standard AMG algorithm to these matrices did not lead to a single instance of a converging solver.

Figure 3 shows the performance profiles for total time to solution (at left) and complexity (at right) for the problems of Table 8 using these three preconditioning strategies. It should be noted that for many of the smaller problems in this set, all

TABLE 8
Names, dimensions, and numbers of nonzero entries for the nonsymmetric Harwell–Boeing test set.

Name	n	nnz
BP1000	822	4661
BP1200	822	4726
BP1400	822	4790
BP1600	822	4841
BP200	822	3802
BP400	822	4028
BP600	822	4172
BP800	822	4534
FS5411	541	4285
FS5412	541	4285
FS5413	541	4285
FS5414	541	4285
FS6801	680	2646
FS6802	680	2646
FS6803	680	2646
FS7601	760	5976
FS7602	760	5976
FS7603	760	5976
GEMAT11	4929	33185
GEMAT12	4929	33111
GRE1107	1107	5664
GRE512	512	2192
JPWH991	991	6027
LNS3937	3937	25407
LNS511	511	2796
LNSP3937	3937	25407
LNSP511	511	2796
MAHINDAS	1258	7682
MCFE	765	24382

Name	n	nnz
NNC1374	1374	8606
NNC666	666	4044
ORANI678	2529	90158
ORSIRR1	1030	6858
ORSIRR2	886	5970
ORSREG1	2205	14133
PDE9511	961	4681
PORES2	1224	9613
PORES3	532	3474
PSMIGR1	3140	543162
PSMIGR2	3140	540022
PSMIGR3	3140	543162
SAYLR3	1000	3750
SAYLR4	3564	22316
SHERMAN1	1000	3750
SHERMAN2	1080	23094
SHERMAN3	5005	20033
SHERMAN4	1104	3786
SHERMAN5	3312	20793
SHL0	663	1687
SHL200	663	1726
SHL400	663	1712
STEAM2	600	13760
WATT1	1856	11360
WATT2	1856	11550
WEST0655	655	2854
WEST0989	989	3537
WEST1505	1505	5445
WEST2021	2021	7353

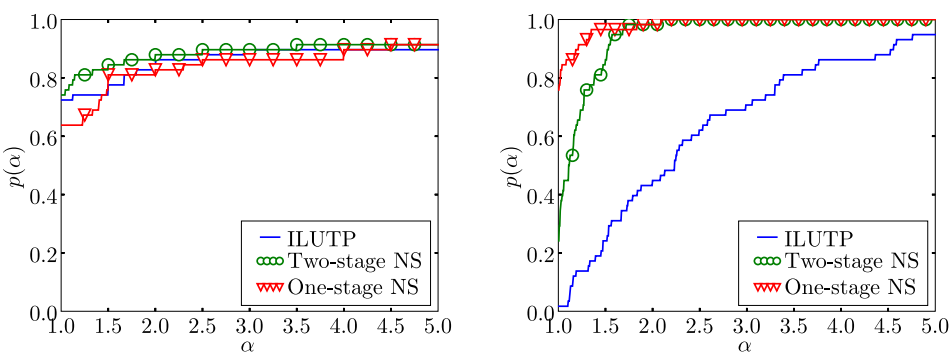


FIG. 3. *Performance profiles for total time to solution (left) and complexity (right) for the three nonsymmetric preconditioning strategies on the Harwell–Boeing test problems. Solution is taken to mean a reduction of the residual by a relative factor of* 10^6*, within* 1000 *iterations.*

three preconditioners require very little time, so that a relative difference of 50% may reflect only 0.01 seconds of CPU time.

For all three approaches, we see similar profiles for total time to solution. For the parameters chosen, ILUTP tends to require more significant setup time, while the ARMS approaches are more balanced between their setup and solve times. A

TABLE 9

Names, dimensions, and numbers of nonzero entries for the circuit and semiconductor device simulation matrices. In the left column are the Bomhof, Hamm, Schenk_IBMSDS, and Wang collections. At right are the matrices from the Schenk_ISEI set.

Name	n	nnz
circuit_1	2624	35823
circuit_2	4510	21199
circuit_3	12127	48137
circuit_4	80209	307604
add20	2395	17319
add32	4960	23884
bcircuit	68902	375558
hcircuit	105676	513072
memplus	17758	126150
scircuit	170998	958936
2D_27628_bjtcai	27628	442898
2D_54019_highK	54019	996414
3D_28984_Tetra	28984	599170
3D_51448_3D	51448	1056610
ibm_matrix_2	51448	1056610
matrix_9	103430	2121550
matrix-new_3	125329	2678750
swang1	3169	20841
swang2	3169	20841
wang1	2903	19093
wang2	2903	19093
wang3	26064	177168
wang4	26068	177196

Name	n	nnz
barrier2-1	113076	3805068
barrier2-2	113076	3805068
barrier2-3	113076	3805068
barrier2-4	113076	3805068
barrier2-9	115625	3897557
barrier2-10	115625	3897557
barrier2-11	115625	3897557
barrier2-12	115625	3897557
igbt3	10938	234006
nmos3	18588	386594
ohne2	181343	11063545
para-4	153226	5326228
para-5	155924	5416358
para-6	155924	5416358
para-7	155924	5416358
para-8	155924	5416358
para-9	155924	5416358
para-10	155924	5416358

significant difference, however, is seen in terms of the preconditioner complexities for these problems, on the right of Figure 3. Here, we see that the single-stage partitioning algorithm results in the least fill for nearly 70% of the problems considered. The longer setup times of ILUTP are reflected in its slowly increasing profile in this figure; it required more than twice the minimum memory on more than half of the problems in this set.

4.5. Circuit and semiconductor simulation matrices. As a final test set, we consider matrices arising in semiconductor and circuit simulation, obtained from the University of Florida Sparse Matrix collection [13]. The matrices are taken from five sets within this collection: Bomhof [6], Hamm, Schenk_IBMSDS [33], Schenk_ISEA [33], and Wang [23]. The dimensions and numbers of nonzero entries for these problems are listed in Table 9; the matrices range in dimension from 2395 to 181343, and in number of nonzero entries from 17 thousand to 11 million.

For these matrices, as in the previous section, we compare the preconditioners produced by the ARMS methodology using both the new single-stage nonsymmetric partitioning approach and the two-stage nonsymmetric partitioning approach of [28], along with an ILUTP approach. For the single-stage partitioning, $\theta = 0.5$ is chosen, along with a drop tolerance of 0.01 and maximum fill per row in the ILUT factors of five times the average number of nonzeros per row of A_{ff} on all but the coarsest level, where a drop tolerance of 0.0001 and a fill factor of ten are used. For the two-stage partitioning approach, we use the parameters from [28, section 6.5], with a relative diagonal-dominance threshold of 0.1, a drop tolerance of 0.01, and an allowable fill factor of three on all but the coarsest grid, where a drop tolerance of 10^{-5} and fill factor of twenty is used. For both preconditioners, we allow rescaling of A_{ff} on all

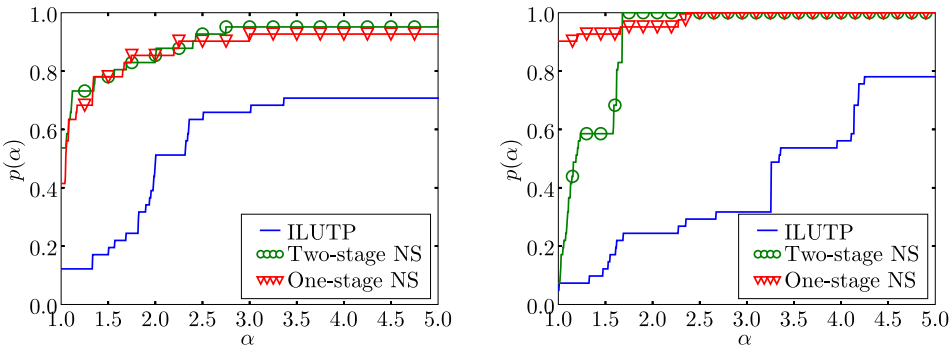


FIG. 4. *Performance profiles for total time to solution (left) and complexity (right) for the three nonsymmetric preconditioning strategies with the circuit and semiconductor device simulation matrices. Solution is taken to mean a reduction of the residual by a relative factor of* 10^6*, within* 1000 *iterations.*

grids before the ILUT factors are computed. For the ILUTP preconditioner, a drop tolerance of 0.0001 and a maximum fill factor of twenty are used.

The performance profiles for these problems, shown in Figure 4, indicate a much bigger gap between the ARMS approaches and the ILUTP preconditioner than is seen for the smaller problems of section 4.4. In terms of total time, the performance profiles for the two ARMS based approaches are quite similar, with the single-stage approach being fastest for roughly 40% of the test problems, while the two-stage partitioning yields the fastest solver for approximately 55% of the test problems. The ILUTP preconditioner is notably slower in terms of total time to solution, primarily because of its increased setup costs.

Looking at these results in more detail than is shown in the performance profiles, we see that the difference between the two ARMS based approaches is significant for some of the problems in this test set. On the matrix circuit_3, from the Bomhof set, the preconditioner based on the single-stage approach failed to converge within 1000 iterations, while the two-stage preconditioner converged, but slowly, in 723 iterations. For the matrix circuit_4, both approaches converged in 9 iterations (in about 0.3 seconds), but the single-stage approach needed only 0.5 seconds for setup, compared to 1.2 seconds for the two-stage approach. For the matrix 3D_28984_Tetra, from the Schenk_IBMSDS set, the two-stage preconditioner did not converge within 1000 iterations, yet the single-stage preconditioner converged in only 1 iteration, requiring 0.5 seconds to set up a successful preconditioner, compared to 0.4 seconds for the unsuccessful approach. On the matrix scircuit, from the Hamm set, the new partitioning approach yielded a preconditioner that converged in 32 iterations (and 3.8 seconds) compared to 60 iterations (and 9.3 seconds) for the two-stage approach, yet required only slightly more setup time, 1.4 seconds versus 1.2 seconds. For the wang3 and wang4 matrices, the single-stage partitioning approach yielded preconditioners that are faster both in setup and iteration, yielding a total speedup of over 25%. For many of the problems, however, the differences are less significant, with the single-stage preconditioner being approximately 10% faster on the barrier2 problems, and the two-stage preconditioner approximately 10% faster on the para problems, both from the Schenk_ISEI set.

Some of the speedup seen in iteration times for the single-stage partitioning pre-

conditioners is due to improved preconditioner complexities, as seen on the right of Figure 4. For both ARMS based approaches, the preconditioner complexities are quite good, below one for many problems, but the new partitioning scheme leads to overall lower complexities. Once again, the memory requirements of the ILUTP preconditioners are much higher; for this test set, ILUTP required over three times the amount of memory of the optimal solver on over two-thirds of the problems. These increased memory requirements do not result in large iteration times, as the constructed preconditioners require only a few iterations to sufficiently reduce the residual, but are reflected in the large gap in setup times (and, thus, total solution times) between the ILUTP and ARMS approaches shown on the left of Figure 4. While overall time to solution is an important measure of these preconditioners, the increased memory requirements of the ILUTP approach should not be ignored, especially for large matrices such as these, where the feasibility of storing such large preconditioners must be considered.

5. Ordering the A_{ff} block. While much effort has been focused on the partitioning of rows and columns within the ARMS algorithm, little consideration has been given to the ordering of the rows within the partitions. While ordering of the coarse-scale block has little practical effect, reordering of the A_{ff} block could result in a significant change in the sparsity of the resulting ILUT factors, as is the case in sparse direct methods (see, for example, [17]). Here, we extend the initial study of [29], where it was seen that the use of standard reordering techniques from sparse direct solvers may also result in improvements to the overall efficiency of the ARMS algorithm.

The efficiency of the ARMS process rests on the accurate approximation of the action of the inverse of A_{ff} by its ILUT factors. While diagonal dominance of A_{ff} is an important consideration in being able to find sparse, accurate ILUT factors, it is not the only requirement. Indeed, even if the graph of A_{ff} allows exact LU factors to be defined with no fill beyond its nonzeros, this can occur only if A_{ff} is appropriately ordered. As in sparse direct methods, then, the sparsity of the ILUT factors depends somewhat on the ordering of the A_{ff} block, and reordering this block may be an effective remedy if the ARMS algorithm performs well but has a high complexity.

In this section, experiments with several standard techniques for reordering the A_{ff} block within ARMS are given. As a baseline for comparison, we consider the order in which the F rows are selected by whatever selection procedure is used, as is used in the results of section 4. There are two possible ways to evaluate the success of these reorderings. If the same ILUT tolerances are used for both the ordered and unordered systems, we can compare the combination of the ARMS fill factor (or preconditioner complexity, defined as the total number of nonzeros stored in the ARMS preconditioner on all levels divided by the number of nonzero entries in A) and iteration counts for the resulting solvers. Of particular interest, of course, is the total time to solution, which is, in effect, the product of these two values when computer architecture considerations are neglected. Alternately, the tolerances used to form the ILUT factors could be adjusted so that the ARMS preconditioner complexity using reordering (approximately) matches that of the preconditioner without reordering. In this situation, the relative performance of the two preconditioners can easily be compared based solely on iteration counts, as each iteration has roughly the same cost for both preconditioners. Because of the number of problems and variety of approaches considered, only the first sort of comparison is used here.

Several common reordering techniques are considered. From the METIS tool-

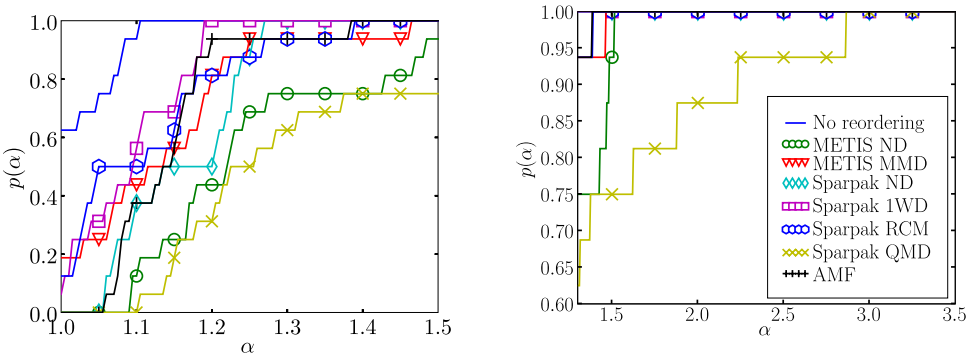


FIG. 5. *Performance profiles for total time to solution of the symmetric PDE based problems, using various reordering techniques with the single-stage nonsymmetric partitioning algorithm. Solution is taken to mean a reduction of the residual by a relative factor of* 10^6 *on all grids but* 1024×1024*, where a reduction by a relative factor of* 10^4 *is used. At right are profiles for* $\alpha \in [1.3, 3.5]$*; the detail at left shows* $\alpha \in [1.0, 1.5]$*; each reordering algorithm is denoted by the same line color and marker in both figures.*

kit [20], we consider the Nodal Nested Dissection and Multiple Minimum Degree orderings. From the SPARSPAK package [17], we consider both Nested and One-way Dissections, along with Reverse Cuthill–McKee and Quotient Minimum Degree reorderings. Finally, we consider the Approximate Minimum Fill technique of [1].

Adding these reorderings to the ARMS setup procedure requires very little change to the overall ARMS algorithm. The partitioning stage in ARMS already defines reorderings of the rows and columns of A (for any partitioning, and not just that described in Algorithm 3) and, implicitly, defines the fine-scale block, A_{ff}. The adjacency graph of A_{ff} is then extracted from that of A, symmetrized (nodes i and j are deemed to be adjacent if either $i \in \text{Adj}(j)$ or $j \in \text{Adj}(i)$), and passed to the reordering algorithm. The ordering computed here is then compounded with that from ARMS over the fine-grid rows and columns, after which the ARMS setup and iterations proceed as usual. In particular, no changes are needed to the partitioning algorithm itself other than the call to compute the reordering, and no changes are needed within the solve phase. The effects of reordering the A_{ff} block on a given level, however, compound on coarser scales; reordering the A_{ff} block affects both the sparsity and entries in the ILUT factors, which are then used to compute the approximate Schur complement on the coarse scale.

Performance profiles for the total time to solution (setup plus solve times) for the symmetric PDE based problems including reordering are shown in Figure 5. The plot at right shows the performance profile for $1.3 \leq \alpha \leq 3.5$, by which point all methods have $p(\alpha) = 1$. In general, the total times to solution using reordering are slightly larger than those without reordering, but almost all of the methods have total time within a factor of 1.5 of the fastest, as seen in the detail at left. Notice that while using the standard ARMS ordering is fastest for many problems, using the METIS Multiple Minimum Degree, Sparspak One-way Dissection, and Reverse Cuthill–McKee reorderings are each the fastest for some problems.

Results for iteration time and complexity are shown in Figure 6. For iteration time (at left), the performance curves are quite similar for all of the different methods. Using the standard ARMS ordering is fastest for only half of the problems, while the METIS Multiple Minimum Degree reordering results in the fastest solver for five

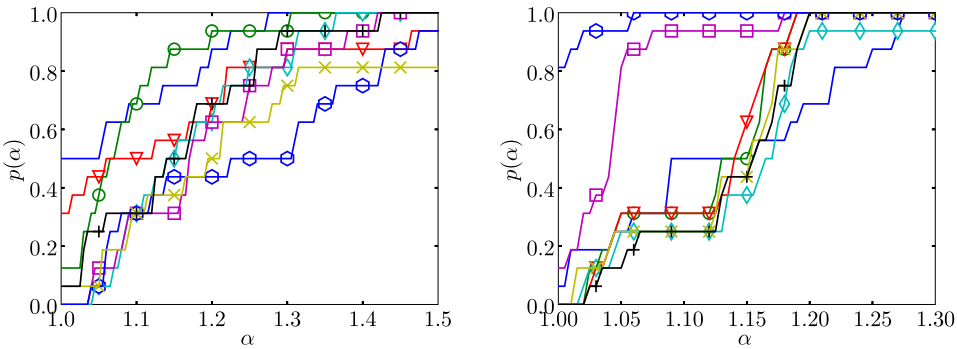


FIG. 6. *Performance profiles for solution time excluding setup (at left) and complexity (at right) for the symmetric PDE based problems, using various reordering techniques with the single-stage nonsymmetric partitioning algorithm. Solution is taken to mean a reduction of the residual by a relative factor of* 10^6 *on all grids but* 1024×1024*, where a reduction by a relative factor of* 10^4 *is used. Each reordering algorithm is denoted by the same line color and marker as in Figure* 5.

problems. Considering total time and iteration time separately distinguishes between solvers that have a slow overall time (as shown in Figure 5) because of the costs of the reordering stage and those that are slow because of some deficiency in the resulting preconditioner. In particular, note that the METIS ND reordering is one of the slowest in terms of total time to solution yet is among the fastest in iteration time. The added setup costs for this reordering obscure the actual performance gains in the solve phase.

On the right in Figure 6 are the performance profiles for preconditioner complexity, a measure of the memory requirements of the preconditioner itself. Overall, the effect of reordering is seen to be small (all preconditioner complexities are within 40% of the minima), but not trivially so. In the detail, we see that the Sparspak Reverse Cuthill–McKee and One-way Dissection algorithms are most effective in reducing the fill, but that all reorderings showed some benefit, in terms of lower complexity, over the original ordering by the ARMS process.

For the circuit and semiconductor problems, the single-stage partitioning scheme already results in low ARMS preconditioner complexities (below one for all but 10 of these problems and below one-half for 17 of them); very slight performance improvements can be expected from any of the reorderings considered. The SPARSPAK One-Way Dissection and Reverse Cuthill–McKee reorderings are most effective in terms of reducing total time to solution, and performance profiles for these two reorderings are shown in Figure 7. These results again highlight the importance of a fast reordering algorithm; nearly all preconditioner complexities are within 10% of the optimal approach for each problem, resulting in only a small possible savings in iteration time. The most effective approaches are those with low overhead in terms of setup costs.

The results in Figures 5, 6, and 7 suggest that there is little practical benefit in compounding the reordering produced by the ARMS partitioning algorithm with one of these techniques. Similar results are seen in [15], where reorderings for a single-level ILU preconditioner for the discrete Poisson problem were considered. Reordering in a multilevel procedure, such as ARMS, potentially has more significant consequences than in a single-level ILU or ILUT preconditioner, as the reordering and factors computed at one level affect choices made at coarser levels; however, no significant benefit

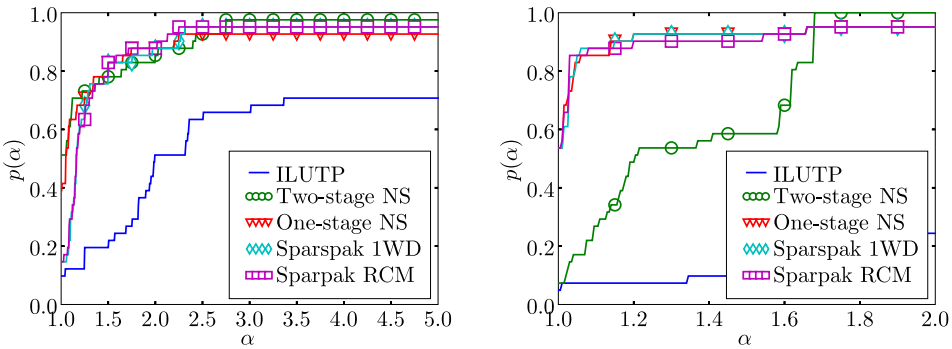


FIG. 7. *Performance profiles for total time to solution (left) and complexity (right) for the ILUTP-preconditioned GMRES and preconditioners using the nonsymmetric partitioning schemes for ARMS, possibly with reordering, on the circuit and semiconductor device simulation problems. Solution is taken to mean a reduction of the residual by a relative factor of* 10^6*, within* 1000 *iterations.*

(direct or indirect) was observed in our tests. In [5], it is observed that the benefit of reordering in incomplete factorizations depends on how far the matrix is from being symmetric and diagonally dominant. With our partitioning, the matrix A_{ff}, whose ILUT factors are computed, is always diagonally dominant, and thus our findings are consistent with those of [5].

6. Conclusions. We present a new greedy algorithm for partitioning matrices within a multilevel preconditioner such as ARMS. This algorithm combines the attractive features of two previous works, in that it is based on a direct greedy approach to finding the largest diagonally dominant A_{ff} block (as in [22]) but allows for nonsymmetric permutations, as these are known to be appropriate for problems with significant nonsymmetry in the operator [28]. We also demonstrate that further reordering of the fine-scale block of the partitioned system, in order to achieve a better approximation of this block by its ILUT factors while emphasizing sparsity, leads to little practical benefit because of the guarantee that our technique offers on the structure of this block.

The new partitioning approach generalizes the symmetric approach introduced previously but does not rely on relative diagonal dominance measures as a previous nonsymmetric partitioning approach does. The setup algorithm is somewhat more complicated than that of the two-stage approach, often resulting in longer times for the preconditioner construction, but generally giving preconditioners with lower complexities. In many cases, the improved performance of the preconditioner compensates for the added expense in forming it, but this is not always so. For problems where memory considerations are more important than total time, the new technique offers better complexity than previous approaches, without a significant impact on solver performance.

Appendix: Detailed implementation of Algorithm 3. The following is a detailed description of the implementation of the single-stage greedy coarsening algorithm sketched in Algorithm 3. These details are discussed in section 3.3.

ALGORITHM 4 (detailed single-stage greedy nonsymmetric partitioning algorithm).

1. Initialize $U_{\text{row}} = U_{\text{col}} = \{1, 2, \ldots, n\}$

2. Initialize $C_{\text{row}} = C_{\text{col}} = \emptyset$ and $F_{\text{row}} = F_{\text{col}} = \emptyset$
3. Compute A^T
4. For all $i \in U_{\text{row}}$,
   (a) Sort $\text{Adj}(i)$ by decreasing $|a_{ij}|$
   (b) Compute $k_i = \text{argmax}_{k \in U_{\text{col}}} |a_{ik}|$
   (c) Compute $l_i = \sum_{j \in F_{\text{col}} \cup U_{\text{col}}} |a_{ij}|$
   (d) Compute $r_i = \sum_{j \in F_{\text{col}}} |a_{ij}|$
   (e) If $a_{ik_i} = 0$, then $C_{\text{row}} = C_{\text{row}} \cup \{i\}$, $U_{\text{row}} = U_{\text{row}} \setminus \{i\}$.
   (f) If $\frac{|a_{ik_i}|}{l_i} \geq \theta$, then make (i, k_i) a diagonal element of A_{ff}:
      i. Make i an F-row: $F_{\text{row}} = F_{\text{row}} \cup \{i\}$, $U_{\text{row}} = U_{\text{row}} \setminus \{i\}$
      ii. Make k_i an F-column: $F_{\text{col}} = F_{\text{col}} \cup \{k_i\}$, $U_{\text{col}} = U_{\text{col}} \setminus \{k_i\}$
      iii. For $m \in U_{\text{row}} \cap \text{Adj}^T(k_i)$,
         - If k_m has already been defined and $k_m = k_i$,
           – Compute new $k_m = \text{argmax}_{k \in U_{\text{col}}} |a_{mk}|$
           – If $a_{mk_m} = 0$, then $C_{\text{row}} = C_{\text{row}} \cup \{m\}$, $U_{\text{row}} = U_{\text{row}} \setminus \{m\}$
         - Update $r_m = r_m + |a_{mk_i}|$
         - If $\frac{|a_{mk_m}|}{r_m} < \theta$, then $C_{\text{row}} = C_{\text{row}} \cup \{m\}$, $U_{\text{row}} = U_{\text{row}} \setminus \{m\}$
   (g) If $\frac{|a_{ik_i}|}{r_i} < \theta$, then $C_{\text{row}} = C_{\text{row}} \cup \{i\}$, $U_{\text{row}} = U_{\text{row}} \setminus \{i\}$
5. For $j \in U_{\text{col}}$, compute $w_j = \sum_{i \in U_{\text{row}}} \frac{|a_{ij}|}{|a_{ik_i}|}$
6. While $U_{\text{col}} \neq \emptyset$ and $U_{\text{row}} \neq \emptyset$,
   (a) Let $j^\star \approx \text{argmax}_{j \in U_{\text{col}}} \{w_j\}$ and remove $j^\star$ from U_{col}: $C_{\text{col}} = C_{\text{col}} \cup \{j^\star\}$, $U_{\text{col}} = U_{\text{col}} \setminus \{j^\star\}$
   (b) For $i \in U_{\text{row}} \cap \text{Adj}^T(j^\star)$,
      i. Update $l_i = l_i - |a_{ij^\star}|$
      ii. If $k_i = j^\star$,
         A. Compute new $k_i = \text{argmax}_{k \in U_{\text{col}}} |a_{ik}|$
         B. If $a_{ik_i} = 0$, then make i a C-row: $C_{\text{row}} = C_{\text{row}} \cup \{i\}$, $U_{\text{row}} = U_{\text{row}} \setminus \{i\}$
         C. If $a_{ik_i} \neq 0$ and $\frac{|a_{ik_i}|}{r_i} < \theta$, then,
            - For each $j \in U_{\text{col}} \cap \text{Adj}(i)$, update column weight, $w_j = w_j - \frac{|a_{ij}|}{|a_{ik_i^{\text{old}}}|}$
            - Make i a C-row: $C_{\text{row}} = C_{\text{row}} \cup \{i\}$, $U_{\text{row}} = U_{\text{row}} \setminus \{i\}$
         D. If $a_{ik_i} \neq 0$ and $\frac{|a_{ik_i}|}{r_i} \geq \theta$, then, for $j \in U_{\text{col}} \cap \text{Adj}(i)$, update column weight,

$$w_j = w_j - \frac{|a_{ij}|}{|a_{ik_i^{\text{old}}}|} + \frac{|a_{ij}|}{|a_{ik_i^{\text{new}}}|}$$

      iii. If $i \in U_{\text{row}}$ and $\frac{|a_{ik_i}|}{l_i} \geq \theta$, then make (i, k_i) a diagonal element of A_{ff}:
         A. Make i an F-row: $F_{\text{row}} = F_{\text{row}} \cup \{i\}$, $U_{\text{row}} = U_{\text{row}} \setminus \{i\}$
         B. Make k_i an F-column: $F_{\text{col}} = F_{\text{col}} \cup \{k_i\}$, $U_{\text{col}} = U_{\text{col}} \setminus \{k_i\}$
         C. For $j \in U_{\text{col}} \cap \text{Adj}(i)$, update column weight, $w_j = w_j - \frac{|a_{ij}|}{|a_{ik_i}|}$
         D. For $m \in U_{\text{row}} \cap \text{Adj}^T(k_i)$
            - If k_m has already been defined and $k_m = k_i$,
              – Compute new $k_m = \text{argmax}_{k \in U_{\text{col}}} |a_{mk}|$
              – If $a_{mk_m} = 0$, then make i a C-row: $C_{\text{row}} = C_{\text{row}} \cup \{m\}$,

$U_{\text{row}} = U_{\text{row}} \setminus \{m\}$
- If $a_{mk_m} \neq 0$, then for each $j \in U_{\text{col}} \cap \text{Adj}(m)$, update column weight,

$$w_j = w_j - \frac{|a_{ij}|}{|a_{ik_i^{\text{old}}}|} + \frac{|a_{ij}|}{|a_{ik_i^{\text{new}}}|}$$

- Update $r_m = r_m + |a_{mk_i}|$
- If $m \in U_{\text{row}}$ and $\frac{|a_{mk_m}|}{r_m} < \theta$,
  - Make m a C-row, $C_{\text{row}} = C_{\text{row}} \cup \{m\}$, $U_{\text{row}} = U_{\text{row}} \setminus \{m\}$.
  - For $j \in U_{\text{col}} \cap \text{Adj}(m)$, update column weight, $w_j = w_j - \frac{|a_{mj}|}{|a_{mk_m}|}$

7. $C_{\text{row}} = C_{\text{row}} \cup U_{\text{row}}$
8. $C_{\text{col}} = C_{\text{col}} \cup U_{\text{col}}$

REFERENCES

[1] P. R. Amestoy, T. A. Davis, and I. S. Duff, *An approximate minimum degree ordering algorithm*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 886–905.

[2] O. Axelsson and P. S. Vassilevski, *Algebraic multilevel preconditioning methods,* I, Numer. Math., 56 (1989), pp. 157–177.

[3] O. Axelsson and P. S. Vassilevski, *Algebraic multilevel preconditioning methods,* II, SIAM J. Numer. Anal., 27 (1990), pp. 1569–1590.

[4] R. E. Bank and C. Wagner, *Multilevel ILU decomposition*, Numer. Math., 82 (1999), pp. 543–576.

[5] M. Benzi, D. B. Szyld, and A. van Duin, *Orderings for incomplete factorization preconditioning of nonsymmetric problems*, SIAM J. Sci. Comput., 20 (1999), pp. 1652–1670.

[6] W. Bomhof and H. van der Vorst, *A parallel linear system solver for circuit simulation problems*, Numer. Linear Algebra Appl., 7 (2000), pp. 649–665.

[7] E. F. F. Botta and F. W. Wubs, *Matrix renumbering ILU: An effective algebraic multilevel ILU preconditioner for sparse matrices*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 1007–1026.

[8] A. Brandt, S. F. McCormick, and J. W. Ruge, *Algebraic multigrid (AMG) for sparse matrix equations*, in Sparsity and Its Applications, D. J. Evans, ed., Cambridge University Press, Cambridge, UK, 1984.

[9] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge, *Adaptive smoothed aggregation (αSA)*, SIAM J. Sci. Comput., 25 (2004), pp. 1896–1920.

[10] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge, *Adaptive algebraic multigrid*, SIAM J. Sci. Comput., 27 (2006), pp. 1261–1286.

[11] R. Bridson and W.-P. Tang, *A structural diagnosis of some IC orderings*, SIAM J. Sci. Comput., 22 (2000), pp. 1527–1532.

[12] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, 2nd ed., SIAM, Philadelphia, 2000.

[13] T. Davis, *University of Florida sparse matrix collection*, http://www.cise.ufl.edu/research/sparse/matrices. NA Digest, vol. 92, no. 42, October 16, 1994; NA Digest, vol. 96, no. 28, July 23, 1996; and NA Digest, vol. 97, no. 23, June 7, 1997.

[14] E. D. Dolan and J. J. Moré, *Benchmarking optimization software with performance profiles*, Math. Program., 91 (2002), pp. 201–213.

[15] I. S. Duff and G. A. Meurant, *The effect of ordering on preconditioned conjugate gradients*, BIT, 29 (1989), pp. 635–657.

[16] R. D. Falgout and P. S. Vassilevski, *On generalizing the algebraic multigrid framework*, SIAM J. Numer. Anal., 42 (2004), pp. 1669–1693.

[17] J. A. George and J. W.-H. Liu, *Computer Solution of Large Sparse Positive-Definite Systems*, Prentice–Hall, Englewood Cliffs, NJ, 1981.

[18] N. I. M. Gould and J. A. Scott, *Complete Results for a Numerical Evaluation of HSL Packages for the Direct Solution of Large Sparse, Symmetric Linear Systems of Equations*, Numerical Analysis Internal Report 2003-2, Rutherford Appleton Laboratory, Oxfordshire, UK, 2003.

[19] N. I. M. Gould and J. A. Scott, *A numerical evaluation of HSL packages for the direct solution of large sparse, symmetric linear systems of equations*, ACM Trans. Math. Software, 30 (2004), pp. 300–325.

[20] G. Karypis and V. Kumar, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM J. Sci. Comput., 20 (1998), pp. 359–392.

[21] S. MacLachlan, T. Manteuffel, and S. McCormick, *Adaptive reduction-based AMG*, Numer. Linear Algebra Appl., 13 (2006), pp. 599–620.

[22] S. MacLachlan and Y. Saad, *A greedy strategy for coarse-grid selection*, SIAM J. Sci. Comput., 29 (2007), pp. 1825–1853.

[23] J. J. H. Miller and S. Wang, *An exponentially fitted finite element method for a stationary convection-diffusion problem*, in Computational Methods for Boundary and Interior Layers in Several Dimensions, J. J. H. Miller, ed., Boole Press, Dublin, 1991, pp. 120–137.

[24] Y. Notay, *Algebraic multigrid and algebraic multilevel methods: A theoretical comparison*, Numer. Linear Algebra Appl., 12 (2005), pp. 419–451.

[25] J. W. Ruge and K. Stüben, *Algebraic multigrid (AMG)*, in Multigrid Methods, S. F. McCormick, ed., Frontiers Appl. Math. 3, SIAM, Philadelphia, 1987, pp. 73–130.

[26] Y. Saad, *ILUT: A dual threshold incomplete ILU factorization*, Numer. Linear Algebra Appl., 1 (1994), pp. 387–402.

[27] Y. Saad, *ILUM: A multi-elimination ILU preconditioner for general sparse matrices*, SIAM J. Sci. Comput., 17 (1996), pp. 830–847.

[28] Y. Saad, *Multilevel ILU with reorderings for diagonal dominance*, SIAM J. Sci. Comput., 27 (2005), pp. 1032–1057.

[29] Y. Saad, A. Soulaimani, and R. Touihri, *Variations on algebraic recursive multilevel solvers (ARMS) for the solution of CFD problems*, Appl. Numer. Math., 51 (2004), pp. 305–327.

[30] Y. Saad and B. Suchomel, *ARMS: An algebraic recursive multilevel solver for general sparse linear systems*, Numer. Linear Algebra Appl., 9 (2002), pp. 359–378.

[31] Y. Saad and J. Zhang, *BILUM: Block versions of multielimination and multilevel ILU preconditioner for general sparse linear systems*, SIAM J. Sci. Comput., 20 (1999), pp. 2103–2121.

[32] Y. Saad and J. Zhang, *BILUTM: A domain-based multilevel block ILUT preconditioner for general sparse matrices*, SIAM J. Matrix Anal. Appl., 21 (1999), pp. 279–299.

[33] O. Schenk, S. Roellin, and A. Gupta, *The effects of nonsymmetric matrix permutations and scalings in semiconductor device and circuit simulation*, IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, 23 (2004), pp. 400–411.

SIAM J. SCI. COMPUT.
Vol. 29, No. 5, pp. 2144–2161
© 2007 Society for Industrial and Applied Mathematics

EXTENSIONS OF CERTAIN GRAPH-BASED ALGORITHMS FOR PRECONDITIONING*

DAVID FRITZSCHE†, ANDREAS FROMMER‡, AND DANIEL B. SZYLD†

Abstract. The original TPABLO algorithms are a collection of algorithms which compute a symmetric permutation of a linear system such that the permuted system has a relatively full block diagonal with relatively large nonzero entries. This block diagonal can then be used as a preconditioner. We propose and analyze three extensions of this approach: We incorporate a nonsymmetric permutation to obtain a large diagonal, we use a more general parametrization for TPABLO, and we use a block Gauss–Seidel preconditioner which can be implemented to have the same execution time as the corresponding block Jacobi preconditioner. Experiments are presented showing that for certain classes of matrices, the block Gauss–Seidel preconditioner used with the system permuted with the new algorithm can outperform the best ILUT preconditioners in a large set of experiments.

Key words. block partitioning, PABLO, TPABLO, preconditioning, block Gauss–Seidel, ILU decomposition

AMS subject classifications. 65F50, 65F10, 65F05

DOI. 10.1137/060661284

1. Introduction.

We consider a nonsingular linear system

$$Ax = b,\ A \in \mathbb{R}^{n\times n},\ x, b \in \mathbb{R}^n, \tag{1.1}$$

with n large and A sparse. We do not assume A to be symmetric. Direct methods, based on Gaussian elimination, have reached a mature state for this general system; see, e.g., [18], [19], [22]. They may, however, suffer from severe fill-in so that memory resources may become insufficient and/or execution times may become too large.

Modern iterative methods for the solution of (1.1) work best with a good preconditioner, and at each step, the basic operation is usually one matrix-vector multiplication with A and one application of the preconditioner. A good preconditioner contains as much information about the matrix A as possible. Incomplete factorizations of A (ILU preconditioners) have proven to work well in many cases [4].

The PABLO (parametrized block ordering) and TPABLO (threshold PABLO) approaches to preconditioning proceed in the following manner: a permutation matrix $P \in \mathbb{R}^{n\times n}$ is computed such that the permuted matrix $\widehat{A} = PAP^T$ has a (relatively) dense block diagonal; see [5], [13], [14], [30]. Block sizes are determined dynamically. In the simplest case the preconditioner is taken as the block diagonal $\widehat{D}$ with respect to these blocks; i.e., one solves the permuted system

$$\widehat{A}\widehat{x} = \widehat{b}, \quad \text{where} \quad \widehat{A} = PAP^T,\ \widehat{x} = Px,\ \text{and}\ \widehat{b} = Pb,$$

iteratively using GMRES (or some other Krylov subspace method) with preconditioner $\widehat{D}$; see, e.g., [29], [34], [36] for their description. To solve the preconditioning

*Received by the editors May 29, 2006; accepted for publication (in revised form) April 16, 2007; published electronically September 28, 2007.
http://www.siam.org/journals/sisc/29-5/66128.html

†Department of Mathematics, Temple University, Philadelphia, PA 19122-6094 (fritzsche@temple.edu, szyld@temple.edu). Supported in part by the National Science Foundation under grant DMS-0207525 and by the U.S. Department of Energy under grant DE-FG02-05ER25672.

‡Faculty of Mathematics and Science, Bergische Universität Wuppertal, D-42097 Wuppertal, Germany (frommer@math.uni-wuppertal.de).

system $\widehat{D}s = r$, a (complete) LU decomposition $\widehat{D} = \widehat{L}\,\widehat{U}$ is computed. Since this reduces to an LU decomposition on each of the diagonal blocks, we can use readily available packages. For small and dense blocks we use level 3 BLAS-based LAPACK [2] routines and for large and sparse blocks we use UMFPACK [16]. Moreover, the LU decomposition of $\widehat{D}$ could be done in parallel.

Whereas PABLO determines the blocks for $\widehat{D}$ by just using structural information of A given through its associated directed graph, the TPABLO variants also take the numerical values in A into account.

In the present paper, which builds upon [28], we propose three extensions of the PABLO and TPABLO preconditioners. First, we apply a nonsymmetric permutation to put large entries on the diagonal. We discuss this in section 2. In section 3 we introduce new parametrizations to be used with PABLO and TPABLO. We describe the resulting algorithmic framework, termed XPABLO, in detail and analyze its computational complexity. In other words XPABLO is more general, it includes new different criteria to produce the blocking permutation, and it reduces to the previous versions for specific parametrizations. In section 4 we discuss how one can use either the lower or the upper block triangular part of the matrix as a preconditioner at the same cost as the block diagonal preconditioner.

After reviewing several practical issues related to the implementation in section 5, we finally give results of numerical experiments in section 6, including a comparison with ILU preconditioning. In section 7, we present some concluding remarks and an outlook to further development.

In addition to the references already mentioned, several authors have explored reorderings and partitioning techniques to improve the robustness and performance of different types of preconditioners. Among them, we cite [6], [7], [8], [9], [24]; see also the sections on the effect of orderings in the survey [4].

We conclude this introduction by noting that PABLO variants have been applied successfully in the construction of preconditioners in settings that are different from the one presented in this paper. A slightly modified version of PABLO has been used successfully in a parallel computing setting to build block diagonal preconditioners for difficult problems in computational fluid dynamics (CFD) and chemical engineering; see, e.g., [23]. The threshold variants TPABLO and XPABLO can be used as a method to find blocks with large entries in dense or sparse matrices. Specifically, in [26] and [27], XPABLO is used to find such blocks to build preconditioners for the CSYM method [12] for solving systems with complex symmetric matrices. The situation there is that an efficient preconditioner has to be factorized into its complex symmetric singular value decomposition, so that block diagonal preconditioners seem to be the only practical way. In the examples presented in [26] the iteration count is reduced by up to 30% when using this approach.

2. Nonsymmetric permutations and diagonal scalings. Let σ be the permutation associated with the permutation matrix $Q \in \mathbb{R}^{n\times n}$; i.e., for all $x \in \mathbb{R}^n$

$$(Qx)_i = x_{\sigma(i)}, \quad i = 1, \ldots, n.$$

A [illegible] of the nonsingular matrix $A \in \mathbb{R}^{n\times n}$ is any permutation σ for which

$$(QA)_{ii} = a_{\sigma(i),i} \neq 0, \quad i = 1, \ldots, n.$$

A maximum transversal exists for the class of structurally nonsingular matrices which contains the class of nonsingular matrices; see, e.g., [19]. In [20], two different types of

transversals with respect to different notions of a "weight" were introduced according to the following definition.

DEFINITION 2.1. [illegible] σ

(i) [illegible] bottleneck transversal [illegible]

$$\min_{i=1,\ldots,n} |a_{\sigma(i),i}|$$

(ii) [illegible] maximum product transversal [illegible]

$$\prod_{i=1}^{n} |a_{\sigma(i),i}|$$

The paper [20] proposes and analyzes various algorithms to compute bottleneck transversals. Extensive testing shows that in practice the computational complexity of these algorithms is reported to behave like $\mathcal{O}(n+nz)$, where nz is the total number of nonzeros in the matrix, although the theoretical worst case upper bound may be much larger. Algorithms for computing bottleneck and maximum product transversals have been incorporated in the HSL-library as algorithm MC64; see [20], [21] for their description. As a by-product, when computing a maximum product transversal, MC64 also delivers diagonal scaling matrices $\hat{C}$ and $\hat{R}$ such that

$$\hat{A} = Q(\hat{R}A\hat{C}) \tag{2.1}$$

is an I-matrix; i.e., $|\hat{A}_{ij}| \leq 1$ for all i, j and $\hat{A}_{ii} = 1$, $i = 1, \ldots, n$.

The rationale for these different approaches to obtaining diagonals with large weight is heuristic: Large diagonals tend to decrease the need for pivoting in a direct elimination method. ILU-type preconditioners can also benefit greatly [6], [35]. Moreover, an iterative method with diagonal preconditioner may be expected to converge more rapidly if the diagonal is large compared to the off-diagonal part of the matrix.

In this paper, we use maximum product transversals and we always apply the scaling computed by MC64; i.e., we always transform A to an I-matrix. For our test problems in section 6 this always gave us better results than using the usually cheaper bottleneck transversal.

3. XPABLO: An extension of PABLO and TPABLO. PABLO, TPABLO, and XPABLO compute a (symmetric) permutation for A, and a block partition of it, based on information of the (directed) graph of A and the size of the entries of A. Let us quickly recall that the directed graph or digraph $G(A) = (V, E)$ associated with A is given as

$$V = \{1, \ldots, n\}, \quad E = \{(i, j) : a_{ij} \neq 0\}.$$

Let $nz = |E|$, the cardinality of E. Using a somewhat nonstandard terminology, we say that vertex $i \in V$ is [illegible] with edge $e = (k, \ell)$ if $i = k$ or $i = \ell$. This terminology makes no distinction between the starting point k and the end point ℓ of the edge e. The two vertices k and ℓ incident to (k, ℓ) are called [illegible]. Given a set of vertices $W \subseteq V$, the adjacency set $\mathrm{adj}(W)$ contains all vertices which are not in W but which are adjacent to a vertex in W; i.e.,

$$\mathrm{adj}(W) = \big\{j \in V : j \notin W \text{ and } j \text{ is adjacent to some } i \in W\big\}.$$

To describe the PABLO algorithms, we need additional notation for subgraphs. If $G = (V, E)$ and $V' \subseteq V$, the induced subdigraph $G|_{V'}$ is $G' = (V', E')$ with $E' = \{(i, j) \in E : i, j \in V'\}$. Frequently, we use quantities that refer to an induced subdigraph $G' = G|_{V'}$. We indicate this using the [illegible] $|_{V'}$. For example, $\mathrm{adj}|_{V'}(W)$ refers to the adjacency set in the subdigraph induced by V'. We write $|E|_{V'}$ to denote the cardinality of E'.

PABLO, TPABLO, and XPABLO produce a partitioning of V into m disjoint, nonempty subsets V_κ, $\kappa = 1, \ldots, m$ (the "blocks"). These blocks are built one at a time. We now describe how this is done, assuming the following situation: Blocks $V_1, \ldots, V_{\nu-1}$ have already been built. Given the current block V_ν and a candidate vertex $i \in \mathrm{adj}|_{\overline{V}}(V_\nu)$ with $\overline{V} = V \setminus \bigcup_{\kappa=1}^{\nu-1} V_\kappa$, the algorithms decide whether or not to incorporate i into the current block using a criterion τ. The following definitions describe some basic criteria. They can be combined logically to yield a variety of different criteria, including, as we shall see, the traditional PABLO and TPABLO criteria. Algorithm 1 toward the end of the section shows how candidate vertices are selected; see also [30].

DEFINITION 3.1. [illegible] $G = (V, E)$ [illegible] $V' \subseteq V$ [illegible] $G' = (V', E')$ [illegible] fullness $\varphi(V')$ [illegible] V' [illegible]

$$\varphi(V') = \begin{cases} \dfrac{|E'|}{|V'|^2 - |V'|} & \text{if } |V'| > 1, \\ 0 & \text{if } |V'| \leq 1. \end{cases}$$

The fullness thus defined measures the number of edges in G' compared to the maximally possible number $|V'|^2 - |V'|$ for a complete digraph. The definition of $\varphi(\{v\}) = 0$ follows PABLO in the interpretation that a graph with only one vertex is empty, but other definitions may be useful depending on the problems one needs to solve.

Given $\alpha > 0$, in our generic situation, we say (see [30]) that vertex i satisfies the [illegible] (with fullness parameter α) if

$$\varphi(V_\nu \cup \{i\}) \geq \alpha \varphi(V_\nu). \tag{FC}$$

Note that (FC) can be fulfilled even when $\alpha > 1$.

DEFINITION 3.2. [illegible] degree $\deg(i)$ [illegible] i [illegible] $G = (V, E)$ [illegible] [illegible] i

$$\deg(i) = \big|\{e \in E : i \text{ [illegible] } e\}\big|.$$

[illegible] $V' \subseteq V$ [illegible] $\deg|_{V'}(i)$ [illegible] $\deg(i)$ [illegible] $V' \cup \{i\}$

DEFINITION 3.3. [illegible] i [illegible] $G = (V, E)$ [illegible] $\deg(i) > 0$ [illegible] $V' \subseteq V$ [illegible] connectivity [illegible] i [illegible] V' [illegible]

$$\frac{\deg|_{V'}(i)}{\deg(i)}.$$

In our generic situation, given $\beta > 0$, we say that vertex i satisfies the [illegible] (with connectivity parameter β) if

$$\deg|_{V_\nu}(i) \geq \beta \deg|_{\overline{V}}(i). \tag{CC}$$

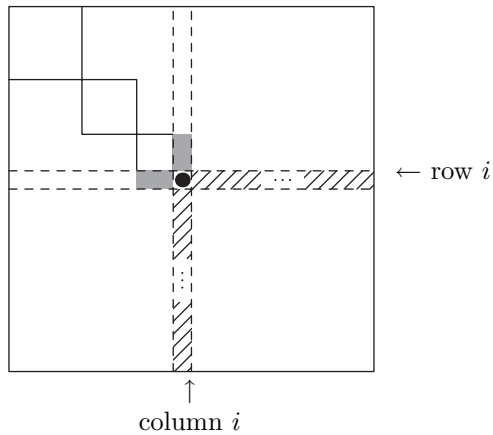


FIG. 3.1. *Illustration of the fullness criterion (FC) and the connectivity criterion (CC).*

The connectivity criterion means that in $G|_{\overline{V}}$ at least a fraction β of all edges incident with i have their other incident vertex in V_ν. The criterion is never met if $\beta > 1$ (and $\deg(i) > 0$).

Figure 3.1 illustrates the fullness and the connectivity criterion for the digraph $G(A)$ in terms of the pattern of the matrix A. The matrix is assumed to be already symmetrically permuted such that the blocks built up so far (there are three such blocks) appear first. The black diagonal entry corresponds to the candidate vertex to be included into the third block. The fullness criterion requires that the grey parts of row i and column i must not be too empty. The connectivity criterion means that the hatched parts of these rows and columns should not contain too many elements as compared to the grey parts. Row and column i are considered only together—not individually. The white parts of row i and column i are not taken into consideration when deciding on vertex i.

The PABLO algorithm of [30] adds a candidate vertex $i \in \mathrm{adj}|_{\overline{V}}(V_\nu)$ to the current block V_ν if and only if v satisfies the fullness , the connectivity criterion for some prechosen parameters α and β. We formulate this by saying that PABLO uses the criterion τ defined as

$$\tau = \mathrm{FC} \vee \mathrm{CC}. \tag{PABLO}$$

This combined PABLO criterion uses only structural information of A.

In its threshold counterparts TPABLO1 and TPABLO2 (see [5], [13]), numerical values of the matrix entries are also taken into account. For a precise statement, and also to formulate our generalization XPABLO, we introduce additional notation and terminology. From now on we consider the digraph $G(A)$ as an edge-weighted digraph where edge $e = (i, j)$ has weight $w(e) = |a_{ij}|$.

DEFINITION 3.4. , $A \in \mathbb{R}^{n\times n}$, $\gamma > 0$, $A^{>\gamma} \in \mathbb{R}^{n\times n}$,

$$(A^{>\gamma})_{ij} = \begin{cases} a_{ij} & , |a_{ij}| > \gamma, \\ 0 & , \end{cases}$$

We use the superscript notation $^{>\gamma}$ in an intuitive manner at various places. For example, if $G = G(A)$ is the digraph of A, then $G^{>\gamma} = G(A^{>\gamma})$. Another example is the following definition.

DEFINITION 3.5. [illegible] $G = (V, E)$ [illegible] $V' \subseteq V$ [illegible] $\gamma \geq 0$ [illegible] threshold fullness $\varphi^{>\gamma}(V')$ [illegible] V' [illegible] $\varphi(V')$ [illegible] V' [illegible] $G^{>\gamma}$

In our generic situation, given a threshold parameter $\gamma > 0$ and a threshold fullness parameter $\theta \in [0, 1]$, we say that i satisfies the [illegible] if

$$\varphi^{>\gamma}(V_\nu \cup \{i\}) \geq \theta. \tag{TFC}$$

In contrast to the plain fullness criterion, the threshold fullness criterion just measures the fullness the new block $V_\nu \cup \{i\}$ has in $A^{>\gamma}$ without relating it to the fullness of V_ν.

DEFINITION 3.6. [illegible] $G = (V, E)$ [illegible] $V' \subseteq V$ [illegible] $\gamma \geq 0$ [illegible] i [illegible] $\deg|_{V'}(i) > 0$ [illegible] threshold connectivity [illegible] i [illegible] V' [illegible]

$$\frac{\deg|^{>\gamma}_{V'}(i)}{\deg|_{V'}(i)}.$$

Threshold connectivity compares the number of heavy edges of i in $G|_{V'}$ to all such edges. As opposed to plain connectivity according to Definition 3.3, it does not consider any edges going to nodes outside V'.

In our generic situation, given a threshold connectivity parameter ζ, we say that the [illegible] holds if

$$\deg|^{>\gamma}_{V_\nu}(i) \geq \zeta \deg|_{V_\nu}(i). \tag{TCC}$$

This criterion measures how many of the new entries of the current block are large. A useful choice for the threshold connectivity parameter is $\zeta = 1/2n$. For this value of ζ the right-hand side of (TCC) is always strictly less than one since $\deg|_{V_\nu}(i) < 2n$ for all $i \in V$. Hence the threshold connectivity criterion holds if and only if $\deg|^{>\gamma}_{V_\nu}(i) \geq 1$.

In addition to the threshold parameter γ, XPABLO also accepts an additional threshold parameter δ, which is used to filter out nonzero matrix entries with very small magnitude, both from the graph $G(A)$ and from the threshold graph $G(A^{>\gamma})$; i.e., XPABLO really works on the graphs $G(A^{>\delta})$ and $G(A^{>\max(\gamma,\delta)})$. So we have $0 \leq \delta < \gamma$ in any useful application. Note that δ changes only the graphs on which XPABLO is operating and not the underlying matrix.

We now have four criteria at hand upon which we can decide whether to include a new vertex to a current block or not. These criteria can be logically combined resulting in various XPABLO criteria, denoted by τ. The following list shows how XPABLO is reduced to PABLO or TPABLO by choosing a specific criterion τ and specific values for some of the parameters.

1. With $\tau = \text{FC} \vee \text{CC}$, XPABLO is equivalent to PABLO.
2. With $\tau = (\text{FC} \vee \text{CC}) \wedge \text{TCC}$ and $\zeta = 1/(2n)$, XPABLO is equivalent to TPBALO1.
3. With $\tau = (\text{FC} \vee \text{CC}) \wedge \text{TFC}$ and $\theta = 1$, XPABLO is equivalent to TPABLO2.

The default criterion for XPABLO is $\tau = \text{FC} \vee \text{CC} \vee \text{TCC}$ with $\zeta = 1/(2n)$; see section 5.2 for a more detailed discussion of XPABLO default parameters.

Once XPABLO has determined a partitioning of V into blocks $V_1, \ldots, V_m$, let π be a permutation that groups the vertices $1, \ldots, n$, from V into these blocks; i.e., $V_1 = \{\pi(1), \ldots, \pi(k_1)\}$, $V_2 = \{\pi(k_1 + 1), \ldots, \pi(k_2)\}, \ldots, V_m = \{\pi(k_{m-1} + 1), \ldots, \pi(k_m)\}$, with $k_j = \sum_{i=1}^{j} |V_j|$. Then the permuted matrix PAP^T is naturally partitioned

input: a digraph $G(A) = (V, E)$ and a criterion τ
output: a partitioning $\{V_1, \ldots, V_m\}$ of V
set $C := V$, $Q := \emptyset$, $B := \emptyset$ and set $\kappa := 1$
set $\deg|_B(i) := 0$ and $\deg|_B^{>\gamma}(i) := 0$ for all $i \in V$
while $C \neq \emptyset$ **do**
  remove a vertex i from C, and place it in B
  set $|E|_B := 0$, $|E|_B^{>\gamma} := 0$, $\varphi(B) := 0$, $\varphi^{>\gamma}(B) := 0$
  for all vertices j adjacent to i **do**
    move j from C to Q if $j \in C$
    $\deg|_B(j) := \deg|_B(j) + \mathrm{mult}(i, j)$
    $\deg|_B^{>\gamma}(j) := \deg|_B^{>\gamma}(j) + \mathrm{mult}^{>\gamma}(i, j)$
  end for
  while $Q \neq \emptyset$ **do**
    remove a vertex i from Q
    if i fulfills the XPABLO criterion τ **then**
      insert i into B
      set $|E|_B := |E|_B + \deg|_B(i)$
      set $|E|_B^{>\gamma} := |E|_B^{>\gamma} + \deg|_B^{>\gamma}(i)$
      update $\varphi(B)$ and $\varphi^{>\gamma}(B)$ {this requires $|E|_B$ and $|E|_B^{>\gamma}$}
      for all vertices j adjacent to i **do**
        move j from C to Q if $j \in C$
        $\deg|_B(j) := \deg|_B(j) + \mathrm{mult}(i, j)$
        $\deg|_B^{>\gamma}(j) := \deg|_B^{>\gamma}(j) + \mathrm{mult}^{>\gamma}(i, j)$
      end for
    else
      insert vertex i into C
    end if
  end while
  set $V_\kappa := B$ and $\kappa := \kappa + 1$ {finish current block}
  for all vertices $i \in B$ **do**
    for all vertices j adjacent to i **do**
      set $\deg|_B(j) := 0$ and $\deg|_B^{>\gamma}(j) := 0$
      $\deg(j) := \deg(j) - \mathrm{mult}(i, j)$ {needed for (CC)}
    end for
  end for
  set $B := \emptyset$
end while

ALGORITHM 1. *XPABLO with criterion τ based on (CC), (FC), (TCC), and (TFC).*

into m blocks of size $|V_j|$, the jth diagonal block corresponding to the vertex set V_j. The threshold criteria may now be interpreted as placing large entries on these diagonal blocks. This is made more precise in the following proposition which follows immediately upon inspection of the various criteria.

PROPOSITION 3.7.

1. $\tau = (_ \vee _) \vee$ $\zeta = 1/(2n)$ PAP^T γ
2. $\tau = (_ \vee _) \wedge$ $\zeta = 1$ PAP^T $\geq \gamma$

Algorithm 1 now describes the XPABLO algorithm in more detail. It uses three data structures C, Q, and B to manage the vertices. The set B denotes the current block being built. During the algorithm, quantities needed for τ, such as $\deg|_{V'}(i)$, $\deg|_{V'}^{>\gamma}(i)$, etc., are always updated immediately. The notation $\mathrm{mult}(i, j)$ ($\in \{0, 1, 2\}$) for two nodes i, j refers to the number of directed edges between i and j.

The time complexity of Algorithm 1 is given in the following result.

PROPOSITION 3.8. $\mathcal{O}(n + |E|)$ τ $\mathcal{O}(1)$

We assume that $G = G(A)$ is given stored in an adjacency list representation including the values for $\mathrm{mult}(i,j)$; see, e.g., [19]. Note that for the common sparse matrix formats $G(A)$ can be obtained with cost $\mathcal{O}(n+|E|)$.

Scanning the adjacency list representation, we build up $G^{>\gamma}$ and $\mathrm{mult}^{>\gamma}(i,j)$ with complexity $\mathcal{O}(|E|)$. Taking Q as a queue, C as a doubly linked list with an external index, and B as a singly linked list, all insert and remove operations we use with these lists as well as their flushing can be done with cost $\mathcal{O}(1)$. As was already shown in [30], the entire PABLO algorithm has then complexity $\mathcal{O}(n+|E|)$. The main observation to obtain this result is to realize that each node i is inserted into Q at most $\deg(i)$ times.

By assumption, the cost for testing criterion τ is $\mathcal{O}(1)$ as it is in PABLO. Therefore, the additional work in XPABLO as compared to PABLO is that we have to update the new quantities $|E|_B^{>\gamma}$, $\varphi^{>\gamma}(B)$, and $\deg|_B^{>\gamma}$ used in (TCC) and (TFC). These updates follow the ones to be done for $|E|_B$, $\varphi(B)$, and $\deg|_B$ by PABLO. In total, this means $\mathcal{O}(n)$ extra operations to update $|E|_B^{>\gamma}$ and $\varphi^{>\gamma}(B)$ and $\mathcal{O}(|E|)$ extra operations to update $\deg|_B^{>\gamma}$. So this is an additional cost of $\mathcal{O}(n+|E|)$, which establishes the proposition. ∎

Let us note that in our practical implementations we indeed build up $G^{>\gamma}$ explicitly. This is in contrast to the implementation of TPABLO in [13]. As is explained in [13], the implementation there is not more than d times as costly as PABLO when d is the maximum degree in G. We prefer to build up $G^{>\gamma}$ explicitly, since then TPABLO requires only about twice the work that PABLO does, so that our implementation is in most cases faster than that in [13].

We defer a detailed discussion on how to choose an XPABLO criterion τ and the various parameters (α, β, γ, θ, ζ) to the sections on practical issues and on numerical results (sections 5 and 6).

4. Efficient block Gauss–Seidel preconditioning. Let A denote the matrix obtained after the preprocessing steps and after applying the XPABLO permutation P of section 3, i.e., $A \leftarrow PQ(RAC)P^T$. Then A has a block structure

$$A = \begin{bmatrix} A_{11} & \cdots & A_{1m} \\ \vdots & \ddots & \vdots \\ A_{m1} & \cdots & A_{mm} \end{bmatrix}, \qquad A_{ij} \in \mathbb{R}^{n_i \times n_j},$$

with $n_i = |V_i|$ being the size of the blocks corresponding to the XPABLO permutation P. Each diagonal block $D_i = A_{ii}$ in the block diagonal $D = \mathrm{diag}(D_1, \ldots, D_m)$ should be (relatively) full and should have (relatively) large nonzero entries. Even with A nonsingular, some of the diagonal blocks D_i can be singular; see section 5.1 for a discussion of this issue. For the moment we assume all D_i to be nonsingular.

We can compute the (row pivoted) LU factorization $D_i = \Pi_i L_i' U_i'$ for each block, so that $D = \Pi\, L'\, U'$, $\Pi = \mathrm{diag}(\Pi_i)$, $L' = \mathrm{diag}(L_i')$, and $U' = \mathrm{diag}(U_i')$, and we use this LU factorization of D when it comes to solving linear systems of the form

$$Ds = r$$

in a preconditioned iterative method with preconditioner D. This block diagonal preconditioning approach (with PABLO, TPABLO1, and TPABLO2) was used in [14], where the numerical results were quite encouraging. Note that when the diagonal block D_i is dense, we can use the optimized dense matrix linear algebra code from

LAPACK to perform these factorizations very efficiently [2]. When they are sparse, a sparse factorization is performed; we use the package UMFPACK [16].

An obvious extension of this idea is to use block Gauss–Seidel preconditioning, i.e., to include a block triangular part of A into the preconditioner. In the following discussion we will use the lower block triangular part, i.e., a forward block Gauss–Seidel iteration, as the preconditioner. It is easy to see that Proposition 4.1 below also holds when the preconditioner M is the upper block triangular part of A.

Note that block Gauss–Seidel iteration without acceleration by a Krylov subspace method was used in the original PABLO paper [30]. Using the established block structure the block Gauss–Seidel preconditioner M can be written as

$$M = \begin{bmatrix} D_1 & 0 & \cdots & 0 \\ A_{21} & D_2 & & \vdots \\ \vdots & & \ddots & 0 \\ A_{m1} & A_{m2} & \cdots & D_m \end{bmatrix}. \tag{4.1}$$

The usual interfaces for (left) preconditioned sparse iterative solvers separate the operation $z = M^{-1}Ax$ into a matrix-vector multiplication $y = Ax$ and a solve operation $z = M^{-1}y$. Viewed in this way, computing $M^{-1}Ax$ seems to be much more expensive than computing $D^{-1}Ax$. But it is not more expensive at all if we do not separate the operations, as we show in the next result. This result is basically a form of the well-known Eisenstat trick [25], [31], [32, pp. 208, 225]. Using this trick, we could also implement efficient block SOR preconditioners, but this is not pursued in this paper.

PROPOSITION 4.1. [illegible] $A \in \mathbb{R}^{n\times n}$ [illegible] $m \times m$ [illegible] $D_i = A_{ii}$ $i = 1, \ldots, m$ [illegible] $D = \operatorname{diag}(D_i)$ [illegible] M [illegible] A [illegible] $D^{-1}Ax$ [illegible] $M^{-1}Ax$ [illegible]. Let $z = M^{-1}Ax$ and $\tilde{z} = D^{-1}Ax$. For any vector $w \in \mathbb{R}^n$ we write w_i to denote the ith block, conformal with the block structure of A. We split A as $A = D - L - U$ into the block diagonal part $D = \operatorname{diag}(D_i)$ (same as before), the block lower triangular part L with $L_{ij} = -A_{ij}$ for $i > j$, and the block upper triangular part U with $U_{ij} = -A_{ij}$ for $i < j$. Both L and U have a zero block diagonal.

Using this notation the block triangular preconditioner is $M = D - L$, and we can rewrite the preconditioned matrix-vector multiplication as

$$\begin{aligned} M^{-1}Ax &= (D-L)^{-1}(D-L-U)x \\ &= (D-L)^{-1}(D-L)x - (D-L)^{-1}Ux \\ &= x - (D-L)^{-1}Ux. \end{aligned}$$

If we set $y = Ux$ and $v = (D-L)^{-1}y$, we get $z_i = x_i - v_i$, $i = 1, \ldots, m$, where

$$v_i = D_i^{-1}\left(y_i + \sum_{j=1}^{i-1} L_{ij}v_j\right) \qquad \text{and} \qquad y_i = \sum_{j=i+1}^{m} U_{ij}x_j.$$

We can further simplify the formula for v_i to

$$v_i = -D_i^{-1}\left(\sum_{j=1}^{i-1} A_{ij}v_j + \sum_{j=i+1}^{m} A_{ij}x_j\right). \tag{4.2}$$

Following the same procedure, we write $\tilde{z} = D^{-1}Ax = x - \tilde{v}$ and get $\tilde{z}_i = x_i - \tilde{v}_i$, where

$$\tilde{v}_i = -D_i^{-1}\left(\sum_{j=1}^{i-1} A_{ij}x_j + \sum_{j=i+1}^{m} A_{ij}x_j\right). \tag{4.3}$$

But the computations (4.2) and (4.3) use the exact same number of operations if we assume that dense vectors are used. ∎

5. Practical issues. Before reporting on our numerical experiments, we address some practical issues when using XPABLO in a preconditioning framework. We concentrate on the two main issues, namely, numerical stability and the choice of a good set of values for all the different XPABLO parameters.

5.1. Robustness. We consider the issue of numerical instability (or even the singularity) of the diagonal blocks D_ν in the block triangular preconditioner M. If A is a general nonsingular matrix, there is no guarantee that all the blocks D_i are nonsingular, nor that M is nonsingular. We mention that difficulties with singular blocks were very rare in our numerical experiments using the recommended parameters, as described in section 5.2. Problems with singular and nearly singular blocks are handled as follows: If a diagonal block D_ν happens to be singular or numerically singular, i.e., if the numerical factorization fails, we modify the diagonal entries such that the block becomes diagonal dominant.

The problems described cannot occur for the class of H-matrices. Recall that a matrix A is termed an H-matrix if there exist weights u_j, $j = 1, \ldots, n$, $u_j > 0$ such that for all $j = 1, \ldots, n$

$$|a_{ii}| \cdot u_i > \sum_{\substack{j=1,\ldots,n \\ j \neq i}} |a_{ij}| \cdot u_j.$$

The class of H-matrices contains the class of M-matrices, and they arise, e.g., in certain discretizations of (elliptic) boundary value problems; see, e.g., [10]. We have the following result.

PROPOSITION 5.1. [illegible] $A \in \mathbb{R}^{n\times n}$ [illegible] H [illegible] A [illegible] M [illegible] (4.1) [illegible] D_ν [illegible] $\nu = 1, \ldots, m$ [illegible]. Assume first that A is an H-matrix. Then M is also an H-matrix (with the same weights u_j as for A). Therefore, the diagonal blocks D_ν of M are H-matrices as well and they are thus nonsingular. Moreover, their LU factorizations can be performed without pivoting; see, e.g., [1].

If A is symmetric positive definite, each of the blocks D_ν, which are principal submatrices of A, are symmetric positive definite as well. In particular, they are all nonsingular and they admit a Cholesky factorization and an LU factorization without pivoting. ∎

In view of the maximum transversal transformation $A \to Q(RAC)$ and the two-sided XPABLO permutation $A \to PAP^T$, it is important to notice that the H-matrix property, symmetry, and positive definiteness are preserved under the XPABLO permutation. In general, however, these properties will not be conserved under the transversal transformation, although the I-matrix scaling of MC64 will have a tendency to preserve the H-matrix property.

TABLE 5.1
Recommended XPABLO criterion τ and parameter values for block Jacobi and block Gauss–Seidel preconditioning.

Preconditioner	τ	α	β	γ	δ	ζ
Block Jacobi	FC $\vee$ CC $\vee$ TCC	1.1	0.6	$\frac{\sum \lvert a_{ij} \rvert}{nz}$	0.05	$1/(2n)$
Block Gauss–Seidel	FC $\vee$ TCC		–			

5.2. Choosing parameters. We also have to consider carefully the choice of the XPABLO parameters and the actual criterion τ. A first important point in practice is that of fixing a maximum block size in Algorithm 1. Indeed, it may happen that with the criteria and the parameters in use, the algorithm would produce only one block, the whole matrix, and this is clearly not useful. We therefore fix a maximum block size, and we modify Algorithm 1 so that it closes the current block B as soon as it reaches this maximum size. We often also fix a minimum block size. This block size is achieved by merging adjacent blocks which are too small. Note that we do not merge blocks if we would exceed the maximum block size, even if we retain small blocks. For block Jacobi or block Gauss–Seidel preconditioning we recommend setting the minimum block size to 200 and the maximum block size to 1000.

The next, very crucial, issue is that of finding adequate parameters and a suitable XPABLO criterion τ. Good parameters for each given matrix could be found by trying a variety of different parameter settings and then choosing the best one (with respect to the resources being used, time, and storage). Of course, this is too costly in practice. We performed a long series of computations on many test problems, and we can give suggested values for most of the parameters resulting in run times of the preconditioned iterative methods which are quite close to the individually best ones. The suggested choices for both block Jacobi and block Gauss–Seidel preconditioning are presented in Table 5.1. If these recommendations are used, the only difference between XPABLO and TPABLO1 is a small change in τ. XPABLO has $\tau =$ FC $\vee$ CC $\vee$ TCC and TPABLO1 has $\tau =$ (FC $\vee$ CC) $\wedge$ TCC. Nevertheless, in almost all cases XPABLO is clearly to be preferred compared to TPABLO1.

As a last issue, we further discuss our choice for γ, the threshold parameter used in (TCC) and (TFC). Of all the parameter recommendations, the one for γ is the weakest in the sense that it is the most likely candidate for not being a good choice for a particular problem. Therefore, we present a different recommendation, which also gives good overall results and for some problems even better results than the default choice. After having scaled (and permuted) the matrix according to (2.1), it seems reasonable to use γ as a "percentage" parameter; i.e., we chose γ such that a given fraction $\gamma' \in [0, 1]$ of the nonzero elements are dropped when passing from A to $A^{>\gamma}$. Since we know the number $|E|$ of nonzeros in A, finding the corresponding γ is an instance of the k[illegible] : γ is the value of the kth largest element of A with $k = \lfloor \gamma' |E| \rfloor$. Note that the kth largest element problem can be solved with linear complexity; i.e., its cost is $\mathcal{O}(|E|)$; see, e.g., [15]. In practical computations, finding the average of the magnitude of the nonzeros is much simpler and faster.

6. Numerical experiments. In this section we present numerical results using XPABLO as a tool for preconditioning. We start by showing the improvements of XPABLO over PABLO and TPABLO, and then provide a detailed comparison of XPABLO and ILUT as a tool for preconditioning. All experiments were run on an Intel Core 2 processor. The test programs are written in C, C++, and Fortran and were compiled using the GNU Compiler Collection (GCC). Identical optimization

TABLE 6.1
Summary information on the test matrices.

Matrix	n	nz	Group
CAVITY16	4 562	137 887	DRIVCAV
CAVITY26	4 562	138 040	DRIVCAV
EX19	12 005	259 577	FIDAP
EX35	19 716	227 872	FIDAP
GARON1	3 175	84 723	Garon
GARON2	13 535	373 235	Garon
RAEFSKY2	3 242	293 551	Simon
RAEFSKY3	21 200	1 488 768	Simon
SHYY41	4 720	20 042	Shyy
SHYY161	76 480	329 762	Shyy
BARRIER2-1	113 076	2 129 496	Schenk ISEI
BARRIER2-3	113 076	2 129 496	Schenk ISEI
IGBT3	10 938	130 500	Schenk ISEI
NMOS3	18 588	237 130	Schenk ISEI
OHNE2	181 343	6 869 939	Schenk ISEI
PARA-4	153 226	2 930 882	Schenk ISEI
PARA-8	155 924	2 094 873	Schenk ISEI
2D_54019_HIGHK	54 019	486 129	Schenk IBMSDS
3D_51448_3D	51 448	537 038	Schenk IBMSDS
IBM_MATRIX_2	51 448	537 038	Schenk IBMSDS
MATRIX_9	103 430	1 205 518	Schenk IBMSDS

flags were used in all cases. This allows us to compare the execution times and not only the iteration counts. The test matrices are from the University of Florida Sparse Matrix collection [17]. Some basic properties of our test matrices are summarized in Table 6.1. For most matrices we use the provided right-hand side b. If no right-hand side is provided, we use $b := Ae$, where $e = (1, \ldots, 1)^T$ is the vector of all ones.

Preliminary tests had shown that the XPABLO framework can perform well for problems stemming from CFD and semiconductor device simulation. Therefore, the test matrices were specifically chosen from these application areas. In Table 6.1 the matrices above the horizontal line are CFD problems, and the matrices below the line are device simulation problems. We note that PABLO was not designed specifically to solve these kinds of problems. As noted in section 1, a modified version of PABLO has been used before for problems in CFD [23]. More information on solving systems from semiconductor device simulation can be found in [35].

To illustrate the different phases of the XPABLO framework, Figure 6.1 shows matrix plots of the matrix GARON1 during four different phases. The bottom right plot shows the matrix of the system we are actually solving. The structure of this system is shown in more detail in Figure 6.2.

Table 6.2 shows how the execution times and iteration counts using the XPABLO framework compare to PABLO- and TPABLO1-based preconditioning. Additionally the improvements gained by using MC64 instead of the default row-column scaling (RCS) is shown. RCS just scales the largest entry in each row and column to one. The following scaling and permutation algorithms are used:

- P: RCS + PABLO,
- T: RCS + TPABLO1,
- X: RCS + XPABLO,
- MX: MC64 + XPABLO.

All parameters are set to the values described in section 5; in particular, γ is set to the average of all nonzero entries. Only the criterion τ is varied. For MX

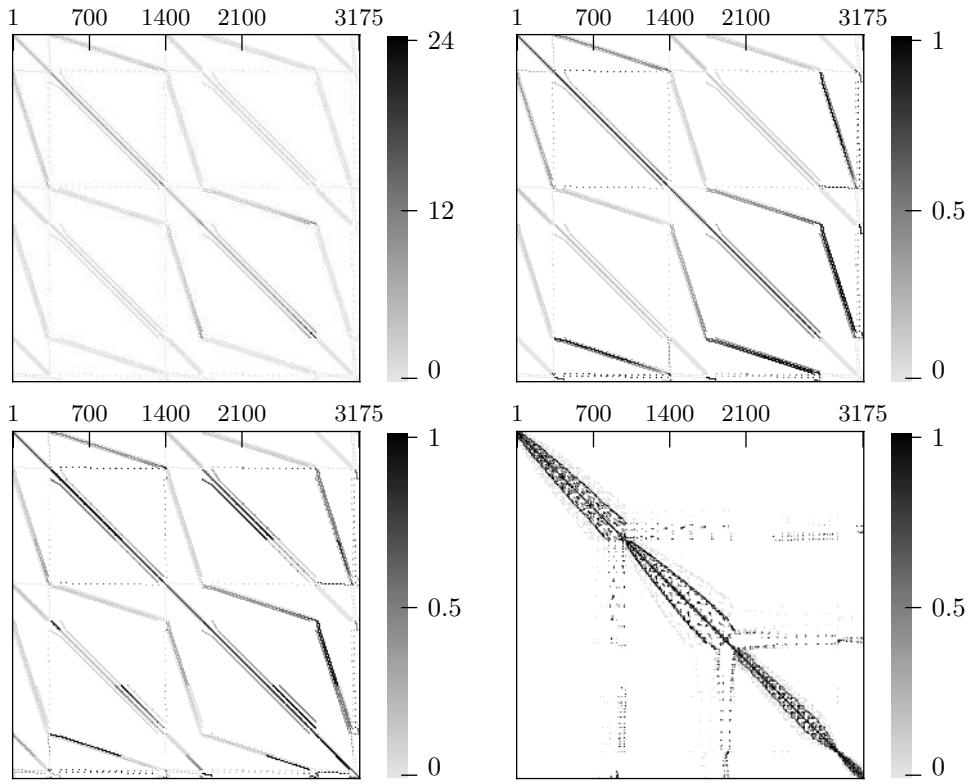


FIGURE 6.1. *Matrix plots of* GARON1 *during four different phases of the XPABLO framework for preconditioning. Top left: The original matrix. Top right: The matrix scaled by* MC64*, but not yet permuted to be an I-matrix. Bottom left: After scaling and permuting by* MC64*. The matrix is now an I-matrix. Bottom right: The matrix after the XPABLO permutation. Note the difference in scale between the top left and the other plots.*

we additionally use XPABLO in conjunction with MC64. GMRES(50) is used as the solver. The preconditioner is the lower block triangular part of the scaled and permuted matrix, using the blocks found by (T/X)PABLO. A dash indicates that we did not observe convergence after 20 cycles of GMRES(50), i.e., after 1000 iterations. For each matrix the numbers in bold indicate the smallest execution time and the lowest iteration number. More details on the stopping criterion are given later. The robustness of XPABLO can be readily appreciated.

We now describe how we compare XPABLO with ILUT as a tool for preconditioning. We use ILUPACK [11] for our experiments with ILUT. The ILUT implementation has two parameters—the drop tolerance τ and the elbow space ℓ. We write ILUT(τ, ℓ) to denote ILUT preconditioning with the specified parameters. In the experiments the elbow space is chosen large enough so that no overflow occurs. We denote this by writing ILUT(τ, ∞). In preliminary tests the three drop tolerances have been found to be good values for the selected problems. We use both XPABLO and ILUT in conjunction with MC64; see section 2. We precede ILUT additionally with a reverse Cuthill–McKee ordering (RCM); this follows the recommendations in [7], [8], and [35]. We mention here that RCM is the default ordering in several software packages, e.g., in [37]. We do not use RCM together with XPABLO as our experiments have shown that it does not further improve the convergence.

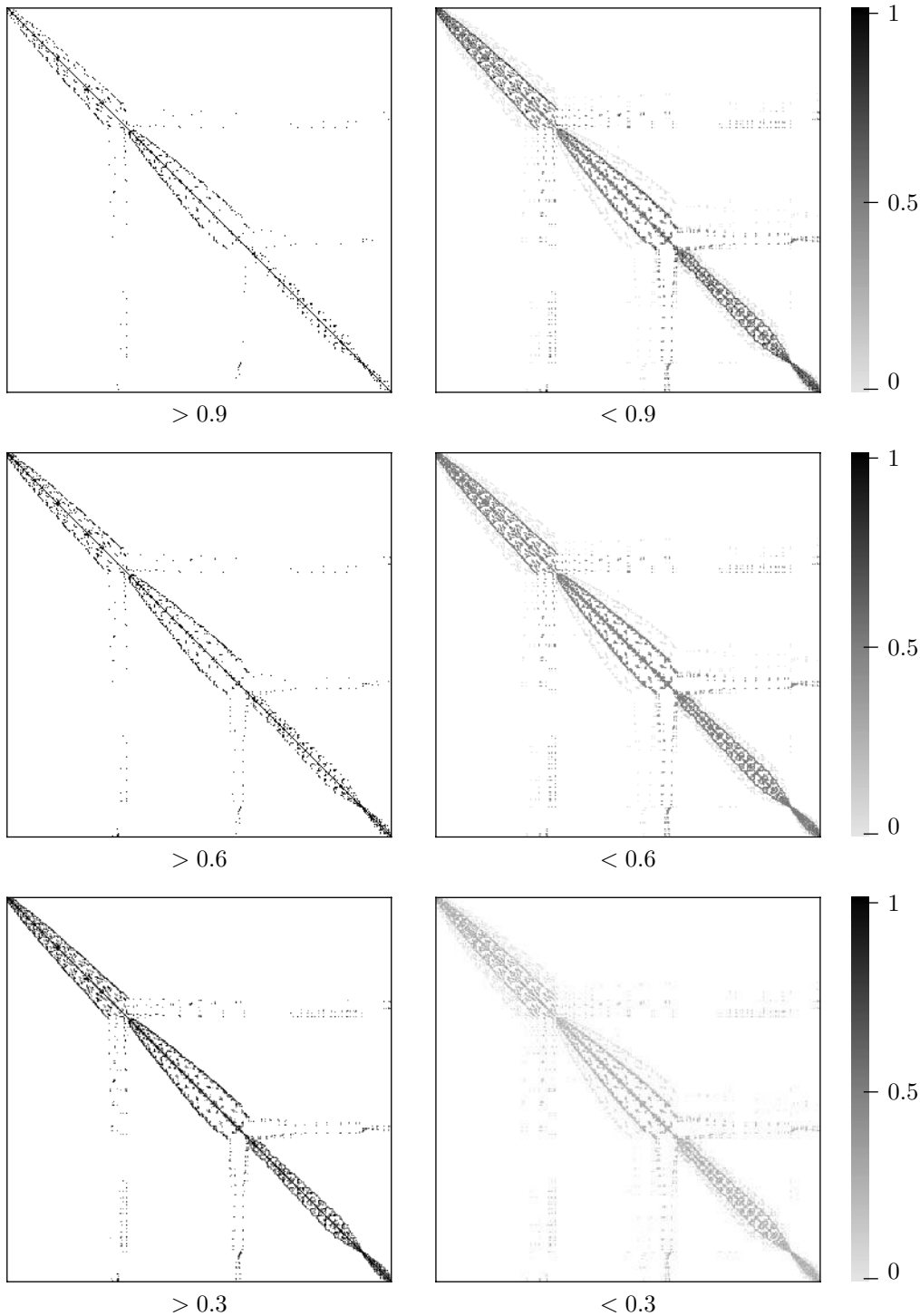


FIGURE 6.2. *Matrix plots of* GARON1 *after scaling and permutation by* MC64 *and XPABLO. Left column: The plots show only matrix entries with a magnitude greater than the given value (*0.9, 0.6 *or* 0.3*). Right column: The plots show the matrix entries less than the given magnitude.*

TABLE 6.2
XPABLO compared to PABLO and TPABLO.

Matrix	Times in seconds				Total iterations			
	P	T	X	MX	P	T	X	MX
CAVITY16	–	1.2	1.3	**0.59**	–	**1**	2	47
CAVITY26	–	–	1.3	**0.48**	–	–	**2**	23
EX19	–	–	5.5	**4.5**	–	–	506	**408**
EX35	–	–	–	**1.3**	–	–	–	**30**
GARON1	**0.24**	–	0.4	0.36	**2**	–	41	36
GARON2	–	–	3.9	**3.1**	–	–	171	**124**
RAEFSKY2	1.9	2	**1.1**	1.2	243	243	**46**	**46**
RAEFSKY3	–	–	15	**14**	–	–	310	**296**
SHYY41	0.14	0.13	0.11	**0.1**	25	20	**2**	**2**
SHYY161	**1.7**	1.7	2.7	1.7	11	**7**	48	14
BARRIER2-1	–	–	–	–	–	–	–	–
BARRIER2-3	–	–	–	–	–	–	–	–
IGBT3	–	–	1.5	**1.4**	–	–	138	**135**
NMOS3	–	6.1	**1.4**	1.5	–	548	**41**	47
OHNE2	–	–	–	**98**	–	–	–	**245**
PARA-4	–	–	**15**	42	–	–	**21**	170
PARA-8	–	–	–	**26**	–	–	–	**84**
2D_54019_HIGHK	–	–	**7.3**	7.7	–	–	**147**	160
3D_51448_3D	–	–	5.2	**4.8**	–	–	64	**55**
IBM_MATRIX_2	–	–	**4.6**	4.7	–	–	49	**47**
MATRIX_9	–	–	22	**16**	–	–	176	**100**

As already mentioned, we use GMRES(50) as the iterative solver in all our experiments. The stopping criterion is

$$\|r_k\|/\|b\| < \sqrt{\epsilon}, \tag{6.1}$$

where r_k is the residual at the kth iteration and ϵ is the machine epsilon, i.e., $\sqrt{\epsilon} \approx 10^{-8}$ since we use IEEE 754 double precision arithmetic. We also did tests with the alternative stopping criterion

$$\frac{\|r_k\|_2}{\sqrt{\|A\|_\infty \|A\|_1}\,\|x_k\|_2 + \|b\|_2} \le \sqrt{\epsilon},$$

which is the default criterion in ILUPACK [11], based on the analysis in [3]; see also [33]. Since the results were very similar and the full test cannot be done at each iteration, we show only results using the relative residual stopping criterion (6.1). The solver was stopped if no convergence is reached after 20 cycles of GMRES(50), i.e., after 1000 iterations. We use the following preconditioners:

- J: MC64 + XPABLO + block Jacobi,
- LX: MC64 + XPABLO + forward block Gauss–Seidel,
- UX: MC64 + XPABLO + backward block Gauss–Seidel,
- I1: MC64 + RCM + ILUT($10^{-2}, \infty$),
- I2: MC64 + RCM + ILUT($10^{-3}, \infty$),
- I3: MC64 + RCM + ILUT($10^{-4}, \infty$).

The results are summarized in Table 6.3. For XPABLO-based preconditioners we use the recommended parameters described in section 5. A dash indicates that we did not observe convergence. For each matrix the bold numbers indicate the smallest execution time and the lowest iteration number.

As it can be observed, the XPABLO-based preconditioners can perform better than the ILUT-based ones in many cases, and in some cases much better. In some

TABLE 6.3
Total CPU times for solving a linear system using preconditioned GMRES(50) including all preprocessing and the corresponding iteration numbers.

	Times in seconds						Total iterations					
	XPABLO			ILUT			XPABLO			ILUT		
Matrix	J	LX	UX	I1	I2	I3	J	LX	UX	I1	I2	I3
CAVITY16	1.4	**0.59**	–	–	1.5	1.3	193	47	–	–	88	**27**
CAVITY26	3.6	**0.48**	–	–	–	1.3	654	23	–	–	–	**21**
EX19	–	4.5	**1**	–	–	–	–	408	**28**	–	–	–
EX35	1.5	1.3	**1.1**	–	–	–	39	30	**12**	–	–	–
GARON1	0.68	**0.36**	2.5	–	0.74	0.69	90	**36**	703	–	101	45
GARON2	7.6	**3.1**	–	–	–	–	350	**124**	–	–	–	–
RAEFSKY2	3.5	1.2	0.86	**0.61**	2	5.2	223	46	23	28	13	**7**
RAEFSKY3	35	14	**4.2**	5.3	9.1	22	737	296	44	92	14	**7**
SHYY41	1.2	0.1	–	**0.09**	0.12	0.1	739	**2**	–	68	70	61
SHYY161	–	1.7	**1.6**	–	–	–	–	14	**1**	–	–	–
BARRIER2-1	–	–	**8.8**	–	–	–	–	–	**13**	–	–	–
BARRIER2-3	–	–	**9.1**	–	–	–	–	–	**14**	–	–	–
IGBT3	3.4	1.4	0.64	0.83	**0.42**	0.43	334	135	27	115	35	**23**
NMOS3	3.5	1.5	1.6	0.78	**0.7**	0.78	173	47	61	51	27	**16**
OHNE2	210	98	**20**	60	46	69	531	245	**10**	218	85	42
PARA-4	81	42	**13**	20	20	–	369	170	**15**	161	91	–
PARA-8	61	26	**13**	14	15	27	248	84	**18**	131	74	44
2D_54019_HIGHK	17	7.7	2.7	**0.73**	2.2	13	347	160	13	13	**7**	21
3D_51448_3D	8.3	4.8	**3.1**	–	–	–	131	55	**14**	–	–	–
IBM_MATRIX_2	7.6	4.7	**3.2**	15	94	180	119	47	15	**1**	**1**	**1**
MATRIX_9	33	16	**8.2**	12	41	190	253	100	22	**1**	**1**	**1**

examples the XPABLO-based preconditioners perform better, although the iteration counts of the ILUT-based preconditioners are lower. The reason for this is that times for finding and, more importantly, factorizing the preconditioner are not the same in both cases.

7. Conclusion. While XPABLO is not always competitive with sophisticated ILU preconditioners like ILUT, we have seen that XPABLO can perform better than ILUT for many CFD and semiconductor device simulation problems. Furthermore, XPABLO-based preconditioners have much more parallelization potential than ILU-based preconditioners.

Acknowledgments. We thank Michele Benzi and the anonymous referee for their constructive comments. Their help and suggestions are very much appreciated.

REFERENCES

[1] G. ALEFELD, *Über die Durchführbarkeit des Gaußschen Algorithmus bei Gleichungen mit Intervallen als Koeffizienten (On the feasibility of the Gaussian algorithm for equations with interval coefficients)*, Computing Supplementum, 1 (1977), pp. 15–19 (in German).

[2] E. ANDERSON, Z. BAI, C. BISCHOF, S. BLACKFORD, J. DEMMEL, J. DONGARRA, J. DU CROZ, A. GREENBAUM, S. HAMMARLING, A. MCKENNEY, AND D. SORENSEN, *LAPACK Users' Guide*, 3rd ed., Software Environ. Tools 9, SIAM, Philadelphia, 1999.

[3] M. ARIOLI, I. DUFF, AND D. RUIZ, *Stopping criteria for iterative solvers*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 138–144.

[4] M. BENZI, *Preconditioning techniques for large linear systems: A survey*, J. Comput. Phys., 182 (2002), pp. 418–477.

[5] M. BENZI, H. CHOI, AND D. B. SZYLD, *Threshold ordering for preconditioning nonsymmetric problems*, in Scientific Computing, Proceedings of the Workshop, (Hong Kong, 1997), G. Golub, S.-H. Lui, F. Luk, and R. Plemmons, eds., Springer-Verlag, Singapore, 1997, pp. 159–165.

[6] M. Benzi, J. C. Haws, and M. Tůma, *Preconditioning highly indefinite and nonsymmetric matrices*, SIAM J. Sci. Comput., 22 (2000), pp. 1333–1353.
[7] M. Benzi, W. Joubert, and G. Mateescu, *Numerical experiments with parallel orderings for ILU preconditioners*, Electron. Trans. Numer. Anal., 8 (1999), pp. 88–114.
[8] M. Benzi, D. B. Szyld, and A. van Duin, *Orderings for incomplete factorization preconditioning of nonsymmetric problems*, SIAM J. Sci. Comput., 20 (1999), pp. 1652–1670.
[9] M. Benzi and M. Tůma, *Orderings for factorized sparse approximate inverse preconditioners*, SIAM J. Sci. Comput., 21 (2000), pp. 1851–1868.
[10] A. Berman and R. J. Plemmons, *Nonnegative Matrices in the Mathematical Sciences*, Classics Appl. Math., SIAM, Philadelphia, 1994. Originally published by Academic Press, New York, 1979.
[11] M. Bollhöfer and Y. Saad, *ILUPACK—preconditioning software package, release v*1.0, *May* 14, 2004, available online at http://www.tu-berlin.de/ilupack/.
[12] A. Bunse-Gerstner and R. Stöver, *On a conjugate gradient-type method for solving complex symmetric linear systems*, Linear Algebra Appl., 287 (1999), pp. 105–123.
[13] H. Choi and D. B. Szyld, *Application of threshold partitioning of sparse matrices to Markov chains*, in IEEE International Computer Performance and Dependability Symposium IPDS'96, Urbana-Champaign, IL, 1996, pp. 158–165.
[14] H. Choi and D. B. Szyld, *Threshold Ordering for Preconditioning Nonsymmetric Problems with Highly Varying Coefficients*, Tech. report 96-51, Department of Mathematics, Temple University, Philadelphia, PA, 1996.
[15] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed., MIT Press, Cambridge, MA, 2001.
[16] T. A. Davis, *Algorithm* 832*: UMFPACK V*4.3*: An unsymmetric-pattern multifrontal method*, ACM Trans. Math. Software, 30 (2004), pp. 196–199. Also available online at http://www.cise.ufl.edu/research/sparse/umfpack/.
[17] T. A. Davis, *University of Florida Sparse Matrix Collection*. Available online at http://www.cise.ufl.edu/research/sparse/matrices/.
[18] I. S. Duff, *MA*57*—A code for the solution of sparse symmetric definite and indefinite systems*, ACM Trans. Math. Software, 30 (2004), pp. 118–144.
[19] I. S. Duff, A. M. Erisman, and J. K. Reid, *Direct Methods for Sparse Matrices*, Monographs on Numerical Analysis, Clarendon Press, Oxford, UK, 1986.
[20] I. S. Duff and J. Koster, *The design and use of algorithms for permuting large entries to the diagonal of sparse matrices*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 889–901.
[21] I. S. Duff and J. Koster, *On algorithms for permuting large entries to the diagonal of a sparse matrix*, SIAM J. Matrix Anal. Appl., 22 (2001), pp. 973–996.
[22] I. S. Duff and J. A. Scott, *A parallel direct solver for large sparse highly unsymmetric linear systems*, ACM Trans. Math. Software, 30 (2004), pp. 95–117.
[23] L. C. Dutto, W. G. Habashi, and M. Fortin, *Parallelizable block diagonal preconditioners for the compressible Navier-Stokes equations*, Comput. Methods Appl. Mech. Engrg., 117 (1994), pp. 15–47.
[24] L. C. Dutto, *The effect of ordering on preconditioned GMRES algorithm, for solving the compressible Navier-Stokes equations*, Internat. J. Numer. Methods Engrg., 36 (1993), pp. 457–497.
[25] S. C. Eisenstat, *Efficient implementation of a class of preconditioned conjugate gradient methods*, SIAM J. Sci. Statist. Comput., 2 (1981), pp. 1–4.
[26] S. Fischer, *Über auf Takagi-Faktorisierungen beruhende Präkonditionierer für CSYM (On Takagi-factorization based preconditioners for CSYM)*, Ph.D. thesis, Department of Mathematics and Science, University of Wuppertal, Wuppertal, Germany, 2006 (in German).
[27] S. Fischer and A. Frommer, *Preconditioning CSYM*, in preparation.
[28] D. Fritzsche, *Graph theoretical methods for preconditioners*, Master's thesis, Department of Mathematics and Science, University of Wuppertal, Wuppertal, Germany, 2004.
[29] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, Frontiers Appl. Math. 17, SIAM, Philadelphia, 1997.
[30] J. O'Neil and D. B. Szyld, *A block ordering method for sparse matrices*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 811–823.
[31] J. M. Ortega, *Efficient implementations of certain iterative methods*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 882–891.
[32] J. M. Ortega, *Introduction to Parallel and Vector Solution of Linear Systems*, Plenum Press, New York, 1988.
[33] C. C. Paige and Z. Strakoš, *Residual and backward error bounds in minimum residual Krylov subspace methods*, SIAM J. Sci. Comput., 23 (2002), pp. 1898–1923.

[34] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.
[35] O. Schenk, S. Röllin, and A. Gupta, *The effects of unsymmetric matrix permutations and scalings in semiconductor device and circuit simulation*, IEEE Transactions on CAD of Integrated Circuits and Systems, 23 (2004), pp. 400–411.
[36] V. Simoncini and D. B. Szyld, *Recent computational developments in Krylov subspace methods for linear systems*, Numer. Linear Algebra Appl., 14 (2007), pp. 1–59.
[37] R. S. Tuminaro, M. Heroux, S. A. Hutchinson, and J. N. Shadid, *Official Aztec User Guide, version* 2.1, Tech. report SAND99-8801J, Sandia National Laboratory, Albuquerque, NM, 1999. Available online at http://www.cs.sandia.gov/CRF/aztec1.html.

SIAM J. SCI. COMPUT.
Vol. 29, No. 5, pp. 2162–2188
© 2007 Society for Industrial and Applied Mathematics

NEARLY OPTIMAL PRECONDITIONED METHODS FOR HERMITIAN EIGENPROBLEMS UNDER LIMITED MEMORY. PART II: SEEKING MANY EIGENVALUES*

ANDREAS STATHOPOULOS† AND JAMES R. MCCOMBS†

Abstract. In a recent companion paper, we proposed two methods, GD+k and JDQMR, as nearly optimal methods for finding one eigenpair of a real symmetric matrix. In this paper, we seek nearly optimal methods for a large number, nev, of eigenpairs that work with a search space whose size is $O(1)$, independent from nev. The motivation is twofold: avoid the additional $O(nevN)$ storage and the $O(nev^2N)$ iteration costs. First, we provide an analysis of the oblique projectors required in the Jacobi–Davidson method and identify ways to avoid them during the inner iterations, either completely or partially. Second, we develop a comprehensive set of performance models for GD+k, Jacobi–Davidson type methods, and ARPACK. Based both on theoretical arguments and on our models we argue that any eigenmethod with $O(1)$ basis size, preconditioned or not, will be superseded asymptotically by Lanczos-type methods that use $O(nev)$ vectors in the basis. However, this may not happen until $nev > O(1000)$. Third, we perform an extensive set of experiments with our methods and against other state-of-the-art software that validates our models and confirms our GD+k and JDQMR methods as nearly optimal within the class of O(1) basis size methods.

Key words. eigenvalues, Jacobi–Davidson, ARPACK, Newton, conjugate gradient, optimality, preconditioning

AMS subject classification. 65F15

DOI. 10.1137/060661910

1. Introduction.

The numerical solution of large, sparse, Hermitian, or real symmetric eigenvalue problems is one of the most computationally intensive tasks in a variety of applications. The challenge is twofold. First, the matrix size, N, is routinely more than a million, while an order of a billion has also been tried [43]. Second, many applications, including electronic structure calculations, require the computation of hundreds or even thousands of extreme eigenpairs. Often the number of required eigenpairs, nev, is described as a small percentage of the problem size. In such cases, orthogonalization of nev vectors, an $O(nev^2N)$ task, becomes $O(N^3)$, making the scaling to larger problem sizes practically infeasible.

Iterative methods are the only means of addressing these large problems. Yet, iterative methods may converge slowly, especially as the problem size grows, and must store the iteration vectors for computing eigenvector approximations. Beyond challenges in execution time, the storage demands of these applications can be staggering. Over the last decade, iterative methods have been developed [14, 51, 54, 30, 29, 49, 41, 57] that can effectively use the large arsenal of preconditioners for linear systems and converge nearly optimally to an eigenpair under limited memory requirements.

The quest for optimality under limited memory is a natural one. On symmetric linear systems, Krylov methods such as conjugate gradient (CG) achieve optimal convergence through a three term recurrence. For eigenvalue problems, the Lanczos method can produce the optimal space through a three term recurrence, but the

*Received by the editors June 3, 2006; accepted for publication (in revised form) May 14, 2007; published electronically September 28, 2007. This work was supported by National Science Foundation grants ITR/DMR 0325218 and ITR/AP-0112727.
http://www.siam.org/journals/sisc/29-5/66191.html
†Department of Computer Science, College of William and Mary, Williamsburg, VA 23187-8795 (andreas@cs.wm.edu, mccombjr@cs.wm.edu).

vectors must be stored or recomputed. With preconditioning, traditional Lanczos does not apply directly. Restarting techniques can be employed so that approximations are obtained from a search space of limited size, but at the expense of convergence.

When seeking one eigenpair, a useful way to approach the problem is by considering it as a nonlinear problem, where both the eigenvalue and eigenvector are unknown. In [57], we studied the problem from two nonlinear viewpoints: the inexact Newton and the limited memory quasi-Newton [40]. The two methods we have developed, the JDQMR and the GD+k, can be related to the above respective viewpoints, which allowed us to argue for their near-optimal convergence. The former is a Jacobi–Davidson (JD) method where the inexact inner iteration is stopped dynamically and optimally. The latter is a generalized Davidson (GD) method with a recurrence-based restarting that delivers convergence extremely close to, and sometimes indistinguishable from, the optimal method (i.e., without restarting).

When seeking many eigenpairs, it is an open question whether optimality can be achieved under limited memory. If one eigenvalue is known exactly, the corresponding eigenvector can be obtained optimally through a CG iteration [30, 57]. If nev eigenvalues are known, one may think that the analogue optimality is to run nev separate CG iterations. This is the approach taken by most limited memory, preconditioned eigensolvers for small nev values. Yet, it is clearly suboptimal because a method that stores the CG iterates from each run would converge in much fewer iterations. For example, when the number of CG iterations is $O(N)$, the former approach takes $O(nev\ N)$, while an unrestarted Lanczos would take no more than N.

In this paper, we focus on methods that do not allow their memory requirements to grow unbounded. We further distinguish between two "classes" of these methods: The first allows the size of the search space to be a small multiple of nev, typically $2nev$ or $1.5nev$. This class concedes the $O(nev^2)$ factor in the hope of much fewer iterations. The second class restricts the search space to a constant size (e.g., 20 vectors) and uses locking to obtain all nev eigenvectors. Although the number of iterations increases, orthogonalization and iteration costs decrease dramatically. This memory-based classification is slightly different from the one in [5], but it better captures the performance characteristics of solvers for large nev. The holy grail in this area has been to obtain nev eigenpairs in linear to nev scaling.

We make three contributions in this paper. First, after discussing the merits of current state-of-the-art methods for finding many eigenvalues, we analyze the effects of projecting against converged eigenvectors in the correction equation of the JD method, and we propose variants that avoid both the extra storage and the second orthogonalization. In particular, the unpreconditioned JDQMR-000 variant requires no orthogonalization during inner iterations and thus has the potential of overcoming the $O(nev^2)$ scaling barrier.

Second, we perform a detailed complexity analysis of the two classes of methods, and, based also on experimental observations, we develop a complete set of models that describes the relative performance between JD variants, GD+k, and implicitly restarted Lanczos (IRL) as implemented in ARPACK. The analysis not only reveals the relative asymptotic behavior but also provides performance crossover points between methods that can be used for dynamic method selection within a multimethod software, such as PRIMME, which we have recently developed [34].

Third, we conduct an extensive set of experiments, with and without preconditioning, which complements the ones in [5] by comparing our GD+k and JDQMR variants against software that implements current state-of-the-art methods such as JDBSYM,

BLOPEX, and ARPACK. The experiments show that our proposed JDQMR variants and GD+k are clearly the best methods in their class and confirm the crossover points and the asymptotic behaviors from our models. They also demonstrate the almost linear scaling of the JDQMR-000 variant without preconditioning up to several hundreds of eigenvalues.

2. State-of-the-art eigenmethods for many eigenvalues. Throughout this paper we assume that we seek the nev algebraically smallest eigenvalues λ_i and their corresponding eigenvectors $\mathbf{x}_i$, $i = 1, \ldots, nev$, of a symmetric matrix A of size N.

2.1. Lanczos-based methods. It should be no surprise that unrestarted Lanczos is still considered a state-of-the-art method without preconditioning. It is the optimal method in terms of number of iterations, for one or more eigenvalues, it is theoretically well understood, and it has several efficient, highly robust implementations [58, 45, 21, 9]. For finding one eigenvalue JD methods are now preferred to Lanczos, not only because of preconditioning, but also because of nearly optimal convergence that does not require unlimited storage [57]. For many eigenvalues, however, the ability of unrestarted Krylov spaces to capture large parts of the extremes of the spectrum of A remains unparalleled. This optimal convergence comes with several drawbacks. Frequent selective and/or partial orthogonalizations are required to avoid dealing with ghost eigenvalues. Also, besides storing or recomputing the vector iterates, the method needs to store a tridiagonal matrix of a size equal to the number of iterations, which can be tens of thousands, and solve for nev of its eigenpairs. Finally, Lanczos has difficulty obtaining multiple eigenvalues. Block versions of Lanczos have been proposed as a robust and cache efficient alternative [8, 19] but usually increase the total execution time (see [22] for a recent survey).

In the rest of the paper, we do not consider unrestarted Lanczos methods, because they do not satisfy our limited memory criterion and cannot use preconditioning without some form of inner-outer iteration [57, 38]. In doing so, we realize that there could be cases where the unrestarted methods are faster than the alternatives we present. In addition, we do not discuss the powerful shift-and-invert Lanczos [45], because it requires the exact inversion of the matrix.

To reduce storage and iteration costs, the Lanczos method can be restarted through the implicit restarting [53] or the equivalent thick restarting [36, 56, 59] technique. When the maximum size m_{max} for the search space is reached, the nev required Ritz vectors are computed and replace the search space. The ARPACK software is a high quality, efficient implementation of the implicitly restarted Arnoldi and Lanczos methods [33], and it has become the default eigensolver in many applications.

Restarting impairs the optimality of Lanczos. Implicit (or thick) restarting with nev (or more) vectors improves convergence but may not be sufficient with small basis sizes. Indeed, it is known that IRL is a polynomially accelerated simultaneous iteration with a polynomial of degree $m_{max} - nev$ [32]. Hence, inner-outer methods can be more effective for small nev. On the other hand, ARPACK requires that $m_{max} > nev$ (and typically $m_{max} = 2nev$), which implies that for really large nev the algorithm approaches again an unrestarted Lanczos with full orthogonalization, with its convergence benefits but high iteration costs.

2.2. Newton approaches. We can view the eigenvalue problem as a constrained minimization problem for minimizing the Rayleigh quotient $\mathbf{x}^T A\mathbf{x}$ on the unit sphere or equivalently minimizing $\mathbf{x}^T A\mathbf{x}/\mathbf{x}^T\mathbf{x}$ [14]. For many eigenpairs, the same formulation applies for minimizing the trace of a block of vectors [48], working with nev-

dimensional spaces [2]. As we discussed in [57], most eigenmethods can be interpreted through the inexact Newton viewpoint or the quasi-Newton viewpoint.

2.2.1. The inexact Newton approach. The exact Newton method for eigenproblems, applied on the Grassmann manifold to enforce normalization of the eigenvectors, is equivalent to Rayleigh quotient iteration (RQI) [14]. It is well known that solving the Hessian equation to full accuracy is not needed at every Newton step. Inexact Newton methods attempt to balance good convergence of the outer Newton method with an inexact solution of the Hessian equation. Extending inexact Newton methods to RQI has attracted a lot of attention in the literature [46, 31, 52, 20, 49].

In [57] we argued that a more appropriate representative of the inexact Newton minimization for eigenvalue problems is the JD method [51]. See also [1] for a theoretical discussion on Newton variants. Given an approximate eigenvector $\mathbf{u}^{(m)}$ and its Ritz value $\theta^{(m)}$, the JD method obtains an approximation to the eigenvector error by approximately solving the correction equation

$$(I - \mathbf{u}^{(m)}\mathbf{u}^{(m)T})(A - \eta I)(I - \mathbf{u}^{(m)}\mathbf{u}^{(m)T})\mathbf{t}^{(m)} = -\mathbf{r}^{(m)} = \theta^{(m)}\mathbf{u}^{(m)} - A\mathbf{u}^{(m)}, \tag{2.1}$$

where η is a shift close to the wanted eigenvalue. The next (inexact) Newton iterate is then $\mathbf{u}^{(m+1)} = \mathbf{u}^{(m)} + \mathbf{t}^{(m)}$. The pseudoinverse of the Hessian is considered to avoid the singularity when $\eta \approx \lambda$ and also to avoid yielding back $\mathbf{t}^{(m)} = \mathbf{u}^{(m)}$ when the equation is solved accurately with $\eta = \theta^{(m)}$. The latter problem could cause stagnation in the classical or the GD methods [10, 37, 55, 42, 61]. The GD method obtains the next iterate as $\mathbf{t}^{(m)} = K^{-1}\mathbf{r}^{(m)}$, where the preconditioner K approximates $(A - \eta I)$ directly—not as an iterative solver.

JD and GD mehods are typically used with subspace acceleration, where the iterates $\mathbf{t}^{(m)}$ are accumulated in a search space from which eigenvector approximations are extracted through Rayleigh–Ritz or some other projection technique [44, 35, 25]. This is particularly beneficial, especially when looking for more than one eigenpair.

The challenge in the JD method is to identify the optimal accuracy to solve each correction iteration. In [41] Notay proposed a dynamic stopping criterion based on monitoring the growing disparity in convergence rates between the eigenvalue residual and linear system residual of CG. The norm of the eigenvalue residual was monitored inexpensively through a scalar recurrence (see [24] for recent extensions). In [57] we proposed JDQMR that extends JDCG by using symmetric QMR [16] as the inner method. The advantages are as follows: (a) The smooth convergence of sQMR allows for robust and efficient stopping criteria. (b) It can handle indefinite correction equations, which is important also for large *nev*. (c) sQMR, unlike MINRES, can use indefinite preconditioners, which are often needed for interior eigenproblems. We also argued that JDQMR-type methods cannot converge more than three times more slowly than the optimal method and usually are significantly less than two times slower. Coupled with the very low sQMR costs, JDQMR has proven to be one of the fastest and most robust methods for $nev = 1$.

When seeking many eigenvalues, the Newton method can be applied on the *nev*-dimensional Grassmann manifold to compute directly the invariant subspace (see [48] and [2] for a more recent review). Practically, however, the Grassmann RQI approach proposed in [2] is simply a block JD method. To see this, note first that the JD search space is kept orthonormal; second, a Rayleigh–Ritz is performed at every outer step, and therefore the Sylvester equations stemming from the Newton approach are decoupled as *nev* independent correction equations for each vector in the block. The

open computational question is how to solve these *nev* linear systems most efficiently, and whether to use a block method at all.

The problem is highly related to ongoing linear systems research for multiple right-hand sides [50, 7, 26, 23]. It could be argued that seed methods that build one large Krylov space and reuse it to solve all the *nev* equations are the best choice. In our case, the disadvantage of seed methods is that they need to store this large space, which is exactly what we try to avoid by restarting the outer JD iteration and by using short term recurrence methods for the correction equation. Moreover, the multiple right-hand sides are not related in any way, so it is unclear how much one system can benefit from the Krylov space of another.

Block methods that solve all the correction equations simultaneously do not consistently improve the overall runtime. On some occasions, however, when the eigenvalues are multiple or highly clustered, certain JD implementations may benefit from a small block size, because the fast, targeted convergence of Newton could cause eigenvalues to converge out of order and thus the required ones to be missed.

With large *nev*, however, the near optimal convergence of the JDQMR has to be repeated *nev* times, much like the *nev* independent CGs we mentioned in the introduction. In this case, and assuming a very small block size, the role of a larger subspace acceleration is to obtain better approximations for nearby eigenpairs while JD converges to the targeted eigenpair. Although the convergence rate of QMR cannot improve further, increasing the basis size gives increasingly better initial guesses for the eigenpairs to be targeted next. In this paper, we avoid this continuum of choices and focus only on constant, limited memory basis sizes.

2.2.2. The quasi-Newton approach. An alternative to Newton is the use of the nonlinear conjugate gradient (NLCG) method for eigenproblems [14]. It is natural to consider a method that minimizes the Rayleigh quotient on the whole space $L = \left\{\mathbf{u}^{(m-1)}, \mathbf{u}^{(m)}, \mathbf{r}^{(m)}\right\}$ instead of only along one search direction. The method

$$\mathbf{u}^{(m+1)} = \text{RayleighRitz}\left(\left\{\mathbf{u}^{(m-1)}, \mathbf{u}^{(m)}, \mathbf{r}^{(m)}\right\}\right), \ m > 1, \tag{2.2}$$

is often called locally optimal conjugate gradient (LOCG) [13, 28] and seems to consistently outperform other NLCG-type methods. For numerical stability, $\mathbf{u}^{(m)} - \tau^{(m)}\mathbf{u}^{(m-1)}$ for some weight $\tau^{(m)}$ can be used instead of $\mathbf{u}^{(m-1)}$. When used with multivectors $\mathbf{u}^{(m)}, \mathbf{r}^{(m)}$, this is the well-known LOBPCG method [30].

LOCG can be accelerated by minimizing over the subspace of more iterates $\mathbf{u}^{(m)}$. When the subspace reaches its maximum size we must restart. If we restart through IRL, however, we lose the single important direction ($\mathbf{u}^{(m-1)}$) that offers the excellent convergence to LOCG. This was first observed in [39] for the Davidson method, although under a different viewpoint. In [54] we offered a theoretical justification and an efficient implementation that combined this technique with thick restarting for the GD method. In [57], we noted the connection of our method, which we call GD+k, to the quasi-Newton and in particular to the limited memory BFGS method [18, 40].

GD(m_{min}, m_{max})+k uses a basis of maximum size m_{max}. When m_{max} is reached, we compute the m_{min} smallest Ritz vectors, $\mathbf{u}_i^{(m)}$, and also k of the corresponding Ritz vectors $\mathbf{u}_i^{(m-1)}$ from step $m-1$. An orthonormal basis for this set of m_{min}+k vectors becomes the restarted basis. If the block size in GD/JD is b, it is advisable to keep k $\geq b$ to maintain good convergence for all block vectors. The special case of block GD(b, $3b$)+b is equivalent mathematically to LOBPCG with the same block size. In [57], we showed that convergence of the GD+k for one eigenvalue is often

ALGORITHM 2.1. [illegible] m_{min}, m_{max} [illegible]

(1) start with $\mathbf{v}_0$ starting vector
(2) $\mathbf{t}^{(0)} = \mathbf{v}_0,\ l = m = nmv = 0,\ X = [\,]$
(3) **while** $l < nev$ **and** $nmv < max_num_matvecs$
(4) Orthonormalize $\mathbf{t}^{(m)}$ against $\mathbf{v}_i, i = 1, \ldots, m$ and $\mathbf{x}_i, i = 1, \ldots, l$
(5) $m = m + 1$, $nmv = nmv + 1$, $\mathbf{v}_m = \mathbf{t}^{(m-1)}$, $\mathbf{w}_m = A\mathbf{v}_m$
(6) $H_{i,m} = \mathbf{v}_i^T \mathbf{w}_m$ for $i = 1, \ldots, m$
(6.0) $s_i^{old} = s_i,\ i = 1, \ldots, m,$
(7) compute eigendecomposition $H = S\Theta S^T$ with $\theta_1 \le \theta_2 \le \cdots \le \theta_m$
(8) $\mathbf{u}^{(m)} = Vs_1$, $\theta^{(m)} = \theta_1$, $\mathbf{w}^{(m)} = Ws_1$
(9) $\mathbf{r}^{(m)} = \mathbf{w}^{(m)} - \theta^{(m)}\mathbf{u}^{(m)}$
(10) **while** $\|\mathbf{r}^{(m)}\| \le tol$
(11) $\lambda_{l+1} = \theta^{(m)}$, $X = [X, \mathbf{u}^{(m)}]$, $l = l + 1$
(12) **if** $(l = nev)$ **stop**
(13) $m = m - 1$, $H = 0$
(14) **for** $i = 1, \ldots, m$
(15) $\mathbf{v}_i = Vs_{i+1}$, $\mathbf{w}_i = Ws_{i+1}$
(16) $H_{ii} = \theta_{i+1}$, $s_i = e_i$, $\theta_i = \theta_{i+1}$
(17) **end for**
(18) $\mathbf{u}^{(m)} = \mathbf{v}_1$, $\theta^{(m)} = \theta_1$, $\mathbf{r}^{(m)} = w_1 - \theta^{(m)}\mathbf{u}^{(m)}$
(19) **end while**
(20) **if** $m \ge m_{max}$ then
(20.0) Orthogonalize $s_i^{old},\ i = 1, \ldots, \mathrm{k}$, among themselves
and against $s_i,\ i = 1, \ldots, m_{min}$,
(20.1) Compute $H_{sub} = s^{oldT} H s^{old}$
(20.2) Set $s = [s_1, \ldots, s_{m_{min}}, s_1^{old}, \ldots, s_{\mathrm{k}}^{old}]$
(21) $H = 0$
(22) **for** $i = 2, \ldots, m_{min} + \mathrm{k}$
(23) $\mathbf{v}_i = Vs_i$, $\mathbf{w}_i = Ws_i$, $H_{ii} = \theta_i$
(24) **end for**
(25) $\mathbf{v}_1 = \mathbf{u}^{(m)}$, $\mathbf{w}_1 = \mathbf{w}^{(m)}$, $H_{11} = \theta^{(m)}$, $m = m_{min}$
(25.0) $H(m_{min} + 1 : m_{min} + \mathrm{k}, m_{min} + 1 : m_{min} + \mathrm{k}) = H_{sub}$
(25.1) $m = m_{min} + \mathrm{k}$
(26) **end if**
(27) Precondition the residual $\mathbf{t}^{(m)} = Prec(\mathbf{r}^{(m)})$
(28) **end while**

ALGORITHM 2.1.

indistinguishable from the optimal method. Yet, higher iteration costs than JDQMR make it less competitive for very sparse operators.

When seeking many eigenvalues, we have never found the use of a block size $b > 1$ beneficial, even with a small subspace acceleration. Then, as with JDQMR, each eigenpair has to be targeted and converged independently. The role of subspace acceleration is perhaps more important for GD+k than for JDQMR, as it provides both its local optimal convergence and the initial guesses for nearby eigenvalues.

2.3. The GD+k and the JDQMR algorithms. The GD algorithm can serve as the basic framework for implementing both JDQMR and GD+k. Algorithm 2.1

depicts a version of the basic GD+k algorithm for finding *nev* smallest eigenpairs. It implements Rayleigh–Ritz and locking and is the basis for the performance models in section 4. The implementation of +k restarting is shown at steps numbered with decimal points. Both steps (20.0) and (20.1) apply on vectors of size m_{max}, and therefore the cost of the GD(m_{min}, m_{max})+k implementation is the same as that of GD(m_{min}+k, m_{max}). In our experience, k = 1 or 2 is always sufficient, obviating the use of larger m_{min} and hence being less expensive. Although not explored here, Algorithm 2.1 can be viewed as a block method by implementing vectors as multivectors.

Step (27) differentiates between eigenmethods. By returning $\mathbf{t}^{(m)} = \mathbf{r}^{(m)}$ or the preconditioned residual, $\mathbf{t}^{(m)} = K^{-1}\mathbf{r}^{(m)}$, we have the GD+k method. Step (27) can also return a JD correction vector. Without inner iterations, a preconditioner K can be inverted orthogonally to the space $Q = [X, \mathbf{u}^{(m)}]$ and applied to $\mathbf{r}^{(m)}$. Assuming $(Q^T K^{-1} Q)$ is invertible, the pseudoinverse of this preconditioner can be written as

$$\left((I - QQ^T)K(I - QQ^T)\right)^+ = (I - K^{-1}Q(Q^T K^{-1}Q)^{-1}Q^T)K^{-1}(I - QQ^T) \tag{2.3}$$

$$= K^{-1}(I - Q(Q^T K^{-1}Q)^{-1}Q^T K^{-1})(I - QQ^T). \tag{2.4}$$

When this preconditioner is used in an iterative method on (2.1), a significant extra storage for $K^{-1}Q$ is required to avoid doubling the number of preconditioning operations. In [15, 51, 6] it is shown that the JD method can be implemented with one projection with Q per iteration. In the following section we show that this may not be sufficient with right preconditioning and high convergence tolerance. Our JDQMR algorithm uses the GD+k as the underlying outer method and, at step (27), calls the symmetric, right preconditioned QMR as described in detail in [57].

3. Avoiding the JD oblique projectors. When eigenvectors converge in the JD method, they are locked in the array X, and the algorithm targets the next higher eigenpair. As the shift in the correction equation moves inside the spectrum, the equation becomes increasingly indefinite. Although this is not prohibitive for the QMR solver, it usually is unnecessarily expensive, as QMR has to rebuild the lower eigendirections in X sufficiently so that it can converge to the correction which should be orthogonal to X. The JD authors have observed that it is always beneficial to project X from the correction equation. They also mention that the preconditioner needs also to be projected in the same way, so that the image of its operation matches the domain of the projected matrix A. This reasoning, however, is not of practical use because the vectors can be considered embedded in $\Re^N$. A better approach would be to consider the effect of these projectors on the conditioning of the correction equation, and whether it justifies the significant additional expense—either a second preconditioning operation per iteration or an array that stores $K^{-1}X$.

There is little doubt that the skew projection of (2.3) should be used for the Ritz vector $\mathbf{u}^{(m)}$. First, there is a constant additional cost to QMR, which is independent from *nev*. More importantly, an extremely accurate preconditioner may cause the problems seen in Davidson. A recent analysis by Notay [42] shows that, although such problems are rare, they could arise. The Davidson problem is specific to the vector $\mathbf{u}^{(m)}$ and cannot occur if orthogonality to the converged eigenvectors X is not enforced in the correction equation. However, the effectiveness of the preconditioner may change. To simplify notation in this section, we assume that the matrix A and the preconditioner K encapsulate both the shifting with η and the projection (2.3) for $\mathbf{u}^{(m)}$, and focus on $Q = X$. Define also the projector for any matrix B for which

TABLE 3.1
Projection alternatives to the classical JD correction equation (with right preconditioning). The 0/1 *string characterizes whether there is a projection on the left of A, on the right of A, and whether the right projection is skew projection or not. Theoretically JD corresponds to* (111) *although it is typically implemented as* (011).

(Left skew right)	Operator	(Left skew right)	Operator
111	$PAP_{K^{-1}}K^{-1}$	011	$AP_{K^{-1}}K^{-1}$
101	$PAPK^{-1}$	001	APK^{-1}
100	PAK^{-1}	000	AK^{-1}

Q^TBQ is not singular,

$$P_B = (I - BQ(Q^TBQ)^{-1}Q^T), \tag{3.1}$$

and note that if $P = (I - QQ^T)$, in addition to (2.3), (2.4) it holds that

$$PP_BBP = P_BB = BP_B^T = PBP_B^TP. \tag{3.2}$$

Based on the above, the appropriate formulation for right preconditioning of the JD projected operator in the correction equation is

$$PAP(PKP)^+ = PAP_{K^{-1}}K^{-1} \tag{3.3}$$

$$= PAK^{-1}P_{K^{-1}}^T. \tag{3.4}$$

Computationally the above formulation is expensive because at every QMR step it requires two orthogonalizations with Q, from both left and right of A, and a backsolve with the factors of the small matrix $(Q^TK^{-1}Q)^{-1}$. Moreover, to avoid an extra preconditioning operation per iteration, the vectors $K^{-1}Q$ must be stored, which is a very unreasonable storage requirement for a large number of eigenvalues. Naturally, the question is whether some projectors can be avoided, in their skew form or altogether.

Equation (3.3) suggests several variants of a projected operator based on whether we operate with a projector on the left and/or on the right of A, and whether we relax the requirement for a right skew projector, replacing it with P. Table 3.1 summarizes the variants. Note that the cases $PAK^{-1}P_{K^{-1}}^T$ and $AK^{-1}P_{K^{-1}}^T$ (from (3.4)) are equivalent to (111) and (011), respectively, so they are not included. Additionally, when the QMR iterates are orthogonal to Q, a P projection on the right of the preconditioner is unnecessary; hence we have not included the case $PAK^{-1}P$. Finally, we do not include the variant $AK^{-1}P$ because it maintains no orthogonality of the QMR iterates. We discuss each column of the table separately.

3.1. No *P* projection on the left of *A*. Case (000) does not work orthogonally to Q, and it is expected to require a sharply increasing number of inner iterations as more eigenvalues are locked, especially with less effective preconditioners. There is a notable exception. When K has the same eigenvectors as A (e.g., if K is a polynomial of A or simply $K = I$), QMR starts with the residual $\mathbf{r}^{(m)} \perp Q$ and thus all its iterates will stay in $Q^\perp$. This property is sometimes exploited in methods with polynomial acceleration [60] (see also [4] in the context of an a priori known null space in Maxwell's equations). Floating point arithmetic and the fact that the eigenvectors in Q are converged to *tol*, not to machine precision, will eventually introduce Q components that QMR will have to remove by additional iterations. However, a few additional iterations are a small price to pay for removing the limiting scalability factor

$O(nev^2N)$ of orthogonalization. In all our experiments, unpreconditioned JDQMR-000 (no projections) achieves an almost linear scaling with nev, both in convergence and in time.

Case (011) is the recommended way to implement right preconditioning in JD. For any vector v, we have $P_{K^{-1}}K^{-1}v \in Q^{\perp}$, and thus $AP_{K^{-1}}K^{-1}v \in Q^{\perp}$, requiring no additional left orthogonalization. This formulation, however, may have disadvantages when the required relative convergence tolerance is much larger than machine precision (e.g., $tol \geq \|A\|\sqrt{\epsilon_{\text{machine}}}$). In such cases, large Q components may appear early in the QMR iterations, delaying its convergence. Finally, we note that the same concern holds for the (001) case, which we do not examine further because even if we fixed this concern, for example, by using case (101), it would still not be competitive for the reason described in section 3.2.2.

3.2. P projection on the left of A. Cases that project with P on the left of A guarantee that all QMR iterates will be exactly in $Q^{\perp}$, regardless of how close Q is to an A-invariant subspace. Case (111) implements accurately the JD projection, but it is very expensive and storage demanding. Cases (101) and (100) approximate equations (3.3) and (3.4), respectively, by replacing the skew projectors with P, thus avoiding the additional storage. In addition, (100) does not require the right projector, so it is the least expensive method that can maintain the QMR iterates in $Q^{\perp}$. If conditioning of the matrix in (100) does not differ much from the matrix in (111), case (100) should be preferred.

To examine the condition numbers of the projected matrices, we note first that when B has a nontrivial null space, its condition number is defined in its range as $\kappa(B) = \|B\|\|B^{+}\|$. We also need the following elementary property for pseudoinverses.

PROPOSITION 3.1. [illegible] A B [illegible]

$$(AB)^{+} = B^{+}A^{+}.$$

[illegible] The pseudoinverse of a matrix A is defined as $A^{+} = V\Sigma^{-1}U^{T}$, where $A = U\Sigma V^{T}$ is the economy size SVD of A which consists only of the nonzero singular values and vectors. Similarly, the pseudoinverse for $B = YMZ^{T}$ is $B = ZM^{-1}Y^{T}$. Let $X \in \Re^{N,k}$, $k < N$, be an orthonormal basis for the range of A and B. Because A, B are symmetric, their SVDs can be described as $A = U\Sigma(XQ_A)^{T}$ and $B = XQ_BMZ^{T}$ for some orthogonal matrices $Q_A, Q_B \in \Re^{k,k}$. By substituting $Y = XQ_B$ and $V = XQ_A$ and using a known property of pseudoinverses, we have

$$\begin{aligned}(AB)^{+} &= (U\Sigma V^{T}YMZ^{T})^{+} = Z(\Sigma V^{T}YM)^{-1}U^{T} \\ &= ZM^{-1}Y^{T}Y(V^{T}Y)^{-1}\Sigma^{-1}U^{T} = B^{+}Y(V^{T}Y)^{-1}\Sigma^{-1}U^{T} \\ &= B^{+}XQ_B(Q_A^{T}Q_B)^{-1}\Sigma^{-1}U^{T} = B^{+}XQ_A\Sigma^{-1}U^{T} = B^{+}A^{+}. \qquad \square\end{aligned}$$

Using Proposition 3.1 with a two norm and (3.3), (3.4), we can derive the condition numbers for cases (111), (101), and (100):

$$\begin{aligned}\kappa_{111} &= \|PAP(PKP)^{+}\|\|PKP(PAP)^{+}\| = \|PAK^{-1}P_{K^{-1}}^{T}\|\|PKA^{-1}P_{A^{-1}}^{T}\|, \\ \kappa_{101} &= \|PAPK^{-1}P\|\|(PK^{-1}P)^{+}(PAP)^{+}\| = \|PAPK^{-1}P\|\|P_KKPA^{-1}P_{A^{-1}}^{T}\|, \\ \kappa_{100} &= \|PAK^{-1}P\|\|(PAK^{-1}P)^{+}\| = \|PAK^{-1}P\|\|P_{KA^{-1}}KA^{-1}\|.\end{aligned}$$

There are obvious similarities between the condition numbers, but no one emerges as the default winner. The effect of the projectors $P_{K^{-1}}^{T}, P_{A^{-1}}^{T}$, and $P_{KA^{-1}}$ depends on how close A, K, K^{-1}, and KA^{-1} are to Q-invariant. We therefore examine the following two special, extreme cases.

3.2.1. Special case: Q are exact eigenvectors of A. This assumption leads to $PA = AP$, $P_{KA^{-1}} = P_K$, $P^T_{A^{-1}} = P$, and, by considering (3.2), to the following simplifications:

$$\kappa_{111} = \|PAK^{-1}P^T_{K^{-1}}\|\|PKA^{-1}P\|,$$
$$\kappa_{100} = \kappa_{101} = \|PAK^{-1}P\|\|P_KKA^{-1}P\|.$$

The differences between κ_{111} and κ_{100} may be better understood if we consider how close Q is to an invariant subspace of K. If we denote as $R_K = KQ - Q(Q^TKQ)$ the residual of Q in K, we have $P_K = P - R_K(Q^TKQ)^{-1}Q^T$, and κ_{100} becomes

$$\begin{aligned}\kappa_{100} &= \|PAK^{-1}P\|\|PKA^{-1}P - R_K(Q^TKQ)^{-1}Q^TKPA^{-1}\| \\ &= \|PAK^{-1}P\|\|PKA^{-1}P - R_K(Q^TKQ)^{-1}R_K^TA^{-1}\|. \end{aligned} \tag{3.5}$$

Similarly, with $R_{K^{-1}} = K^{-1}Q - Q(Q^TK^{-1}Q)$ and $P^T_{K^{-1}} = P - Q(Q^TK^{-1}Q)^{-1}R_{K^{-1}}$,

$$\kappa_{111} = \|PAK^{-1}P - AR_{K^{-1}}(Q^TK^{-1}Q)^{-1}R^T_{K^{-1}}\|\|PKA^{-1}P\|. \tag{3.6}$$

Moreover, let $\delta_0 = \|R_K(Q^TKQ)^{-1}R_K^TA^{-1}\|$ and $\delta_1 = \|AR_{K^{-1}}(Q^TK^{-1}Q)^{-1}R^T_{K^{-1}}\|$. Then, $\delta_0 = O(\|R_K\|^2)$ and $\delta_1 = O(\|R_{K^{-1}}\|^2)$, and through some basic algebra we can derive

$$|\kappa_{111} - \kappa_{000}| \le \delta_0\|P_KKA^{-1}P\| + \delta_1\|PAK^{-1}P^T_{K^{-1}}\| = O(\|R_K\|^2 + \|R_{K^{-1}}\|^2). \tag{3.7}$$

Equation (3.7) implies that the differences between the two methods disappear with better preconditioners, when Q becomes close to K-invariant (depending on the squares of the residuals). When Q is far from K-invariant the two condition numbers may not be close. However, there is no reason why κ_{111} should be smaller than κ_{100}. If we assume, according to (3.5), (3.6), that smaller residuals may lead to smaller condition numbers, then obtaining $\|R_K\| \ll \|R_{K^{-1}}\|$ would depend on the norms of $\|K\|$ and $\|K^{-1}\|$ and on what part of the K spectrum is approximated by Q and how well. In extensive experiments on practical cases we have noticed negligible differences between cases (111) and (100).

3.2.2. Special case: Q inexact eigenvectors but $K = A$. In this extreme case, $\kappa_{111} = \kappa_{100} = 1$, but κ_{101} can be much worse. To see why, let the residuals $R_A = AQ - Q(Q^TAQ)$ and $R^T_{A^{-1}} = Q^TA^{-1} - (Q^TA^{-1}Q)Q^T$, and consider the norms for κ_{101}:

$$\begin{aligned}\|PAPK^{-1}P\| &= \|P(A - AQQ^T)A^{-1}P\| \\ &= \|P(I - (R_A + Q(Q^TAQ))Q^TA^{-1})P\| \\ &= \|P - P(R_A + Q(Q^TAQ))(R^T_{A^{-1}} + (Q^TA^{-1}Q)Q^T)P\| \\ &= \|P - R_AR^T_{A^{-1}}\|, \\ \|P_KKPA^{-1}P^T_{A^{-1}}\| &= \|P + R_A(Q^TAQ)^{-1}(Q^TA^{-1}Q)^{-1}R^T_{A^{-1}}\|. \end{aligned}$$

Obviously, interleaving an inaccurate eigenprojector between A and K can cause κ_{101} to be far from 1, despite the double orthogonalization expense. The same effects remain, only exaggerated, if the left projection is avoided as in the (001) case.

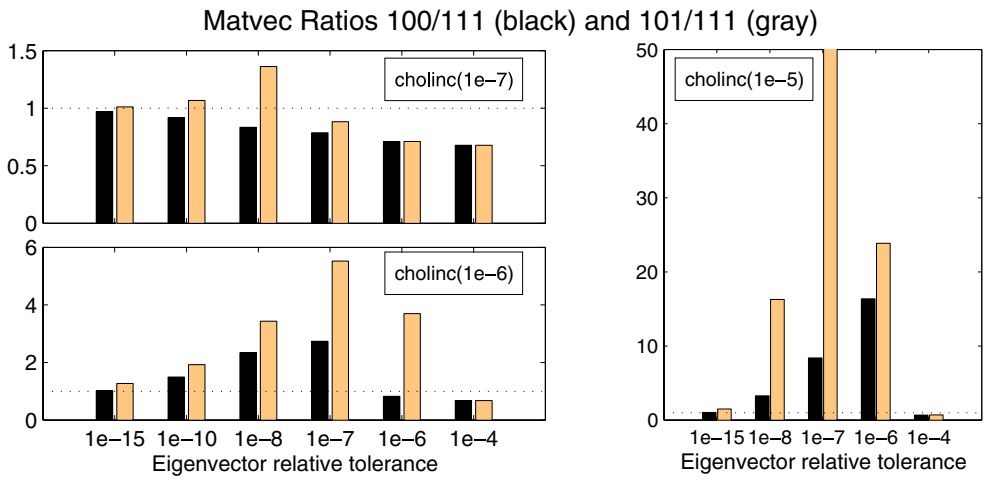


FIG. 3.1. *We look for* 20 *lowest eigenpairs of the matrix PLAT*362 *from Harwell Boeing. We show the ratios of the number of matrix-vector products (including the preconditioner application) of method* (100) *over* (111) *(black bars), and* (101) *over* (111) *(gray bars). The three boxes depict results from preconditioning with the MATLAB* `cholinc`*, with thresholds* `1e-7, 1e-6`*, and* `1e-5`*, respectively. In each box, the x-axis varies the relative tolerance for computed eigenvectors; i.e., the residual of* Q *is less than* $\|A\|_F tol$*. The example was picked because it shows two possible anomalies. First, decreasing both eigenvector accuracy (around* $\sqrt{\epsilon_{machine}}$*) and the quality of the preconditioner could necessitate the use of* (111)*. Second, case* (101) *shows a much higher sensitivity than* (100) *to the accuracy of the eigenvalues. Even in this "anomaly" case, with very good or very bad eigenvector accuracy, method* (100) *takes fewer iterations than* (111)*.*

3.2.3. General case. When both Q and K are not accurate, it is hard to quantify the relative merits of κ_{111} and κ_{100}. However, the Q are not random vectors but approximate eigenvectors to a certain threshold (typically at least as good as single precision). Therefore, we expect our results of section 3.2.1 to apply in general. In particular, (3.5), (3.6) suggest that method (100) should be preferred over the expensive (111) with good preconditioners. This is in contrast to the situation for the Ritz vector $\mathbf{u}^{(m)}$, where skew projectors are especially needed when $K \approx A$.

In our extensive experiments, we never found it necessary to use method (111) when the eigenvectors were computed relatively accurately. In fact, method (100) consistently gave lower iteration numbers. Despite the lack of theoretical assurance, we have observed similar behavior even with smaller eigenvector accuracy (10^{-7} relative tolerance in the next section). Situations can arise, however, where (111) could be beneficial. We have been able to identify only one such case. Figure 3.1 shows the results from this "anomaly" case. Even so, method (100) is still competitive and often better than (111) except for a small range of carefully picked parameters. The figure also demonstrates the disadvantages of method (101).

4. Modeling the relative asymptotic behavior between methods. The challenge in relative modeling is that practical models must include information about time complexity, type of operations and their performance on certain hardware, and cost of the operators (matvec and preconditioner), but most importantly they must capture the relative convergence behavior of the methods. Insight on the latter arises from lengthy experimentation with the methods on a variety of problems. In the following, we do not separate the experimentation from the modeling process as the two

are complementary. Instead, for each pair of methods, we first model their observed relative convergence behavior. Then, we couple it with time complexity models to predict their relative asymptotic behavior for large *nev*. Because the meaning of iteration differs among methods, we normalize all models as cost per operator application (that includes matvec and preconditioner, if available).

4.1. Experimentation environment. In our experiments we use actual production codes, not prototypes, that implement state-of-the-art methods. Our GD+k and JDQMR methods are implemented in C, in the package PRIMME: PReconditioned Iterative MultiMethod Eigensolver [34]. From various methods available in PRIMME, we compare only GD+k and the following JDQMR variants (following notation from the previous section): JDQMR-000, JDQMR-100, JDQMR-011, JDQMR-111. With the exception of forcing these projector configurations, the default parameters provided by PRIMME are used in all the experiments. These defaults include the use of locking, block size of 1, $m_{min} = 6$, $m_{max} = 18$, k = 1 with preconditioning, and k = 2 without it. The methods converge when the residual norm of each of the *nev* required eigenpairs is less than $\|A\|_F tol$, where $\|A\|_F$ is the Frobenius norm of A.

The only other symmetric JD implementation available is JDBSYM [17]. It is also written in C, it is a block method, but it stops the inner iteration using the scheme proposed in [51, 12]. JDBSYM provides several choices of inner iterative methods. We opt for the symmetric QMR to facilitate a comparison with JDQMR. We use the same m_{min}, m_{max} for the JD basis, block size of 1, a maximum number of 200 inner iterations, TOLDECAY = 1.5, symmetric preconditioning OPTYPE, and strategy = 0. In certain cases strategy = 1 was necessary to achieve convergence. JDBSYM finds only eigenvalues closest to a shift τ—not extreme ones. In all our tests, we provide τ as a small, left perturbation of the precomputed λ_1 and let JDBSYM switch to using the Ritz values as shifts when EPS_TR $= 10^{-3}\|A\|_F/\sqrt{N}$. Convergence is declared when all residual norms fall below $\|A\|_F tol$.

From the class of Lanczos methods we use the ARPACK software [33]. The basis size for ARPACK is chosen as $\max(40, 2nev)$ for $nev < 400$ and $1.5nev$ for $nev \geq 400$. The tolerance *tol* is provided directly to ARPACK.

Finally, we compare against BLOPEX, a C implementation of LOBPCG [27]. We chose to implement a wrapper around BLOPEX(b) that uses locking to compute *nev* eigenvalues a block, b, at a time. After some experimentation, we found $b = 10$ to be the best choice for most problems. For large *nev*, BLOPEX(*nev*) was several times slower than BLOPEX(10). We ask for convergence tolerance of $\|A\|_F tol$.

All methods start with the same random initial guess. We have run experiments for two different tolerances: *tol* = 1e-15 and *tol* = 1e-7. For BLOPEX we report only results for *tol* = 1e-7, as it could not produce results with the lower tolerance. We run experiments on an Apple G5 with 1 GB of memory and two 2GHz processors, each with 512 MB L2 cache. The C codes are compiled using the gcc-4.0.0 compiler with -O3 flag, and the ARPACK is compiled with the g77 compiler. We link with the Apple vecLib library that includes optimized versions of BLAS/LAPACK libraries.

We use eight matrix problems (as shown in Table 4.1), six from the University of Florida [11] and the FEAP [3] collections, and two standard seven-point three-dimensional Laplacian matrices generated by SPARSKIT [47] with zero Dirichlet boundary conditions. The matrix sizes are far beyond toy problems, yet they make finding hundreds or thousands of eigenvalues tractable on a workstation. They also represent different applications, operators, and sparsities. The smallest side of the spectrum is hard to obtain for all matrices, while the largest side is easier for several

TABLE 4.1
The matrices used in the experiments, their size, nonzero elements per row, and their source.

Matrix	N	Nz/row	Source	Matrix	N	Nz/row	Source
torsion1	40000	4.94	UF	Cone_A	22032	65.04	FEAP
Andrews	60000	12.67	UF	Plate33K_A0	39366	23.22	FEAP
cfd1	70656	25.84	UF	Lap7pt15K	12167	6.74	SKIT
finan512	74752	7.99	UF	Lap7pt125K	110592	6.88	SKIT

TABLE 4.2
The parameters used to model the various methods, their meaning, and range of values.

Name	Description	Value
$OP(OP_p)$	Cost of the matvec (matvec+preconditioner)	
c_a	ARPACK iteration cost per eigenvalue	$14N$
c_g	Cost of reorthogonalizing locked vectors in GD	$4N$
c_p	Cost of projecting the locked vectors in QMR	$2N$
c_q	Symmetric QMR cost per step	$21N$
c_{qr}, c_{gr}	Outer iteration costs for JDQMR/GD+1	(see section 4.5)
C	$MV_{jdqmr000}/(MV_{arp}nev)$	[0.1, 0.25]
f	$f = c_g/c_a$	4/14
f_p	$f_p = MV_{jdqmr100}/MV_{jdqmr000}$	
k_{inn}^{xyz}	Number of inner iterations in JDQMR-xyz	> 1
γ	$\gamma = MV_{jdqmr100}/MV_{GD+1}$	[1.1, 2.4]

of them. Finally, Table 4.2 provides a look-up reference for the parameters used in our modeling, their meaning, and their typical range of values.

4.2. JDQMR-000 versus ARPACK. Because the size of the ARPACK basis increases with nev, so does its effectiveness. Thus, the number of matrix vector operations per eigenvalue found by ARPACK is expected to decrease rapidly with nev. In contrast, the number of matvecs per eigenvalue found for JDQMR-000 is expected to be at least constant or increase slightly for highly interior eigenpairs because of the loss of implicit orthogonality during inner iterations. Although it is difficult to model this decrease/increase of matvecs for the two methods, surprisingly, we can relate empirically the ratio of their number of matvecs. The left graph in Figure 4.1 shows that for sufficiently large nev, the matvec ratio between ARPACK and JDQMR follows a power law. Interestingly, fitting the ratios to a power law curve results in the model

$$MV_{jdqmr000} = C\ nev\ MV_{arp}, \tag{4.1}$$

with $0.1 < C < 0.25$. The same observations hold for the experiments in Figure 4.2.

ARPACK obtains this excellent asymptotic convergence with nev, at the expense of increasing orthogonalization and restarting costs. Based solely on flop counts as a measure of performance and considering only higher order terms and their constants, we have that between restarts ARPACK requires $\sum_{k=nev}^{2nev} 8kN = 3/2nev^2 8N$ flops for (re)orthogonalization and $2nev^2N$ flops for restarting. Averaging over the nev steps between restarts, we obtain $14nevN$ flops per matrix-vector operation. Let $c_a = 14N$ be the constant capturing these costs independently from nev. This facilitates a more general model, as the constants in the flop counts may change depending on implementation and hardware platform. Letting OP denote the cost of the matrix vector operator, the approximate cost function per step for ARPACK is

$$ArpPerMV = c_a\ nev + OP.$$

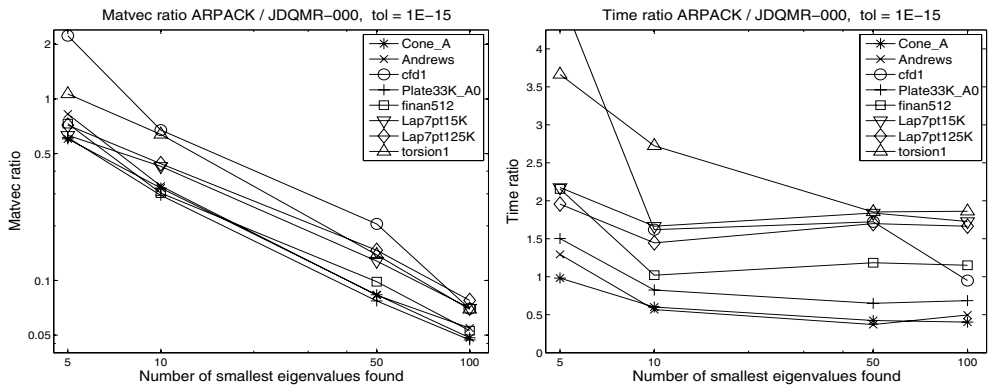


FIG. 4.1. *Relative performance of ARPACK over JDQMR-000 for finding nev smallest eigenvalues of 8 matrices. The left graph shows the matvec ratios. The right graph shows time ratios.*

For JDQMR-000 the cost of the inner iteration is that of the QMR method,

$$InnerPerMV = c_q + OP.$$

If flop count is used as a performance measure, $c_q = 21N$. The outer JD step, which includes orthogonalization against all $l = 1, \ldots, nev$ converged eigenvalues, occurs only every k_{inn} inner iterations. To avoid counting the matvec of the outer step twice, we include it in the cost of the inner method. Then, we have the following approximate cost model per operator application:

$$OuterPerMV = c_g \; nev/k_{inn}.$$

To provide insight on the size of the constant c_g we can use an accounting method to analyze the average cost per step over all $l = 1, \ldots, nev$. Because the outer iteration is an inexact Newton method, each eigenvalue is obtained with roughly the same number of outer iterations. Then, if m_{op} is the total number of matvecs, one eigenvalue is found every $m_{op}/(k_{inn}nev)$ outer iterations. Therefore, the total orthogonalization cost throughout the execution of the method is $m_{op}/(k_{inn}nev)\sum_{l=0}^{nev-1} 8lN = m_{op}4(nev - 1)N/k_{inn}$. Averaging over all m_{op} matvecs, we obtain $c_g = 4N$.

The outer JD step also incurs costs for orthogonalization of the basis, restarting, and residual computation. These can be twice as expensive as the ARPACK costs if the basis sizes were the same. But with locking, these outer JD costs do not scale with nev, so they are not included in c_g. Thus, asymptotically, $f = c_g/c_a = 4/14 < 1$, but f increases for small nev.

Considering the ratio of the overall JDQMR-000 and ARPACK costs and substituting the matvec model from (4.1), we have

$$r_{arp}^{j000} = \frac{MV_{jdqmr000}(c_q + OP + c_g \; nev/k_{inn})}{MV_{arp}\;(c_a nev + OP)} = \frac{C(c_q + OP)}{c_a + OP/nev} + \frac{Cnev\; c_g/k_{inn}}{c_a + OP/nev}. \tag{4.2}$$

Asymptotically, for large nev and not too dense operators, we have $OP/nev \ll c_a$, and the ratio can be approximated as

$$r_{arp}^{j000} \approx C\;\frac{c_q + OP}{c_a} + \frac{f\; C\; nev}{k_{inn}}. \tag{4.3}$$

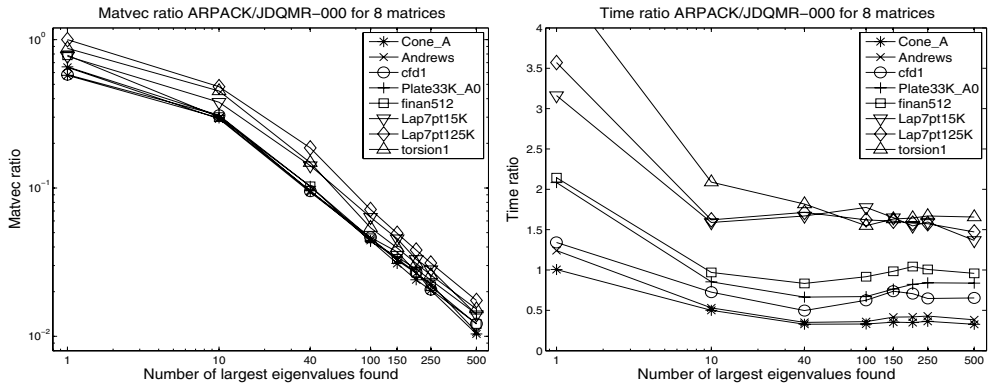


FIG. 4.2. *Relative performance of ARPACK over JDQMR-000 for finding nev largest eigenvalues for tol* $= 1e-15$. *The left graph shows the matvec ratios. The right graph shows time ratios.*

The first summand in (4.3) determines when JDQMR-000 cannot be faster than ARPACK, regardless of k_{inn}, because of too expensive an operator. The first summand, and thus the ratio, is always greater than 1 if $OP > c_a/C - c_q$. In our experiments $C \in [0.1, 0.25]$, [illegible] 000 [illegible] $OP < 119N$ [illegible] $OP < 35N$ (for $C = 0.1$ or $C = 0.25$, respectively). Yet, for many important problems stemming from finite difference or finite element discretizations of PDEs, the matrices are sufficiently sparse to favor JDQMR-000.

Equation (4.3) states that even for those sufficiently sparse matrices, [illegible] 000 [illegible] *nev* Initially, when only some of the eigenvectors have converged, JDQMR-000 spends most of its time in the inner iteration. As more eigenvectors converge, the orthogonalization of the outer step becomes increasingly expensive and will eventually dominate.

Some intuition can be gained by two extreme examples. First, consider a hypothetical case with the least competitive choice of parameters for JDQMR, $C = 0.25$ and $k_{inn} = 10$. The second summand of (4.3) remains less than 0.8 for $nev < 112$. Second, consider an actual experiment with the Lap7pt15K matrix, as shown in Figure 4.3. This case is favorable for JDQMR-000, but it also demonstrates how well the model captures the relative behavior of the two methods. For this case, we empirically measure $C = 0.1$, $k_{inn} = 50$. Also, because of the seven point three-dimensional Laplacian, the operator cost is $14N$. With these values, the model becomes $r = 0.25 + 0.00057\ nev$, dictating that JDQMR-000 is faster than ARPACK for $nev < 1313$, which is close to the crossover point in the experiment. Moreover, for smaller nev values, the first summand dominates, and this is the reason why in Figures 4.1 and 4.2 the time ratio of the two methods for this matrix looks constant for $40 < nev < 500$.

When seeking only a few eigenvalues, JDQMR-000 is always faster than ARPACK, not so much because it avoids the projectors, but because of the nearly optimal convergence for one eigenpair and the cheaper QMR iteration ($21N$ versus $14N$ nev). In (4.2) the second summand is negligible if $nev \ll k_{inn}$, and so $r_{arp}^{j000} < 1$ for $nev = 1$, and r_{arp}^{j000} increases monotonically with nev. Then, $r_{arp}^{j000} \to Cnev$ as $OP \to \infty$, and so, assuming a worst case $C = .25$, JDQMR-000 should always be faster for $nev < 4$, regardless of operator cost. In practice, this model is not meant to be accurate for

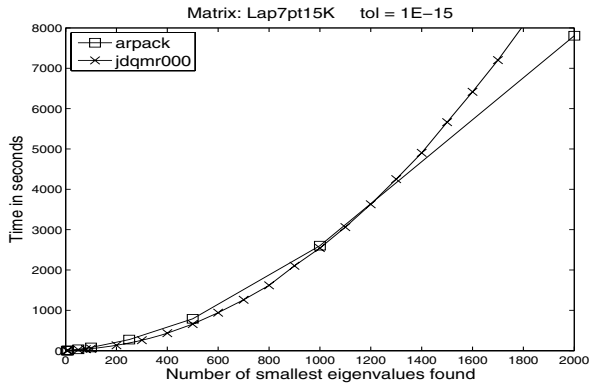


FIG. 4.3. *Time for finding nev smallest eigenvalues for ARPACK and JDQMR*-000. *Although much faster initially, JDQMR*-000 *succumbs to orthogonalization costs of the outer step at* 1100.

small nev, but the intuition it provides agrees with all the experiments we have performed. Moreover, for very high cost operators, GD+k provides a more efficient choice than JDQMR.

In summary, JDQMR-000 is faster than ARPACK for sufficiently sparse matrices and up to a certain nev, which can be $O(100s)$. Beyond that nev, using a larger basis with ARPACK is preferable. For very small nev, JDQMR-000 is always preferable.

4.3. Preconditioned JDQMR-100 versus unpreconditioned JDQMR-000. With a "good" preconditioner, JDQMR-000 can converge quickly on the indefinite correction equation, similarly to an exact shift and invert, obviating the projection with locked eigenvectors. With less accurate preconditioners, our results have been inconclusive. JDQMR-000 would sometimes converge quickly to many extreme eigenvalues, only to slow down appreciably, and unpredictably, on the next unconverged one. For this reason, we do not consider the preconditioned JDQMR-000 further.

Depending on the preconditioner, JDQMR-100 reduces the number of matvecs over unpreconditioned JDQMR-000 by a factor $f_p < 1$: $MV_{jdqmr100} = f_p MV_{jdqmr000}$. The number of matvecs per eigenvalue found by JDQMR-100 may increase with nev because as the shift in the correction equation moves inside the spectrum, the preconditioner, which usually does not change, may not be equally effective. A commensurate increase in matvecs of the unpreconditioned JDQMR-000, however, keeps the f_p roughly constant for different nev. This is depicted for a sample matrix in Figure 4.4, where JDQMR-100 requires 12 times fewer matvecs than the unpreconditioned JDQMR-000 for any nev. Moreover, since the number of outer iterations does not vary substantially, we assume that $k_{inn}^{100} = f_p k_{inn}^{000}$.

In addition to the c_q QMR cost, each JDQMR-100 inner iteration involves a projection with locked eigenvectors (c_p nev) and an operator OP_p which includes matvec and preconditioning. In our flop count model, $c_p = 2N$, which is half of c_g, because we do not reorthogonalize when applying the projector. Reorthogonalization against locked eigenvectors occurs at every outer step for a total of $c_g nev$ cost. Hence, we arrive at the model for the time ratio of the two methods:

$$r_{j000}^{j100} = \frac{MV_{jdqmr100}\ (c_q + OP_p + c_p nev + c_g\ nev/k_{inn}^{100})}{MV_{jdqmr000}\ (c_q + OP + c_g\ nev/k_{inn}^{000})}$$

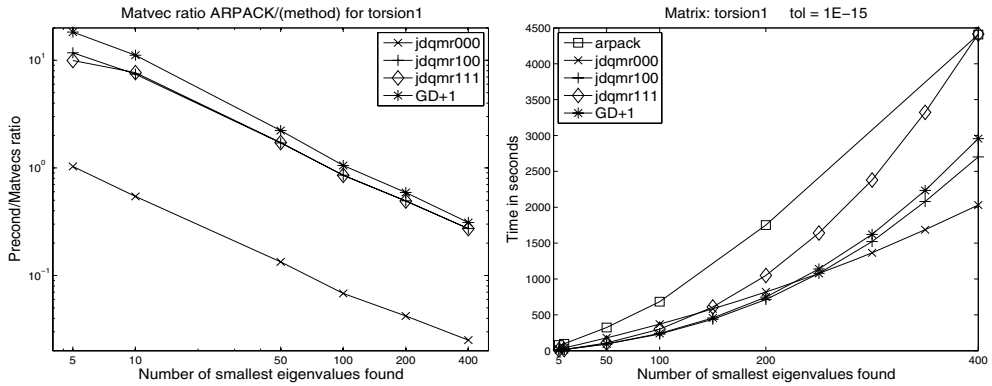


FIG. 4.4. *Matvec ratio of ARPACK over five methods (left graph) and time (right graph) for finding nev smallest eigenvalues. All methods use ILUT*$(40, 1e-3)$ *preconditioning, except for ARPACK and JDQMR-000. Methods that avoid projections are better—even unpreconditioned.*

$$= \frac{f_p(c_q + OP_p + c_p nev) + c_g\ nev/k_{inn}^{000}}{c_q + OP + c_g\ nev/k_{inn}^{000}}. \tag{4.4}$$

For small nev and for preconditioners that are not disproportionally expensive, the ratio is close to f_p, and JDQMR-100 delivers the benefits of preconditioning. For large nev, orthogonalization becomes dominant: $r_{j000}^{j100} \to f_p + f_p k_{inn}^{000}/2 = f_p + k_{inn}^{100}/2$, so since $k_{inn}^{100} > 1$, [illegible] 000 [illegible] 100

Intuitively, the more effective and relatively cheaper the preconditioner, the more eigenvalues JDQMR-100 finds faster before JDQMR-000 becomes beneficial. Figure 4.4 shows an example of this behavior for the torsion1 matrix.

4.4. Preconditioned JDQMR-100 versus ARPACK. Multiplying (4.2) with (4.4) we obtain a time ratio for JDQMR-100 over ARPACK as follows:

$$r_{arp}^{j100} = C f_p\ \frac{c_q + OP_p}{c_a} + C f nev \left(\frac{f_p}{2} + \frac{1}{k_{inn}^{000}} \right), \tag{4.5}$$

where $f = c_g/c_a$ as previously, and OP/nev is negligible. In the example in Figure 4.4, the preconditioned JDQMR-100 has not crossed the ARPACK curve for $nev = 400$, but at that value their growth rates have become similar. For this example, we empirically obtained $f_p = 1/12$, $k_{inn}^{000} = 45$, and $C = 0.1$. Then, according to the model the curves will cross at $nev = 511$, although it is the approximate asymptotic behavior which is of interest here and not the exact crossover point. The curve for the two-projection JDQMR-111 already crosses the ARPACK curve at $nev = 400$.

The last two models suggest that [illegible], for which it may be preferable to switch to unpreconditioned JDQMR-000 (if the matrix operator is inexpensive) or ARPACK (if the matrix operator is expensive). The crossover for that switch, however, depends on the preconditioner and may be a very large nev number.

4.5. JDQMR-100 versus GD+1. We conclude this section with a discussion on the relative performance behavior of GD+1 and JDQMR-100. As we observed in all our experiments and conjectured in [57], the ratio of matvecs $\gamma = MV_{jdqmr1000}/MV_{GD+1}$ is typically $\gamma \in [1, 2.4]$ and usually closer to 1.5. Because

TABLE 4.3

JDQMR-100 versus GD+1: The crossover cost in flops of the combined matrix and preconditioner (OP_p). JDQMR-100 is faster for OP_p costs less than the given values. GD + 1 is faster otherwise. For a certain γ and k_{inn} we also provide the crossover as a function of nev.

	k_{inn} :	10	30	∞
$\gamma = 1.5$	$OP_P <$	$(323 + 0.8nev)N$	$(363 + 1.6nev)N$	$(383 + 2nev)N$
	$nev = 10$	$331N$	$379N$	$403N$
	$nev = 100$	$403N$	$523N$	$583N$
$\gamma = 2$	$OP_P <$	$(141 - 0.8nev)N$	$(168 - 0.3nev)N$	$(181 - 8\frac{nev}{k_{inn}})N$
	$nev = 10$	$133N$	$165N$	$181N$
	$nev = 100$	$61N$	$141N$	$181N$

of similar costs, to compare the two methods for small nev, we must complement the asymptotic model with the outer JD costs that do not depend on nev (residual computation and restarting).

We use the accounting method of section 4.2 to extend the flop-count cost model in [57] to $nev > 1$. The remaining costs that do not depend on nev are given for GD+1 by $c_{gr} = (11.4m_{max} + 18)N$, and for the outer iteration of JDQMR-100 by $c_{qr} = (11.4m_{max} - 3)N$. Hence, the time ratio can be modeled as

$$r_{GD+1}^{j100} = \gamma \frac{(c_q + OP_p + c_p nev + (c_{qr} + c_g\ nev)/k_{inn})}{(c_{gr} + c_g\ nev + OP_p)} \longrightarrow \gamma \left(\frac{1}{2} + \frac{1}{k_{inn}} \right).$$

JDQMR-100 is faster than GD+1 asymptotically if $\gamma \in [1, 2)$ and $k_{inn} > 2\gamma/(2 - \gamma)$, which is the most typical case. For example, for $\gamma = 1.5$ and $k_{inn} = 10$, JDQMR-100 is 10% faster than GD+1, while with $k_{inn} = \infty$ it can be up to 25% faster. If $\gamma > 2$, then GD+1 is always faster. It is important to note that in either case, [illegible]

This asymptotic performance comparison is valid only for nev that are significantly larger than OP_p/N, which includes the cost of both the matrix and the preconditioning operators and thus can be very expensive. Moreover, in the realm of such large nev, JDQMR-000 is always better than JDQMR-100, and ARPACK is eventually faster than JDQMR-000, so the asymptotic comparison is less relevant. In Figure 4.4, using an expensive preconditioner, the two methods are very close, with GD+1 winning slightly for $nev < 150$, and JDQMR-100 winning slightly for $nev > 200$.

For small nev, the winner is determined by the cost of the operator and the number of inner iterations that JDQMR-100 performs. Considering our flop model for the above constants, including the c_{gr} and c_{qr}, and a typical basis size $m_{max} = 18$, we can obtain [illegible] 100 [illegible] +1

$$OP_p < \frac{1}{\gamma - 1} \left(223.2 - \gamma(21 + 202.2/k_{inn}) + (4 - 2\gamma - 4\gamma/k_{inn})nev\right) N.$$

Table 4.3 shows the cost for OP_p beyond which GD+1 becomes faster than JDQMR-100. Despite the asymptotic behavior, with expensive preconditioners the case for GD+1 is compelling even for 100 eigenvalues. Our experiments in the next section use an ILUT preconditioner with a rather large fill-in, and as expected the GD+1 is the fastest method.

We emphasize that we do not advocate the use of flop-counts in the models. They serve only to obtain qualitative information on the trade-offs between methods. Yet, our models are built on top of much more general cost variables (such as c_g or c_q) and operator costs (OP, OP_p) that depend on the implementation and hardware platform.

More importantly, they can be measured not only a priori but also at runtime, thus enabling a dynamic and possibly autonomic method selection that depends on the problem at hand. This is a direction of future research.

5. Comparisons with other methods. In this section we provide further experiments with and without preconditioning, for high and low tolerances, and for both sides of the spectrum. We have already demonstrated the relative asymptotic strengths between GD+k, JDQMR-100, and JDQMR-000. The goals of the following experiments are (a) to show that by avoiding the oblique projector in JDQMR-100, convergence is almost identical to (and sometimes better than) JDQMR-011, and JDQMR-111, but in significantly less time and/or storage, (b) to compare with JDBSYM and BLOPEX, and (c) to confirm some modeling predictions in various cases.

5.1. Without preconditioning. We look for the smallest 100 eigenvalues, and ask for *tol*=1e-15. In Figure 5.1 the left graph shows the linear scaling of matvecs with the number of eigenvalues found for all JDQMR and GD+k methods, except for a slight increase in JDQMR-000. JDBSYM has problems scaling to many eigenvalues for this matrix. The right graph shows the quadratic $O(nev^2)$ behavior in methods with projections, while JDQMR-000 achieves (at least up to 100 eigenvalues) what was designed for: almost linear scaling with *nev*. The matrix, however, has 65 elements per row, yielding an expensive $130N$ operator. Thus, the significant reduction in matvecs from the large ARPACK basis is hard to match. Notice also that JDQMR-100 and JDQMR-011 converge identically.

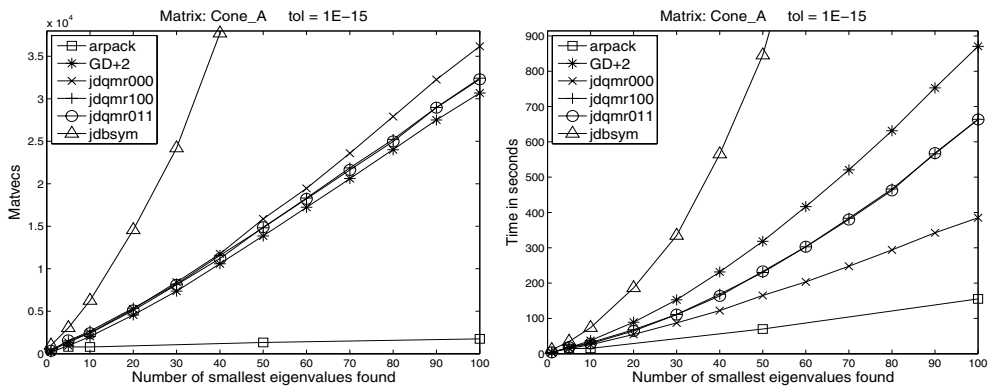


FIG. 5.1. *Matvecs (left graph) and time (right graph) of six methods for nev smallest eigenvalues.*

In Figure 5.2, all JD/GD methods converge very similarly, which means that the stopping criteria of JDBSYM work well in this case. Most methods, particularly JDQMR-000, are better than ARPACK up to 50 eigenvalues, but for $nev = 100$ the much larger ARPACK basis captures some parts of the spectrum that smaller bases could not. ARPACK takes fewer matvecs to find 100 eigenpairs than 50, matching the time of JDQMR-000. This spectrum dependent behavior is hard to model in general.

In Figure 5.3, a sparser matrix, finan512, is considered, and so JDQMR-000 is several times faster than any other JD/GD method and better than ARPACK, exactly as predicted by our model. Again JDBSYM is only slightly slower than JDQMR-100 and JDQMR-011. The last two are again identical.

In Figure 5.4, we consider the largest matrix in the group, whose eigenvalues are all of multiplicity 3 or 6. JDBSYM cannot converge in tractable time for this matrix

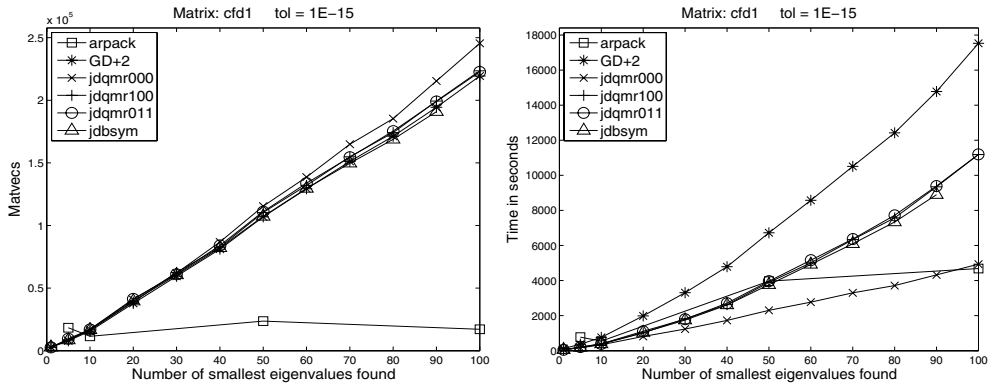


FIG. 5.2. *Matvecs (left graph) and time (right graph) of six methods for nev smallest eigenvalues.*

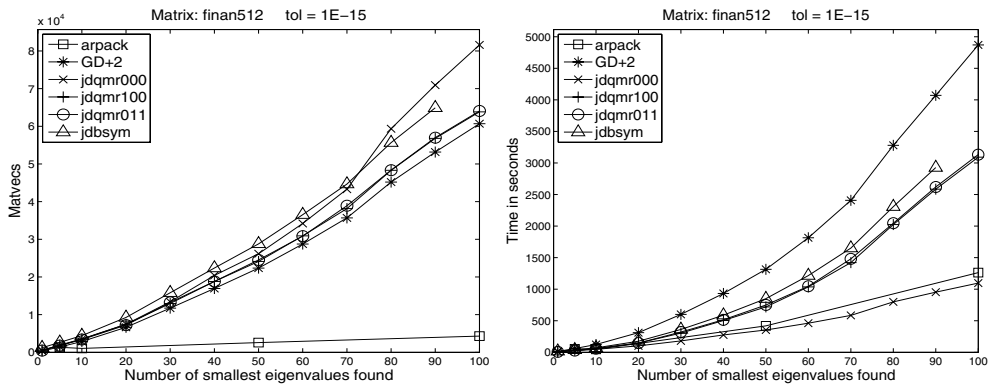


FIG. 5.3. *Matvecs (left graph) and time (right graph) of six methods for nev smallest eigenvalues.*

with strategy = 1, and strategy = 0 performed worse. The sparsity of this Laplacian makes JDQMR-000 significantly faster than all other methods. In all four examples, GD+2 offers the fastest convergence among JD methods in terms of matvecs but not execution time because of its more expensive iteration. Results for the 100 smallest eigenvalues from all matrices were summarized in Figure 4.1.

In the previous examples BLOPEX could not reach the required *tol*. Figure 5.5 shows results from the Cone_A matrix, but with *tol*=1e-7. As with *tol* =1e-15, JDBSYM does not converge well for this matrix. BLOPEX with block size of 10 not only is slower than GD+2/JDQMR methods, but its convergence does not scale linearly with *nev*. For this matrix, the relative behavior of the rest of the methods is similar to *tol* =1e-15 (Figure 5.1).

In Figure 5.6 we observe a significant deterioration of the performance of ARPACK over the *tol* =1e-15 case in Figure 5.2. Closer scrutiny of the iterations between the two figures reveals they are about the same. ARPACK does not benefit from the higher threshold, still computing almost all 100 eigenpairs to full accuracy. We observed this behavior of ARPACK with high tolerances in the majority of our experiments. Surprisingly, JDBSYM is much slower for *tol*=1e-7 than with full accuracy (compare with Figure 5.2). BLOPEX now scales linearly with *nev*, but it is significantly worse than any GD+k/JDQMR method, even for the smallest 10 eigenpairs.

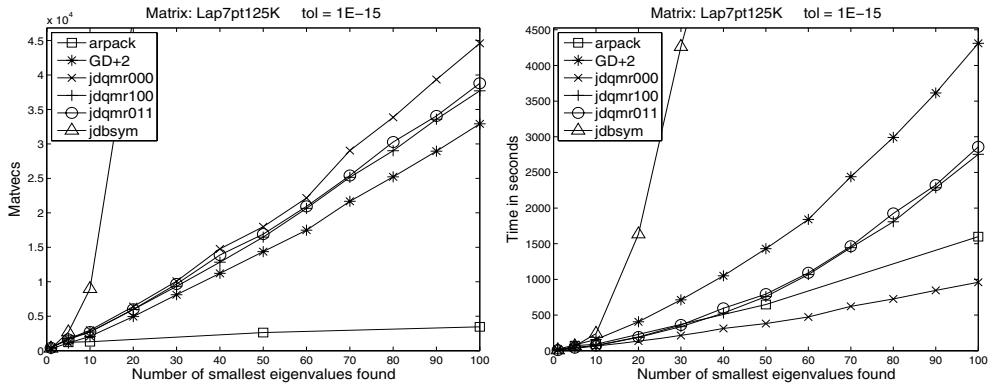


FIG. 5.4. *Matvecs (left graph) and time (right graph) of six methods for nev smallest eigenvalues.*

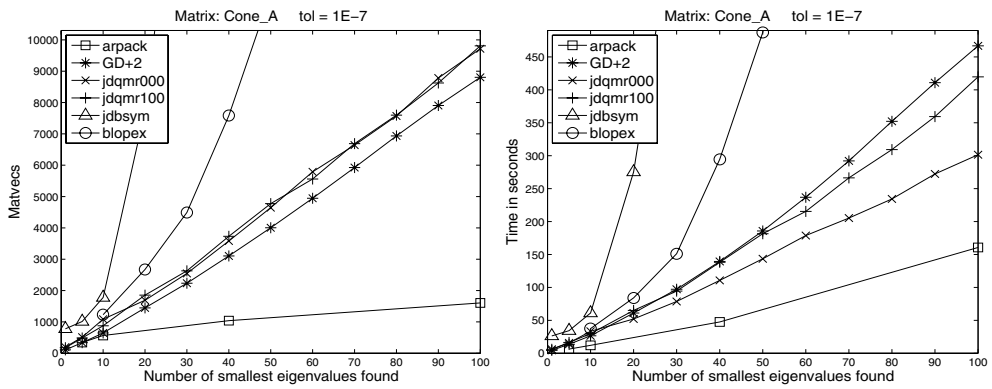


FIG. 5.5. *Matvecs (left) and time (right) of six methods with* $tol = 1e-7$ *for smallest eigenvalues.*

Figure 5.7 reports similar results in execution time for the finan512 and Lap7pt125K matrices. For the finan512, BLOPEX fails to converge to more than 10 eigenpairs, and JDBSYM is again far slower than the *tol* =1e-15 case. The JDQMR and GD+k methods are consistent both in robustness and their relative behavior.

To see whether BLOPEX would perform better on easier spectra, we run experiments with *tol*=1e-7, seeking the 500 largest eigenvalues of the matrices. In Figure 5.8 we give sample results from two matrices. For Plate33K_A0, the largest side of the spectrum is slightly easier to compute, but BLOPEX is still not competitive. For Cone_A, the largest side is much easier. In terms of matvecs (not shown in the figure) BLOPEX is competitive with JDQMR-000 up to 100 eigenvalues, then convergence deteriorates slowly, and the method fails for the 350th eigenvalue. In terms of time, the orthogonalization penalty is apparent on the BLOPEX method.

5.2. With preconditioning. For our preconditioning experiments, we use the ILUT preconditioner from the SPARSKIT library [47]. We have observed that although matvecs always decreased with ILUT, for some matrices execution time did not. Next, we report results from the three matrices on which unpreconditioned JDQMR/GD+k methods were slower than the ARPACK method. For these three cases, ILUT resulted in gains. Also, for the ILUT factorization to complete successfully, we had to shift the matrices slightly away from singularity.

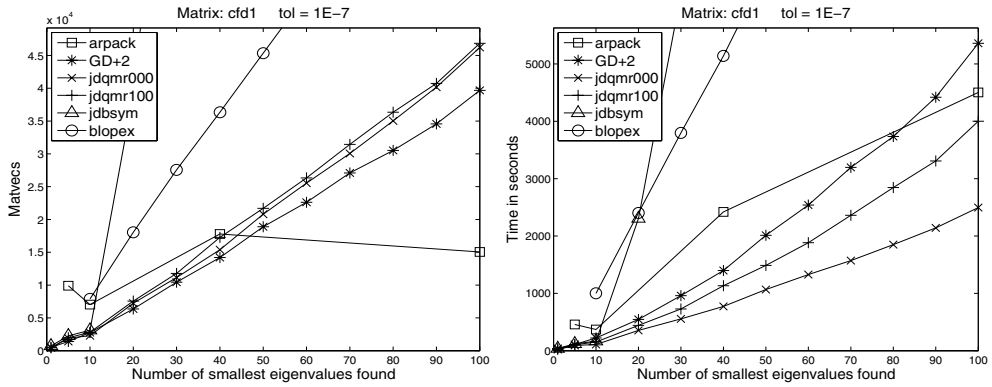


FIG. 5.6. *Matvecs (left) and time (right) of six methods with* $tol = 1e-7$ *for smallest eigenvalues.*

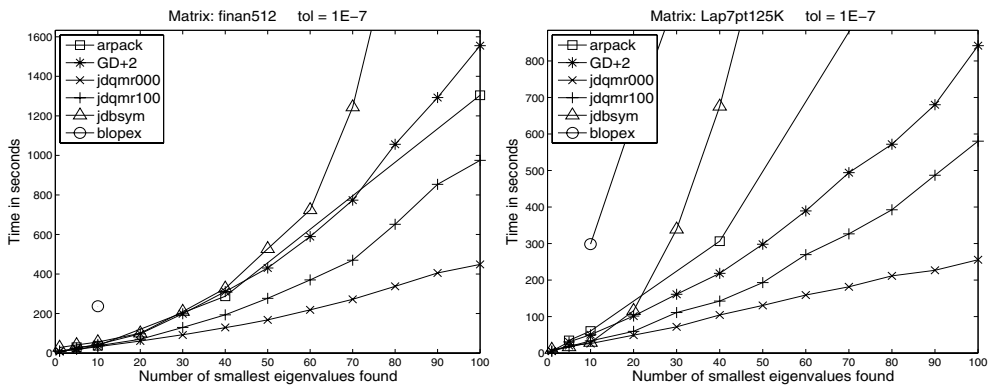


FIG. 5.7. *Execution times of six methods on two matrices with* $tol = 1e-7$ *for smallest eigenvalues.*

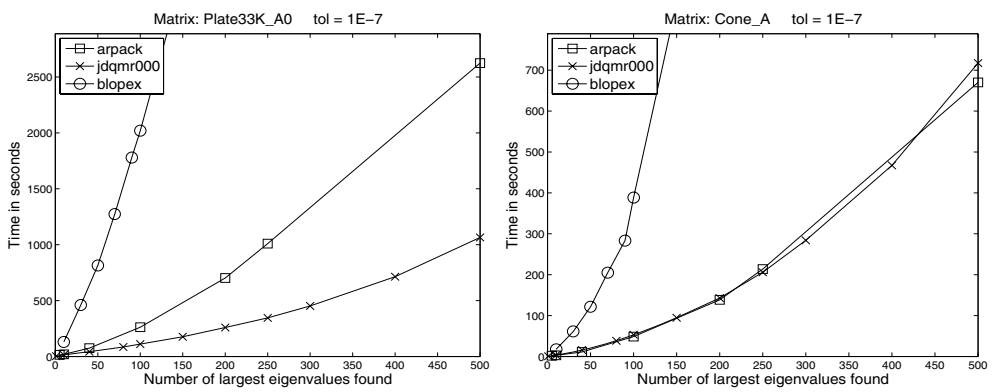


FIG. 5.8. *Times for three methods on two matrices with* $tol = 1e-7$ *for nev largest eigenvalues.*

In Figure 5.9, the ILUT(80,1e-4) of the shifted cfd1 matrix $A+0.01I$ is computed. All preconditioned methods improve and become much better than ARPACK. All JDQMR-100/111/011 variants converge identically, but the 111 takes more time for

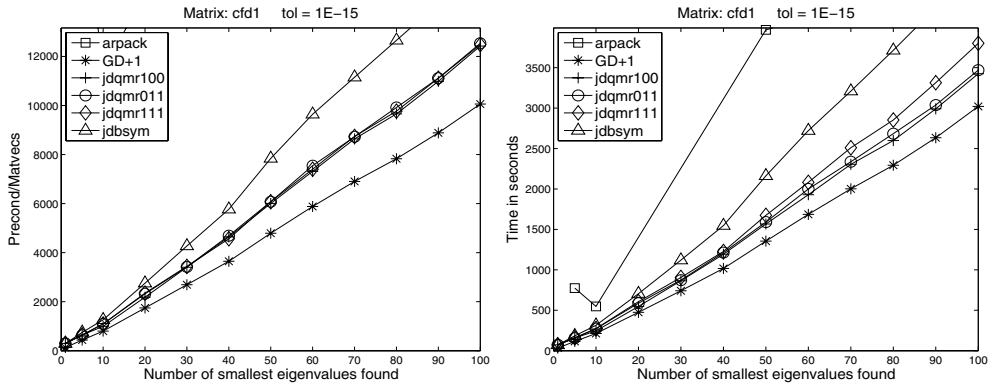


FIG. 5.9. *Matvecs (left) and time (right) with* $ILUT(80, 1e-4)$ *preconditioner. Smallest nev.*

large *nev*. JDBSYM also improves but less than JDQMR-100. Because of large fill-in, the ILUT preconditioner is expensive, and therefore the method with smallest matvecs wins, i.e., GD+1. The number of matvecs for ARPACK is large and out of scale.

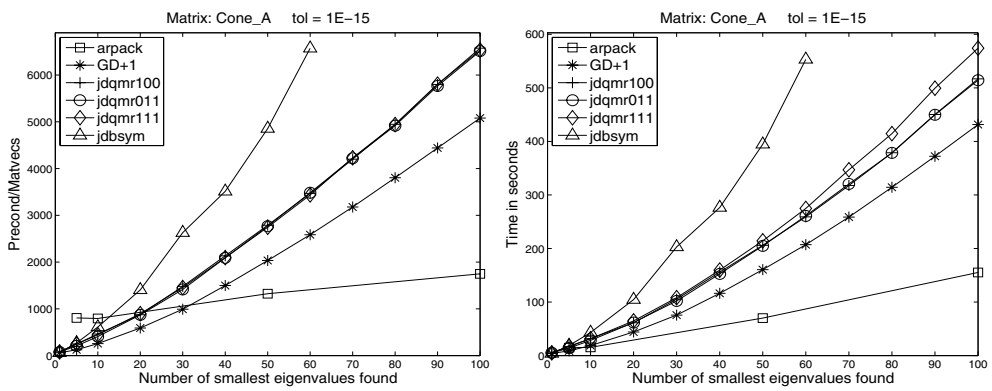


FIG. 5.10. *Matvecs (left) and time (right) with* $ILUT(40, 1e-6)$ *preconditioner. Smallest nev.*

In Figure 5.10, the ILUT(40,1e-6) of the shifted Cone_A matrix $A + 3e6I$ is computed. Although preconditioning reduces matvecs appreciably for JD methods, they still grow linearly with *nev*. As a consequence, the time graph for this experiment shows that ARPACK is faster than any other method beyond $nev = 10$. Observations about the relative behavior of the other methods hold in this example, too. Experiments with the Plate33K_A0 matrix have shown similar convergence behaviors and crossover points, so they are not reported.

Despite the use of preconditioning, BLOPEX was not able to reach convergence to *tol* =1e-15 in any of the previous cases. Therefore, we conclude this section with three experiments that include preconditioning but use a higher tolerance, 1e-7.

In Figure 5.11, with an ILUT(20,1e-6) of the unshifted Laplacian, neither JDBSYM nor BLOPEX were competitive. The graphs also include the unpreconditioned JDQMR-000, the time of which is identical to preconditioned JDQMR-100.

Figure 5.12 shows timing results from Cone_A with the same preconditioner as

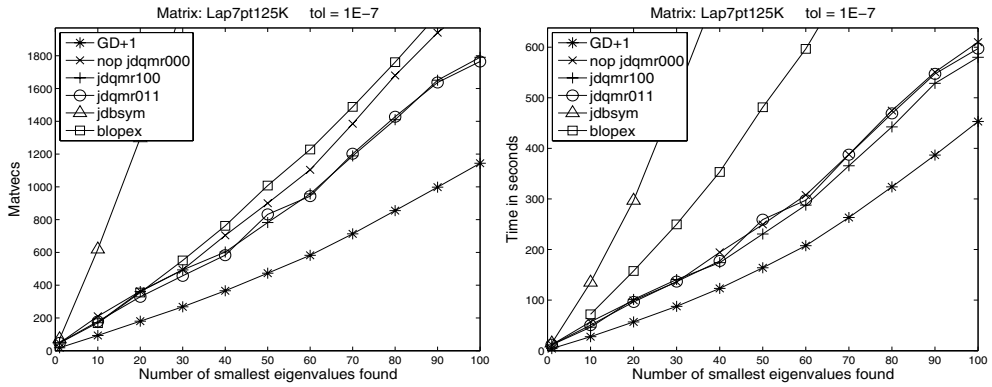


FIG. 5.11. *Matvecs (left) and time (right) with* $ILUT(20, 1e-6)$. *Smallest nev and tol* $= 1e-7$.

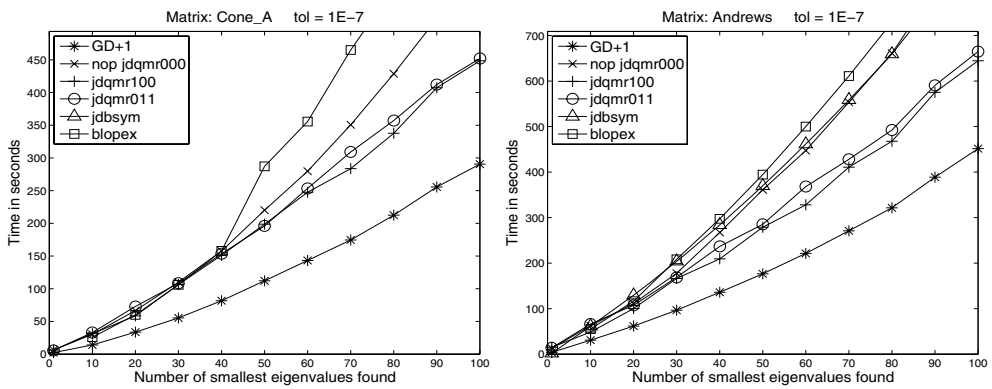


FIG. 5.12. *Times for two matrices, smallest nev, and tol* $= 1e-7$. $ILUT(40, 1e-6)$ *for both matrices.*

before and from the matrix Andrews shifted as $(A-1\text{e-}6I)$. For Cone_A, BLOPEX is marginally faster than JDQMR-100 for 10 eigenvalues, but its convergence deteriorates slowly beyond that and more rapidly after $nev = 40$. JDBSYM converged very slowly for this example. For the Andrews matrix, BLOPEX and JDBSYM are competitive with JDQMR-100 for 10 eigenvalues, but they grow appreciably slower with *nev*, following the curve of the unpreconditioned JDQMR-000. In all our preconditioning experiments, the heavier operators made GD+1 the fastest method.

6. Conclusions. In our research we seek nearly optimal methods for obtaining one or many eigenpairs of Hermitian or real symmetric matrices under limited memory. In an earlier companion paper we identified GD+k and JDQMR as two nearly optimal methods with complementary strengths for obtaining one eigenvalue. In this paper, we offer arguments for why the same near-optimality is not achievable when looking for many eigenpairs.

We have also offered ways to alleviate the quadratic scaling bottleneck stemming from the orthogonalization against converged eigenvectors. Specifically, without preconditioning, our JDQMR-000 method performs no projections during the inner JD iteration and thus achieves nearly linear scaling with *nev* up to the point where *nev* becomes much larger than the number of inner iterations performed, shifting the bot-

tleneck to the outer step. JDQMR-000 outperforms all other JD/GD variants, and for sparse enough matrices even outperforms ARPACK up to thousands of eigenvalues. With preconditioning, our analysis suggests that the oblique projection in the JD correction equation is unnecessary for converged eigenvectors. The resulting method, JDQMR-100, is faster and requires half the storage of other JD variants.

Based on a detailed complexity analysis and extensive experimental observations, we have developed a comprehensive set of models that describes the relative performance between JD variants, GD+k, and ARPACK. The asymptotic analysis has provided the following valuable insight: $O(1)$ $O(nev)$ In practice, $O(1)$ basis sizes with or without preconditioning can be used extremely effectively up to hundreds or thousands of eigenvalues (as our JDQMR-000 shows). Our models also provide performance crossover points between methods that can be used for dynamic, and possibly autonomic, method selection within a multimethod software, such as PRIMME.

Finally, we have provided a large set of experiments on a variety of problems and conditions and have compared these against state-of-the-art methods and software, such as JDBSYM and BLOPEX. Invariably, our GD+k and JDQMR methods are far more efficient and robust and seem to be nearly optimal within the class of $O(1)$ basis size.

Acknowledgments. The comments from the referees have contributed to a clearer and more concise presentation. We thank M. Hochstenbach for pointing out some connections with a method in [4].

REFERENCES

[1] P.-A. Absil, C. G. Baker, and K. A. Gallivan, *A truncated-CG style method for symmetric generalized eigenvalue problems*, J. Comput. Appl. Math., 189 (2006), pp. 274–285.

[2] P.-A. Absil, R. Mahony, R. Sepulchre, and P. Van Dooren, *A Grassmann-Rayleigh quotient iteration for computing invariant subspaces*, SIAM Rev., 44 (2002), pp. 57–73.

[3] M. F. Adams, *Evaluation of three unstructured multigrid methods on 3D finite element problems in solid mechanics*, Internat. J. Numer. Methods Engrg., 55 (2002), pp. 519–534.

[4] P. Arbenz, R. Geus, and S. Adam, *Solving Maxwell eigenvalue problems for accelerating cavities*, Phys. Rev. ST Accel. Beams, 4 (2001), 022001.

[5] P. Arbenz, U. L. Hetmaniuk, R. B. Lehoucq, and R. S. Tuminaro, *A comparison of eigensolvers for large-scale 3D modal analysis using AMG-preconditioned iterative methods*, Internat. J. Numer. Methods Engrg., 64 (2005), pp. 204–236.

[6] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, eds., *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, Software Environ. Tools 11, SIAM, Philadelphia, 2000.

[7] T. F. Chan and W. L. Wan, *Analysis of projection methods for solving linear systems with multiple right-hand sides*, SIAM J. Sci. Comput., 18 (1997), pp. 1698–1721.

[8] J. Cullum and W. E. Donath, *A block Lanczos algorithm for computing the q algebraically largest eigenvalues and a corresponding eigenspace of large, sparse, symmetric matrices*, in Proceedings of the 1974 IEEE Conference on Decision and Control, IEEE Control Systems Society, Piscataway, NJ, 1974, pp. 505–509.

[9] J. Cullum and R. A. Willoughby, *Lánczos Algorithms for Large Symmetric Eigenvalue Computations, Vol.* 2, *Programs*, Progress in Scientific Computing 4, Birkhäuser Boston, Boston, 1985.

[10] E. R. Davidson, *The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices*, J. Comput. Phys., 17 (1975), pp. 87–94.

[11] T. Davis, *University of Florida Sparse Matrix Collection*, Technical report, University of Florida, Gainesville, FL. NA Digest, vol. 92, no. 42, October 16, 1994, NA Digest, vol. 96, no. 28, July 23, 1996, and NA Digest, vol. 97, no. 23, June 7, 1997.

[12] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *Inexact Newton methods*, SIAM J. Numer. Anal., 19 (1982), pp. 400–408.

[13] E. G. D'yakonov, *Iteration methods in eigenvalue problems*, Math. Notes, 34 (1983), pp. 945–953.
[14] A. Edelman, T. A. Arias, and S. T. Smith, *The geometry of algorithms with orthogonality constraints*, SIAM J. Matrix Anal. Appl., 20 (1998), pp. 303–353.
[15] D. R. Fokkema, G. L. G. Sleijpen, and H. A. van der Vorst, *Jacobi–Davidson style QR and QZ algorithms for the reduction of matrix pencils*, SIAM J. Sci. Comput., 20 (1998), pp. 94–125.
[16] R. W. Freund and N. M. Nachtigal, *A New Krylov-Subspace Method for Symmetric Indefinite Linear Systems*, Technical report, AT&T Bell Laboratories, Murray Hill, NJ, 1994.
[17] R. Geus, *JDBSYM*, http://people.web.psi.ch/geus/software.html.
[18] P. H. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, New York, 1986.
[19] G. H. Golub and R. Underwood, *The block Lanczos method for computing eigenvalues*, in Mathematical Software III, J. R. Rice, ed., Academic Press, New York, 1977, pp. 361–377.
[20] G. H. Golub and Q. Ye, *Inexact inverse iteration for generalized eigenvalue problems*, BIT, 40 (2000), pp. 671–684.
[21] R. G. Grimes, J. G. Lewis, and H. D. Simon, *A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 228–272.
[22] M. H. Gutknecht, *Block Krylov Space Solvers: A Survey*, http://www.sam.math.ethz.ch/~mhg/talks/bkss.pdf.
[23] M. H. Gutknecht, *Block Krylov space methods for linear systems with multiple right-hand sides: An introduction*, in Modern Mathematical Models, Methods and Algorithms for Real World Systems, Anamaya Publishers, New Delhi, India, 2006.
[24] M. E. Hochstenbach and Y. Notay, *Controlling Inner Iterations in the Jacobi–Davidson Method*, CASA report 07-01, Department of Mathematics, TU Eindhoven, Eindhoven, The Netherlands, 2007. Submitted.
[25] Z. Jia, *A refined iterative algorithm based on the block Arnoldi process for large unsymmetric eigenproblems*, Linear Algebra Appl., 270 (1998), pp. 171–189.
[26] M. E. Kilmer and E. de Sturler, *Recycling subspace information for diffuse optical tomography*, SIAM J. Sci. Comput., 27 (2006), pp. 2140–2166.
[27] A. V. Knyazev, *BLOPEX*, http://www-math.cudenver.edu/~aknyazev/software/BLOPEX.
[28] A. V. Knyazev, *Convergence rate estimates for iteratime methods for symmetric eigenvalue problems and its implementation in a subspace*, in Eigenwertaufgaben in Natur- und Ingenieurwissenschaften und ihre numerische Behandlung (Oberwolfach, 1990), Internat. Ser. Numer. Math. 96, Birkhäuser, Basel, 1991, pp. 143–154.
[29] A. V. Knyazev, *Preconditioned eigensolvers—an oxymoron?*, Electron. Trans. Numer. Anal., 7 (1998), pp. 104–123.
[30] A. V. Knyazev, *Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method*, SIAM J. Sci. Comput., 23 (2001), pp. 517–541.
[31] Y.-L. Lai, K.-Y. Lin, and W.-W. Lin, *An inexact inverse iteration for large sparse eigenvalue problems*, Numer. Linear Algebra Appl., 4 (1997), pp. 425–437.
[32] R. B. Lehoucq, *Implicitly restarted Arnoldi methods and subspace iteration*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 551–562.
[33] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, Software Environ. Tools 6, SIAM, Philadelphia, 1998.
[34] J. R. McCombs and A. Stathopoulos, *PRIMME: PReconditioned Iterative Multimethod Eigensolver*, http://www.cs.wm.edu/~andreas/software/.
[35] R. B. Morgan, *Computing interior eigenvalues of large matrices*, Linear Algebra Appl., 154 (1991), pp. 289–309.
[36] R. B. Morgan, *On restarting the Arnoldi method for large nonsymmetric eigenvalue problems*, Math. Comput., 65 (1996), pp. 1213–1230.
[37] R. B. Morgan and D. S. Scott, *Generalizations of Davidson's method for computing eigenvalues of sparse symmetric matrices*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 817–825.
[38] R. B. Morgan and D. S. Scott, *Preconditioning the Lanczos algorithm for sparse symmetric eigenvalue problems*, SIAM J. Sci. Comput., 14 (1993), pp. 585–593.
[39] C. W. Murray, S. C. Racine, and E. R. Davidson, *Improved algorithms for the lowest eigenvalues and associated eigenvectors of large matrices*, J. Comput. Phys., 103 (1992), pp. 382–389.
[40] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer-Verlag, New York, 1999.

[41] Y. Notay, *Combination of Jacobi–Davidson and conjugate gradients for the partial symmetric eigenproblem*, Numer. Linear Algebra Appl., 9 (2002), pp. 21–44.
[42] Y. Notay, *Is Jacobi–Davidson faster than Davidson?*, SIAM J. Matrix Anal. Appl., 26 (2005), pp. 522–543.
[43] J. Olsen, P. Jørgensen, and J. Simons, *Passing the one-billion limit in full configuration-interaction (FCI) calculations*, Chem. Phys. Lett., 169 (1990), pp. 463–472.
[44] C. C. Paige, B. N. Parlett, and Van der Vorst, *Approximate solutions and eigenvalue bounds from Krylov spaces*, Numer. Linear Algebra Appl., 2 (1995), pp. 115–133.
[45] B. N. Parlett, *The Symmetric Eigenvalue Problem*, Classics Appl. Math. 20, SIAM, Philadelphia, 1997.
[46] A. Ruhe and T. Wiberg, *The method of conjugate gradients used in inverse iteration*, BIT, 12 (1972), pp. 543–554.
[47] Y. Saad, *SPARSKIT: A Basic Toolkit for Sparse Matrix Computations*, Technical report 90-20, Research Institute for Advanced Computer Science, NASA Ames Research Center, Moffet Field, CA, 1990. Software currently available at ftp://ftp.cs.umn.edu/dept/sparse/.
[48] A. Sameh and Z. Tong, *The trace minimization method for the symmetric generalized eigenvalue problem*, J. Comput. Appl. Math., 123 (2000), pp. 155–175.
[49] V. Simoncini and L. Eldén, *Inexact Rayleigh quotient-type methods for eigenvalue computations*, BIT, 42 (2002), pp. 159–182.
[50] V. Simoncini and E. Gallopoulos, *An iterative method for nonsymmetric systems with multiple right-hand sides*, SIAM J. Sci. Comput., 16 (1995), pp. 917–933.
[51] G. L. G. Sleijpen and H. A. van der Vorst, *A Jacobi–Davidson iteration method for linear eigenvalue problems*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 401–425.
[52] P. Smit and M. H. C. Paardekooper, *The effects of inexact solvers in algorithms for symmetric eigenvalue problems*, Linear Algebra Appl., 287 (1999), pp. 337–357. Special issue celebrating the 60th birthday of Ludwig Elsner.
[53] D. C. Sorensen, *Implicit application of polynomial filters in a K-step Arnoldi method*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 357–385.
[54] A. Stathopoulos and Y. Saad, *Restarting techniques for (Jacobi-)Davidson symmetric eigenvalue methods*, Electr. Trans. Numer. Anal., 7 (1998), pp. 163–181.
[55] A. Stathopoulos, Y. Saad, and C. F. Fischer, *Robust preconditioning of large, sparse, symmetric eigenvalue problems*, J. Comput. Appl. Math., 64 (1995), pp. 197–215.
[56] A. Stathopoulos, Y. Saad, and K. Wu, *Dynamic thick restarting of the Davidson, and the implicitly restarted Arnoldi methods*, SIAM J. Sci. Comput., 19 (1998), pp. 227–245.
[57] A. Stathopoulos, *Nearly optimal preconditioned methods for Hermitian eigenproblems under limited memory. Part* I: *Seeking one eigenvalue*, SIAM J. Sci. Comput., 29 (2007), pp. 481–514.
[58] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, England, 1965.
[59] K. Wu and H. Simon, *Thick-restart Lanczos method for large symmetric eigenvalue problems*, SIAM J. Matrix Anal. Appl., 22 (2000), pp. 602–616.
[60] Y. Zhou, Y. Saad, M. L. Tiago, and J. R. Chelikowsky, *Self-consistent-field calculations using Chebyshev-filtered subspace iteration*, J. Comput. Phys., 219 (2006), pp. 172–184.
[61] Y. Zhou, *Studies on Jacobi-Davidson, Rayleigh quotient iteration, inverse iteration generalized Davidson and Newton updates*, Numer. Linear Algebra Appl., 13 (2006), pp. 621–642.

SIAM J. SCI. COMPUT.
Vol. 29, No. 5, pp. 2189–2209
© 2007 Society for Industrial and Applied Mathematics

ALGEBRAIC MULTIGRID PRECONDITIONING OF HIGH-ORDER SPECTRAL ELEMENTS FOR ELLIPTIC PROBLEMS ON A SIMPLICIAL MESH*

LUKE OLSON†

Abstract. Algebraic multigrid is investigated as a solver for linear systems that arise from high-order spectral element discretizations. An algorithm is introduced that utilizes the efficiency of low-order finite elements to precondition the high-order method in a multilevel setting. In particular, the efficacy of this approach is highlighted on simplexes in two and three dimensions with nodal spectral elements up to order $n = 11$. Additionally, a hybrid preconditioner is also developed for use with discontinuous spectral element methods. The latter approach is verified for the discontinuous Galerkin method on elliptic problems.

Key words. spectral element, discontinuous Galerkin, algebraic multigrid

AMS subject classifications. 65N15, 65N30, 65N55

DOI. 10.1137/060663465

1. Introduction.

In recent years spectral element methods have increased in popularity [18, 9]. Their success is often attributed to the ability to approximate solutions to partial differential equations (PDEs) efficiently by requiring fewer degrees-of-freedom than low-order approaches (e.g., bilinear finite elements). Moreover, high-order methods yield more accurate solutions in problems with long time integration [14] and are also beneficial in multiscale simulation [17]. Combined with a compatible discretization method such as the discontinuous Galerkin (DG) method, the method is used successfully in a variety of applications [20, 15].

The overall efficiency of high-order methods, however, remains an open question, particularly when a solution to a system of equations is needed. While fewer degrees-of-freedom are required to achieve a certain accuracy in the solution compared with low-order methods, the degrees-of-freedom are more intimately coupled, and the resulting matrix is more dense. The reduced locality impacts both the amount of storage and the complexity of the solver.

The use of high-order elements also changes many matrix properties, impacting the success of an iterative method. The condition of the matrix is high compared to that of low-order methods resulting in slower convergence (e.g., $\mathcal{O}(N^4/h^{-2})$ for two dimensional (2D) elliptic problems [26]), and the systems are not M-matrices, a valuable theoretical property for algebraic multigrid.

The multigrid method has found success for a variety of applications [32], particularly in the scope of elliptic problems. Algebraic-based multigrid methods [30] in particular have emerged as robust solvers, relying only on the graph of the matrix to achieve a scalable algorithm. The primary goal of this paper is to investigate the use of algebraic multigrid (AMG) with low-order finite elements as a preconditioner of high-order spectral element discretizations of elliptic problems.

*Received by the editors June 21, 2006; accepted for publication (in revised form) June 26, 2007; published electronically September 28, 2007. This work was partially supported by the NSF under award DMS-0612448.

http://www.siam.org/journals/sisc/29-5/66346.html

†Department of Computer Science, 201 North Goodwin Avenue, University of Illinois at Urbana-Champaign, Urbana, IL 61801 (lukeo@uiuc.edu).

The contributions of this paper are twofold. First, an AMG algorithm for solving systems arising from nodal spectral elements on simplexes in two and three dimensions is highlighted. The solution technique utilizes a low-order finite element preconditioner based on a local Delaunay triangulation of the interpolatory nodes. Scalable performance of the multigrid preconditioner is verified. Second, a new hybrid preconditioner is identified for use in a nonconforming setting. In particular, we extend the successful AMG preconditioner in the continuous setting to an effective solution method in the discontinuous framework and highlight the necessary components for achieving a scalable algorithm.

The remainder of the paper is organized as follows. Sections 1.1 and 1.2 briefly detail the spectral element method and the common AMG framework used throughout the paper. In section 1.3, previous results are summarized which provide a basis for the algorithmic details and performance outlined in section 2. The preconditioner for the discontinuous method is introduced in section 3, and we summarize the scope of this research in section 4.

1.1. Discretization. The focus of this paper is on spectral elements of nodal type: basis functions defined by Lagrange interpolating polynomials over a set of nodes. While this framework has many benefits, including direct representation of the solution as coefficients and a decoupling of degrees-of-freedom in the case of inter-element connectivity, it does not possess a direct hierarchical representation. Spectral methods of modal type often do form a hierarchical set and are more directly amenable to multilevel solution techniques. The so-called spectral multigrid method developed computationally and theoretically in [22, 27, 23] has proven to be efficient, yielding bounded convergence factors in one dimension and only marginal growth in two dimensions. The results in [12] highlight the versatility of the p-multigrid approach. As discussed in section 1.3, we follow an approach that also utilizes a low-order basis, but we do so through well-established theory on the spectral equivalence of the low-order finite element and high-order spectral element operators.

Let $\Omega \in \mathbb{R}^d$ denote the domain for $d = 2, 3$. Initially we focus on the elliptic PDE

$$-\nabla \cdot \nabla u(\mathbf{x}) = f(\mathbf{x}) \quad \text{in } \Omega, \tag{1.1a}$$

$$u(\mathbf{x}) = g(\mathbf{x}) \quad \text{on } \partial\Omega, \tag{1.1b}$$

where AMG has been successful for low-order finite element methods. The associated weak problem for the Galerkin method becomes the following: Find $u_{h,n} \in V$ so that

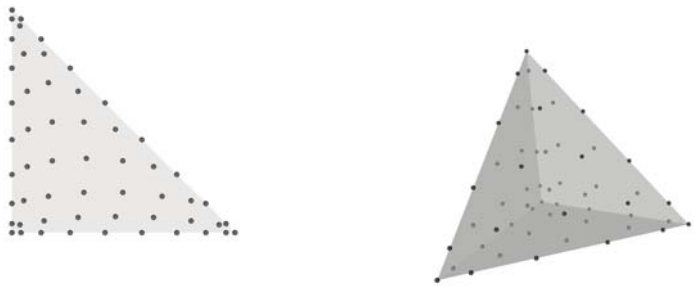
$$a(u_{h,n}, v_{h,n}) = \ell(v_{h,n}) \quad \forall\, v_{h,n} \in V, \tag{1.2}$$

where

$$a(u, v) = \int_\Omega \nabla u \cdot \nabla v \, d\mathbf{x}, \quad \ell(v) = \int_\Omega f(\mathbf{x}) v \, d\mathbf{x}, \tag{1.3}$$

and V denotes the spectral element basis. The system of equations $Au = b$ that results from (1.3) is the focus of the first portion of this paper.

As explained above, the basis functions V are nodal. As a reference element, we use the d-simplex $\hat{\kappa}_d = \{(\xi_1, \ldots, \xi_d) \in \mathbb{R}^d : \xi_i \in [-1, 1], \sum_{i=1}^d \xi_i < 0\}$ and denote the complete space of n-degree polynomials on $\hat{\kappa}_d$ with minimal dimension as P_n^d. The dimension of this space [13] is $\dim P_n^d \equiv N_n^d = \binom{d+n}{d}$. Over $\hat{\kappa}$ we define $\phi_{n,i}(\mathbf{x}) \in \hat{V}$ as a Lagrange interpolating polynomial with the property $\phi_{n,i}(\mathbf{x}_j) = \delta_{ij}$, where δ_{ij} is the Kronecker delta function, $\mathbf{x}_i$ are nodal locations, and i ranges from 1 to $N = N_n^d$.



(a) $\hat{\kappa}_2$ (b) $\hat{\kappa}_3$

FIG. 1.1. *Nodal sets with* $n = 10$ (1.1(a)) *and* $n = 6$ (1.1(b)) *on the reference elements.*

In one dimension, the Gauss–Legendre–Lobatto (GLL) nodes offer a highly stable and complete set interpolation points for a Lagrange basis. For higher dimensions, a natural set of nodal locations is not available. Much progress has been made in defining stable and accurate nodal sets [31, 33]; we employ nodal locations based on electrostatics [13] due to their availability and near-optimal numerical properties. The results in this paper were largely invariant with respect to small variations in the nodal locations, namely using the Fekete nodal set [31] or Warp&Blend nodes [33]. In two dimensions, the electrostatic nodal set we use enforces GLL nodes on each edge of $\hat{\kappa}_2$, letting the remaining $N_n^2 - 3N_n^1 - 3$ interior nodes be determined by symmetry and electrostatic distribution. Similarly, in three dimensions, we impose the 2D distribution on each face of the tetrahedron and allow the remaining nodal locations in the interior to be determined also from electrostatics. The values are determined in preprocessing or from a library and only for the reference element $\hat{\kappa}_d$. An example is illustrated in Figure 1.1. As with GLL nodes in one dimension, the electrostatic nodes (equal to GLL nodes in one dimension) are clustered in the corners.

1.2. Algebraic multigrid. We consider the solution of high-order $A_{sp}u = b$ using algebraic-based multigrid methods together with standard Krylov acceleration methods such as the conjugate gradient (CG). The AMG algorithm we adhere to is the classical Ruge–Stüben (RS) [30] algorithm, with a few modifications, which are mentioned throughout this paper. The MG cycling we use is the familiar V-cycle summarized for two levels by the following:

1.	Presmooth ν_1 times on the fine grid Ω^0	$u^0 \leftarrow S(u^0; \nu_1)$.
2.	Restrict residual $r^0 = b^0 - Au^0$ to the coarse grid Ω^1	$r^1 = Rr^0$.
3.	Solve the coarse grid problem $A^1e^1 = r^1$ on Ω^1	$e^1 \approx (A^1)^{-1}r^1$.
4.	Interpolate the coarse-grid error to Ω^0	$e^0 = Pe^1$.
5.	Correct the previous approximation on Ω^0	$u^0 \leftarrow u^0 + e^0$.
6.	Postsmooth ν_2 on the fine grid Ω^0	$u^0 \leftarrow S(u^0; \nu_1)$.

The setup phase is more tedious and impacts on the application to high-order discretizations. Unknown u_i is connected to u_j if $a_{ij} \neq 0$. The size of a_{ij} indicates the impact of this connection, relating how much influence the error at j has on the error at i in the relaxation process. We say that u_i is strongly connected [30] to u_j if

$$-a_{ij} \geq \theta \max_{k \neq i} -a_{ik}, \tag{1.4}$$

where a threshold of $\theta = 0.25$ is used in our computations, a typical selection for

elliptic problems. We refer to [30] for details on the selection of coarse grids.

For a given C/F splitting of Ω^0, the next step in the setup phase is to define interpolation. The θ parameter in the definition of the strength-of-connection controls the density of coarse grids. Likewise, in the definition of interpolation, we need to ensure that corresponding coarse-grid operators A^k are sufficiently sparse. In section 2, long-range connections are not used in interpolation: only direct connections are used in the definition. However, it does not yield effective interpolation for all problems, as highlighted in section 3.

Although AMG is well suited as a solution technique, it is also used for many problems as an effective preconditioner. This process can be viewed as acceleration since Krylov steps can often reduce error components that the multigrid cycle is unable to annihilate. Furthermore, each step of the Krylov method is relatively inexpensive, requiring on the order of one matrix-vector multiply.

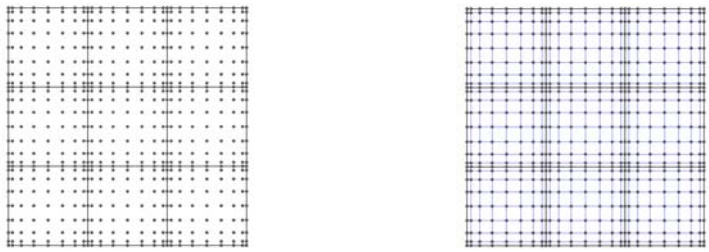
ALGORITHM 1. AMG-PCG ALGORITHM.

1 $r_{new} = r_0 = b - Ax_0$
2 $MG_{\text{setup}}(\hat{A})$
3 $p = z = MG_{\text{cycle}}(\hat{A}, r_0)$
4 while $\|r_{new}\|/\|r_0\| > \delta$
5 $\quad \alpha := (r, z)/(Ap, p)$
6 $\quad x_{new} := x + \alpha p$
7 $\quad r_{new} := r - \alpha Ap$
8 $\quad z_{new} := MG_{\text{cycle}}(\hat{A}, r_{new})$
9 $\quad \beta := (r_{new}, z_{new})/(r, z)$
10 $\quad p_{new} := z_{new} + \beta p$
11 $\quad$ update r, z, p

The construction of $\hat{A}$ in Algorithm 1 is central to the algorithm introduced in sections 2 and 3. Specifically, steps 2, 3, and 8 require the formulation of $\hat{A}$ along with the definition of the associated multigrid process. The reference to MG_{cycle} in Algorithm 1 consists of one V-cycle for the computations reported in this paper, but the use of multiple cycles or W-cycles may be easily incorporated.

1.3. AMG vis-à-vis high order. The strategy proposed in this paper relies on the ability of low-order finite element operators to precondition the high-order spectral element discretization. Let A_s be the matrix from a spectral element discretization, and let A_f be the resulting matrix from a low-order method defined over a tessellation of the nodal set in the spectral element discretization. The concept is well developed with early contributions in [24]. In [24], it is shown that using a finite-difference operator to precondition a matrix from a *spectral* method is effective in one dimension. The use of finite elements is detailed in [8] for one dimension, and the effectiveness for two dimensions is detailed in [10]. Norm dependent bounds on $A_f^{-1}A_s$ are further developed in [25], the impact of advection and reaction terms are studied in [25, 11], and the 2D and 3D Helmholtz problem is studied in [4]. The condition number in higher dimensions, however, does not in general remain bounded, and the effectiveness of preconditioning is decreased when using simplexes [10]. Still, the condition of the system is improved and is suitable for Krylov iterative methods. An extensive study focusing on Schwarz methods in this context can be found in [21] and has motivated much the effort here.

As an example of the low-order preconditioner, consider the 3×3 spectral element mesh of order $n = 10$ displayed in Figure 1.2(a). The nodal set used in each element is a tensor set of the one dimensional (1D) GLL nodes. An effective preconditioner



(a) A_s (b) A_f

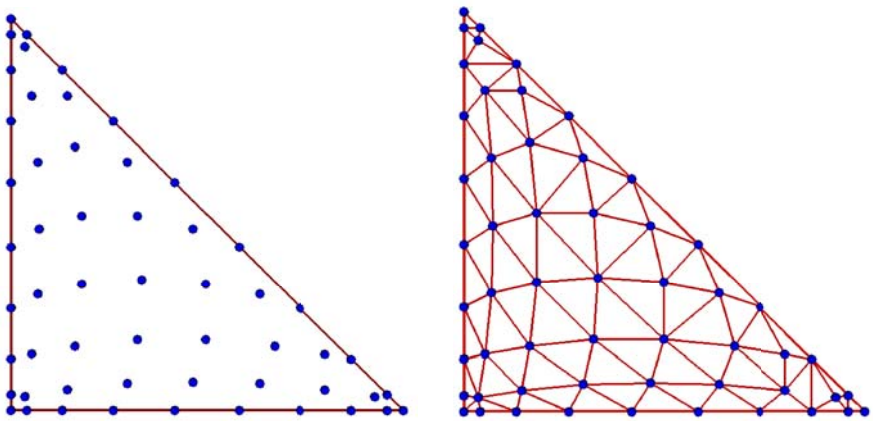
FIG. 1.2. *Meshes corresponding to the spectral element operator A_s and to the finite element preconditioner A_f, respectively.*

utilizes the nodal locations, forming a bilinear finite element mesh resulting in A_f. The effect is illustrated in Figure 1.2(b). The performance using low-order finite element preconditioning with AMG is studied for quadrilaterals in [16]. One AMG V-cycle is used to approximate the action of A_f^{-1} to precondition the spectral element matrix.

Two aspects are highlighted in [16] which motivate the algorithms developed in this paper. First, for both Poisson's equations and Stoke's flow (H^1-elliptic least-squares form), the use of a bilinear finite element preconditioner with AMG as illustrated in Figure 1.2 scales with spectral element order. The preconditioner is known to be effective; however, the inexact and inexpensive preconditioning of AMG is sufficient. More precisely, CG iterations exhibit a linear increase with spectral order n, while AMG preconditioned CG (AMG-PCG) iterations remain invariant, needing only 12–14 iterations in the cases tested. Furthermore, the average convergence factor of the AMG-PCG iterations also scaled with n, remaining constant at around 0.34. Also notable regarding this simulation is that the AMG complexity (detailed below in section 2) remains bounded, although somewhat high.

The subtle nature of complexities is the basis of the second observation in [16]. While AMG-PCG applied directly to the spectral operator A (i.e., $\hat{A} = A$) yields attractive convergence factors of around 0.1 for all spectral orders, the cost of each AMG cycle heavily depends on n. The overall cost associated in preconditioning with $\hat{A} = A$ scales as $\mathcal{O}(n^4)$, whereas preconditioning with $\hat{A} = A_f$ scales only as $\mathcal{O}(n^2)$. The underlying statement of [16] is that AMG-PCG iterations can be performed directly on the spectral element operator when $n < 4$ and that $\hat{A} = A_f$ preconditioning is beneficial if $n \geq 4$. The later case is even more remarkable since explicit construction of the spectral element operator A is not needed, saving memory and reducing the $\mathcal{O}(n^4)$ construction cost.

The algorithms proposed in this paper follow a similar process; however, we extend the results, implementation, and analysis in a number of directions. The methodology is extended to simplexes and for nontensor nodal sets—based on electrostatics introduced in section 1.1—requiring an unstructured local tessellation utilizing a reference Delaunay triangulation. Additionally, the quality of the meshes is considered and simulations are performed both in two and in three dimensions. A new complexity measure is also introduced to account for the use of a low-complexity, low-order preconditioner in the context of a high-order solver. The new measure highlights the efficacy of low-order finite element AMG preconditioning. Discontinuous spectral elements are also considered, and a novel multilevel preconditioner is introduced.



(a) $n = 10$ (b) $n = 1$

FIG. 2.1. *Local Delaunay tessellation of the reference element for* $n = 10$.

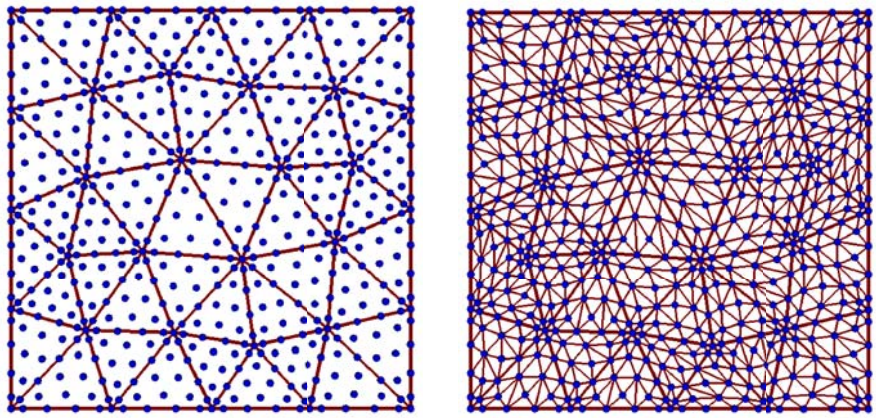
2. Continuous methods. In this section we investigate the use of Algorithm 1 for high-order spectral elements on triangular and tetrahedral meshes. A continuous Galerkin method is used (1.3) to discretize the model problem (1.1).

Central to the algorithm is the ability to use a low-order finite element operator as a preconditioner: $\hat{A} = A_f$. For triangles and for tetrahedrons, the nontensor nodal locations (see Figure 1.1) require additional attention, whereas in the case of quadrilaterals, a straightforward structured mesh is used to locally tessellate the nodal set (see Figure 1.2). Here, we use a Delaunay triangulation of the nodal set on the reference element, mapping the local tessellation to each element in the spectral element mesh to construct the global finite element mesh. The Delaunay triangulation is used since it maximizes the minimum angle of the local elements as depicted in Figure 2.1. Moreover, since the Delaunay operation is executed in preprocessing over just the reference element, it does not add to the complexity of the algorithm. The nonuniqueness of tetrahedral tessellations did not result in problematic elements for the nodal sets tested, although postprocessing could easily be included to ensure the exclusion of so-called sliver elements. Since the quality of the tessellation is enforced only on the reference element, the area ratios and minimum angles are not preserved over the global grid. Yet, given a reasonable initial spectral element mesh, we can assume for d-simplexes that the mesh qualities are preserved since the map Jacobian is constant. On the other hand, as with quadrilaterals, the aspect ratio of the mesh can become skewed particularly at high order due to the grouping of nodes in corners. An example is illustrated in Figure 2.2.

2.1. AMG measures. A classical Ruge–Stüben implementation of AMG is used (see section 1.2). We assume a C/F splitting and define interpolation by considering $Ae \approx 0$ for smooth error. That is,

$$a_{ii}e_i \approx -\sum_{j\in C_i^s} a_{ij}e_j - \sum_{j\in F_i^s} a_{ij}e_j - \sum_{j\in N_i^w} a_{ij}e_j, \tag{2.1}$$

where C_i^s are C-points that strongly influence F-point i, F_i^s are F-points that strongly influence i, and N_i^w are points that weakly influence i. First, weak connections N_i^w



(a) $n = 6$ (b) $n = 1$

FIG. 2.2. *Meshes corresponding to the spectral element operator A and to the finite element preconditioner A_f, respectively.*

are distributed to the diagonal as in the original RS description [30] (cf. [3]):

$$\left(a_{ii} + \sum_{j \in N_i^w} a_{ij}\right) e_i \approx - \sum_{j \in C_i^s} a_{ij} e_j - \sum_{j \in F_i^s} a_{ij} e_j. \tag{2.2}$$

Intuitively this is compatible with the high-order discretization as weak connections may be large in magnitude but of the same sign as the diagonal, thus preserving the dominance of the F-point i in the interpolation process. Numerically the performance is only slightly improved.

The remaining term in (2.2) is perhaps more important than the weak connections: F-points strongly influencing i. We also adhere to the original algorithm and distribute direct strong connections along all nodes in C_i^s to ensure that interpolation is reasonable in the event that a strength of an edge has been misidentified for a positive off-diagonal entry.

The remaining parameters in the AMG algorithm are standard and largely invariant with spectral order for the problems tested. We run V(1,1)-cycles in all tests performed, using pointwise Gauss–Seidel over C-points followed by F-points in prerelaxation and the reverse ordering for postrelaxation in order to preserve symmetry of the AMG preconditioner, a necessary attribute for PCG.

Measuring the performance of Algorithm 1 is not straightforward. Typical tools in analyzing the performance of multigrid include the convergence factor, ρ, which is computed as the average reduction in the residual in each iteration. In each test, we drive Algorithm 1 until the relative residual is reduced by a factor of 10^{-6}. One component of a scalable algorithm is to maintain a constant convergence factor (as low as 0.05 for some problems). The other key element is the complexity of each step in the process. For AMG we consider the *operator complexity* for a matrix A^0:

$$\chi_{\text{op}} = \frac{\sum_{\ell=0}^{\ell_{max}} \aleph(A^\ell)}{\aleph(A^0)}, \tag{2.3}$$

where $\aleph(A)$ is the number of nonzeros in matrix A and ℓ_{max} is the number of levels to be visited in the AMG cycle. The operator complexity is more instructive than

the *grid complexity*, which similarly measures the number of variables on all levels compared with the finest. Additionally, we define the *cycle complexity* as

$$\chi_{\text{cyc}} = \frac{\sum_{\ell=0}^{\ell_{max}} \nu_\ell \cdot \aleph(A^\ell)}{\aleph(A^0)}, \tag{2.4}$$

where ν_ℓ is the number of relaxation sweeps on level ℓ. If we consider one relaxation sweep on the fine grid as one work unit, then χ_{cyc} approximates the amount of work per cycle; for a V(1,1)-cycle, χ_{op} and χ_{cyc} differ by a factor of approximately 2. As we intend to iterate on both spectral and finite element matrices, we denote by χ^s_{cyc} and χ^f_{cyc} the respective cycle complexities.

The results listed in [16] indicate that the cycle complexities χ^f_{cyc} using finite element preconditioning are much larger than the complexities of AMG applied directly to the spectral element operator χ^s_{cyc}. Moreover, χ^s_{cyc} scales with polynomial order, while χ^f_{cyc} grows slightly. If we consider the complexity of the overall algorithm, the results are even more compelling. The fundamental difficulty with χ^f_{cyc} is that $A_f u = b$ is not the target system and does not dominate the process. Rather, the matrix-vector multiply in the CG wrapper involves spectral element operator A_s and comprises much of the cost. This identifies the suboptimal performance indicated by χ^f_{cyc}. Additionally, due to the extreme density of the high-order operator on the finest level, the spectral complexity χ^s_{cyc} is deceptively small. To extend these definitions, we propose the following hybrid operator complexity which accounts for the finite element operator within the spectral PCG wrapper (Algorithm 1):

$$\chi^\star_{\text{op}} = \frac{\aleph(A^0) + \sum_{\ell=0}^{\ell_{max}} \aleph(A^\ell_f)}{\aleph(A^0)} \tag{2.5}$$

$$= \frac{\aleph(A^0) + \chi^f_{\text{op}} \cdot \aleph(A^0_f)}{\aleph(A^0)}. \tag{2.6}$$

A similar definition for $\chi^\star_{\text{cyc}}$ follows.

Combining the work per cycle (2.6) with the convergence factor ρ, we define the overall work as the number of work units to reduce the residual by a factor of 10:

$$W = -\frac{\chi_{\text{cyc}}}{\log_{10} \rho} \tag{2.7}$$

or

$$W^\star = -\frac{\chi^\star_{\text{cyc}}}{\log_{10} \rho}. \tag{2.8}$$

2.2. Numerical tests. In this section we present numerical results validating the scalability of Algorithm 1 with the measures introduced in the previous section. In particular, we refer to four situations listed in Table 2.1.

The performance results in Figures 2.3(c) and 2.4 are for a spectral element unstructured triangulation of the unit square with 574 elements. The mesh quality is high, with a normalized minimum angle of 0.7 (over a scale of 0.0 to 1.0) and an area ratio of approximately 0.5. The solver performance was consistent for both coarser and finer spectral element meshes, with nearly constant element diameters across the tessellation; for brevity we present only this particular case.

TABLE 2.1
List of methods.

CG-A_s	The classical CG method applied to the target system of equations $A_s u = b$.
AMG-A_s	AMG V(1,1)-cycles as a solver for $A_s u = b$, rather than used as a preconditioner.
PCG-AMG-A_s	AMG-A_s with acceleration. Algorithm 1 with $\hat{A} = A$.
PCG-AMG-A_f	AMG preconditioned CG using low-order finite element preconditioning. Algorithm 1 with $\hat{A} = A_f$.

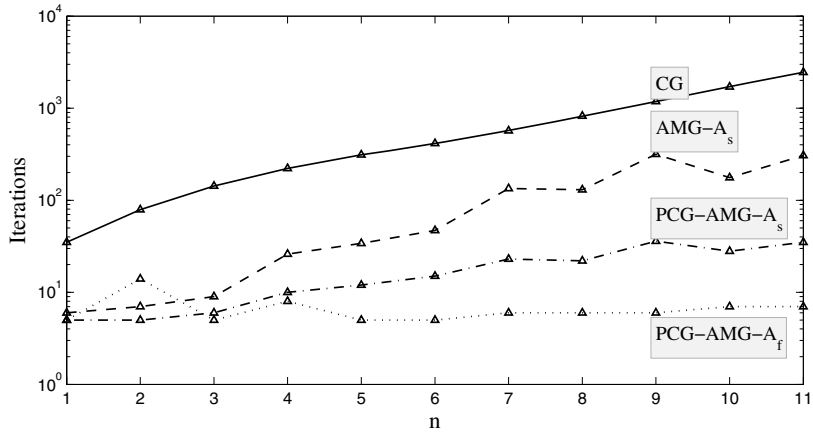
The methods in Table 2.1 are compared as the order of the spectral element is increased from linear ($n = 1$) to very high-order ($n = 11$). Figures 2.3(a)–2.3(c) highlight the efficiency of low-order preconditioning with AMG. As a benchmark, pure CG iterations (CG-A_s) are also listed in the first figure. From Figure 2.3(b) it is evident that AMG alone is unable to handle orders beyond $n = 6$. This is consistent with previous attempts where direct application of AMG was found to become ineffective at $n \approx 4$ [29]. Furthermore, both Figures 2.3(a) and 2.3(b) confirm the scalability (in spectral order) for low-order preconditioning. Krylov acceleration alone is unable to effectively capture problematic eigenmodes in the faulty AMG-A_s. The level curve for PCG-AMG-A_f compared with the nontrivial slope in the PCG-AMG-A_s approach confirms that the low-order operator A_f combined with AMG is accurately preconditioning the spectral operator in the CG algorithm.

Figures 2.3(a) and 2.3(b) do not fully address the complexity of the algorithm. The complexity of steps 2 and 8 in Algorithm 1 will vary depending on $\hat{A}$. For both AMG-A_s and PCG-AMG-A_s, we base the work (2.7) on complexity χ^s_{cyc} as in (2.4). The PCG-AMG-A_f algorithm with $\hat{A} = A_f$ requires the hybrid complexity $\chi^\star_{\text{cyc}}$ defined by $\chi^\star_{\text{op}}$ in (2.6), and the associated work is defined in (2.8). The work is presented in Figure 2.3(c). Since AMG is able to maintain a modest level of complexity for the low-order operator as n increases, the overall complexity remains close to 1.0, while the work increases significantly when AMG is applied to the spectral operator A_s.

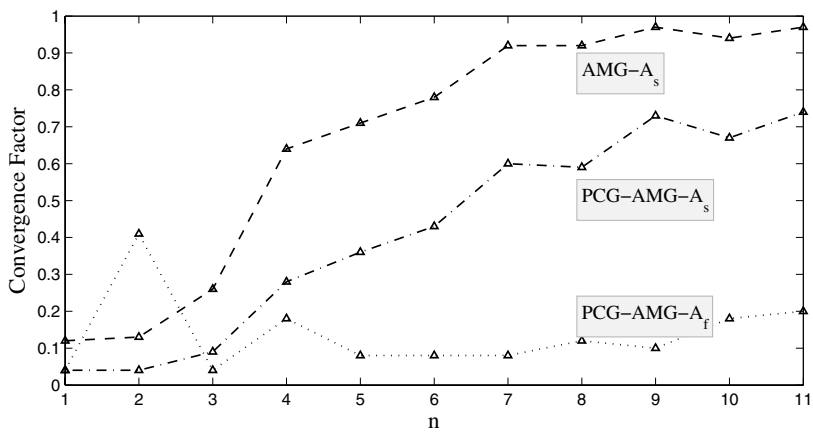
The low complexity associated with AMG applied to A_f is predictable, as the density of the operator is much less than that of the spectral operator A_s (cf. Figure 2.4): the number of connections per node in the graph of A_f was on average 6, and the maximum number of connections increases from 8 to 12 with n. The notable AMG advantage is the ability to handle large variations in the size of the elements. The area ratios of the finite element mesh decrease with $\mathcal{O}(n^4)$ (cf. Figure 2.4) as the polynomial order increases, while the minimum angle degrades only slightly, being preserved by the local Delaunay process. This view captivates the necessity of using an algebraic-based solver in the context of low-order preconditioning on unstructured grids.

We extend this approach to three dimensions in a similar manner over a spectral element unstructured mesh of 188 elements. The performance for the methods listed in Table 2.1 are presented in Figure 2.5 and reveal a trend similar to that of two dimensions. AMG combined with CG acceleration effectively preconditions the spectral element operator A_s when combined with A_f, while use of AMG with A_s yields both poor convergence and high complexities.

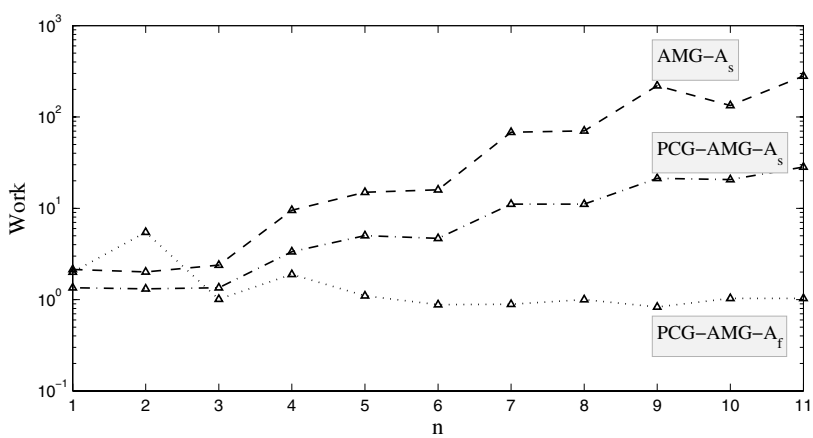
Similar to the case of quadrilaterals [16], AMG should be used directly with A_s when the spectral order is low $n \leq 4$, while low-order preconditioning is beneficial for high-order $n > 4$. When matrix-free application of the spectral operator A_s is available, then $\hat{A} = A_f$ gains even more appeal, although construction and management



(a) Iterations



(b) Convergence Factors



(c) Total Work

FIG. 2.3. *Performance measures in two dimensions: The convergence factors (ρ), iterations to convergence, and work per digit of accuracy are listed for the methods in Table* 2.1.

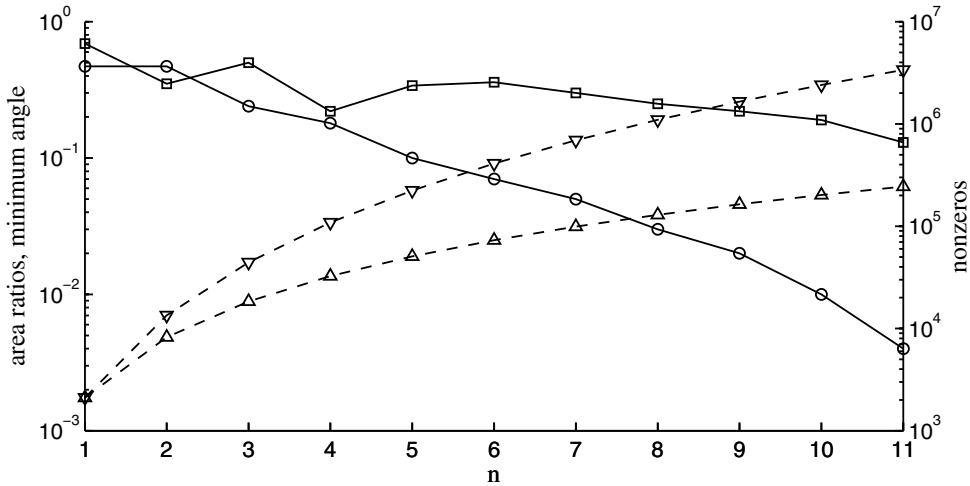
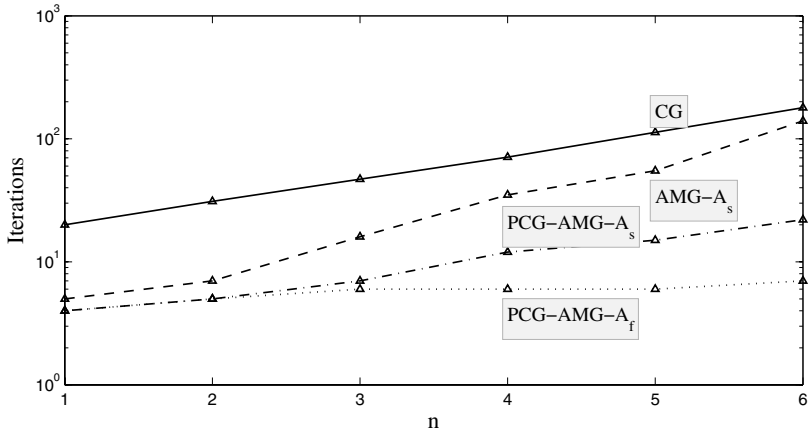


FIG. 2.4. *Mesh diagnostics: The number of connections in both A_s (∇) and in A_f ($\triangle$) along with the normalized minimum angle ($\square$) and the area ratio ($\circ$) in the finite element mesh.*

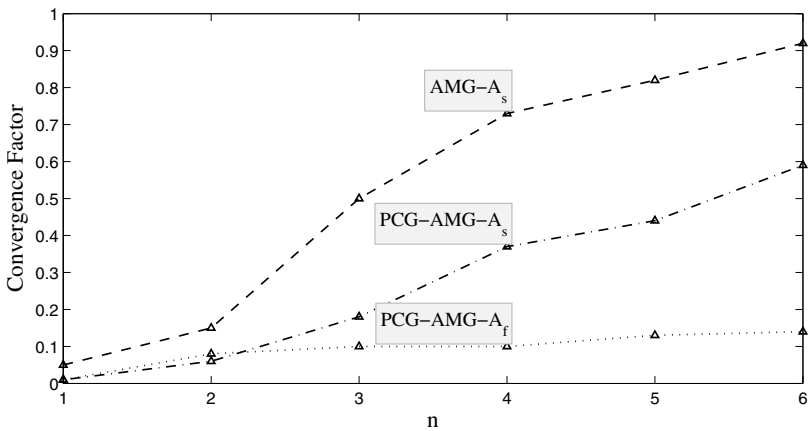
of the low-order finite element operator should be considered. From Figure 2.3(c), we notice a similar separation at $n = 4$, with PCG-AMG-A_f remaining relatively inexpensive. Moreover, the 3D simulations show this threshold to be at slightly lower orders—at $n \approx 2$ or 3—as illustrated in Figure 2.5(c).

The deterioration in convergence for classical AMG applied directly to A_s is largely due to the underlying M-matrix assumptions of the method. Both strength-of-connection and interpolation construction in RS-AMG are motivated through principles based on an M-matrix. Indeed, much of the early convergence theory on AMG relies on the system being an M-matrix [30], and the high-order matrices A_s considered here do not have this property: $a_{ij} > 0$ for some (i, j) (apart from the linear case). A well-known theoretical tool to generalize the theory is the concept of an *essentially positive* matrix [2]. For positive off-diagonal connections a_{ij}, if there exists a d-path with sufficiently large negative weight, then the AMG convergence results still hold [32]. Computationally, however, the adjustments for essentially positive matrices (i.e., basing coarse-grid selection and interpolation on negative weights only) have little impact. The results vary little whether positive connections are ignored or collapsed.

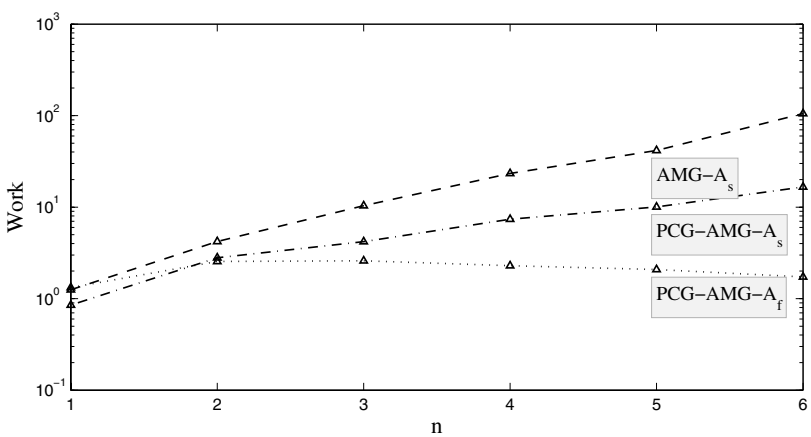
The notable change in performance for AMG at certain polynomial orders (e.g., $n = 4$ in two dimensions) is that the matrices are no longer essentially positive (with a path length of two). As an example, in the 2D case with a effective grid size of $h \approx 1/4$, the matrix is heavily populated with off-diagonals from $n = 2$, but the essentially positive property is not lost until $n = 5$ (see Figure 2.6). The number of positive edges in the graph of A_s grows with $\mathcal{O}(n^4)$, making classical AMG strength measures increasingly less effective. The $n = 4$ case coincides also with the deterioration of AMG convergence, thus confirming these M-matrix diagnostics. One common problem in utilizing very long-range connections in interpolation is that the computation is overwhelmed by the operator complexities.



(a) Iterations



(b) Convergence Factors



(c) Total Work

FIG. 2.5. *Performance measures in three dimensions: The convergence factors (ρ), iterations to convergence, and work per digit of accuracy are listed for the methods in Table* 2.1.

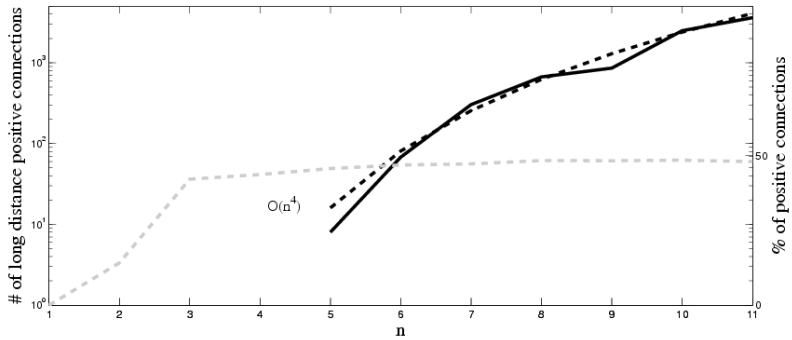


FIG. 2.6. *The increase in positive off-diagonal connections that do not have a sufficiently large negatively weighted path of length two (solid line). The dark dashed line is the $\mathcal{O}(n^4)$ reference, and the dashed gray line indicates the percentage of off-diagonals that are positive.*

3. Discontinuous methods. The DG method was initially introduced as a solution method for solving hyperbolic and convection dominated problems [6]. More recently, the DG methodology has successfully been extended to a variety of problems, becoming a robust and popular framework. In this section we describe the DG approach followed throughout this paper.

A variety of DG methods have been proposed in recent years for various problems [1]; we follow a *local* DG method (LDG) [7], which is described as follows. Recalling the model elliptic problem (1.1), let $\mathbf{q} = \nabla u$ and rewrite the problem as a first-order system:

$$\mathbf{q} - \nabla u = 0 \quad \text{in } \Omega, \tag{3.1a}$$

$$-\nabla \cdot \mathbf{q} = f \quad \text{in } \Omega, \tag{3.1b}$$

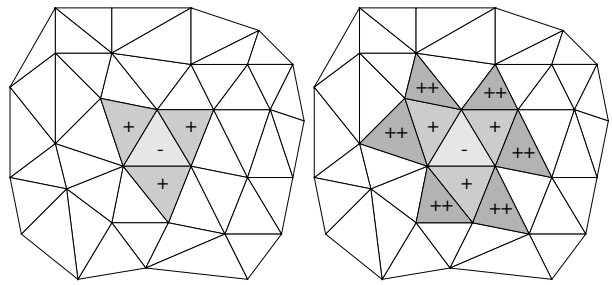
$$u(\mathbf{x}) = g \quad \text{on } \partial\Omega. \tag{3.1c}$$

Central to the DG methodology is the discontinuity in the approximation between elements; the DG varieties are distinguished by the way interelement connectivity is imposed. Let $\mathcal{K}^h$ be a tessellation of Ω, and consider the weak form of (3.1) over each $\kappa \in \mathcal{K}^h$:

$$\int_\kappa \mathbf{q} \cdot \psi \, d\mathbf{x} + \int_\kappa u \nabla \cdot \psi \, d\mathbf{x} = \oint_{\partial\kappa} u\psi \cdot \mathbf{n}_\kappa \, d\mathbf{x}, \tag{3.2a}$$

$$\int_\kappa \mathbf{q} \cdot \nabla\phi = \oint_{\partial\kappa} \mathbf{q} \cdot \mathbf{n}_\kappa \, d\mathbf{x} + \int_\kappa f\phi \, d\mathbf{x}, \tag{3.2b}$$

where $\phi(\mathbf{x})$ and $\psi(\mathbf{x})$ are scalar and vector test functions, respectively, and $\mathbf{n}_\kappa$ is the outward normal on κ. The interelement connectivity is now determined by definition of numerical flux on each edge. On element κ, define u^- to be the value of u interior to the element, and define u^+ to be the value of u in the adjacent neighboring element. For a scalar function u and vector function $\mathbf{q}$, the *jump* and the *average* between neighboring elements are respectively defined as $[\![u]\!] = u^-\mathbf{n}^- + u^+\mathbf{n}^+$, $\{\!\{u\}\!\} = \frac{1}{2}(u^- + u^+)$, $[\![\mathbf{q}]\!] = \mathbf{q}^- \cdot \mathbf{n}_{k^-} + \mathbf{q}^+ \cdot \mathbf{n}_{k^+}$, $\{\!\{\mathbf{q}\}\!\} = \frac{1}{2}(\mathbf{q}^- + \mathbf{q}^+)$. We define the numerical fluxes u^* and $\mathbf{q}^*$ independently of ∇u, allowing the formulation of the weak problem (3.2) independently of the flux $\mathbf{q}(\mathbf{x})$. In general, the numerical fluxes for the



(a) LDG (b) Brezzi

FIG. 3.1. *Stencil width for the LDG method compared with Brezzi and related DG methods. The width is one for the case of LDG, while dependence extends to two layers (++) for other common DG methods.*

LDG method are defined as [1]

$$u^* = \{\!\{u\}\!\} + \beta \cdot [\![u]\!], \qquad \mathbf{q}^* = \{\!\{\mathbf{q}\}\!\} - \beta [\![\mathbf{q}]\!] - \eta_k [\![u]\!], \tag{3.3}$$

which replace u and $\mathbf{q}$ in the elemental boundary terms of (3.2). With this, we strengthen the weak problem slightly using an additional integration-by-parts to arrive at

$$\int_\kappa \mathbf{q} \cdot \psi \, d\mathbf{x} - \int_\kappa \nabla u \cdot \psi \, d\mathbf{x} = \oint_{\partial\kappa} (u^* - u)\psi \cdot \mathbf{n}_\kappa \, d\mathbf{x}, \tag{3.4a}$$

$$- \int_\kappa \nabla \cdot \mathbf{q}\phi = \oint_{\partial\kappa} (\mathbf{q}^* - \mathbf{q}) \cdot \mathbf{n}_\kappa \, d\mathbf{x} + \int_\kappa f\phi \, d\mathbf{x} \tag{3.4b}$$

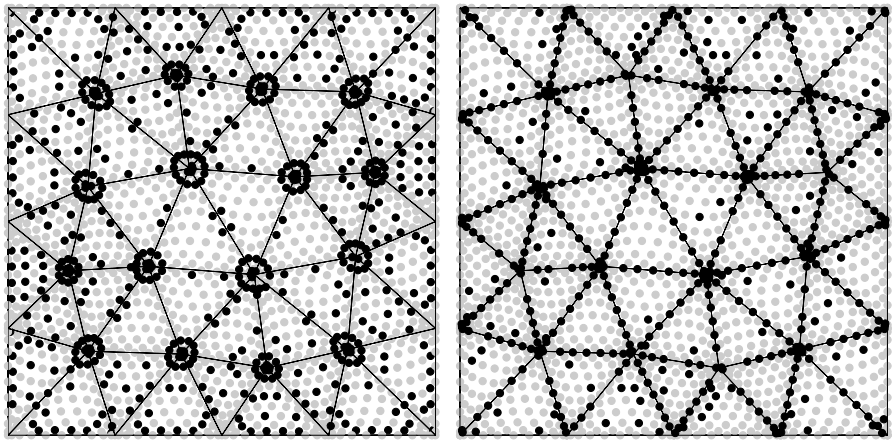
for all $\kappa \in \mathcal{K}$. The construction leads to the linear system $A^{dg}u = b$, which is the focus of the section.

Choosing $\beta = \mathbf{n}_\kappa/2$ ensures symmetry and locality of the associated stiffness matrix [1] and is a standard choice which leads to the *local* DG method.

3.1. AMG modification. While the LDG method does exhibit many advantageous properties in terms of the approximation both mathematically and computationally, it introduces additional complexities in the algebraic system. The issues concerning AMG are illustrated in this section, including modifications to the basic RS algorithm.

A key component of the LDG formulation is the so-called penalty term, identified by the coefficient η_κ in (3.3). From an approximation standpoint, η_κ offers stability control as it moderates the effect of the jump in u, $[\![u]\!]$. Moreover, it is well known that the spectrum of the linear system is influenced by the η_κ parameter, impacting many iterative solution methods. For elliptic problems, we can expect a relationship of $\text{cond}(A) = \mathcal{O}(\eta)$ between the condition number and η_κ, with $\eta_\kappa = \eta$ for all $\kappa \in \mathcal{K}$ [5]. With a well-balanced penalty term of $\eta_\kappa = \frac{n}{h^2}$ (cf. [34]), the preconditioner proposed below performed well and exhibited no dependence on slight variations in η_κ. This may be partly due to the results reported in [5] indicating less variability with respect to the penalty parameter in the condition number at higher orders.

Our choice in DG schemes was partly based on the fact that LDG limits the breadth of dependence. Figure 3.1 illustrates this effect. The result is that (1) the locality of AMG interpolation will be more effective for tighter stencils and (2) a low-order preconditioner will be more easily identified since construction can be arguably constrained in the first level to interelement connectivity only.



(a) Continuous Galerkin (b) Discontinuous Galerkin

FIG. 3.2. *AMG C/F splitting for $n = 11$. The C-points are represented by gray circles and the F points by black circles.*

Still, central to DG is the concept of additional degrees-of-freedom along element boundaries. AMG applied directly to the spectral operator in the continuous Galerkin method yields adequate results; we cannot expect such straightforward application in the context of DG methods, as the connections between degrees of freedom along a boundary play a much different role than connectivity with interior nodes. Indeed, Figure 3.2(a) identifies this subtlety for the case of $n = 11$. Figure 3.2(a) is the C/F splitting for the continuous methods analyzed in the previous section, while 3.2(b) depicts the C/F splitting for the first-level. The latter case results in a nonconvergent method and, more notably, very high complexities.

The solution is to extend the definition of "strong connections" in the AMG setup phase. In the classical RS algorithm presented earlier, only *direct* connections in the graph of the matrix are considered as possible interpolation nodes. In the DG framework, this results in element boundary nodes being interpolated only from nodes interior to the element and from nodes on adjacent element boundaries. Conversely, in the continuous Galerkin setting, the scope of interpolation extends into both elements for points along an edge. We accomplish this extension by considering long-range connections in the graph for interpolation (see Figure 3.3).

To consider long-range influence, we use an approach based on relaxation in defining the strong connections since using relaxation circumvents the need for an M-matrix. The implementation was introduced in [28] but has not been used extensively in practice. For node i, we seek a measure of influence from some node j (not necessarily connected to i). There are three cases in which we would like to account for long-range influence; three examples are illustrated in Figure 3.4. In Figure 3.4(a), we account for the strength of the direct connection i to j. In Figure 3.4(b), we see that i will benefit from having two connections to a pair $\{j, k\}$ with a strong connection. The influence of k on j will either strengthen the existing connection to i or possibly balance the connection, leaving the influence of j on i as weak. Finally, Figure 3.4(b) identifies the most common example of long-range influence: no connection between i and j, but a common connection to node k.

To identify the influence in the cases of Figure 3.4 we employ the following algo-

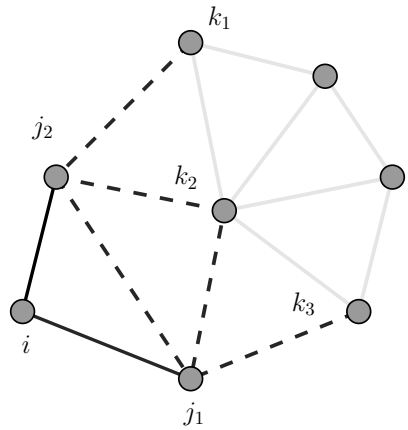


FIG. 3.3. *Long-range connections: Node i is directly connected to j indices, while having long-range connections to k indices. The direct connections are solid black, and the long-range (or direct-direct) connections are dashed black.*

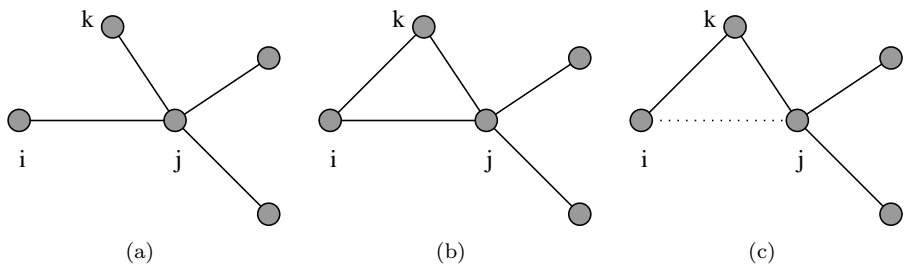


FIG. 3.4. *Three cases for long-range influence: only direct, common direct, and indirect.*

rithm:

1. Initialize vector v to be 1 at node j and zero otherwise.
2. Relax at i.
3. Relax at each k.
4. Relax again at i.

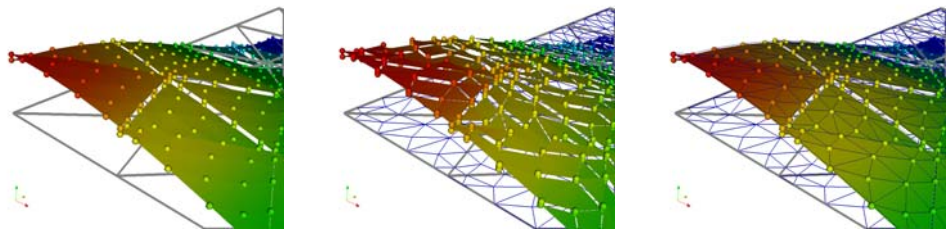
Since we consider smooth error to have little variation in the direction of a strong connection, a large value of v_i will indicate a strong connection between i and j. Classical RS interpolation is then implemented.

3.2. Hybrid preconditioner. The construction of a low-order preconditioner resembles the low-order finite element preconditioner studies in section 2; however, we introduce several modifications to account for the discontinuity. Coupled with the modified AMG algorithm in section 3.1, the process yields a scalable method.

The preconditioner we present is motivated by three components:

1. Low-order preconditioning is effective in the simplex (cf. section 2).
2. Continuous, linear finite elements are effective coarse spaces in domain decomposition for low-order DG methods [19].
3. Interelement connectivity in the spectral element DG method is closely associated with the spectrum of the matrix (cf. (3.4)).

Combining steps 1–3 into a suitable algorithm provides the basis of our approach. Given the bilinear form $a^{dg}(u, v)$ defined by the LDG method (3.4), we have the



(a) Spectral Element (b) Finite Element (c) Hybrid

FIG. 3.5. *Grid vectors in a spectral element, finite element, and locally projected (hybrid) space.*

following weak problem defined: Find $u, v \in V^{dg}$ so that

$$a^{dg}(u,v) = \ell(v) \qquad \forall\, v \in V^{dg}, \tag{3.5}$$

where $V^{dg} = V_s^{dg}$ defines the discontinuous nodal spectral element basis—Lagrange interpolating polynomials over electrostatic nodes as defined in section 1.1—over the spectral element mesh $\mathcal{K}_s = \mathcal{K}$. This leads to the matrix equation $A_s^{dg} u = f$. Similarly, we define (3.5) with a discontinuous linear finite element space V_f^{dg} over the locally constructed finite element mesh $\mathcal{K}_f$ of section 2 (cf. Figure 2.2), resulting in a (low-order) matrix equation $A_f^{dg} u = b$.

Motivated by [19] we construct operator C to project the LDG problem into a locally *continuous* finite element space. Denote by $V^\star$ a hybrid space of continuous finite elements within each spectral element. $V^\star$ is defined as

$$V^\star = \{u \in V_f^{dg} \,:\, u \in H_0^1(\kappa) \,\forall\, \kappa \in \mathcal{K}_s\}. \tag{3.6}$$

Functions in $V^\star$ are continuous across local elements but remain discontinuous across spectral elements $\kappa \in \mathcal{K}_s$. Defining $C : V_f^{dg} \to V^\star$ by averaging the jump, $u \leftarrow (u^+ + u^-)/2$, we arrive at the hybrid preconditioner matrix

$$A^\star = C A_f^{dg}. \tag{3.7}$$

The preconditioner is illustrated in Figure 3.5 using an example grid vector. The underlying spectral element mesh with for $n = 5$ is apparent in Figure 3.5(a), while the discontinuous finite element problem $A_f^{dg} u = b$ is evident by the jumps and locally refined mesh in Figure 3.5(b). Figure 3.5(c) highlights the hybrid preconditioner, where jumps are maintained between spectral elements and the local finite element space is continuous.

With the hybrid preconditioner defined, the advantage of using long-range connections as proposed in section 3.1 can be seen in Figure 3.6. The connectivity between an element boundary node i and nodes in adjacent elements (e.g., j) is limited in the case of direct connections. However, with long-range connections, the possible influence is more extensive, mimicking the case of continuous elements.

3.3. Numerical tests. We confirm the efficacy of the hybrid preconditioner proposed in the previous section by considering the model problem (1.1) in two dimensions. An LDG method is constructed based on (3.4), and we test the performance as the order of the spectral elements increases by considering the complexity measures presented in section 2.1.

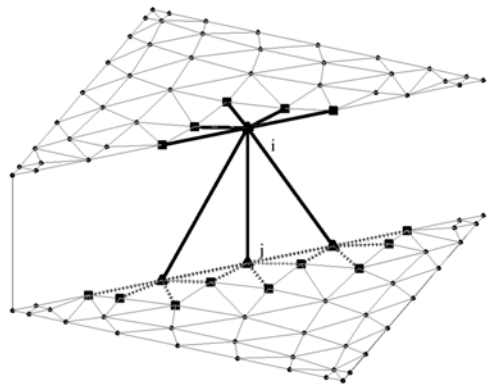


FIG. 3.6. *Long-range connections in the hybrid preconditioner: node i and node j with direct connections (solid) and additional long-range connections (dotted).*

TABLE 3.1

AMG preconditioning for DG: Polynomial order n is compared with the number of iterations of Algorithm 1, the average convergence factor ρ, the total work per digit accuracy $W^\star$, the number of nonzeros in the matrices, and the total number of degrees-of-freedom in the spectral element problem N.

n	Iterations	ρ	$W^\star$	$\aleph(A_s^{dg})$	$\aleph(A^\star)$	N
1	19	0.46	9.77	3420	3420	138
2	8	0.18	3.33	13680	14076	276
3	9	0.21	3.19	38000	35100	460
4	9	0.20	2.57	85500	59184	690
5	6	0.09	1.61	167580	98244	966
6	9	0.18	1.87	297920	141408	1288
7	11	0.26	2.05	492480	194508	1656
8	12	0.29	2.02	769500	251712	2070
9	12	0.29	1.88	1149500	321372	2530
10	13	0.34	2.05	1655280	399312	3036
11	12	0.30	1.68	2311920	478908	3588

We present in Table 3.1 the performance measures for Algorithm 1 with $\hat{A} = A^\star$. As the polynomial order increases, the number of iterations needed to converge along with the average convergence factor remain generally bounded. Furthermore, the complexity measure $\chi^\star_{\text{op}}$ justified in section 2.1 combined with the average convergence factor yields a low, bounded work measure $W^\star$, confirming the scalability of the method with respect to spectral order. Additional simulations on both coarser and finer spectral element grids generated nearly identical trends.

The performance listed in Table 3.1 is encouraging,[1] particularly in view of the number of nonzeros in the matrix. The ratios of nonzeros (columns 5 and 6) to problem size N (column 7) is plotted in Figure 3.7. The complexity of the fine grid preconditioning operator $A^\star$ remains low, while the density in the spectral operator A_s^{dg} grows with n, as expected.

Finally, the C/F splitting in AMG for the fine-grid hybrid preconditioner is presented in Figure 3.8. Comparing with Figure 3.2, the long-range connections used for

[1]The case of $n = 1$ is trivial. The additional and unnecessary work in constructing the preconditioner is evident in the results, which are provided for completeness.

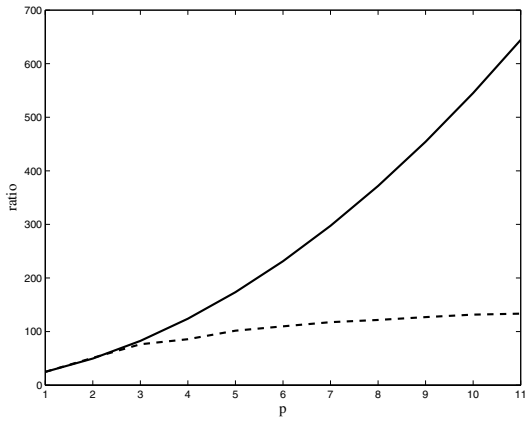


FIG. 3.7. *The ratio of nonzeros in matrix A_s^{dg} (solid) and in matrix $A^\star$ (dashed) to the matrix size N.*

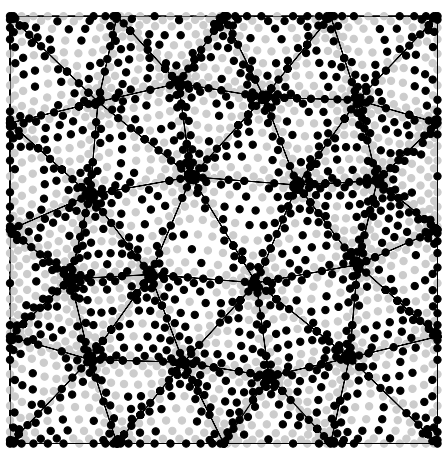


FIG. 3.8. *AMG C/F splitting for $n = 11$ for the matrix of the hybrid discontinuous preconditioner. The C-points are represented by gray circles and the F points by black circles.*

the hybrid preconditioner are able to produce a more compatible splitting, resulting in both better convergence and lower complexity.

4. Summary. The proposed computational framework is shown to be an effective approach for solving high-order spectral element methods. The algorithm, based on low-order preconditioning, provides a straightforward way of combining two highly successful methodologies: algebraic multigrid and high-order spectral element discretizations.

The present work contributes to the development of scalable solvers for high-order discretizations in several ways. A low-order preconditioner is extended to d-simplexes using a local Delaunay triangulation, while hybrid measures are proposed to accurately weigh the cost of using a low-order multilevel preconditioner in the context of a high-order spectral element discretization. Both 2D and 3D results highlight the efficacy for continuous methods and provide a motivational basis in moving toward discontinuous discretizations.

A new multilevel preconditioner for discontinuous methods is introduced. The approach utilizes an extension to the classical Ruge–Stüben AMG which accounts for long-range connections in the coarse-grid selection phase and in construction of the intergrid transfer operators. The AMG modifications are used in combination with a local low-order continuous finite element preconditioner to efficiently solve the spectral element problem.

Acknowledgment. The author thanks the developers of Sledge++, a discontinuous spectral element library [35], which provided the basis for the numerical results presented in this paper. Additionally, portions of the RS implementation and modifications were developed by John Ruge; the author is appreciative of the many helpful discussions.

REFERENCES

[1] D. N. Arnold, F. Brezzi, B. Cockburn, and L. D. Marini, *Unified analysis of discontinuous Galerkin methods for elliptic problems*, SIAM J. Numer. Anal., 39 (2002), pp. 1749–1779.

[2] A. Brandt, *Algebraic multigrid theory: The symmetric case*, Appl. Math. Comput., 19 (1986), pp. 23–56.

[3] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, 2nd ed., SIAM, Philadelphia, 2000.

[4] C. Canuto and P. Pietra, *Boundary and interface conditions within a finite element preconditioner for spectral methods*, J. Comput. Phys., 91 (1990), pp. 310–343.

[5] P. Castillo, *Performance of discontinuous Galerkin methods for elliptic PDEs*, SIAM J. Sci. Comput., 24 (2002), pp. 524–547.

[6] B. Cockburn, G. E. Karniadakis, and C.-W. Shu, *The development of discontinuous Galerkin methods*, in Discontinuous Galerkin Methods (Newport, RI, 1999), Lecture Notes in Comput. Sci. Eng. 11, Springer, Berlin, 2000, pp. 3–50.

[7] B. Cockburn and C.-W. Shu, *The local discontinuous Galerkin method for time-dependent convection-diffusion systems*, SIAM J. Numer. Anal., 35 (1998), pp. 2440–2463.

[8] M. Deville and E. Mund, *Chebyshev pseudospectral solution of second-order elliptic equations with finite element preconditioning*, J. Comput. Phys., 60 (1985), pp. 517–533.

[9] M. O. Deville, P. F. Fischer, and E. H. Mund, *High-Order Methods for Incompressible Fluid Flow*, Cambridge Monogr. Appl. Comput. Math. 9, Cambridge University Press, Cambridge, UK, 2002.

[10] M. O. Deville and E. H. Mund, *Finite-element preconditioning for pseudospectral solutions of elliptic problems*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 311–342.

[11] M. O. Deville and E. H. Mund, *Finite element preconditioning of collocation schemes for advection-diffusion equations*, in Iterative Methods in Linear Algebra (Brussels, 1991), North–Holland, Amsterdam, 1992, pp. 181–189.

[12] B. T. Helenbrook, D. Mavriplis, and H. L. Atkins, *Analysis of "p"-multigrid for continuous and discontinuous finite element discretizations*, in Proceedings of the 16th AIAA Computational Fluid Dynamics Conference, Orlando, FL, 2003.

[13] J. S. Hesthaven, *From electrostatics to almost optimal nodal sets for polynomial interpolation in a simplex*, SIAM J. Numer. Anal., 35 (1998), pp. 655–676.

[14] J. S. Hesthaven, *High-order accurate methods in time-domain computational electromagnetics. A review*, Adv. Imaging Electron Phys., 17 (2003), pp. 59–123.

[15] J. S. Hesthaven and T. Warburton, *Nodal high-order methods on unstructured grids.* I. *Time-domain solution of Maxwell's equations*, J. Comput. Phys., 181 (2002), pp. 186–221.

[16] J. J. Heys, T. A. Manteuffel, S. F. McCormick, and L. N. Olson, *Algebraic multigrid for higher-order finite elements*, J. Comput. Phys., 204 (2005), pp. 520–532.

[17] M. Iskandarani, D. B. Haidvogel, J. C. Levin, E. Curchitser, and C. A. Edwards, *Multiscale geophysical modeling using the spectral element method*, Computing in Science & Engineering, 4 (2002), pp. 42–48.

[18] G. E. Karniadakis and S. J. Sherwin, *Spectral/hp Element Methods for CFD*, Numer. Math. Sci. Comput., Oxford University Press, New York, 1999.

[19] C. Lasser and A. Toselli, *An overlapping domain decomposition preconditioner for a class of discontinuous Galerkin approximations of advection-diffusion problems*, Math. Comp., 72 (2003), pp. 1215–1238.

[20] I. Lomtev and G. E. Karniadakis, *A discontinuous Galerkin method for the Navier-Stokes equations*, Internat. J. Numer. Methods Fluids, 29 (1999), pp. 587–603.
[21] J. W. Lottes and P. F. Fischer, *Hybrid multigrid/Schwarz algorithms for the spectral element method*, J. Sci. Comput., 24 (2005), pp. 45–78.
[22] Y. Maday and R. Muñoz, *Spectral element multigrid.* II. *Theoretical justification*, J. Sci. Comput., 3 (1988), pp. 323–353.
[23] Y. Maday, R. Muñoz, A. Patera, and E. Rønquist, *Spectral element multigrid methods*, in Iterative Methods in Linear Algebra (Brussels, 1991), North–Holland, Amsterdam, 1992, pp. 191–201.
[24] S. A. Orszag, *Spectral methods for problems in complex geometries*, J. Comput. Phys., 37 (1980), pp. 70–92.
[25] S. V. Parter and E. E. Rothman, *Preconditioning Legendre spectral collocation approximations to elliptic problems*, SIAM J. Numer. Anal., 32 (1995), pp. 333–385.
[26] L. F. Pavarino, E. Zampieri, R. Pasquetti, and F. Rapetti, *Overlapping Schwarz methods for Fekete and Gauss–Lobatto spectral elements*, SIAM J. Sci. Comput., 29 (2007), pp. 1073–1092.
[27] E. M. Rønquist and A. T. Patera, *Spectral element multigrid.* I. *Formulation and numerical results*, J. Sci. Comput., 2 (1987), pp. 389–406.
[28] J. Ruge, *AMGS*01, FORTRAN Library, June, 1997.
[29] J. Ruge, *AMG for higher-order discretizations of second-order elliptic problems*, in Proceedings of the Eleventh Copper Mountain Conference on Multigrid Methods, Copper Mountain, CO, 2003.
[30] J. W. Ruge and K. Stüben, *Algebraic multigrid*, in Multigrid Methods, S. F. McCormick, ed., Frontiers Appl. Math. 3, SIAM, Philadelphia, 1987, pp. 73–130.
[31] M. A. Taylor, B. A. Wingate, and R. E. Vincent, *An algorithm for computing Fekete points in the triangle*, SIAM J. Numer. Anal., 38 (2000), pp. 1707–1720.
[32] U. Trottenberg, C. W. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, San Diego, CA, 2001.
[33] T. Warburton, *An explicit construction for interpolation nodes on the simplex*, J. Engrg. Math., 56 (2006), pp. 247–262.
[34] T. Warburton and M. Embree, *The role of the penalty in the local discontinuous Galerkin method for Maxwell's eigenvalue problem*, Comput. Methods Appl. Mech. Engrg., 195 (2006), pp. 3205–3223.
[35] T. Warburton, J. Hesthaven, and L. Wilcox, *Sledge++ Users' Guide*, http://www.caam.rice.edu/~timwar/TimWarburton/Sledge++.html, 2006.

SIAM J. SCI. COMPUT.
Vol. 29, No. 5, pp. 2210–2223
© 2007 Society for Industrial and Applied Mathematics

APPLYING AUTOMATED MEMORY ANALYSIS TO IMPROVE ITERATIVE ALGORITHMS*

J. M. DENNIS† AND E. R. JESSUP‡

Abstract. In this paper, we describe automated memory analysis, a technique to improve the memory efficiency of a sparse linear iterative solver. Our automated memory analysis uses a language processor to predict the data movement required for an iterative algorithm based upon a MATLAB implementation. We demonstrate how automated memory analysis is used to reduce the execution time of a component of a global parallel ocean model. In particular, code modifications identified or evaluated through automated memory analysis enable a significant reduction in execution time for the conjugate gradient solver on a small serial problem. Further, we achieve a 9% reduction in total execution time for the full model on 64 processors. The predictive capabilities of our automated memory analysis can be used to simplify the development of memory-efficient numerical algorithms or software.

Key words. memory analysis, language processor, sparse linear algebra, ocean modeling

AMS subject classifications. 68-04, 68U01, 68W40, 86A05

DOI. 10.1137/060661533

1. Introduction and motivation.

Many important scientific and engineering computations involve the solution of a set of partial differential equations formulated as a large sparse system of linear equations. The solution of the linear system by an iterative linear solver is often the single most costly component of the application [13, 26]. Historically, iterative solvers have been designed to achieve the best numerical accuracy for a given number of floating-point operations [18, 24, 25]. This approach is based on the assumption that the time to perform those operations dominates the cost of the solver. This assumption is no longer true because, while advances in computer architecture have significantly reduced the cost of a floating-point operation, the memory access cost has not seen nearly as rapid an improvement [21]. We must therefore take a memory-centric approach to the design and implementation of efficient algorithms. To that end, we propose the use of memory analysis as an integral component of the design process. Memory analysis evaluates the interaction of the algorithm with the memory hierarchy by calculating the amount of data that must be loaded from the memory hierarchy for each component of the iterative algorithm. We have developed the Sparse Linear Algebra Memory Model (SLAMM) language processor to automate memory analysis. The SLAMM language processor analyzes an

*Received by the editors May 31, 2006; accepted for publication (in revised form) July 2, 2007; published electronically September 28, 2007. This work was performed by an employee of the U.S. Government or under U.S. Government contract. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights.

http://www.siam.org/journals/sisc/29-5/66153.html

†Computational & Information Systems Laboratory, National Center for Atmospheric Research, Boulder, CO 80307-3000 (dennis@ucar.edu). The work of this author was supported by the National Science Foundation Cooperative under grant NSF01 and by the Department of Energy under CPP Program grant DE-FC03-97ER62402.

‡Department of Computer Science, University of Colorado, Boulder, CO 80309-0430 (jessup@cs.colorado.edu). The work of this author was supported by the National Science Foundation under grants ACI-0072119 and CCF-0430646.

algorithm written in MATLAB code. Unlike a cache monitoring tool which only measures actual data movement of an existing implementation, a SLAMM analysis gives a prediction of the minimum required data movement for an efficient implementation of the algorithm in a compiled language.

Automated memory analysis provides both the ability to rapidly evaluate the memory efficiency of a particular design choice during the design phase and the ability to improve the memory efficiency of a preexisting solver. We illustrate both applications by using SLAMM to reduce the execution time of the Parallel Ocean Program (POP) [27]. In section 2, we provide the background for memory-efficient programming and review related work. In section 3, we describe POP, a parallel ocean model developed at Los Alamos National Laboratory. We describe the two-dimensional (2D) data structure it currently uses and an alternative one-dimensional (1D) data structure in section 3. In section 4, we describe memory analysis, a technique to calculate the amount of data that must be moved through the memory hierarchy for each component of an iterative algorithm. We describe manual memory analysis in section 4.1 and automated memory analysis using SLAMM in section 4.2. Finally, in section 5, we demonstrate how SLAMM is used to increase the memory efficiency of the POP iterative solver and reduce execution time.

2. Background and related work. We propose a technique to simplify the design and implementation of memory-efficient software through automated memory analysis. Our automated memory analysis uses compiler techniques to evaluate an algorithm implemented in MATLAB. The SLAMM language processor inputs MATLAB code marked with SLAMM directives and outputs the same code with additional code blocks to perform memory analysis. Our work differs from the compilers described in [1, 10, 14, 23] because we do not convert MATLAB to another language but rather analyze the code to determine its potential memory efficiency before implementation in a compiled language.

We focus our efforts on improving the execution rate of linear algebra algorithms that are sensitive to the amount of bandwidth between the memory hierarchy and the CPU. Bandwidth is the rate that data is moved through the memory hierarchy in units of cache-lines. We quantify the impact of the memory hierarchy bandwidth on a particular algorithm by comparing the total storage requirement of an algorithm as well as its working set size to the size of the cache. The total storage requirement is the total size in bytes of all variables required by an algorithm. The working set size is the size in bytes of all variables accessed during a particular section or phase of the algorithm. Different implementations of the same algorithm may have the same total storage requirement but may have significantly different working set sizes. The results of matching an algorithm's working set size to cache size can result in significant performance gains. For example, loop blocking [5, 17] of dense linear algebra, which breaks up larger code blocks or loops into smaller blocks in order to match working set size and cache size, is used in the optimized level 2 and 3 BLAS routines [11].

Several researchers have used compiler techniques to predict serial and parallel performance. Cascaval et al. [6] use the Polaris compiler to predict the performance of serial Fortran and C codes. Polaris-based predictions for execution time and cache miss-rates, which are fed back into the compiler to improve code optimization, are accurate to within an average of 20% error. Van Gemund [31] uses the PAMELA compiler to add symbolic cost expressions for data parallel problems, which, in combination with a simple machine model, allow it to predict execution time to within

10% error. Fahringer [12] developed the Parameter-based Performance Prediction Tool (P^3T) which is integrated with the Vienna Fortran compiler [3].

Unlike these previous approaches [6, 12, 31], SLAMM does not analyze Fortran or C source code but rather analyzes MATLAB code. The use of MATLAB allows the analysis of an algorithm before implementation in a compiled language. Analyzing an algorithm written in an interpreted high level language like MATLAB avoids the bias of the implementation details present in a Fortran or C version. The SPIRAL project [22] has a similar basis. Recognizing that information is lost when an algorithm is implemented in Fortran or C, SPIRAL generates optimized code based on mathematical descriptions of FFT and DFT algorithms. The use of a high level description of an algorithm allows SLAMM to accurately predict the minimum possible data movement needed by an algorithm and compares it to the existing data movement. We do not develop a sophisticated analytical model of memory performance because the consumer of SLAMM's analysis is the developer of the algorithm. While it is possible to study the amount of memory that is loaded between the multiple components of the memory hierarchy, we find that a simple model, which is described in section 4, accurately correlates to the amount of data loaded from L2 to L1 cache ($Mbytes_{L1}$) and is sufficient to provide valuable insight into the algorithm developer.

The performance of POP has been analyzed by several groups. Kerbyson and Jones [16] develop an analytical model of expected parallel performance. Snavely et al. [28] employ a convolution-based framework that uses hardware performance counts and MPI library traces to predict parallel performance. Our work does not specifically address parallel performance but rather concentrates on reducing data movement for a single component of the POP model.

3. Parallel Ocean Program. The Parallel Ocean Program (POP) [27] is a global ocean model developed at Los Alamos National Laboratory that is used extensively as the ocean component of the Community Climate System Model [7]. POP uses finite-difference for the baroclinic component of the timestep and a preconditioned conjugate gradient solver to solve for surface pressure in the barotropic component. Parallelism on distributed memory computers is supported through the Message Passing Interface (MPI) [29] standard.

We use POP version 2.0.1 [20] to examine data movement in the barotropic solver using the *test* and $gx1v3$ grids. The test grid is a coarse grid provided with the source code to facilitate the porting of POP to other compute platforms. The gx1v3 grid is higher resolution than the test grid with one degree resolution at the equator. POP using the gx1v3 grid represents 20% of the total compute cycles at the National Center for Atmospheric Research (NCAR) [4].

3.1. Existing data structure. POP uses a three-dimensional (3D) computational mesh. The horizontal dimensions are decomposed into logically rectangular 2D blocks [15]. The computational mesh is distributed across multiple processors by placing one or more 2D blocks on each processor. There are km vertical levels associated with each 2D horizontal block. Blocks that do not contain any ocean points, or land blocks, are eliminated. Table 1 contains the dimensions of the test and gx1v3 grids. The variables nx and ny are the global numbers of points in the x and y directions of the grid, respectively. Table 1 also contains the block sizes $bsize_x$ and $bsize_y$, the number of blocks $nblocks$, and the numbers of ocean, land, and total points in the barotropic solver for several block size configurations. The block size parameters $bsize_x$ and $bsize_y$ correspond to the number of points in the x and y directions for each 2D block. Note that land block elimination reduces the number of 20×24

TABLE 1
Description of POP grids.

Name	Grid $nx \times ny \times km$	$bsize_x \times bsize_y$	*nblocks*	# points Ocean	Land	Total
test	$192 \times 128 \times 20$	16×16	92	15378	8174	23552
gx1v3	$320 \times 384 \times 40$	40×48	64	86354	36526	122880
		20×24	246	86354	31726	118080

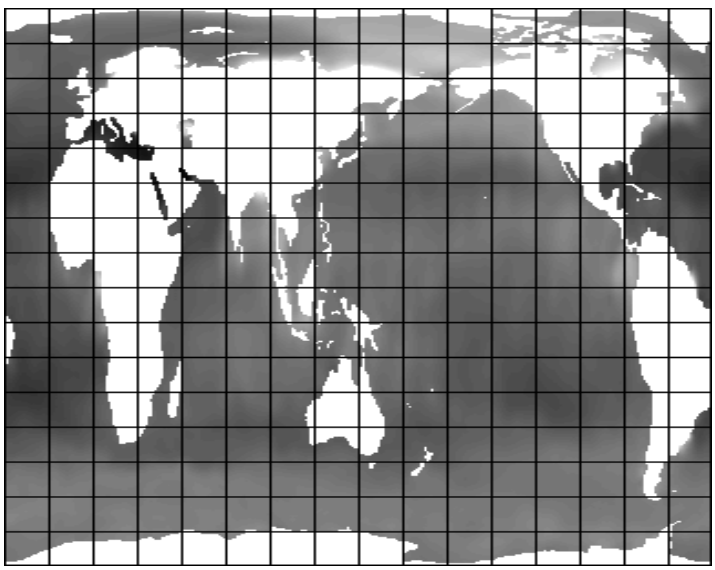


FIG. 1. *The POP gx1v3 grid, where white corresponds to land points, gray to ocean points, and superimposed lines indicate* 20×24 *blocks. Reprinted from* [9] *with permission from IEEE.* ©2007 *IEEE.*

blocks to 246, resulting in a reduction in the number of land points versus the 40×48 configuration.

Figure 1 illustrates the gx1v3 grid decomposed into 2D blocks of size 20×24. The white area in Figure 1 represents land, while ocean is indicated by gray. The black lines in Figure 1 indicate the boundaries of the 20×24 blocks. We next describe the form of the 2D data structure in greater detail.

Each 2D block contains $bsize_x \times bsize_y$ internal grid points and a halo region of width *nghost*. The halo region is used to store grid points that are located in neighboring 2D blocks. Figure 2 illustrates the 2D data structure within the source code for the block from Figure 1 containing the Iberian peninsula. Note that the large white structure on the upper right side of the block in Figure 2 is the Iberian peninsula, while the white structure in the lower right side is the northern coast of Africa. For this particular block, only about 60% of the grid points represent ocean.

We next describe the form of matrix-vector multiply within the conjugate gradient solver. The matrix-vector multiply within the conjugate gradient solver is applied via a 9-point stencil. The code for the 9-point stencil is illustrated in Figure 3. To calculate point $Y(i,j)$, the nine points $X(i,j)$, $X(i,j+1)$, $X(i,j-1)$, $X(i+1,j)$,

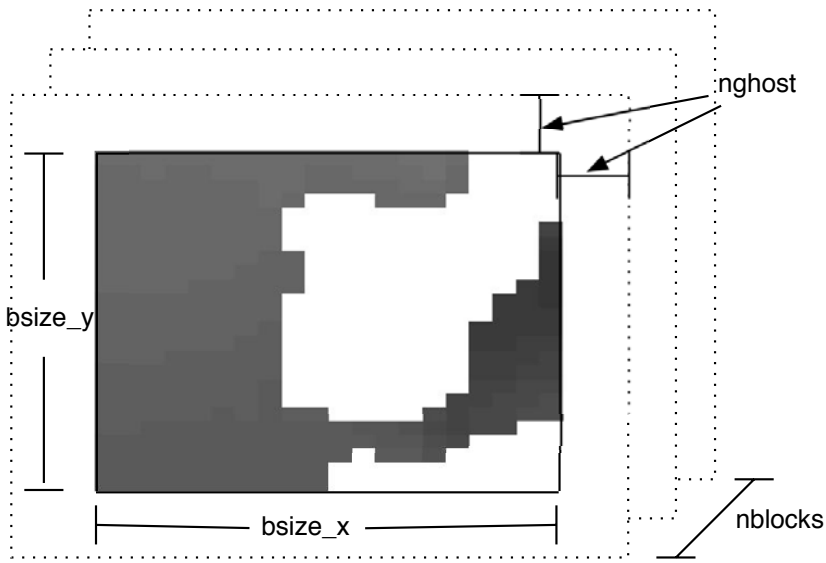


FIG. 2. *A* 20×24 *block of the* $gx1v3$ *grid over the Iberian peninsula. Note the presence of land and ocean points within the block.*

```
do j=this_block%jb,this_block%je
do i=this_block%ib,this_block%ie
   Y(i,j) = A0 (i  ,j  )*X(i  ,j  ) + &
            AN (i  ,j  )*X(i  ,j+1) + &
            AN (i  ,j-1)*X(i  ,j-1) + &
            AE (i  ,j  )*X(i+1,j  ) + &
            AE (i-1,j  )*X(i-1,j  ) + &
            ANE(i  ,j  )*X(i+1,j+1) + &
            ANE(i  ,j-1)*X(i+1,j-1) + &
            ANE(i-1,j  )*X(i-1,j+1) + &
            ANE(i-1,j-1)*X(i-1,j-1)
end do
end do
```

FIG. 3. *The barotropic operator in POP, implemented as a* 9*-point stencil using the* 2*D data structure.*

$X(i-1,j)$, $X(i+1,j+1)$, $X(i+1,j-1)$, $X(i-1,j+1)$, and $X(i-1,j-1)$ are required along with the associated coefficient arrays $A0$, AN, AE, and ANE. The need for a halo region is apparent if we consider the calculation of the corner point $Y(1,1)$. We need among other points $X(0,0)$, which is contained within the halo region. The primary advantage of the 2D data structure is that it provides a regular stride-one access for the matrix-vector multiply. The disadvantage of the 2D data structure is that it includes a large number of grid points that represent land. In effect, a number of explicitly stored zeros are added to the matrix. While it is possible to reduce the number of excessive land points by reducing the block size, smaller blocks have a larger percentage of their total size dedicated to halo points. We present an alternative to the 2D data structure in the next section.

```
is = Mat%Ia(1)
do i=1,n2
  ie = Mat%Ia(i+1)
  tmp = 0.0
     do j=is,ie-1
        tmp = tmp + Mat%A(j)*X(Mat%Ja(j))
     enddo
  Y(i) = tmp
  is = ie
enddo
```

FIG. 4. *The barotropic operator in POP, implemented as compressed sparse row matrix-vector multiply using the* 1*D data structure.*

3.2. An alternate data structure. We next describe a 1D data structure that enables the elimination of a potentially large number of land points present in the 2D data structure. The 1D data structure consists of a 1D array of extent n. The first $nActive$ elements in the 1D array correspond to active ocean points, while the remaining $n - nActive$ points correspond to the off-processor halo needed by the 9-point stencil. In removing the excessive land points, the 1D data structure changes the form of the matrix-vector multiply. Indirect-addressing is now necessary to apply the operator matrix which is stored in a compressed sparse row format. The code for the matrix-vector multiply for the 1D data structure is provided in Figure 4. The need for the $Mat\%Ia$ and $Mat\%Ja$ index arrays has the potential to negate any reduction in data movement achieved by the elimination of land points. We demonstrate in the next section how memory analysis is used to evaluate the impact of alternative data structures on data movement.

4. Memory analysis. We next examine the process of memory analysis. Memory analysis examines the amount of data that must be loaded from the memory hierarchy to the CPU to perform an operation. Consider two fundamental linear algebra operations, a dot product and an AXPY. The dot product of two vectors $\alpha = u'v$, where $v, u \in \mathbb{R}^n$ and $\alpha \in \mathbb{R}$, requires two vectors of length n. The amount of data that must be loaded from the memory hierarchy, or working set load size (WSL), is

$$WSL^{DOT} = 2\,n\,L_d, \tag{1}$$

where L_d is the size of a double precision floating-point value in bytes. The AXPY operation $u = u + \alpha v$, where $v, u \in \mathbb{R}^n$ and $\alpha \in \mathbb{R}$, requires two vectors of length n and a scalar. The working set size for an AXPY operation is

$$WSL^{AXPY} = (2\,n + 1)\,L_d. \tag{2}$$

The derivation of (1) and (2) is an example of manual memory analysis. While manual memory analysis is trivial for simple linear algebra operations, its complexity grows if we extend the technique to an entire iterative algorithm. We describe manual memory analysis of a complete iterative algorithm in section 4.1, followed by our automated memory analysis procedure in section 4.2.

4.1. Manual memory analysis. We focus on the preconditioned conjugate gradient iterative solver used by POP [27] to update the ocean surface pressure at

```
1. r_0 = b − Ax_0, p_0 = 0, φ_0 = (r_0, r_0)
2. for j = 0, 1, ... until convergence
3.         Mz = r_j                      /* Apply Precon */
4.         φ_{j+1} = (r_j, z)            /* dot product */
5.         β = φ_{j+1}/φ_j
6.         p_{j+1} = z + βp_j            /* AXPY */
7.         q = Ap_{j+1}                  /* MxV */
8.         δ = (p_{j+1}, q)              /* dot product */
9.         α = φ_{j+1}/δ
10.        x_{j+1} = x_j + αp_{j+1}      /* AXPY */
11.        r_{j+1} = r_j − αp_{j+1}      /* AXPY */
12. end
```

FIG. 5. *Preconditioned conjugate gradient algorithm with two inner products (PCG2).*

TABLE 2

Working set sizes (WSL) for each line of the preconditioned conjugate gradient algorithm in Figure 5.

Line	Operation	WSL (bytes)
3	Precon	varies
4	$DOT()$	$2\,n\,L_d$
6	$AXPY()$	$(2\,n+1)\,L_d$
7	$MxV()$	varies
8	$DOT()$	$2\,n\,L_d$
10	$AXPY()$	$(2\,n+1)\,L_d$
11	$AXPY()$	$(2\,n+1)\,L_d$
2–12	TOTAL	$(10\,n+3)\,L_d$
		$+\,WSL^{MxV}$
	for 1 iteration	$+\,WSL^{Precon}$

each timestep. In Figure 5, we provide a common form of the preconditioned conjugate gradient solver which uses two products (PCG2). Iterative solvers are composed of basic linear algebra operations. Thus, we compute the working set load size of the solver from the working set load size of its constituent operations. In Table 2, we provide the working set sizes for each line of the PCG2 algorithm in Figure 5. Note that the PCG2 algorithm requires several dot products (DOT) and AXPY operations (AXPY), a single sparse matrix-vector product (MxV), and the application of the preconditioner (Precon) for each iteration.

We first determine the working set load size for the sparse matrix-vector multiply for the 2D and 1D data structures followed by the application of the preconditioner. Consider the matrix-vector multiply for the 2D data structure. The working set load size of a matrix-vector multiply $y = Ax$, where A is a sparse coefficient matrix, is

$$WSL = sizeof(y) + sizeof(x) + sizeof(A), \tag{3}$$

where $sizeof()$ is a function that returns the size of its argument in bytes. An examination of Figure 3, which contains the code for the matrix operator for the 2D data structure, indicates that arrays $A0$, AN, AE, and ANE form the coefficient matrix A. Because $sizeof(A) = 4 \cdot sizeof(x)$, we use (3) to calculate working set load size for the 2D matrix-vector product

$$WSL_{2D}^{MxV} \equiv (2\,n_{2D} + 4\,n_{2D})\,L_d = 6\,n_{2D}\,L_d, \tag{4}$$

```
%SLM start caseStudy;              % line 1
n=200; u=rand(n,1); v=rand(n,1);   % line 2
%SLM start dotprod;                % line 3
alpha=u'*v                         % line 4
%SLM end dotprod;                  % line 5
%SLM print dotprod;                % line 6
%SLM end caseStudy;                % line 7
```

FIG. 6. *MATLAB code that computes the dot product of two vectors with SLAMM directives.*

where

$$n_{2D} = nblocks \cdot (bsize_x + 2 \cdot nghost)\,(bsize_y + 2 \cdot nghost).$$

The working set load size for the matrix-vector multiply for the 1D data structure is different than for the 2D data structure because $sizeof(A)$ is different. In the case of the 1D data structure, we must account for both the number of nonzero entries and the size of the indirect-addressing arrays. The WSL_{1D}^{MxV} is therefore

$$WSL_{1D}^{MxV} \equiv (nnz_{1D} + 2\,n_{1D})\,L_d + (nnz_{1D} + n_{1D})\,L_i, \tag{5}$$

where n_{1D} is the number of ocean points from column 5 of Table 1, nnz_{1D} is the number of nonzero values in the matrix $Mat\%A$ in Figure 4, and L_i is the size of an integer in bytes.

We next calculate the working set load size for the application of the preconditioner. Because POP uses a scaled diagonal preconditioner,

$$WSL^{Precon} = 2\,n\,L_d. \tag{6}$$

Using (4), (5), and (6) and the equations provided in Table 2 we can calculate the working set load sizes for both the 2D- and 1D-based solvers. Deriving and evaluating the analytical expression for data movement in this way is a time-consuming, laborious, and error-prone process that can be automated using language processing techniques. In the next section we describe the automated memory analysis procedure, which greatly simplifies memory analysis.

4.2. Automated memory analysis. We next describe how the Sparse Linear Algebra Memory Model (SLAMM) language processor is used to automate the memory analysis procedure. We use the dot product described in section 4 as an example. Figure 6 contains MATLAB code that calculates the dot product of two vectors $u, v \in \mathbb{R}^n$. Note that lines 2 and 4 in Figure 6 are the MATLAB code to initialize and calculate the dot product, respectively. The remaining lines in Figure 6 with the prefix $\%SLM$ are directives that control the SLAMM language processor. Lines 1 and 7 delineate the beginning and end of a body of MATLAB code. Lines 3 and 5 also delineate a body of code and identify it with the symbolic name *dotprod*. Line 6 is a SLAMM directive requesting the printing of the memory analysis of the dotprod code block.

Based on the input MATLAB code in Figure 6, the SLAMM language processor analyzes and generates modified output code which contains both the original code and new blocks of code that calculate memory usage properties. Figure 7 shows the resulting automated memory analysis of code body dotprod from Figure 6. Note that the total storage requirement (SR) and the working set load size (WSL) are printed

```
SLAMM Memory Analysis for Body: dotprod
       TOTAL: Storage Requirement  Kbytes (SR) : 3.13
       TOTAL: Loaded from L2 -> L1 Kbytes (WSL): 3.12 +- 0.00
                           DGEMM    Kbytes      : 0.00 +- 0.00
                     Sparse Ops     Kbytes      : 0.00 +- 0.00
```

FIG. 7. *The SLAMM generated predictions for dot product code in Figure* 6.

in Kbytes. SLAMM separately indicates the component of data movement due to dense matrix-matrix multiplication, indicated by DGEMM, as well as sparse matrix operations. We differentiate between the different components of WSL to indicate the fraction of the algorithm that might benefit from specific optimization techniques like the use of an optimized DGEMM subroutine. Fundamentally, the SLAMM language processor derives and evaluates (1) automatically. While automated memory analysis is certainly not necessary for a simple operation like a dot product, its advantages become apparent when it is applied to a complete iterative algorithm that may contain a large number of linear algebra operations. We demonstrate the application of automated memory analysis to evaluate the memory efficiency of the POP preconditioned conjugate gradient solver in the next section.

5. Results. Automated memory analysis can be used to evaluate a design decision before implementation in a compiled language and to evaluate the quality of a particular implementation. We first describe the methodology and compute platforms used to validate the accuracy of the SLAMM-based predictions. We instrumented each version of the solvers with a locally developed performance profiling library (Htrace), which is based on the PAPI [19] hardware performance counter API. Htrace calculates data movement by tracking the number of cache-lines moved through the different components of the memory hierarchy. We focus on three primary microprocessor compute platforms that provide counters for cache-lines loaded from the memory hierarchy to the L1 cache: Sun Ultra II [30] (Ultra II), IBM POWER 4 [2] (POWER4), and MIPS R14K [32] (R14K). A description of the cache configuration for each compute platform is provided in Table 3.

TABLE 3
Description of the microprocessor compute platforms and their cache configurations.

CPU	Ultra II	POWER4	R14K
Company	SUN	IBM	SGI
MHz	400	1300	500
L1 data-cache	32KB	32KB	32KB
L2 cache	4 MB	1440 KB	8 MB
L3 cache	–	32 MB	–

We first demonstrate how automated memory analysis is used to evaluate the choice of data structure for POP. We implemented the PCG2 algorithm of Figure 5 in MATLAB using both the 2D and 1D data structures described in sections 3.1 and 3.2, respectively. Using the SLAMM directives described in the previous section, we delineate one iteration of each of the PCG2 algorithms. The predicted data movement (WSL_P) for the 2D data structure (PCG2+2D) and the 1D data structure (PCG2+1D) versions for the test grid are provided in the second column of Table 4. SLAMM predicts that the use of the 1D data structure would reduce the amount of data loaded from the L2 to the L1 cache ($Mbytes_{L1}$) by 34% versus the existing 2D

data structure.

TABLE 4

$Mbytes_{L1}$ for a single iteration of the preconditioned conjugate gradient solver in POP using the test grid. The values of WSL_P and WSL_M are in Kbytes.

Solver implementation	WSL_P	vers.	Ultra II		POWER4		R14K	
			WSL_M	error	WSL_M	error	WSL_M	error
PCG2+2D	4902	v1	5163	5.3%	5068	3.4%	5728	16.9%
		v2	4905	0.1%	4865	−0.7%	4854	−1.0%
PCG2+1D	3218		3164	−1.7%	3335	3.7%	3473	7.9%
Reduction	34%		39%		34%		39%	

The measured data movement (WSL_M) or $Mbytes_{L1}$ the amount of data loaded from the L2 to the L1 cache for the existing 2D data structure implementation (PCG2+2D v1), an optimized 2D version (PCG2+2D v2), and the 1D version are provided for each of the compute platforms in Table 4. We measure 10 iterations and report the average measured data movement. While the discrepancies between the measured and predicted working set load sizes for the PCS2+2D v1 solver are minimal for both the Ultra II and POWER4 platforms, the measured value of 5728 Kbytes for the R14K is 17% greater than the predicted value of 4902 Kbytes. The difference in data movement between the three compute platforms may be due to additional code transformations performed by the Ultra II and POWER4 compilers. That the PCG2+2D v1 solver is loading 17% more data from the memory hierarchy than necessary on the R14K is an indication that it is possible to improve the quality of the implementation.

An examination of the source code for the PCG2+2D v1 solver indicates that a minor change to the dot product calculation reduces data movement. Code blocks that correspond to lines 6 to 8 of the PCG2 algorithm in Figure 5 for the v1 and v2 versions of the solver are provided in Figure 8. The function *operator* applies the 9-point stencil from Figure 3, and the array *LMASK* is an array that masks out points that correspond to land points. In version v1 of the do loop, a temporary array WORK0 is created that contains the pointwise product of two vectors Q and P. Outside the do loop, the dot product of LMASK and the WORK0 array is calculated by *global_sum*. If the size of data accessed in the do loop is larger than the L1 cache, then a piece of the WORK0 array at the end of the do loop is no longer located in the L1 cache and must be reloaded to complete the calculation.

In version v2 of the do loop, in Figure 8, a scalar temporary *delta_local* is added to accumulate each block's contribution to the dot product of Q and P. An additional function *local_sum* is used that applies the land mask to complete the dot product. Finally, we replace the function global_sum with a call to *gsum*. The subroutine gsum, when executed on a single processor, is an assignment of delta_local to delta. Because version v2 of the code block does not access WORK0 outside the do loop, it potentially reduces data movement.

Both dot product calculations (lines 4 and 8 of Figure 5) in the PCG2 algorithm were rewritten to create the v2 solver. The WSL_M values in Table 4 for the v1 and v2 versions of the PCG2+2D solver on the R14K indicate that the rearranged dot product calculations reduce data movement by 18%. Note that data movement is also reduced on the Ultra II and POWER4 platforms but to a lesser extent.

The previous description is a demonstration of how SLAMM is used to guide performance tuning. Table 4 also provides the relative error between predicted and measured data movement for the solvers. SLAMM predicts data movement to within an

```
! ================================
! code block: solver v1
! ================================
do iblock=1,nblocks
     P(:,:,iblock) = Z(:,:,iblock) + P(:,:,iblock)*beta
     Q = operator(P,iblock)
     WORK0(:,:,iblock) = Q(:,:,iblock)*P(:,:,iblock)
enddo
delta=global_sum(WORK0,LMASK)

! ================================
! code block: solver v2
! ================================
delta_local=0.d0
do iblock=1,nblocks
     P(:,:,iblock) = Z(:,:,iblock) + P(:,:,iblock)*beta
     Q = operator(P,iblock)
     WORK0 = Q(:,:,iblock)*P(:,:,iblock)
     delta_local = delta_local + local_sum(WORK0,LMASK(:,:,iblock))
enddo
delta=gsum(delta_local)
```

FIG. 8. *A code block that implements lines* 6 *to* 8 *of the preconditioned conjugate gradient algorithm in Figure* 5 *for the* $v1$ *and* $v2$ *solvers.*

average error of 0.6% for the PCG2+2D v2 solver and within 4.4% for the PCG2+1D solver. We provide the actual percentage reductions in data movement for the 1D versus 2D solver for each compute platform. Table 4 indicates that the actual percentage reductions in data movement are very similar to the predicted reductions. Table 4 clearly demonstrates that it is possible for automated memory analysis to accurately predict the amount of data movement required for an algorithm. Such a capability provides a priori knowledge of the memory efficiency of a particular design choice before implementation in a compiled language.

We next examine the impact of reducing data movement on execution time. The timestep of POP includes a baroclinic and a barotropic component. The barotropic component is composed of a single linear solver for surface pressure. We execute POP using the test grid on a single processor of each compute platform for a total of 20 timesteps. This configuration requires an average of 69 iterations of the PCG2 algorithm per timestep. In Table 5, we provide the barotropic execution time in seconds using the three implementations of the solver. We include the execution time for the initial implementation of the solver using 2D data structures to accurately reflect the overall impact automated memory analysis has on execution time. Note that the PCG1+1D solver consistently has a lower execution time than either of the 2D data structure–based solvers. The last row in Table 5 contains the percentage reduction in barotropic execution time for the PCG2+2D v1 versus the PCG2+1D solver. Table 5 indicates that code modifications either evaluated or identified by automated memory analysis reduce execution time by an average of 46%. Curiously, a comparison of Tables 4 and 5 indicates that the percentage reduction in execution time is even larger than the percentage reduction in data movement. The reason for the discrepancy is unclear.

We next examine the impact of the 1D data structures on parallel execution time

TABLE 5
Barotropic execution time for 20 *timesteps of POP in seconds using the test grid on a single processor.*

Solver implementation	Ultra II	POWER4	R14K
PCG2+2D v1	21.17	4.57	8.58
PCG2+2D v2	20.49	4.01	7.97
PCG2+1D	12.74	2.11	4.61
Reduction	39%	54%	46%

TABLE 6
Total execution time for 200 *timesteps with the gx*1*v*3 *grid on* 64 *POWER*4 *processors.*

	Solver implementation			
	PCG2+2D v1	PCG1+2D v1	PCG2+1D	PCG1+1D
Total time (sec)	86.2	81.5	78.8	73.9

using the gx1v3 grid. We wrote a generalized gather-scatter routine to provide parallel execution under MPI for the 1D version of the solver. Recently, an alternative preconditioned conjugate gradient algorithm which uses a single dot product (PCG1) [8] was added to POP. The PCG1 algorithm provides a performance advantage for parallel execution because it eliminates one of the distributed dot product calculations. Because we want to realistically estimate the impact the 1D data structure has on execution time, we provide timing results for both PCG2 and PCG1 algorithms using the 2D and 1D data structures in Table 6.

We configure POP such that the computational costs of a production run are accurately characterized. We measure a total of 200 timesteps on 64 IBM POWER4 processors, with an average of 151 iterations per timestep. The execution time for the time stepping loop in seconds for the four solver implementations is provided in Table 6. These results indicate that use of the PCG1+1D solver versus the PCG1+2D solver reduces total POP execution time by 9%. A 9% reduction in total execution time of POP is significant because it has been extensively studied and optimized [15, 28]. Further, POP consumes approximately 2.4 million CPU hours every year at NCAR. A 9% reduction in execution time eliminates the need for 216,000 CPU hours per year.

6. Conclusion. We demonstrate how automated memory analysis is used to performance tune the Parallel Ocean Program. We automate memory analysis using the SLAMM language processor, which accurately predicts the amount of data loaded from the L2 cache to the L1 cache to within a relative error of 10%. The accurate predictive capability of SLAMM enables the identification of excessive data movement in the existing 2D data structure–based solver, and it allows us to evaluate the use of an alternative 1D data structure. We observe that the reduction in data movement predicted by SLAMM for the 1D versus the 2D solver is 34% and compares favorably with a measured 36% reduction. The reduction in data movement results in a similar reduction in single processor execution time. Finally we demonstrate how use of the 1D data structure reduces total execution time of POP on 64 processors using a one degree resolution grid by 9%, which represents a substantial savings in compute time.

REFERENCES

[1] P. BANERJEE, N. SHENOY, A. CHOUDHARY, S. HAUCK, C. BACHMANN, M. HALDAR, P. JOISHA, A. JONES, A. KANHARE, A. NAYAK, S. PEIRIYACHERI, M. WALKEN, AND D. ZARETSKY,

A MATLAB compiler for distributed, heterogeneous, reconfigurable computing systems, in Proceedings of the International Symposium on FPGA Custom Computing Machines (FCCM-2000), Napa Valley, CA, 2000, pp. 39–48.

[2] S. Behling, R. Bell, P. Farrell, H. Holthoff, F. O'Connell, and W. Weir, *The POWER4 Processor Introduction and Tuning Guide*, IBM Redbooks, 2001; available online from www.redbooks.ibm.com/redbooks/pdfs/sg247041.pdf.

[3] S. Benkner, *VFC: The Vienna Fortran compiler*, Scientific Programming, 7 (1999), pp. 67–81.

[4] T. Bettge, *Personal communication*, 2006.

[5] S. Carr and K. Kennedy, *Blocking linear algebra codes for memory hierarchies*, in Proceedings of the Fourth SIAM Conference on Parallel Processing for Scientific Computing, SIAM, Philadelphia, 1990, pp. 400–405.

[6] C. Cascaval, L. DeRose, D. A. Padua, and D. A. Reed, *Compile-time based performance prediction*, in LCPC '99: Proceedings of the 12th International Workshop on Languages and Compilers for Parallel Computing, Springer-Verlag, London, 2000, pp. 365–379.

[7] W. D. Collins, C. M. Bitz, M. L. Blackmon, G. B. Bonan, C. S. Bretherton, J. A. Carton, P. Chang, S. C. Doney, J. J. Hack, T. B. Henderson, J. T. Kiehl, W. G. Large, D. S. McKenna, B. D. Santer, and R. D. Smith, *The community climate system model: CCSM*3, J. Climate, CCSM Special Issue, 11 (2005), pp. 2122–2143.

[8] E. F. D'Azevedo, V. L. Eijkhout, and C. H. Romine, *LAPACK Working Note* 56: *Reducing Communication Costs in the Conjugate Gradient Algorithm on Distributed Memory Multiprocessor*, Report CS-93-185, University of Tennessee, Knoxville, 1993.

[9] J. M. Dennis, *Inverse space-filling curve partitioning of a global ocean model*, in Proceedings of the 21st IEEE International Parallel and Distributed Processing Symposium (Long Beach, CA, 2007), CD-ROM, IEEE, Piscataway, NJ, 2007.

[10] L. DeRose and D. Padua, *Techniques for the translation of Matlab programs into Fortran* 90, ACM Trans. Programming Languages Systems, 21 (1999), pp. 286–323.

[11] J. Dongarra, J. DuCroz, S. Hammarling, and I. Duff, *Algorithm* 679: *A set of level* 3 *Basic Linear Algebra Subprograms*, ACM Trans. Math. Software, 16 (1990), pp. 18–28.

[12] T. Fahringer, *Estimating Cache Performance for Sequential and Data Parallel Programs*, Tech. Report TR 97-9, Institute for Software Technology and Parallel Systems, University of Vienna, Vienna, Austria, 1997.

[13] M. R. Field, *Optimizing a parallel conjugate gradient solver*, SIAM J. Sci. Comput., 19 (1998), pp. 27–37.

[14] P. G. Joisha and P. Banerjee, *Static array storage optimization in Matlab*, in ACM SIGPLAN Conference on Programming Language Design and Implementation, San Diego, CA, 2003, pp. 258–268.

[15] P. W. Jones, P. H. Worley, Y. Yoshida, J. B. White, III, and J. Levesque, *Practical performance portability in the Parallel Ocean Program (POP)*, Concurrency Comput. Prac. Exper., 17 (2005), pp. 1317–1327.

[16] D. J. Kerbyson and P. W. Jones, *A performance model of the Parallel Ocean Program*, Internat. J. High Performance Comput. Appl., 19 (2005), pp. 261–276.

[17] M. S. Lam, E. E. Rothberg, and M. E. Wolf, *The cache performance and optimizations of blocked algorithms*, SIGOPS Oper. Syst. Rev., 25 (1991), pp. 63–74.

[18] D. P. O'Leary, *The block conjugate gradient algorithm and related methods*, Linear Algebra Appl., 29 (1980), pp. 293–322.

[19] *PAPI: Performance Application Programming Interface: User's Guide*; available online from http://icl.cs.utk.edu/papi, 2005.

[20] *The Parallel Ocean Program (POP)*; available online from http://climate.lanl.gov/Models/POP, 2006.

[21] D. Patterson, T. Anderson, N. Cardwell, R. Fromm, K. Keeton, C. Kozyrakis, R. Thomas, and K. Yellick, *A case for intelligent RAM*, IEEE Micro, 17 (2) (1997), pp. 34–44.

[22] M. Püschel, J. M. F. Moura, J. Johnson, D. Padua, M. Veloso, B. W. Singer, J. Xiong, F. Franchetti, A. Gačić, Y. Voronenko, K. Chen, R. W. Johnson, and N. Rizzolo, *SPIRAL: Code generation for DSP transforms*, Proc. IEEE, 93 (2005), pp. 232–275.

[23] M. J. Quinn, A. Malishevsky, N. Seelam, and Y Zhao, *Preliminary results from a parallel Matlab compiler*, in Proceedings of the International Parallel and Distributed Processing Symposium, IEEE Computer Society, Orlando, FL, 1998, pp. 81–87.

[24] Y. Saad, *Krylov subspace methods for solving large unsymmetric linear systems*, Math. Comp., 37 (1981), pp. 105–126.

[25] H. D. Simon, *The Lanczos Algorithm for Solving Symmetric Linear Systems*, Ph.D. thesis, University of California, Berkeley, CA, 1982.

[26] H. D. Simon and A. Yeremin, *A new approach to construction of efficient iterative schemes for massively parallel applications: Variable block CG and BiCG methods and variable block Arnoldi procedure*, in Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing, SIAM, Philadelphia, 1993, pp. 57–60.

[27] R. D. Smith, J. K. Dukowicz, and R. C. Malone, *Parallel ocean general circulation modeling*, Phys. D, 60 (1992), pp. 38–61.

[28] A. Snavely, X. Gao, C. Lee, L. Carrington, N. Wolter, J. Labarta, J. Gimenez, and P. Jones, *Performance modeling of HPC applications*, in Parallel Computing (PARCO2003), Elsevier, Dresden, Germany, 2003, pp. 777–784.

[29] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra, *MPI: The Complete Reference: Volume* 1*, The MPI Core*, MIT Press, Cambridge, MA, 2000.

[30] Sun Microsystems, *The Ultra*2 *Architecture: Technical White Paper*; available online from http://pennsun.essc.psu.edu/customerweb/WhitePapers/, 2005.

[31] A. J. C. van Gemund, *Symbolic performance modeling of parallel systems*, IEEE Trans. Parallel Distrib. Systems, 14 (2003), pp. 154–165.

[32] S. Vellas, *Scalar Code Optimization* I; available online from http://sc.tamu.edu/help/origins/sgi_scalar_r14k_opt.pdf, 2005.

SIAM J. SCI. COMPUT.
Vol. 29, No. 5, pp. 2224–2239
© 2007 Society for Industrial and Applied Mathematics

BLOCK LOCALLY OPTIMAL PRECONDITIONED EIGENVALUE XOLVERS (BLOPEX) IN HYPRE AND PETSC*

A. V. KNYAZEV†, M. E. ARGENTATI†, I. LASHUK†, AND E. E. OVTCHINNIKOV‡

Abstract. We describe our software package Block Locally Optimal Preconditioned Eigenvalue Xolvers (BLOPEX) recently publicly released. BLOPEX is available as a stand-alone serial library, as an external package to PETSc (Portable, Extensible Toolkit for Scientific Computation, a general purpose suite of tools developed by Argonne National Laboratory for the scalable solution of partial differential equations and related problems), and is also built into *hypre* (High Performance Preconditioners, a scalable linear solvers package developed by Lawrence Livermore National Laboratory). The present BLOPEX release includes only one solver—the Locally Optimal Block Preconditioned Conjugate Gradient (LOBPCG) method for symmetric eigenvalue problems. *hypre* provides users with advanced high-quality parallel multigrid preconditioners for linear systems. With BLOPEX, the same preconditioners can now be efficiently used for symmetric eigenvalue problems. PETSc facilitates the integration of independently developed application modules, with strict attention to component interoperability, and makes BLOPEX extremely easy to compile and use with preconditioners that are available via PETSc. We present the LOBPCG algorithm in BLOPEX for *hypre* and PETSc. We demonstrate numerically the scalability of BLOPEX by testing it on a number of distributed and shared memory parallel systems, including a Beowulf system, SUN Fire 880, an AMD dual-core Opteron workstation, and IBM BlueGene/L supercomputer, using PETSc domain decomposition and *hypre* multigrid preconditioning. We test BLOPEX on a model problem, the standard 7-point finite-difference approximation of the 3-D Laplacian, with the problem size in the range of 10^5–10^8.

Key words. conjugate gradient, iterative method, preconditioning, multigrid, domain decomposition, parallel computing, eigenvalue, LOBPCG, BLOPEX, *hypre*, PETSc, BlueGene, Beowulf

AMS subject classifications. 65F15, 65N25, 65N55, 65Y05

DOI. 10.1137/060661624

1. Introduction.

We describe a new software package, Block Locally Optimal Preconditioned Eigenvalue Xolvers (BLOPEX) revision 1.0, recently publicly released. BLOPEX is available as a stand-alone serial library, which can be downloaded from http://math.cudenver.edu/~aknyazev/software/BLOPEX/, as an external package to the Portable, Extensible Toolkit for Scientific Computation (PETSc) (see Balay et al. [2]), and is built into the High Performance Preconditioners (*hypre*) library (see Falgout, Jones, and Yang [8, 9]). Jose Roman has also written a Scalable Library for Eigenvalue Problem Computations (SLEPc) (see Hernandez, Roman, and Vidal [10]) interface to our *hypre* BLOPEX.

*Received by the editors May 31, 2006; accepted for publication (in revised form) July 2, 2007; published electronically September 28, 2007. Preliminary results of this work have been presented at the 11th and 12th Copper Mountain Conferences on Multigrid Methods and Preconditioning (2003), the SIAM Conference on Parallel Processing for Scientific Computing (2004), and the 16th International Conference on Domain Decomposition Methods (2005), and have appeared in Knyazev and Argentati [20] and Lashuk et al. [23].

http://www.siam.org/journals/sisc/29-5/66162.html

†Department of Mathematical Sciences, University of Colorado at Denver and Health Sciences Center, P.O. Box 173364, Campus Box 170, Denver, CO 80217-3364 (Andrew.Knyazev@cudenver.edu, http://math.cudenver.edu/~aknyazev/, Merico.Argentati@cudenver.edu, http://math.cudenver.edu/~rargenta/, Ilya.Lashuk@cudenver.edu). The work of these authors was partially supported by NSF awards CNS 0420985, DMS 0208773, and DMS 0612751 and by the Lawrence Livermore National Laboratory.

‡Harrow School of Computer Science, University of Westminster, London HA1 3TP, UK (Evgueni.Ovtchinnikov@na-net.ornl.gov).

The native *hypre* BLOPEX version takes advantage of powerful *hypre* multigrid preconditioners, both algebraic multigrid (AMG, called BoomerAMG in *hypre*) and geometric or structured multigrid (SMG). BLOPEX in PETSc gives the PETSc community easy access to the customizable code of a preconditioned eigensolver.

Currently, BLOPEX includes only one solver—the Locally Optimal Block Preconditioned Conjugate Gradient (LOBPCG) method (see Knyazev [17, 18]) for eigenproblems $Ax = \lambda Bx$ with large, possibly sparse, symmetric matrix A and symmetric positive definite matrix B. Preconditioned eigenvalue solvers in general (see Knyazev [16, 17]) and in particular the LOBPCG method have recently attracted attention as potential competitors to (block-) Lanczos methods. LOBPCG has been implemented using different computer languages in a number of other software packages such as: C++ in Anasazi Trilinos (see Arbenz et al. [1] and Heroux et al. [11]) and in NGSolve (see Zaglmayr [30, sections 7.2–7.4]); C in PRIMME (see Stathopoulos and McCombs [25]); FORTRAN 77 in PLTMG (see Bank [3]); FORTRAN 90 in ABINIT (see Bottin et al. [5]) and in PESCAN (see Tomov et al. [26]); and Python in SciPy by Robert Cimrman and in PYFEMax by Peter Arbenz and Roman Geus. LOBPCG has been independently tested in Yamada et al. [27, 28] for the Fermion–Habbard model on earth simulator CDIR/MPI; in Arbenz et al. [1] and Borzí and Borzí [4] using AMG preconditioning for vibration problems; and in Tomov et al. [26] and Yang, Meza, and Wang [29] for electronic structure calculations. Our *hypre* BLOPEX version has been used in Bramble, Kolev, and Pasciak [6] for the Maxwell problem.

Section 2 contains a complete and detailed description of the LOBPCG algorithm as implemented in BLOPEX. We discuss our specific abstract implementation approach in section 3. Parallel performance on distributed and shared memory systems using domain decomposition and multigrid preconditioning is discussed in section 4. Technical information, in the appendices, includes descriptions of variables, installation and testing instructions, a list of computers used for testing, and acronyms.

2. LOBPCG and its implementations. In subsection 2.1, we briefly describe the ideas behind the LOBPCG method, mostly following Knyazev [17, 18]. In subsection 2.2, we present a complete description of the LOBPCG algorithm as implemented in BLOPEX. Deflation by hard and soft locking is suggested in subsection 2.3.

2.1. LOBPCG ideas and description.

2.1.1. The problem and the assumptions. We consider the problem of computing the m smallest eigenvalues and the corresponding eigenvectors of the generalized eigenvalue problem $Ax = \lambda Bx$, with symmetric (Hermitian in the complex case) matrices A and B, the latter assumed to be positive definite. For a given approximate eigenvector x, we use the traditional Rayleigh quotient $\lambda(x) = (x, Ax)/(x, Bx)$ in the standard scalar product as an approximation to the corresponding eigenvalue. We emphasize that only matrix B is assumed to be positive definite. There is no such requirement on A; moreover, in all formulas in this section we can replace A with $A + \alpha B$ and the formulas are invariant with respect to a real shift α.

To accelerate the convergence, we introduce a preconditioner T, which is typically a function that for a given vector x produces Tx. The existing theory (see, e.g., [18, 21]) of the LOBPCG method is based on several assumptions: T needs to be linear, symmetric, and positive definite. Under these assumptions a specific convergence rate bound is proved in [18, 21]. Numerical tests suggest that *all these requirements on T are necessary* in order to guarantee this convergence rate. The use of preconditioning that does not satisfy the requirements above not only can slow down the convergence,

but may also lead to a breakdown of the method and severe instabilities in the code, resulting in incorrect results—however, this should not be mistakenly interpreted as bugs in the code.

2.1.2. Single-vector LOBPCG. For computing only the smallest eigenpair, i.e., if $m = 1$, the LOBPCG method takes the form of a 3-term recurrence:

$$\begin{array}{l} x^{(i+1)} = w^{(i)} + \tau^{(i)} x^{(i)} + \gamma^{(i)} x^{(i-1)}, \\ w^{(i)} = T(Ax^{(i)} - \lambda^{(i)} Bx^{(i)}),\ \lambda^{(i)} = \lambda(x^{(i)}) = (x^{(i)}, Ax^{(i)})/(Bx^{(i)}, x^{(i)}) \end{array} \tag{2.1}$$

with properly chosen scalar iteration parameters $\tau^{(i)}$ and $\gamma^{(i)}$. The easiest, and perhaps the most efficient, choice of parameters is based on an idea of *local optimality* (see Knyazev [14, 17, 18]), namely, select $\tau^{(i)}$ and $\gamma^{(i)}$ that minimize the Rayleigh quotient $\lambda(x^{(i+1)})$ by using the Rayleigh–Ritz procedure in the 3-D trial subspace spanning $x^{(i)}$, $w^{(i)}$, and $x^{(i-1)}$. We do not describe the Rayleigh–Ritz procedure using an arbitrary basis of the trial subspace here and assume that the reader is familiar with it.

If $\gamma^{(i)} = 0$, method (2.1) turns into the classical preconditioned locally optimal steepest descent method, so one can think of method (2.1) as an attempt to accelerate the steepest descent method by adding an extra term to the iterative recurrence; see, e.g., Knyazev [14, 17]. The general idea of accelerating an iterative sequence by involving the previous approximation is very natural and is well known. Let us note here that if instead of the previous approximation $x^{(i-1)}$ we use the previous preconditioned residual $w^{(i-1)}$ in formula (2.1), we get a different method that in practical tests does not converge as fast as method (2.1).

When implementing the Rayleigh–Ritz procedure in the trial subspace spanning $x^{(i)}$, $w^{(i)}$, and $x^{(i-1)}$, we need to be careful in choosing an appropriate basis that leads to reasonably conditioned Gram matrices. The current eigenvector approximation $x^{(i)}$ and the previous eigenvector approximation $x^{(i-1)}$ get closer to each other in the process of iterations, so special measures need to be taken in method (2.1) to overcome the potential numerical instability due to the round-off errors. Using both vectors $x^{(i)}$ and $x^{(i-1)}$ as basis vectors of the trial subspace leads to ill-conditioned Gram matrices, and the Rayleigh–Ritz method can produce spurious eigenpairs.

A simple fix is suggested in [18]: the 3-term recurrence that contains the current eigenvector approximation, the preconditioned residual, and the implicitly computed difference between the current and the previous eigenvector approximations:

$$\begin{array}{lcll} x^{(i+1)} & = & w^{(i)} + \tau^{(i)} x^{(i)} & + \gamma^{(i)} p^{(i)},\ w^{(i)} = T(Ax^{(i)} - \lambda^{(i)} Bx^{(i)}), \\ p^{(i+1)} & = & w^{(i)} & + \gamma^{(i)} p^{(i)},\ p^{(0)} = 0,\ \lambda^{(i)} = \lambda(x^{(i)}). \end{array} \tag{2.2}$$

We have $x^{(i+1)} \in \mathrm{span}\{w^{(i)},\ x^{(i)},\ p^{(i)}\} = \mathrm{span}\{w^{(i)},\ x^{(i)},\ x^{(i-1)}\}$ as the directions $p^{(i+1)} = x^{(i+1)} - \tau^{(i)} x^{(i)}$; therefore, the new formula (2.2) is mathematically equivalent to (2.1) in exact arithmetic. We choose in (2.2) the scalar iteration parameters $\tau^{(i)}$ and $\gamma^{(i)}$ as above, i.e., minimizing the Rayleigh quotient $\lambda(x^{(i+1)})$.

We note that formula (2.2) is not quite the ultimate fix; e.g., it is possible in principle that at the initial stage of the iterations $\tau^{(i)}$ is small, so that $p^{(i+1)}$ is too close to $x^{(i+1)}$, and the Rayleigh–Ritz procedure may fail. The proper choice of the basis in the trial subspace $\mathrm{span}\{w^{(i)}, x^{(i)}, p^{(i)}\}$ for the Rayleigh–Ritz procedure in (2.2), having in mind that vectors $w^{(i)}$ and $p^{(i)}$ converge to zero, is not a trivial issue; see Hetmaniuk and Lehoucq [12] for possible causes of LOBPCG instability.

The locally optimal choice of the step sizes in the LOBPCG allows an easy generalization from the single-vector version (2.1) or (2.2) to the block version described next in subsection 2.1.3, where a block of m vectors is iterated simultaneously. We return to the discussion of the choice of the basis in subsection 2.2, where it becomes even more vital with the increase of the block size m.

2.1.3. The LOBPCG basics. A block version of LOBPCG for finding the $m \geq 1$ smallest eigenpairs is suggested in Knyazev [16, 17]:

$$x_j^{(i+1)} \in \operatorname{span}\left\{x_1^{(i-1)},\ x_1^{(i)},\ T(A-\lambda_1^{(i)}B)x_1^{(i)},\ldots,\ x_m^{(i-1)},\ x_m^{(i)},\ T(A-\lambda_m^{(i)}B)x_m^{(i)}\right\},$$

where $x_j^{(i+1)}$ is computed as the jth Ritz vector, $j = 1,\ldots,m$, corresponding to the jth smallest Ritz value in the Rayleigh–Ritz procedure on a $3m$-dimensional trial subspace. The block method has the same problem of having close vectors in the trial subspace as the single-vector version (2.1) discussed in the previous subsection, for which [18] suggested the same fix using the directions p.

As in other block methods, the block size should be chosen, if possible, to provide a large gap between first m eigenvalues and the rest of the spectrum, as this typically leads to a better convergence; see [18, 21, 24]. Block methods generally handle clusters in the spectrum and multiple eigenvalues quite well, and the LOBPCG is no exception. An attempt should be made to include the whole cluster of eigenvalues in the block, while for multiple eigenvalues this is not essential at all. The smallest eigenvalues often converge faster, and the few eigenvalues with the noticeably slower convergence usually are the largest in the block and typically correspond to a cluster of eigenvalues that is not completely included in the block. A possible cure is to use a flexible block size a bit larger than the number of eigenpairs actually wanted and to ignore poorly converging eigenpairs.

The block size m should not be so big that the costs of the Rayleigh–Ritz procedure in the $3m$-dimensional trial subspace dominate the costs of iterations. If a large number of eigenpairs is needed, deflation is necessary; see subsection 2.3. The optimal choice of the block size m is problem-, software-, and computer-dependent; e.g., it is affected by such details as the amount of the CPU cache and the use of optimized BLAS library functions in the multivector implementation; see section 3.

2.2. The detailed description of the LOBPCG algorithm. The description of the LOBPCG algorithm as implemented in our BLOPEX-1.0 code follows:

Input: m starting linearly independent vectors in $X \in \mathbb{R}^{n\times m}$, l linearly independent constraint vectors in $Y \in \mathbb{R}^{n\times l}$, devices to compute $A*X$, $B*X$, and $T*X$.

1. Allocate memory $W, P, Q, AX, AW, AP, BX, BW, BP \in \mathbb{R}^{n\times m}$, $BY \in \mathbb{R}^{n\times l}$.
2. Apply the constraints to X:
   $BY = B*Y$; $X = X - Y*\left(Y^T*BY\right)^{-1}*((BY)^T*X)$.
3. B-orthonormalize X: $BX = B*X$; $R = \mathrm{chol}(X^T*BX)$; $X = X*R^{-1}$;
   $BX = BX*R^{-1}$; $AX = A*X$. (Note: "chol" is the Cholesky decomposition.)
4. Compute the initial Ritz vectors: solve the eigenproblem
   $(X^T*AX)*TMP = TMP*\Lambda$;
   and compute $X = X*TMP$; $AX = AX*TMP$; $BX = BX*TMP$.
5. Define the index set J of active iterates to be $\{1,\ldots,m\}$.
6. **for** $k = 0,\ldots,MaxIterations$:
7. Compute the residuals: $W_J = AX_J - BX_J*\Lambda_J$.
8. Exclude from the index set J the indices that correspond to residual

vectors for which the norm has become smaller than the tolerance.
If J, then becomes empty; exit loop.
9. Apply the preconditioner T to the residuals: $W_J = T * W_J$.
10 Apply the constraints to the preconditioned residuals W_J:
$W_J = W_J - Y * \left(Y^T * BY\right)^{-1} * ((BY)^T * W_J)$.
11. Compute BW_J and B-orthonormalize W_J: $BW_J = B * W_J$;
$R = \text{chol}(W_J^T * BW_J)$; $W_J = W_J * R^{-1}$; $BW_J = BW_J * R^{-1}$.
12. Compute AW_J: $AW_J = A * W_J$.
13. **if** $k > 0$
14. B-orthonormalize P_J: $R = \text{chol}(P_J^T * BP_J)$; $P_J = P_J * R^{-1}$;
15. Update $AP_J = AP_J * R^{-1}$; $BP_J = BP_J * R^{-1}$.
16. **end if**
Perform the Rayleigh–Ritz procedure:
Compute symmetric Gram matrices:
17. **if** $k > 0$
18. $$gramA = \begin{bmatrix} \Lambda & X^T * AW_J & X^T * AP_J \\ \cdot & W_J^T * AW_J & W_J^T * AP_J \\ \cdot & \cdot & P_J^T * AP_J \end{bmatrix}.$$
19. $$gramB = \begin{bmatrix} I & X^T * BW_J & X^T * BP_J \\ \cdot & I & W_J^T * BP_J \\ \cdot & \cdot & I \end{bmatrix}.$$
20. **else**
21. $$gramA = \begin{bmatrix} \Lambda & X^T * AW_J \\ \cdot & W_J^T * AW_J \end{bmatrix}.$$
22. $$gramB = \begin{bmatrix} I & X^T * BW_J \\ \cdot & I \end{bmatrix}.$$
23. **end if**
24. **Solve the generalized eigenvalue problem:**
$gramA * C = gramB * C * \Lambda$, where the first m eigenvalues in increasing order are in the diagonal matrix Λ and the $gramB$-orthonormalized eigenvectors are the columns of C.
Compute Ritz vectors:
25. **if** $k > 0$
26. Partition $C = \begin{bmatrix} C_X \\ C_W \\ C_P \end{bmatrix}$ according to the number of columns in X, W_J, and P_J, respectively.
27. Compute $P = W_J * C_W + P_J * C_P$;
$AP = AW_J * C_W + AP_J * C_P$; $BP = BW_J * C_W + BP_J * C_P$.
28. $X = X * C_X + P$; $AX = AX * C_X + AP$; $BX = BX * C_X + BP$.
29. **else**
30. Partition $C = \begin{bmatrix} C_X \\ C_W \end{bmatrix}$ according to the number of columns in X and W_J, respectively.
31. $P = W_J * C_W$; $AP = AW_J * C_W$; $BP = BW_J * C_W$.
32. $X = X * C_X + P$; $AX = AX * C_X + AP$; $BX = BX * C_X + BP$.
33. **end if**
37. **end for**

Output: The eigenvectors X and the eigenvalues Λ.

For description of all LOBPCG BLOPEX variables see Appendix A.

In the complex case, the transpose needs to be replaced by the adjoint. Only double-precision real arithmetic is supported in the current release 1.0 of BLOPEX.

The algorithm is *matrix-free* in the sense that the matrices A and B are not needed in the algorithm and not stored in the code, but rather are accessed only through matrix-vector product functions provided by the user. Matrix-free codes are needed in applications where the matrices A and B are never explicitly formed, e.g., in finite element method software packages for partial differential equations.

The algorithm uses only one block application of the preconditioner T (step 9) and one block matrix-vector product Bx (step 11) and Ax (step 12) per iteration. This is achieved by storing and manipulating $9m$ vectors ($6m$ vectors if $B = I$). If there is not enough memory for $9m$ vectors, or if m is so large that linear algebra operations with $9m$ vectors are more costly than multiplication by A and B, e.g., if A and B are very sparse, the algorithm can be modified so that only $3m$ vectors are stored, but then additional block matrix-vector products Ax and Bx are required per iteration. Such a modification currently is not yet available in BLOPEX.

Choosing a good basis for the Rayleigh–Ritz method becomes even more delicate with a larger number m of vectors in the block. Ill-conditioned Gram matrices in the Rayleigh–Ritz procedure may result in algorithm failure, inaccurate computations of the Ritz pairs, or spurious Ritz pairs. As it can be seen in the LOBPCG algorithm in steps 11–22, we explicitly compute X, P, and W and form the basis of the Rayleigh–Ritz procedure in the following way: X is left untouched since it already has B-orthonormal columns, made of the Ritz vectors, while the vectors inside both W and P are B-orthonormalized in steps 11 and 14, correspondingly, so we put the identities on the block diagonal of the matrix $gramB$ (steps 19 and 22).

For the B-orthonormalization of W and P in steps 11 and 14, the cheapest orthonormalization technique, using the Cholesky decomposition of the Gram matrix, has been chosen. This technique is known to be of poor quality in the presence of the round-off errors. Nevertheless, in our experience, the Cholesky-based B-orthonormalization of W and P reduces the risk of the Rayleigh–Ritz procedure failure while being considerably cheaper than the stable B-orthonormalization of all $3m$ vectors in the basis described, e.g., in Hetmaniuk and Lehoucq [12].

Let us note that matrices P and W converge to zero; what is even worse is that different columns of P and W converge to zero with different speeds, so matrices plugged into the Cholesky decomposition are extremely poorly scaled. It is crucial for the stability that the code of the Cholesky decomposition used here is numerically invariant with respect to matrix scaling; otherwise, the explicit normalization of columns of P and W is necessary prior to the call of the Cholesky decomposition. Such a subtle property as scaling invariance, which we rely on here, apparently may be affected by aggressive optimization flags during the compilation of the Cholesky decomposition code.

We also have tested (in MATLAB) a version in which we explicitly compute the block diagonal of the matrix $gramB$ instead of just setting the blocks to be the identities. Such an approach is similar to a known idea of repeated orthonormalization and results in more attainable accuracy. However, in our tests we have found that the attainable accuracy of our Algorithm 2.2 is already adequate, so this is not implemented in our BLOPEX LOBPCG.

Numerical software development is often an act of a balanced compromise between performance, reliability, and accuracy. Performance has been our main priority in

BLOPEX release 1.0, but we preserve reasonable reliability for common situations and achieve the accuracy typical for large-scale applications in engineering. Design of efficient and reliable implementations of the LOBPCG algorithm for BLOPEX is underway.

2.3. LOBPCG deflation by hard and soft locking. When several eigenvectors are computed simultaneously, it is often the case that some eigenvectors converge faster than others. To avoid the unnecessary computational work, one "locks" the eigenvectors that have already converged within a required tolerance while continuing to iterate the others. Locking approximately computed eigenvectors is not well studied theoretically. In this subsection, we introduce hard and soft locking and explain how locking is used in the LOBPCG algorithm, following the presentation of Knyazev [19]. The idea of soft locking is very natural and may have been known to some practitioners, developing codes for symmetric eigenvalue problems, even prior to Knyazev [19], where the term "soft locking" was apparently first introduced, but we are not aware of any earlier references.

A commonly used locking technique, known as deflation by restriction, is to keep the locked eigenvectors unchanged and those that are still iterated orthogonal to the locked ones. This technique is usually called "locking," but here it is referred to as "hard locking" to distinguish it from a new type of locking described next.

A different technique, called here "soft locking," can be used in simultaneous iterations combined with the Rayleigh–Ritz procedure. The idea of this technique is to remove the residuals of the locked vectors from the computation but continue using the vectors themselves in the Rayleigh–Ritz procedure, which can change them. The orthogonality of the soft locked vectors to iterative vectors follows from the well-known fact that the Ritz vectors corresponding to different Ritz values are orthogonal, and those corresponding to multiple Ritz values can be chosen to be orthogonal.

Compared to hard locking, soft locking is computationally more expensive. The advantage of soft locking is that it provides more accurate results. First and foremost, as the number of hard locked vectors increases, the attainable accuracy of the iterated ones may decrease, which may prevent them from reaching the required tolerance. With soft locking, the attainable accuracy of the iterated vectors does not depend on the number and the accuracy of soft locked ones. Second, if the Rayleigh quotient of kth locked vectors is less than the $(k+1)$th exact eigenvalue, then the accuracy of the first k locked vectors will increase in the course of the iterations, owing to the participation in the Rayleigh–Ritz procedure, even though their role in the iterative loop is now reduced to only this procedure. Soft-locking accuracy can be analyzed using known error bounds for the Rayleigh–Ritz method (see, e.g., [13, 15, 22]).

Below we describe the hard and soft locking as used in the BLOPEX implementation of LOBPCG and present preliminary numerical results demonstrating the effectiveness of soft locking compared to traditional hard locking.

Let us consider the single-vector LOBPCG method with no locking as described by (2.1), which we repeat here for the reader's convenience:

$$x^{(i+1)} = w^{(i)} + \tau^{(i)}x^{(i)} + \gamma^{(i)}x^{(i-1)},\ w^{(i)} = T(Ax^{(i)} - \lambda^{(i)}Bx^{(i)}).$$

Let P be a projector on the complement to the span of already computed eigenvectors. In LOBPCG with hard locking, we redefine $w^{(i)} = PT(Ax^{(i)} - \lambda^{(i)}Bx^{(i)})$ so that all iterates are now within the range of P if the initial guess in chosen within the range. Traditionally, P is orthogonal in the B-based scalar product and the complement is orthogonal also in the B-based scalar product, since this allows the easiest

and most inexpensive implementation. This is what we use in step 10 of the LOBPCG algorithm in subsection 2.2. The constraints, given by the columns of the matrix Y, are in this case the already approximately computed eigenvectors that we want to deflate.

If the locked eigenvectors are computed exactly, then it is easy to analyze the influence of classical hard locking and to show that it is equivalent to simply removing the locked eigenpairs from consideration. If the locked eigenvectors are only approximate, the range of P is not exactly an invariant subspace of the matrix pencil $A-\lambda B$, and the analysis of the influence of the approximation quality on locking becomes much more complicated; see, e.g., D'yakonov and Knyazev [7].

As the number of locked vectors increases, their inaccuracy may affect the attainable accuracy of iterated vectors. To avoid this, we use soft locking in the LOBPCG algorithm in subsection 2.2. We denote by X_J the matrix that contains only the active (not soft locked) iterative eigenvector approximations as columns, while all m iterative vectors X are included in the Rayleigh–Ritz procedure. Only X_J participates in iterations; i.e., having $\Lambda_J^{(i)}$ and $X_J^{(i)}$ we compute only $W_J^{(i)} = T(AX_J^{(i)} - BX_J^{(i)}\Lambda_J^{(i)})$, where $\Lambda_J^{(i)}$ is a diagonal matrix of approximate eigenvalues corresponding to $X_J^{(i)}$. We include the complete X, but only W_J and P_J, in the basis of the Rayleigh–Ritz procedure in steps 18–22 and 26–32 of the LOBPCG algorithm in subsection 2.2.

The positive influence of soft locking is especially noticeable when the stopping tolerance is not so small. Figure 2.1 presents the results of soft locking when the residual norm reaches 10^{-1} in LOBPCG with $m = 10$ for a model problem. We observe that the active eigenpairs continue to converge and that the eigenpairs, which are already soft locked, still improve in accuracy.

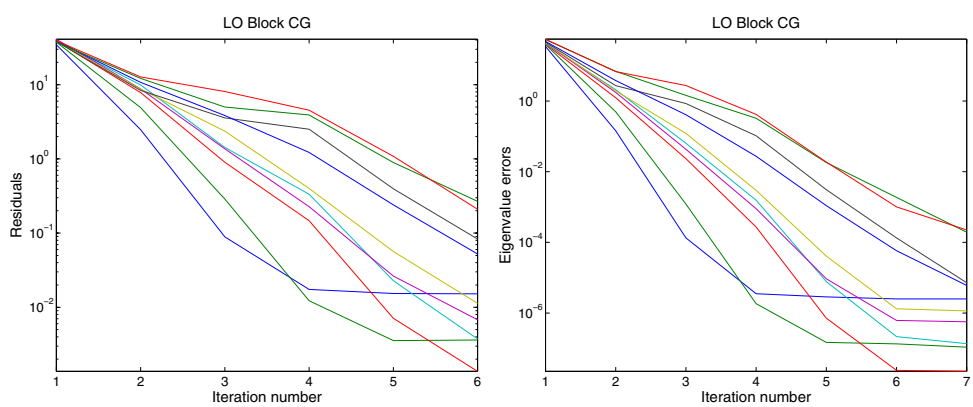


FIG. 2.1. *Residual norms (left) and eigenvalue errors (right) for* 10 *eigenpairs in LOBPCG with soft locking. Soft locked (the bottom* 6*) eigenpairs continue to improve.*

To summarize, we use the classical hard locking in LOBPCG through the constraints and soft locking inside the LOBPCG code. If the user wants to compute so many eigenpairs that it is not reasonable to include all of them at once in the block, the user should determine an appropriate block size m for the given hardware and call LOBPCG with the block size m several times, putting the previously computed eigenvectors into the constraints for the next LOBPCG call. The LOBPCG MATLAB code includes an example of such a call in the help information.

3. Abstract BLOPEX implementation for PETSc and *hypre*. PETSC and *hypre* are software libraries for solving large systems on massively parallel computers. Our native PETSc BLOPEX version gives the PETSc user community easy access to the customizable code of a modern preconditioned eigensolver and an opportunity to easily call *hypre* preconditioners from PETSc. The BLOPEX built-in *hypre* version efficiently takes direct advantage of powerful *hypre* AMG preconditioner BoomerAMG and its geometric multigrid preconditioners.

The BLOPEX library currently includes only the LOBPCG eigensolver. The BLOPEX code is written in C-language and calls a few LAPACK subroutines for dense matrix operations. The matrix-vector multiply and the preconditioner call are done through user-supplied functions. The main LOBPCG code is abstract in the sense that it works only through an interface that determines the particular software environment, e.g., PETSc or *hypre*, in order to call parallel (multi)vector manipulation routines. A block diagram of the main modules is given in Figure 3.1.

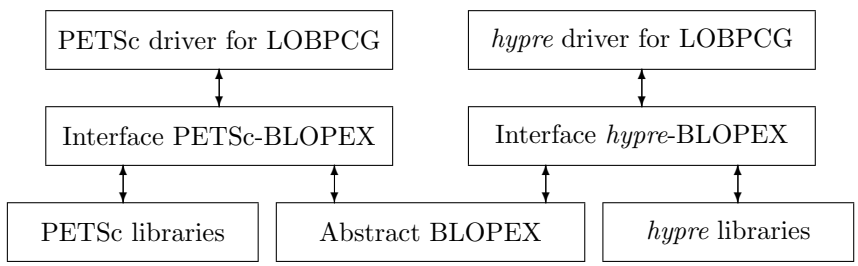


FIG. 3.1. *BLOPEX hypre and PETSc software modules.*

In the abstract part of BLOPEX, the blocks of vectors (X, P, etc.) are represented by an abstract data object we call "multivector." A particular implementation of the multivector is outside of our code, so we only provide an interface to it. Ideally, all operations involving multivectors should be implemented in a block fashion to minimize the number of exchanges between the processors and to take advantage of a highly optimized matrix-matrix multiplication routine `dgemm` in BLAS when computing Gram matrices and updating multivectors. However, to our knowledge such an object is currently not available to users in PETSc and *hypre*, so we had to temporarily use the multivector interfaces as references to individual vectors. When properly implemented multivectors become available in PETSc and *hypre*, we plan to change our interface codes to use the true multivectors, which is expected to lead to greater speed-up due to a smaller number of communications and efficient use of BLAS.

4. BLOPEX LOBPCG numerical results in PETSc and *hypre*. Here we present results which were accumulated over a considerable length of time. Despite our efforts, the reader may still notice some inconsistency, e.g., some tests may use different tolerances. All tests are performed in double-precision real arithmetic.

4.1. The basic accuracy of the algorithm. In these tests BLOPEX LOBPCG computes the smallest 50 eigenvalues of 3-D 7-point $200\times200\times200$ and $200\times201\times202$ Laplacians using a direct application of the *hypre* BoomerAMG preconditioning. In the first case we have eigenvalues with multiplicity, and in the second case the eigenvalues are distinct but clustered. The initial approximations are chosen randomly. We set the stopping tolerance (the norm of the maximum residual) equal to 10^{-6}. The

numerical output (for the complete data, see Knyazev and Argentati [20]) is compared to the exact eigenvalues

$$\lambda_{i,j,k} = 4\left[\sin\left(\frac{i\pi}{2(n_x+1)}\right)^2 + \sin\left(\frac{j\pi}{2(n_y+1)}\right)^2 + \sin\left(\frac{k\pi}{2(n_z+1)}\right)^2\right]$$

of the 3-D 7-point Laplacians on the $n_x \times n_y \times n_z$ cube with $1 \le i \le n_x$, $1 \le j \le n_y$, and $1 \le k \le n_z$. In both tests for all eigenvalues the maximum relative error is less than 10^{-8}, so LOBPCG is cluster robust, i.e., it does not miss (nearly) multiple eigenvalues, which supports the conclusion of Borzí and Borzí [4].

4.2. Performance versus number of inner iterations. We can execute a preconditioner $x = Tb$ either directly or by calling PCG to solve a linear system, which in our tests is $Ax = b$, so T approximates A^{-1}. Increasing the number of "inner" iterations of the PCG will accelerate the overall convergence, but only if we do not make too many iterations, since even if $T = A^{-1}$, the convergence of the LOBPCG method is still linear. In other words, for a given matrix A and a particular choice of a preconditioner T, there should be an optimal finite (maybe zero, corresponding to the direct application of T) number of inner iterations.

We try to find this optimal number for the Schwarz-PCG and BoomerAMG-PCG preconditioners in *hypre* and PETSc for the 7-point 3-D $100 \times 100 \times 100$ Laplacian with block size $m = 1$. We measure the execution time as we vary the quality of the preconditioner by changing the maximum number of inner iterations in the corresponding PCG solver. PETSc and *hypre* built-in Schwarz algorithms are different but demonstrate similar behavior. The *hypre* multigrid BoomerAMG and PFMG preconditioners are called from *hypre* test drivers. The *hypre* BoomerAMG preconditioner in addition is called through PETSc. The results for these tests involving different multigrid preconditioners are also similar.

We find that for this problem the optimal number of inner iterations is approximately 10–15 for Schwarz-PCG, but BoomerAMG-PCG works best if BoomerAMG is applied directly as a preconditioner, without even initializing the PCG solve function. Our explanation of this behavior is based on two facts. First, the Schwarz method with the default parameters used here is somewhat cheaper, but not of such good quality when compared to BoomerAMG in these tests. Moreover, the costs for matrix-vector and multivector linear algebra in LOBPCG are relatively small compared to the costs of the BoomerAMG application but is comparable to the costs of applying the Schwarz preconditioner here. Second, one PCG iteration is less computationally expensive compared to one LOBPCG iteration because of the larger number of linear algebra operations with vectors in the latter. A single direct application of BoomerAMG as the preconditioner in LOBPCG results in enough improvement in convergence to make it the best choice, while Schwarz requires more iterations that are less time consuming if performed using PCG rather than directly in LOBPCG.

In the next set of numerical tests for the same eigenproblem, we use the block size $m = 10$, and we vary the type of PCG preconditioner available in the *hypre* IJ interface together with the number of inner PCG iterations. For the complete test data, see Knyazev and Argentati [20]; here we provide only a summary. We observe that both the number of outer iterations and the execution time can be decreased significantly by choosing the maximum number of inner iterations optimally. For this problem, the BoomerAMG preconditioner applied directly has the fastest run time and converges in the smallest number of iterations, 31. If we use no preconditioning,

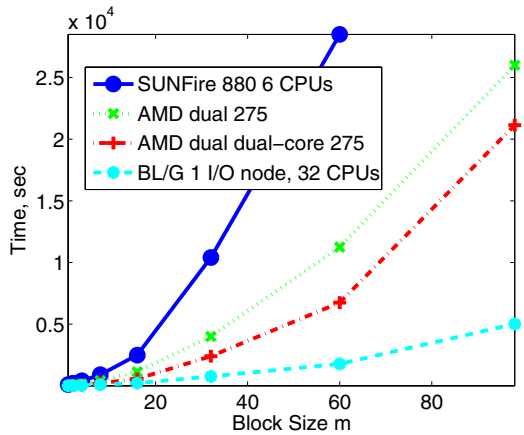


FIG. 4.1. *LOBPCG performance versus block size. Preconditioner: hypre BoomerAMG.*

then in 500 iterations, which take 10 times longer, we still do not reach the required tolerance 10^{-10}, which illustrates the importance of preconditioning.

4.3. LOBPCG performance versus block size. We test *hypre* and PETSc LOBPCG on a 7-point 3-D Laplacian with $128 \times 128 \times 128 \approx 2M$ unknowns using the *hypre* BoomerAMG preconditioner by increasing the block size m from 1 to 98. We run the tests on a shared memory SUN Fire880 system, dual- and single-core AMD Opterons Dual 275 CPUs servers, and a single I/O node of IBM BG/L with 32 CPUs. We find that the *hypre* and PETSc LOBPCG run times are very close and use the same tolerance 10^{-8}; with PETSc the extra overhead is less than a few percent of the total times compared to a direct use of *hypre*, as expected. The growth of the CPU time with the increase of the block size m is approximately linear for this problem up to $m = 16$; then the complexity term $m^2 n$ becomes visible and the slower convergence rate for larger eigenvalues starts noticeably affecting the overall timing, as shown in Figure 4.1. An efficient implementation of the multivector, e.g., such as that used in the ABINIT version of the LOBPCG (see Bottin et al. [5]), is expected to significantly improve performance of the LOBPCG with respect to the block size.

4.4. Scalability. For the 7-point 3-D Laplacian we vary the problem size proportionally to the number of processors. We directly apply *hypre* BoomerAMG or PFMG multigrid preconditioner. First, on the MCR cluster the eightfold increase in the number of dual-CPU nodes does not noticeably increase the LOBPCG CPU time, which is approximately 50 sec for BoomerAMG and 180 sec for PFMG with $m = 1$ and tolerance 10^{-8}. The PFMG takes more time overall compared to BoomerAMG because of the larger convergence factor, but each iteration of the former is 50% faster than that of the latter, as we observe in Figure 4.2, which displays the CPU time per iteration for LOBPCG *hypre* code with *hypre* BoomerAMG and PFMG preconditioners in our second test on the Beowulf cluster. We see that the 32-fold increase in the number of dual-CPU nodes just slightly increases the CPU time.

Finally, we solve the 7-point 3-D Laplacian eigenvalue problems for large matrix sizes and for a variety of block sizes m on the BG/L system. We observe a 10–20% variability in the iteration numbers, and thus in the wall clock, due to the randomness of the initial approximations, so we report here typical average results. Tables 4.1 and 4.2 contain information about LOBPCG runs using *hypre* SMG geometric multigrid and tolerance 10^{-6}.

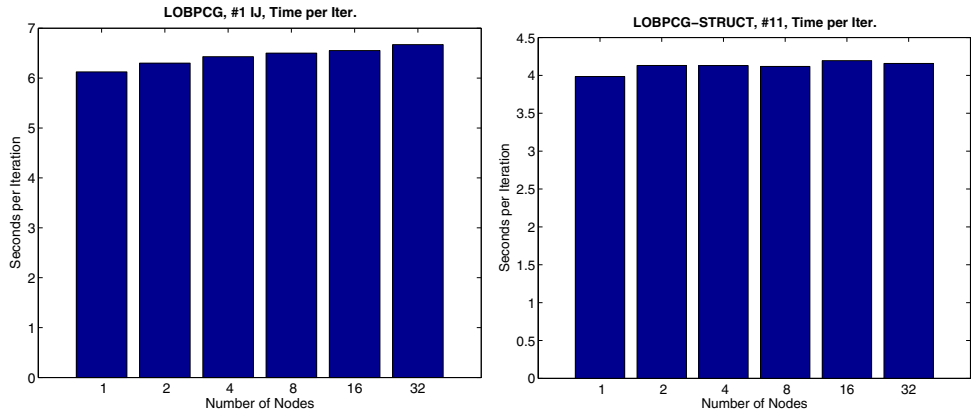


FIG. 4.2. *LOBPCG scalability, 1M unknowns per node on Beowulf.*

TABLE 4.1
Scalability for $80 \times 80 \times 80 = 512{,}000$ *mesh per CPU,* $m = 1/6$.

# CPUs	Matrix Size	Setup (sec.)	# Iter.	Solve time (sec.)
8	4.096 M	7	9 / 19	62 / 580
64	32.768 M	10	8 / 17	62 / 570
512	0.262144 B	15	7 / 13	56 / 500

TABLE 4.2
Scalability for $50 \times 50 \times 50 = 125{,}000$ *mesh per CPU,* $m = 50$.

# CPUs	Matrix size	Setup (sec.)	# Iter.	Solve time (thous. sec.)
8	1 M	1	34	3
64	8 M	5	32	3
512	64 M	10	29	3

5. Conclusions. The BLOPEX software is integrated into the *hypre* library and is easily available in PETSc as an external package. BLOPEX is the only currently available software that solves eigenvalue problems by directly using high quality existing *hypre* and PETSc preconditioners. Our abstract C implementation of the LOBPCG in BLOPEX allows easy deployment with different software packages. Our interface routines are based on *hypre* and PETSc standard interfaces and give the user an opportunity to provide matrix-vector multiply and preconditioned solver functions. The code scalability is tested for model problems of large sizes on parallel systems.

Appendix A. Description of all LOBPCG BLOPEX variables.

X (multivector). The iterative approximation to eigenvectors.

A (function). Multiplies the function argument, which is a multivector, by the first matrix of the eigenproblem.

B (function). Multiplies the function argument, which is a multivector, by the second matrix of the eigenproblem.

T (function). Applies the preconditioner to the function argument, a multivector.

Y (multivector). The given constraint that can be used for hard locking. Iterations run in the B-orthogonal complement to the span of Y.

Λ (diagonal matrix, in the actual code given by a vector of the diagonal entries). The iterative approximation to eigenvalues.

W (multivector). Used to store the residuals (step 7), the preconditioned residuals (step 9), and the preconditioned constrained residuals (step 10 and later).

J (index vector). Denotes the set of indexes of active iterates. The starting value for J is $\{1, \ldots, m\}$. If the norm of the residual r_j for some $j \in J$ has become smaller than the tolerance, this index j is excluded from the set J. By X_J we denote a submatrix of X with the columns, the indexes of which are present in the set J—these are the active iterates. The indexes from $\{1, \ldots, m\}$ that are not present in the set J correspond to iterative approximations to eigenvectors, which have been "soft locked."

P (multivector). The LOBPCG "directions."

gramA and *gramB* (matrices). The Gram matrices in the Rayleigh–Ritz procedure.

C (matrix). The matrix of *gramB*-orthonormalized eigenvectors in the Rayleigh–Ritz procedure, used as the matrix of coefficients to compute the Ritz vectors, the new multivectors X, as linear combinations of the basis X, W_J, and P_J.

AX, BX, AP, BP, etc. (multivectors). Implicitly computed results of the application of the corresponding functions to the corresponding vectors. In the absence of the round-off errors, they would be identical to their targets, e.g., $AX = A \star X$.

R (matrix). The upper triangular matrix of the Cholesky factorization, used for the B-orthogonalization via Cholesky.

Appendix B. Installing and testing BLOPEX with PETSc and *hypre*. Here we give brief but specific information for the reader who wants to install and test our BLOPEX software. This information concerns the current BLOPEX-1.0, PETSc-2.3.2, and *hypre*-2.0.0 releases. We refer to Appendix D for the acronyms and other names not explained here and suggest reading the software manuals for more details.

To install and test BLOPEX and *hypre* in PETSc, download the PETSc source and follow the standard installation procedure with the external packages option. The present procedure is to add "--download-blopex=1 --download-hypre=1" options to the PETSc configuration. BLOPEX and *hypre* sources are then downloaded and compiled automatically. The BLOPEX test driver is already included in PETSc at the src/contrib/blopex/driver directory. For example, in

```
driver -da_grid_x 128 -da_grid_y 128 -da_grid_z 128 -ksp_type preonly -pc_type
hypre -pc_hypre_type boomeramg -n_eigs 10 -seed 2 -tol 1e-6 -itr 5 -full_out 1
```

the eigenproblem for the 7-point 3-D Laplacian is solved by directly using the *hypre* BoomerAMG preconditioner. The last five parameters are self-explanatory.

BLOPEX LOBPCG is built into *hypre* and requires no special configuration options. *hypre* supports four conceptual interfaces: STRUCT, SSTRUCT, FEM, and IJ. LOBPCG has been tested with all but the FEM interface. *hypre*-2.0.0 test drivers for LOBPCG are in the src/test directory. The following are examples with the same options as in the PETSc call above using IJ and STRUCT test drivers:

```
ij -lobpcg -n 128 128 128 -pcgitr 0 -vrand 10 -seed 2 -tol 1e-6 -itr 5 -verb 1
struct -lobpcg -n 128 128 128 -pcgitr 0 -vrand 10 -seed 2 -tol 1e-6 -itr 5 -verb 1
```

Our code can, but is not really intended to, be used with the popular shift-and-invert strategy, since its main advantage compared to traditional eigensolvers is that it uses preconditioning. Our test drivers can call preconditioning directly ("-ksp_type preonly" PETSc option and "-pcgitr 0" *hypre* option) as well as through calls to the *hypre*/PETSc PCG method. In the latter case the action $x = Tb$ of the preconditioner T on a given vector b is performed by calling a few steps of PCG to solve $Ax = b$. The BLOPEX LOBPCG code has been tested with all available *hypre* PCG-capable preconditioners in STRUCT, SSTRUCT, and IJ interfaces, with the PETSc native additive Schwarz, and with the PETSc linked BoomerAMG from *hypre*.

Appendix C. Computers used for testing.

AMD workstations Several single- and dual-core AMD two 64 bit 275 CPUs with 16GB shared memory, running Linux, at CCM CU-Denver.

SUN Fire 880 SUN Solaris 9, 6 UltraSPARC III 64 bit CPUs, 24GB shared memory, at CCM CU-Denver.

Beowulf cluster Linux, 36 nodes, dual PIII 933MHz processors and 2GB memory per node with 7.2SCI Dolphin interconnect, at CCM CU-Denver.

MCR cluster Linux, dual Xeon 2.4GHz 4 GB nodes with Quadrics, at LLNL.

Blue Gene/L (BG/L) IBM single-rack with 1024 compute nodes, organized in 32 I/O nodes with 32 compute nodes each. Every compute node is a dual-core chip, containing two 700MHz PowerPC-440 CPUs and 512 MB of memory. We run using the default 1 CPU for computing and 1 CPU for communication on every compute node. Location: NCAR.

Appendix D. Acronyms and other names.

D.1. Organizations.

CCM CU-Denver Center for Computational Mathematics, University of Colorado at Denver and Health Sciences Center, Downtown Denver Campus.

LLNL Lawrence Livermore National Laboratory.

NCAR National Center for Atmospheric Research.

D.2. Software packages.

BLOPEX Block Locally Optimal Preconditioned Eigenvalue Xolvers, a library for large-scale eigenvalue problems, developed by the authors of the paper.

PETSc Portable, Extensible Toolkit for Scientific Computation, tools for the scalable solution of partial differential equations and related problems developed by Argonne National Laboratory; see Balay et al. [2].

hypre High Performance Preconditioners, scalable linear solvers package developed by LLNL; see Falgout, Jones, and Yang [8, 9].

SLEPc Scalable Library for Eigenvalue Problem Computations, a library for large eigenvalue problems, an extension of PETSc; see Hernandez, Roman, and Vidal [10].

D.3. *hypre* interfaces and preconditioners.

IJ A Linear-Algebraic Interface for applications with sparse matrices.

BoomerAMG a parallel implementation of an algebraic multigrid.

Schwarz Agglomeration-based domain decomposition preconditioner.

STRUCT A structured grid interface for applications with logically rectangular grids.

SMG A parallel semicoarsening multigrid solver for the linear systems arising from finite difference, finite volume, or finite element discretiza-

tions of the diffusion equation on logically rectangular grids. The code solves both 2-D and 3-D problems with discretization stencils of up to 9-point in 2-D and up to 27-point in 3-D. The algorithm semicoarsens in the z-direction and uses plane smoothing.

PFMG A parallel semicoarsening multigrid solver similar to SMG. The main difference between the two methods is in the smoother: PFMG uses simple pointwise smoothing. As a result, PFMG is not as robust as SMG, but is much more efficient per V-cycle.

SSTRUCT A semistructured grid interface that targets applications with grids that are mostly structured but with some unstructured features.

FEI An unstructured grid interface designed for Finite Element applications.

D.4. Methods.

PCG Preconditioned Conjugate Gradient.

LOBPCG Locally Optimal Block Preconditioned Conjugate Gradient.

AMG Algebraic Multigrid.

D.5. Computer terms.

CPU Central Processing Unit.

MPI Message Passing Interface.

BLAS Basic Linear Algebra Subprograms are routines that provide standard building blocks for performing basic vector and matrix operations.

Acknowledgments. We are grateful to all members of the Scalable Algorithms Group of the Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, and, in particular, to Ulrike Yang, Rob Falgout, Edmond Chow, Charles Tong, and Panayot Vassilevski for their patient support and help. We thank Matthew Knepley and other members of the PETSc team for their help with incorporating our BLOPEX as an external package into the PETSc library. We thank Jose Roman, a member of the SLEPc team, for writing the SLEPc interface to our *hypre* LOBPCG.

REFERENCES

[1] P. Arbenz, U. L. Hetmaniuk, R. B. Lehoucq, and R. S. Tuminaro, *A comparison of eigensolvers for large-scale* 3*D modal analysis using AMG-preconditioned iterative methods*, Int. J. Numer. Meth. Engrg., 64 (2005), pp. 204–236.

[2] S. Balay, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang, *PETSc Users Manual*, Technical report ANL-95/11 - Revision 2.1.5, Argonne National Laboratory, Argonne, IL, 2004.

[3] R. E. Bank, *PLTMG: A Software Package for Solving Elliptic Partial Differential Equations*, Users' Guide 9.0, Department of Mathematics, University of California at San Diego, San Diego, CA, 2004. Available online at http://cam.ucsd.edu/~reb/reports/b15.pdf.gz.

[4] A. Borzì and G. Borzì, *Algebraic multigrid methods for solving generalized eigenvalue problems*, Int. J. Numer. Meth. Engrg., 65 (2006), pp. 1186–1196.

[5] F. Bottin, S. Leroux, A. Knyazev, and G. Zérah, *Large scale ab initio calculations based on three levels of parallelization*, Computational Materials Science, to appear (http://dx.doi.org/10.1016/j.commatsci.2007.07.019). Also available online at http://arxiv.org/abs/0707.3405.

[6] J. H. Bramble, T. V. Kolev, and J. E. Pasciak, *The approximation of the Maxwell eigenvalue problem using a least-squares method*, Math. Comp., 74 (2005), pp. 1575–1598.

[7] E. G. D'yakonov and A. V. Knyazev, *On an iterative method for finding lower eigenvalues*, Russian J. Numer. Anal. Math. Modelling, 7 (1992), pp. 473–486.

[8] R. D. Falgout, J. E. Jones, and U. M. Yang, *Pursuing scalability for hypre's conceptual interfaces*, ACM Trans. Math. Software, 31 (2005), pp. 326–350.

[9] R. D. Falgout, J. E. Jones, and U. M. Yang, *The design and implementation of hypre, a library of parallel high performance preconditioners*, in Numerical Solution of Partial Differential Equations on Parallel Computers, A. M. Bruaset and A. Tveito, eds., Lect. Notes Comput. Sci. Eng. 51, Springer, Berlin, 2006, pp. 267–294.

[10] V. Hernandez, J. E. Roman, and V. Vidal, *SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems*, ACM Trans. Math. Software, 31 (2005), pp. 351–362.

[11] M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring, A. Williams, and K. S. Stanley, *An overview of the Trilinos project*, ACM Trans. Math. Software, 31 (2005), pp. 397–423.

[12] U. Hetmaniuk and R. Lehoucq, *Basis selection in LOBPCG*, J. Comput. Phys., 218 (2006), pp. 324–332.

[13] A. V. Knyazev, *Sharp a priori error estimates of the Rayleigh-Ritz method without assumptions of fixed sign or compactness*, Math. Notes, 38 (1986), pp. 998–1002.

[14] A. V. Knyazev, *A preconditioned conjugate gradient method for eigenvalue problems and its implementation in a subspace*, in Numerical Treatment of Eigenvalue Problems, Vol. 5, Internat. Ser. Numer. Math. 96, Birkhäuser, Basel, 1991, pp. 143–154.

[15] A. V. Knyazev, *New estimates for Ritz vectors*, Math. Comp., 66 (1997), pp. 985–995.

[16] A. V. Knyazev, *Preconditioned eigensolvers—An oxymoron?*, Electron. Trans. Numer. Anal., 7 (1998), pp. 104–123.

[17] A. V. Knyazev, *Preconditioned eigensolvers*, in Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide, Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, eds., SIAM, Philadelphia, 2000, pp. 352–368.

[18] A. V. Knyazev, *Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method*, SIAM J. Sci. Comput., 23 (2001), pp. 517–541.

[19] A. V. Knyazev, *Hard and soft locking in iterative methods for symmetric eigenvalue problems*, Presentation at the Eighth Copper Mountain Conference on Iterative Methods, March 28–April 2, 2004. Available online at http://math.cudenver.edu/~aknyazev/research/conf/cm04_soft_locking/cm04.pdf

[20] A. V. Knyazev and M. E. Argentati, *Implementation of a Preconditioned Eigensolver Using Hypre*, Tech. report, CCM-CU Denver, Denver, CO, 2005. Available online at http://math.cudenver.edu/ccm/reports/rep220.pdf.

[21] A. V. Knyazev and K. Neymeyr, *A geometric theory for preconditioned inverse iteration,* III*: A short and sharp convergence estimate for generalized eigenvalue problems*, Linear Algebra Appl., 358 (2003), pp. 95–114.

[22] A. V. Knyazev and J. E. Osborn, *New a priori FEM error estimates for eigenvalues*, SIAM J. Numer. Anal., 43 (2006), pp. 2647–2667.

[23] I. Lashuk, M. E. Argentati, E. Ovchinnikov, and A. V. Knyazev, *Preconditioned eigensolver LOBPCG in hypre and PETSc*, in Proceedings of the 16th International Conference on Domain Decomposition Methods (2005), O. B. Widlund and D. E. Keyes, eds., Lect. Notes Comput. Sci. Eng. 55, Springer, Berlin, 2007, pp. 635–642.

[24] E. Ovtchinnikov, *Cluster robustness of preconditioned gradient subspace iteration eigensolvers*, Linear Algebra Appl., 415 (2006), pp. 140–166.

[25] A. Stathopoulos and J. R. McCombs, *PRIMME: PReconditioned Iterative Multi-Method Eigensolver: Methods and Software Description*, Tech report WM-CS-2006-08, College of William and Mary, Williamsburg, VA, 2006. Available online at http://www.cs.wm.edu/ andreas/publications/primme.pdf.

[26] S. Tomov, J. Langou, A. Canning, L.-W. Wang, and J. Dongarra, *Comparison of nonlinear conjugate-gradient methods for computing the electronic properties of nanostructure architectures*, in International Workshop on Computational Nanoscience and Technology, Lect. Notes Comput. Sci. 3516, Springer-Verlag, New York, 2005, pp. 317–325.

[27] S. Yamada, T. Imamura, and M. Machida, 16.447 *tflops and* 159*-billion-dimensional exact-diagonalization for trapped Fermion–Hubbard model on the earth simulator*, in Proceedings of the ACM/IEEE Conference on Supercomputing, Seattle, WA, November 12–18, 2005, IEEE Computer Society Press, 2005, p. 44.

[28] S. Yamada, T. Imamura, T. Kano, and M. Machida, *High-performance computing for exact numerical approaches to quantum many-body problems on the earth simulator*, in Proceedings of the ACM/IEEE Conference on Supercomputing (Tampa, FL, November 11–17, 2006), ACM Press, New York, 2006, article 47.

[29] C. Yang, J. C. Meza, and L. Wang, *A constrained optimization algorithm for total energy minimization in electronic structure calculations*, J. Comput. Phys., 217 (2006), pp. 709–721.

[30] S. Zaglmayr, *High Order Finite Element Methods for Electromagnetic Field Computation*, Ph.D. thesis, Johannes Kepler University Linz, Linz, Germany, 2006.

SIAM J. SCI. COMPUT.
Vol. 29, No. 6, pp. 2241–2257
© 2007 Society for Industrial and Applied Mathematics

EFFICIENT NUMERICAL SOLUTION OF CAHN–HILLIARD–NAVIER–STOKES FLUIDS IN 2D*

DAVID KAY† AND RICHARD WELFORD‡

Abstract. A finite element discretization of the variable density Cahn–Hilliard–Navier–Stokes system is presented. This system is then decoupled with the Cahn–Hilliard part solved via the nonlinear multigrid method proposed by the authors in [*J. Comput. Phys.*, 212 (2006), pp. 288–304] and the Navier–Stokes part via a multigrid preconditioned GMRES scheme. This new solver is examined numerically and is shown to be efficient with respect to mesh refinement and robust with respect to the problem parameters.

Key words. Cahn–Hilliard equation, Navier–Stokes equation, finite element, nonlinear multigrid

AMS subject classifications. 65N55, 74S05, 82B26, 82D15

DOI. 10.1137/050648110

1. Introduction.

The Cahn–Hilliard equation [6, 8] is a phenomenological model of phase transitions; see the survey work of Elliott [10] for details. When coupled to the Navier–Stokes equations it provides a model for the dynamics of multiphase fluids that is used to model phenomena such as hydrodynamic effects during spinodal decomposition and the behavior of polymer fluids. The numerical simulation of this model is an area of research currently undergoing intense study (see, for example, [2], [5], [15], [19], [20]), with the majority of the work so far concentrating on finite difference discretizations. However, the *efficient* solution of these equations has yet to be studied in depth. Kim, Kang, and Lowengrub [20] have developed a finite difference multigrid scheme based on FAS (full approximation storage) multigrid for the Cahn–Hilliard part and a projection method for the Navier–Stokes part. The multigrid solver has convergence rates independent of the mesh size h with a slight time-step restriction dependent on the problem parameters.

In this paper we introduce a robust efficient nonlinear multigrid scheme for a finite element discretization of the Cahn–Hilliard–Navier–Stokes (CHNS) problem. The model is well suited to finite elements because of their flexibility and capacity for adaptivity.

Set $c \in [-1, 1]$ to be the concentration order parameter so that $c(x) \approx 1$ (resp., $c(x) \approx -1$) if and only if fluid 1 (resp., fluid 2) is present at the point x. Defining the mean velocity field by

$$\mathbf{u} = \frac{1+c}{2}\mathbf{u}_1 + \frac{1-c}{2}\mathbf{u}_2, \tag{1.1}$$

where $\mathbf{u}_i$ is the velocity field of fluid i, and the total density by

$$\rho = \rho_1 + \rho_2, \qquad \rho_1 = \rho_1^0 \frac{1+c}{2}, \qquad \rho_2 = \rho_2^0 \frac{1-c}{2}, \tag{1.2}$$

*Received by the editors December 21, 2005; accepted for publication (in revised form) September 27, 2006; published electronically October 5, 2007.
http://www.siam.org/journals/sisc/29-6/64811.html

†Department of Mathematics, Mantell Building, University of Sussex, Falmer, Brighton, BN1 9RF, UK (d.a.kay@sussex.ac.uk).

‡Corresponding author. Department of Mathematics, Mantell Building, University of Sussex, Falmer, Brighton, BN1 9RF, UK (r.r.welford@sussex.ac.uk). The work of this author was supported by EPSRC grant GR/R8951/01.

where ρ_i^0 is the density of fluid i, mass conservation ensures that the velocity is divergence free; that is,

$$\nabla \cdot \mathbf{u} = 0. \tag{1.3}$$

We note that the velocity defined by $\rho\mathbf{u} = \rho_1\mathbf{u}_1 + \rho_2\mathbf{u}_2$ which is considered by some authors (e.g., [22]) does not have this property.

Let Ω be a bounded convex polygonal domain in $\mathbb{R}^2$. The variable density CHNS problem is the following [4]:

(**P**) Find $\{c(x,t), w(x,t), \mathbf{u}(x,t), p(x,t)\}$ such that

$$\frac{\partial c}{\partial t} + \mathbf{u}\cdot\nabla c - \frac{1}{Pe}\nabla\cdot\left(\frac{b(c)}{\rho(c)}\nabla\left(\frac{w}{\rho(c)}\right)\right) = 0 \quad \text{in } \Omega_T := \Omega\times(0,T), \tag{1.4a}$$

$$-w + \frac{1}{\epsilon}\Phi'(c) - \epsilon\Delta c = 0 \quad \text{in } \Omega_T, \tag{1.4b}$$

$$\rho(c)\left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}\right) - \nu\Delta\mathbf{u} + \nabla p = Kw\nabla c + \delta K\frac{1-c^2}{4}\nabla\left(\frac{w}{\rho(c)}\right) + \rho(c)\mathbf{g} \quad \text{in } \Omega_T, \tag{1.4c}$$

$$\nabla\cdot\mathbf{u} = 0 \quad \text{in } \Omega_T, \tag{1.4d}$$

$$c(x,0) = c^0(x), \quad \mathbf{u}(x,0) = \mathbf{u}^0(x) \quad \forall\, x\in\Omega, \tag{1.4e}$$

$$\frac{\partial c}{\partial \hat{n}} = \frac{\partial w}{\partial \hat{n}} = 0, \quad \mathbf{u} = \mathbf{f} \quad \text{on } \partial\Omega\times(0,T), \tag{1.4f}$$

where the density is given by

$$\rho(c) = 1 + \frac{\delta}{2}(c-1) \tag{1.5}$$

and $\hat{\mathbf{n}}$ is the outward pointing unit normal to the boundary $\partial\Omega$. Here w represents the chemical potential and p the pressure. The given constants are Pe, ϵ, ν, K, and $\mathbf{g}$, where the Péclet number Pe and viscosity ν are measures of the relative rates of diffusion, the interface parameter ϵ determines the width of the interface separating the two fluids (the interface is of width $O(\epsilon)$), the capillary (or Cahn) number K is a nondimensional measure of the dispersion, and $\mathbf{g}$ is the gravitational field; cf. [22].

The contrast between the two densities is given by $\delta = (\rho_1^0 - \rho_2^0)/\rho_1^0$, where we have assumed throughout that $\rho_1^0 \geq \rho_2^0$. The mobility function b and double well potential Φ are given by

$$b(c) = ([1-c^2]_+)^2 = (\max(1-c^2, 0))^2, \tag{1.6}$$

$$\Phi(c) = \frac{1}{4}(1-c^2)^2. \tag{1.7}$$

Existence and uniqueness for this problem have been studied by Boyer [3]. Some fully discrete finite element discretizations of the equal density problem are analyzed in [13].

REMARK 1.1. *Using the identity* $\nabla(wc) = c\nabla w + w\nabla c$, *the term* $-Kw\nabla c$ *in the Navier–Stokes equation* (1.4c) *can be replaced by* $Kc\nabla w$, *where the additional term has been absorbed into the pressure term. Alternatively using* (1.4b), $-Kw\nabla c$ *can be replaced by* $K\epsilon\nabla c\Delta c$, *where again the extra terms are absorbed into the pressure term.*

These equations are very challenging to solve numerically. Typically the interfacial region separating the two fluids is very narrow, and a high spatial resolution is

required to accurately capture the interface dynamics, which leads to a large nonlinear matrix problem to be solved at each time step. Thus finite elements are a good choice for a discretization scheme as they can easily be used in adaptive meshes which have high interfacial resolution. This will be explored further in future work.

The matrix problem that arises from the discretization is very stiff so an implicit time discretization must be used. Additionally the solution may have several time scales over which it evolves. Thus we are led to seek fast and efficient solvers for the problem. We propose a multigrid scheme for the finite element discretization of the problem (**P**) based upon the multigrid scheme for the Cahn–Hilliard equation given in [18] and the F_p preconditioned GMRES solver for the Navier–Stokes equations of Kay, Loghin, and Wathen [17]. We show via numerical examples the robustness of this solver with respect to the problem parameters, in particular that the convergence rate of the scheme is independent of mesh size h.

The layout of this paper is as follows. We start by defining our notation for use below. In section 2 the finite element discretization is described. The solution method is given in section 3, where the multigrid scheme given in [18] and the F_p preconditioner [17] are summarized. Numerical examples demonstrating the efficiency of the proposed scheme are given in section 4. Finally, our conclusions are drawn in section 5.

1.1. Notation. Let $L^2(\Omega)$ denote the space of square integrable functions with norm denoted by $\|\cdot\|$ and inner product $(\cdot,\cdot)$, and let $L^2_0(\Omega) = \{v \in L^2(\Omega) : \int_\Omega v\, dx = 0\}$. Let $H^s(\Omega)$ be the usual Sobolev space. Let $\mathbb{H}^s(\Omega) := (H^s(\Omega))^2$ and $H^1_0(\Omega) := \{u \in H^1(\Omega) : u = 0 \text{ on } \partial\Omega\}$, and similarly for $\mathbb{H}^1_0(\Omega)$.

2. Finite element discretization. For ease of notation we define the four forms

$$a_1(c,u,w) := \left(\frac{b(c)}{\rho(c)}\nabla u, \nabla w\right), \qquad a_2(\mathbf{u},v,w) := (\mathbf{u}\cdot\nabla v, w),$$

$$a_3(\mathbf{u},\mathbf{v},\mathbf{w}) := ((\mathbf{u}\cdot\nabla)\mathbf{v},\mathbf{w}), \qquad a_4(c,u,\mathbf{v}) := \left(\frac{1-c^2}{4}\nabla u, \mathbf{v}\right),$$

and we additionally define $\hat{w} := w/\rho(c)$. Using these, a weak formulation of (1.4a)–(1.4f) can be written as

($\mathbf{P}^{weak}$) Find $\{c(x,t), w(x,t), \mathbf{u}(x,t), p(x,t)\} \in H^1(\Omega)\times H^1(\Omega)\times \mathbb{H}^1_f(\Omega)\times L^2_0(\Omega)$ such that

$$\left(\frac{\partial c}{\partial t}, \eta\right) + a_2(\mathbf{u},c,\eta) + \frac{1}{Pe}a_1(c,\hat{w},\eta) = 0 \quad \forall\, \eta \in H^1(\Omega), \tag{2.1a}$$

$$-(w,\eta) + \frac{1}{\epsilon}(\Phi'(c),\eta) + \epsilon(\nabla c, \nabla\eta) = 0 \quad \forall\, \eta \in H^1(\Omega), \tag{2.1b}$$

$$\begin{aligned}\left(\rho(c)\frac{\partial \mathbf{u}}{\partial t}, \chi\right) + a_3(\rho(c)\mathbf{u},\mathbf{u},\chi) + \nu(\nabla\mathbf{u},\nabla\mathbf{u}) - (p, \nabla\cdot\chi)\\ = Ka_2(\chi,c,w) + \delta K a_4(c,\hat{w},\chi) + (\rho(c)\mathbf{g},\chi) \qquad \forall\, \chi \in \mathbb{H}^1_0(\Omega),\end{aligned} \tag{2.1c}$$

$$(\nabla\cdot\mathbf{u},\xi) = 0 \quad \forall\, \xi \in L^2(\Omega). \tag{2.1d}$$

REMARK 2.1. *Using the incompressibility constraint* (2.1d) *and the boundary conditions for c, we can integrate the convection term in* (2.1a) *to get an equivalent weak form* ($\widetilde{\mathbf{P}}^{weak}$) *to* ($\mathbf{P}^{weak}$)*, where* (2.1a) *is replaced by*

$$\left(\frac{\partial c}{\partial t}, \eta\right) - a_2(\mathbf{u},\eta,c) + \frac{1}{Pe}a_1(c,\hat{w},\eta) = 0 \qquad \forall\, \eta \in H^1(\Omega). \tag{2.2}$$

Both of these formulations will be considered below.

We make the following assumptions on the finite element mesh: let $\mathcal{T}_h$ be a quasi-uniform partitioning of Ω into disjoint open simplices $\{\kappa\}$ with $h_\kappa := diam(\kappa)$ and $h := \max_{\kappa\in\mathcal{T}^h} h_\kappa$, so that $\overline{\Omega} = \cup_{\kappa\in\mathcal{T}^h}\overline{\kappa}$. This triangulation has been refined from an initial coarse macrotriangulation to a refinement level l, leading to a sequence of nested meshes. Additionally we assume that $\mathcal{T}^h$ is weakly acute; that is, for any pair of adjacent triangles the sum of opposite angles relative to the common side does not exceed π. The mesh is then refined once more to produce a finer triangulation $\mathcal{T}^{h/2}$ which also has these properties. We also need the space associated with the triangulation $\mathcal{T}^{2h}$ which is one level coarser than the triangulation $\mathcal{T}^h$.

Associated with $\mathcal{T}^h$, $\mathcal{T}^{h/2}$, and $\mathcal{T}^{2h}$ are the finite element spaces

$$
\begin{aligned}
S^h &:= \{v \in C(\overline{\Omega}) : v|_\kappa \text{ is a polynomial of degree at most } 1\ \forall\,\kappa \in \mathcal{T}^h\} \subset H^1(\Omega),\\
Y^h &:= \{v \in S^h : (v,1) = 0\},\\
X^h &:= \Big\{v \in C(\overline{\Omega}) : v|_\kappa \text{ is a polynomial of degree at most } 1\ \forall\,\kappa \in \mathcal{T}^{h/2}\\
&\qquad \text{and } v = 0 \text{ on } \partial\Omega\Big\} \subset H^1_0(\Omega),\\
V^h &:= \{v \in C(\overline{\Omega}) : v|_\kappa \text{ is a polynomial of degree at most } 1\ \forall\,\kappa \in \mathcal{T}^{2h},\\
&\qquad\qquad (v,1) = 0\} \subset H^1(\Omega).
\end{aligned}
$$

Let $\pi^h : C(\overline{\Omega}) \mapsto S^h$ be the piecewise linear Lagrange interpolation operator such that $(\pi^h v)(x_j) = v(x_j)$ for all $j = 1,\dots,J$, and similarly for $\pi^{h/2} : C(\overline{\Omega}) \mapsto X^h$.

We define a discrete $L^2(\Omega)$ inner product on $C(\Omega)$ via

$$
(u,v)_h := \int_\Omega [\pi^h(uv)](x_j)\,\mathrm{d}\mathbf{x} \equiv \sum_{j=1}^{J} \hat{M}_{jj} u(x_j) v(x_j) \quad \forall\, u, v \in C(\Omega), \tag{2.3}
$$

where $\hat{M}_{jj} := (\phi_j, 1) \equiv (\phi_j, \phi_j)_h$. It is easy to verify that

$$
(\pi^h \eta, \chi)_h = (\eta, \chi)_h \qquad \forall\, \eta, \chi \in C(\overline{\Omega}). \tag{2.4}
$$

Finally for the discrete time derivative we shall write

$$
\partial\eta^n := \frac{\eta^n - \eta^{n-1}}{\tau}.
$$

We can now discretize the system using continuous piecewise linear elements for the Cahn–Hilliard part (2.1a)–(2.1b) and P1-iso-P2/P1 elements for the Navier–Stokes part (2.1c)–(2.1d). These elements are linear in both the velocity and pressure parts but with the pressures being evaluated on the nodes of $\mathcal{T}^h$ and the velocity on the nodes of $\mathcal{T}^{h/2}$; see Figure 2.1.

These elements are known to satisfy the inf-sup stability condition [9],

$$
\inf_{q\in S^h} \sup_{\mathbf{v}\in(X^h)^2} \frac{(q, \nabla\cdot\mathbf{v})}{\|\mathbf{v}\|_{X^h}\|q\|_{S^h}} \geq k_h \geq k_0 > 0 \quad \forall\, \mathbf{v} \in (X^h)^2,\ q \in S^h, \tag{2.5}
$$

where k_0 is independent of h, with similar results if we replace X^h with S^h and S^h with V^h.

The fully discrete problem that we are left to solve is the following:

($\mathbf{P}^{h,\tau}$) Given $c^{h,0}(x) = \pi^h c_0(x)$, $\mathbf{u}^{h,0}(x) = \pi^{h/2}\mathbf{u}_0(x)$, for $n = 1,\dots,N_T$ find

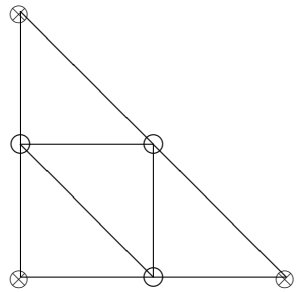


FIG. 2.1. *P1-iso-P2/P1 element:* ∘ = *velocity node,* × = *velocity and pressure node.*

$\{c_h^n, w_h^n, \mathbf{u}_h^n, p_h^n\} \in S^h \times S^h \times (X^h)^2 \times Y^h$ such that

$$(\partial c_h^n, \eta)_h + a_2(\mathbf{u}_h^{n-1}, c_h^n, \eta) + \frac{1}{Pe} a_1(c_h^{n-1}, \hat{w}_h^n, \eta) = 0 \quad \forall\, \eta \in S^h, \tag{2.6a}$$

$$-(w_h^n, \eta)_h + \frac{1}{\epsilon}(\phi(c_h^n) - c_h^{n-1}, \eta)_h + \epsilon(\nabla c_h^n, \nabla \eta) = 0 \quad \forall\, \eta \in S^h, \tag{2.6b}$$

$$(\rho(c_h^n)\partial \mathbf{u}_h^n, \chi) + a_3(\rho(c_h^n)\mathbf{u}_h^{n-1}, \mathbf{u}_h^n, \chi) + \nu(\nabla \mathbf{u}_h^n, \nabla \chi)$$
$$-(p_h^n, \nabla \cdot \chi) = \tilde{\mathbf{f}} \quad \forall\, \chi \in (X^h)^2, \tag{2.6c}$$

$$(\nabla \cdot \mathbf{u}_h^n, \xi) = 0 \quad \forall\, \xi \in S^h \tag{2.6d}$$

with

$$\tilde{\mathbf{f}} = Ka_2(\chi, c_h^n, w_h^n) + \delta K a_4(c_h^n, \hat{w}_h^n, \chi) + (\rho(c_h^n)\mathbf{g}, \chi) \tag{2.7}$$

and where we have used the splitting

$$\Phi'(c_h^n) = (c_h^n)^3 - c_h^{n-1} =: \phi(c_h^n) - c_h^{n-1}. \tag{2.8}$$

This splitting is chosen to ensure the stability of the Cahn–Hilliard part; see [11]. We note that using this discretization scheme on the problem $(\widetilde{\mathbf{P}}^{weak})$ leads to the following discrete scheme:

$(\widetilde{\mathbf{P}}^{h,\tau})$ Given $c^{h,0}(x) = \pi^h c_0(x)$, $\mathbf{u}^{h,0}(x) = \pi^h \mathbf{u}_0(x)$, for $n = 1, \ldots, N_T$ find $\{c_h^n, w_h^n, \mathbf{u}_h^n, p_h^n\} \in S^h \times S^h \times (S^h \cap H^1(\Omega))^2 \times V^h$ such that

$$(\partial c_h^n, \eta)_h - a_2(\mathbf{u}_h^{n-1}, \eta, c_h^n) + \frac{1}{Pe} a_1(c_h^{n-1}, \hat{w}_h^n, \eta) = 0 \quad \forall\, \eta \in S^h, \tag{2.9a}$$

$$-(w_h^n, \eta)_h + \frac{1}{\epsilon}(\phi(c_h^n) - c_h^{n-1}, \eta)_h + \epsilon(\nabla c_h^n, \nabla \eta) = 0 \quad \forall\, \eta \in S^h, \tag{2.9b}$$

$$(\rho(c_h^n)\partial \mathbf{u}_h^n, \chi) + a_3(\rho(c_h^n)\mathbf{u}_h^{n-1}, \mathbf{u}_h^n, \chi) + \nu(\nabla \mathbf{u}_h^n, \nabla \chi)$$
$$-(p_h^n, \nabla \cdot \chi) = \tilde{\mathbf{f}} \quad \forall\, \chi \in (S^h \cap H_0^1)^2, \tag{2.9c}$$

$$(\nabla \cdot \mathbf{u}_h^n, \xi) = 0 \quad \forall\, \xi \in V^h. \tag{2.9d}$$

The discrete schemes $(\mathbf{P}^{h,\tau})$ and $(\widetilde{\mathbf{P}}^{weak})$ are not equivalent since by integrating the convection term in the Cahn–Hilliard equation in $(\widetilde{\mathbf{P}}^{weak})$ by parts we have that

$$\begin{aligned} a_2(\mathbf{u}_h^{n-1}, \eta, c_h^n) &= -(\eta, \nabla \cdot (\mathbf{u}_h^{n-1} c_h^n)) \\ &= -a_2(\mathbf{u}_h^{n-1}, c_h^n, \eta) - (\eta, c_h^n \nabla \cdot \mathbf{u}_h^{n-1}) \\ &\neq -a_2(\mathbf{u}_h^{n-1}, c_h^n, \eta), \end{aligned}$$

as $c_h^n \eta \notin V^h$, so we cannot use (2.9d) to eliminate this term. Thus these two discretizations will give different results, so will investigate both options with respect to the efficiency of the solver.

Since we use a multigrid solver for the convection term in the Navier–Stokes equations we stabilize (2.6c) with a streamline-diffusion method [16]. This involves adding an additional term of the form

$$\gamma(\mathbf{u}_h^{n-1} \cdot \nabla \mathbf{u}_h^n, \mathbf{u}_h^{n-1} \cdot \nabla \chi)$$

into the left-hand side of (2.6c) where γ is the regularization parameter. The local parameter γ_κ is determined by the local mesh Péclet number

$$Pe_\kappa = Re \|\mathbf{u}_h^{n-1}\|_{\infty,\kappa} h_\kappa$$

using

$$\gamma_\kappa = \begin{cases} \frac{1}{2} h_\kappa (1 - \frac{1}{Pe_\kappa}) & \text{if} \quad Pe_\kappa > 1, \\ 0 & \text{if} \quad Pe_\kappa \leq 1. \end{cases}$$

This choice of stabilization aids the convergence of the iterative solver used in the Navier–Stokes equations; see [14], where this choice of parameter is shown to be optimal.

One of the important features of the Cahn–Hilliard equation without flow ($\mathbf{u} \equiv \mathbf{0}$) is that it conserves mass, that is, $\partial c / \partial t = 0$ (take $\eta = 1$ in (2.1a) to show this), and we wish this to be carried over to the discrete case. Both of the discretizations given here can be shown to retain this property as shown in the following lemma.

LEMMA 2.2. *The schemes* $(\mathbf{P}^{weak})$, $(\widetilde{\mathbf{P}}^{weak})$, $(\mathbf{P}^{h,\tau})$, *and* $(\widetilde{\mathbf{P}}^{h,\tau})$ *all conserve mass.*

Proof. For $(\widetilde{\mathbf{P}}^{weak})$ and $(\widetilde{\mathbf{P}}^{h,\tau})$ this comes trivially by taking $\eta = 1$ in the Cahn–Hilliard equation. For $(\mathbf{P}^{weak})$ take $\eta = 1$ in (2.1a), integrate by parts, and use the boundary conditions and the incompressibility constraint (2.1d) to yield the desired result. For $(\mathbf{P}^{h,\tau})$ we take $\eta = 1$ in (2.6a) and, noting that $c_h^n \in S^h$, we can integrate by parts and use (2.6d) to get the desired result. ∎

REMARK 2.3. *We note that if the concentration and velocities were evaluated at the same mesh in scheme* $(\mathbf{P}^{h,\tau})$*, then the pressure would be evaluated on a coarser mesh,* V^h*, so the concentration would not belong to the pressure space, and thus mass would not be conserved. This can be observed in practice and leads to incorrect numerical results. However, such a restriction does not apply for scheme* $(\widetilde{\mathbf{P}}^{\mathbf{h},\tau})$*, which is therefore computationally faster as* c *and* $\mathbf{u}$ *are evaluated on the same mesh of size* h *rather than the velocity being evaluated on the mesh of size* $h/2$*.*

3. Solution of the discrete system. We describe only the scheme $(\mathbf{P}^{h,\tau})$ here; the solution method for $(\widetilde{\mathbf{P}}^{h,\tau})$ is similar, with the relevant changes of finite dimensional spaces.

The system (2.6a)–(2.6d) is solved via a two-stage process. By taking the velocity term in (2.6a) explicitly, the Cahn–Hilliard equations decouple from the Navier–Stokes equations, allowing them to be solved separately. First the Cahn–Hilliard part is solved via a nonlinear multigrid solver (cf. [18]) and then the Navier–Stokes part is solved using these new values for c and w.

3.1. The Cahn–Hilliard solver. The Cahn–Hilliard part requires us to solve the following system:

$$(\partial c_h^n, \eta)_h + a_2(\mathbf{u}_h^{n-1}, c_h^n, \eta) + \frac{1}{Pe} a_1(c_h^{n-1}, \hat{w}, \eta) = 0, \tag{3.1a}$$

$$-(w_h^n, \eta)_h + \frac{1}{\epsilon}(\phi(c_h^n) - c_h^{n-1}, \eta)_h + \epsilon(\nabla c_h^n, \nabla \eta) = 0. \tag{3.1b}$$

Using the stiffness matrix A, mobility dependent stiffness matrix D, convection matrix N, and lumped mass matrix $\hat{M}$ given by

$$A(i,j) := \int_\Omega \nabla\psi_i \cdot \nabla\psi_j \ \mathrm{d}\mathbf{x}, \qquad D(i,j) := \int_\Omega \frac{b}{\rho} \nabla\psi_i \cdot \nabla\psi_j \ \mathrm{d}\mathbf{x},$$

$$N(i,j) := \int_\Omega \mathbf{u}_h^{n-1} \cdot \nabla\psi_i \ \psi_j \ \mathrm{d}\mathbf{x}, \qquad \hat{M}(i,i) := \int_\Omega \psi_i \, \mathrm{d}\mathbf{x},$$

where $S^h = span\{\psi_i\}_{i=1}^{N_p}$, the discrete Cahn–Hilliard problem can be written as

$$\hat{M}c^n + \tau N(\mathbf{u}^{n-1})c^n + \frac{\tau}{Pe} D\hat{w}^n = b, \tag{3.2a}$$

$$-\hat{M}w^n + \frac{1}{\epsilon}\hat{M}\boldsymbol{\phi}(c^n) + \epsilon A c^n = \frac{1}{\epsilon} b, \tag{3.2b}$$

where $b = \hat{M}c^{n-1}$ and $\boldsymbol{\phi}$ is understood to operate pointwise. This problem is solved using the nonlinear successive subspace correction multigrid scheme described in [18]. The mass matrices have been lumped so that the nonlinear term $\hat{M}\boldsymbol{\phi}(c^n)$ is diagonal.

Decompose the finite element space such that

$$S^h = \sum_{l=1}^{L} S_l^h, \tag{3.3}$$

where the $S_l^h \subset S_{l+1}^h$ are obtained by successive (not necessarily uniform) refinements of the triangulation and $S_L^h = S^h$. Project the basis functions for each of these spaces onto the finest grid, given by S^h, using standard prolongation operators (e.g., [27]). The algorithm is then as follows: looping over all of the basis functions η update c and w by

$$c^{n,k+1} = c^{n,k} + \alpha\eta, \quad w^{n,k+1} = w^{n,k} + \beta\eta,$$

where α, β are the solution of

$$(M(c^{n,k} + \alpha\eta), \eta) + \tau(Nc^{n,k}, \eta) + \tau(D(\hat{w}^{n,k} + \beta\hat{\eta}), \eta) - (b, \eta) = 0, \tag{3.4a}$$

$$\epsilon(A(c^{n,k} + \alpha\eta), \eta) + \left(\frac{1}{\epsilon} M\boldsymbol{\phi}(c^{n,k} + \alpha\eta) - \frac{1}{\epsilon} b - M(w^{n,k} + \beta\eta), \eta\right) = 0, \tag{3.4b}$$

with $(x, y) = x^T y$ the Euclidean inner product.

We can eliminate β from these equations to obtain a cubic polynomial in α which is solved using a Newton iteration. This is repeated until all the basis functions have been used in an update and then repeated until the relative residual of (2.6a) has been decreased by a set tolerance, *tol*. One multigrid iteration numerically costs about the same as one Gauss–Seidel iteration [18]; thus any decrease in iteration counts represents a speedup in computational time. This method is seen to be optimally efficient with respect to the problem parameters; see [18], where many numerical examples are presented.

3.2. The Navier–Stokes solver. For the Navier–Stokes part of the scheme we need to solve the following problem at each time step:

$$\frac{1}{\tau}(\mathbf{u}_h^n\chi) + a_3(\rho(c_h^n)\mathbf{u}_h^{n-1}, \mathbf{u}_h^n, \chi) + \nu(\nabla\mathbf{u}_h^n, \nabla\chi) - (p_h^n, \nabla\cdot\chi) = \mathbf{f}, \tag{3.5a}$$

$$(\nabla\cdot\mathbf{u}_h^n, \xi) = 0, \tag{3.5b}$$

where

$$\mathbf{f} := Ka_2(\chi, c_h^n, w_h^n) + \delta Ka_4(c_h^{n-1}, \hat{w}_h^n, \chi) + \left(\rho(c_h^n)\left[\mathbf{g} + \frac{1}{\tau}\mathbf{u}_h^{n-1}\right], \chi\right).$$

This is a generalized Oseen problem, for which solution methods have been extensively studied. We choose the F_p preconditioned GMRES method developed by Kay, Loghin, and Wathen [17].

The F_p preconditioner. Using the stiffness matrix A, convection matrix N, divergence matrices B_x and B_y, and mass matrix M given by

$$A(i,j) := \int_\Omega \nabla\varphi_i\cdot\nabla\varphi_j \ \mathrm{d}\mathbf{x}, \qquad N(i,j) := \int_\Omega \mathbf{u}_h^{n-1}\cdot\nabla\varphi_i \ \varphi_j \ \mathrm{d}\mathbf{x},$$
$$B_x(i,j) := -\int_\Omega \frac{\partial\varphi_i}{\partial x}\psi_j \ \mathrm{d}\mathbf{x}, \qquad B_y(i,j) := -\int_\Omega \frac{\partial\varphi_i}{\partial y}\psi_j \ \mathrm{d}\mathbf{x},$$
$$M(i,j) := \int_\Omega \varphi_i\varphi_j \ \mathrm{d}\mathbf{x},$$

where $X^h = span\{\varphi_i\}_{i=1}^{N_v}$, and additionally letting the matrix N absorb the streamline diffusion term, we can write the Navier–Stokes system as the following matrix problem:

$$\mathcal{A}\mathbf{x} = \hat{\mathbf{f}}, \tag{3.6}$$

where

$$\mathcal{A} = \begin{pmatrix} \mathcal{F} & B^T \\ B & 0 \end{pmatrix}, \qquad \mathcal{F} = \begin{pmatrix} F & 0 \\ 0 & F \end{pmatrix}, \qquad B = (B_x \ B_y),$$
$$F = \frac{1}{\tau}\rho M + \nu A + \rho N, \qquad \mathbf{x} = (\mathbf{u}^n, p^n)^t, \qquad \hat{\mathbf{f}} = (\mathbf{f}, \mathbf{f}_{bc})^t,$$

and the extra term $\mathbf{f}_{bc}$ arises from the imposition of the Dirichlet boundary conditions.

This is a difficult problem to solve as it is a large nonsymmetric indefinite system. We will use GMRES (see [25]) with right preconditioning; i.e., we solve the problem

$$(\mathcal{A}\mathcal{P}^{-1})(\mathcal{P}\mathbf{x}) = \hat{\mathbf{f}}, \tag{3.7}$$

where the preconditioner $\mathcal{P}$ is chosen such that its eigenvalues are clustered and it is easily invertible. The optimal choice of preconditioner is given by [23]

$$\mathcal{P}^{-1} = \begin{pmatrix} \mathcal{F}^{-1} & \mathcal{F}^{-1}B^tS^{-1} \\ 0 & -S^{-1} \end{pmatrix}, \tag{3.8}$$

with $S = B\mathcal{F}^{-1}B^t$, so that

$$\mathcal{A}\mathcal{P}^{-1} = \begin{pmatrix} I & 0 \\ B\mathcal{F}^{-1} & 0 \end{pmatrix}, \tag{3.9}$$

and $\sigma(\mathcal{A}\mathcal{P}^{-1}) = \{1\}$ so the GMRES algorithm converges in at most two iterations. Then at each step we need to solve $\mathcal{P}y = r$, i.e., find v, q such that

$$\begin{pmatrix} v \\ q \end{pmatrix} = \begin{pmatrix} \mathcal{F}^{-1} & \mathcal{F}^{-1}B^tS^{-1} \\ 0 & -S^{-1} \end{pmatrix} \begin{pmatrix} r \\ s \end{pmatrix}. \tag{3.10}$$

This is a two-stage process: first solve for q: $Sq = -s$, then for v: $\mathcal{F}v = r - B^tq$. In practice we replace S and $\mathcal{F}$ with approximations S_* and $\mathcal{F}_*$. The difficulty lies in obtaining a good yet simple approximation for the Schur complement S. Kay, Loghin, and Wathen [17] developed the F_p preconditioner for this purpose. This replaces the inverse of the Schur complement with

$$S_*^{-1} = M_{p*}^{-1}F_pA_{p*}^{-1}, \tag{3.11}$$

where M_{p*}^{-1}, A_{p*}^{-1} are approximations to the inverses of the mass and stiffness matrices on the pressure grid, respectively. M_{p*}^{-1} is obtained by five diagonally scaled conjugate gradient iterations, while A_{p*}^{-1} is obtained via a single multigrid v-cycle applied to A_p, the pressure Laplacian.

This choice of preconditioner is made because it has convergence rates that are independent of mesh size h with only a mild dependence on ν; see [17]. For further details on this scheme see [12, 26].

4. Numerical examples. All of the examples presented were performed on the domain $\Omega = [0,1]^2$, with a solver tolerance of $tol = 1 \times 10^{-6}$. They all use scheme $(\mathbf{P}^{h,\tau})$ except for the final section, where $(\widetilde{\mathbf{P}}^{h,\tau})$ is used.

In the tables of iteration counts the first number is the maximum number of v-cycles for the Cahn–Hilliard solver that were obtained over the course of the simulation, and the second number is the maximum number of GMRES iterations obtained for the Navier–Stokes solver over the course of the simulation. The figures given below illustrate the concentration field c and the velocity streamlines.

4.1. Spinodal decomposition under constant shear. We start by considering the case of two fluids with equal density undergoing spinodal decomposition in a constant shear. The initial data for c is given by a random perturbation of magnitude 0.1 about a mean composition of 0; see Figure 4.1. The velocity is taken to be either $(1,0)^t$ or $(10,0)^t$ throughout. This leads to the stripes forming, and the faster velocity yields more stripes; see Figure 4.2. This first example does not satisfy the boundary conditions for $\mathbf{u}$, but it gives an insight into the performance of the Cahn–Hilliard solver when a velocity is introduced.

The iteration counts for the Cahn–Hilliard solver are given in Table 4.1. It is clear that the iteration counts do not depend on the velocity and are independent of the mesh size, showing that the solver is mesh independent.

4.2. Rayleigh–Taylor instability. Rayleigh–Taylor instability is caused when a heavier fluid is placed upon a lighter one; see Figure 4.3. We examine several cases of this to show how the proposed solver performs during this complex simulation. Since the interesting stages of the evolution do not occur while the solution remains of the form of Figure 4.3, we run this until the initial condition shown in Figure 4.4 is achieved. This is where the most interesting characteristics of the evolution occur such as coalescence and pinch-off. From this initial condition a fixed number of time steps for each problem were calculated, with the various problem parameters varied.

First we examine the effect of the interface width; see Table 4.2. Recalling that the interface width is $O(\epsilon)$, we see that this parameter does not affect the performance

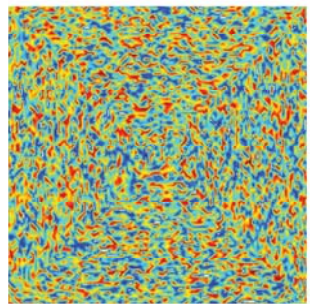


FIG. 4.1. *Initial condition for spinodal decomposition.*

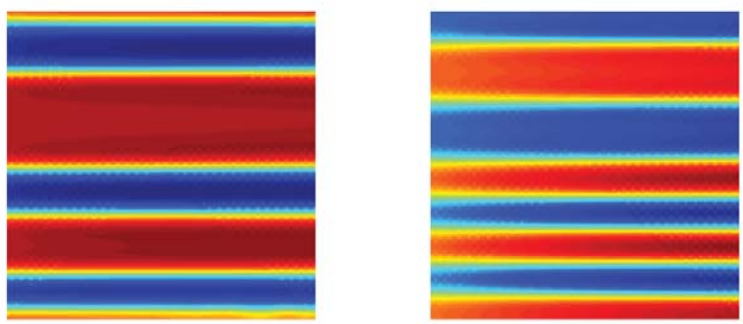


FIG. 4.2. *Fixed shear spinodal decomposition,* $\mathbf{u} = (1, 0)^t$ *and* $\mathbf{u} = (10, 0)^t$.

TABLE 4.1
$\tau = 0.001$, $\epsilon = 0.02$, $Pe = 1000$.

Velocity \ h	1/16	1/32	1/64	1/128
1	4	6	8	9
10	4	6	8	10

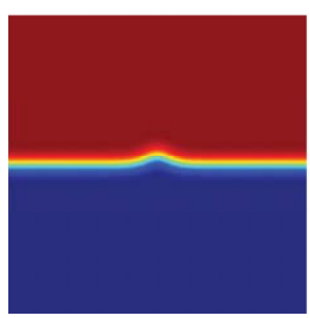


FIG. 4.3. *Starting configuration for Rayleigh–Taylor instability.*

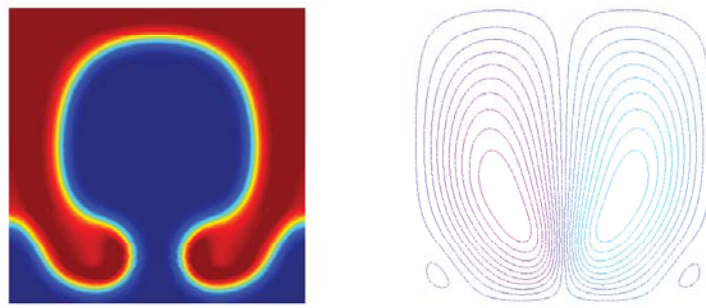


FIG. 4.4. *Initial condition for Rayleigh–Taylor instability.*

TABLE 4.2
$\tau = 0.0005$, $Pe = 1000$, $\nu = 0.01$, $\delta = 0.99$, $K = 0.001$.

$h \setminus \epsilon$	0.08	0.04	0.02	0.01
1/16	5(40)	4(37)	4(37)	5(36)
1/32	6(37)	6(34)	6(34)	7(34)
1/64	7(35)	7(35)	8(35)	8(34)
1/128	8(37)	8(37)	8(37)	8(36)

TABLE 4.3
$\tau = 0.005$, $Pe = 1000$, $\epsilon = 0.04$, $\delta = 0.99$, $K = 0.001$.

$h \setminus \nu$	1	0.1	0.01	0.001
1/16	6(24)	6(27)	6(31)	6(34)
1/32	8(25)	8(28)	7(31)	8(35)
1/64	9(27)	8(30)	8(32)	9(39)
1/128	9(31)	9(32)	9(36)	9(38)

TABLE 4.4
Reynolds numbers.

$h \setminus \nu$	1	0.1	0.01	0.001
1/64	0.04	10.06	385.2	606.2

of the solver. The iteration counts for both the Cahn–Hilliard and Navier–Stokes equations are unaffected by mesh refinement and interface width. Thus with respect to these parameters we can see that the solver performs optimally.

For the second test the viscosity ν was varied; see Table 4.3. Again, with respect to mesh refinement the solver performs well. The increase in iteration counts for the GMRES solver as ν is decreased is due to the Reynolds number increasing; see Table 4.4. This is to be expected, as it is known that the F_p preconditioner has a slight dependence on the Reynolds number [17].

The third test involved varying the densities of the two fluids. The values of δ chosen for this test were 0.5, 0.9, 0.99, and 0.999, representing ratios of fluid densities of 2 : 1, 10 : 1, 100 : 1, and 1000 : 1, respectively. Again we see that the solver performed well with respect to mesh refinement, especially for the higher ratios which are of more practical interest; see Table 4.5.

TABLE 4.5
$\tau = 0.005$, $Pe = 1000$, $\epsilon = 0.04$, $\nu = 0.01$, $K = 0.001$.

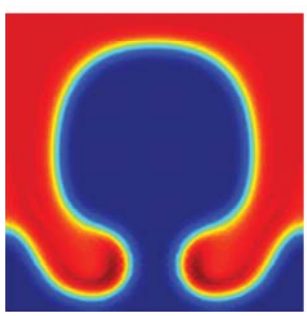
$h \setminus \delta$	0.5	0.9	0.99	0.999
1/16	5(13)	5(21)	6(31)	6(34)
1/32	6(17)	7(22)	7(31)	8(34)
1/64	8(20)	8(24)	8(32)	9(36)
1/128	9(21)	9(26)	9(32)	9(36)

TABLE 4.6
$\tau = 0.005$, $Pe = 1000$, $\epsilon = 0.04$, $\nu = 0.01$, $\delta = 0.99$.

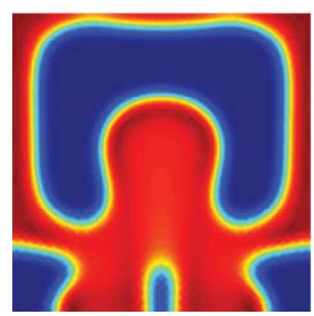
$h \setminus K$	0	0.001	0.01	0.1
1/16	6(31)	6(31)	6(31)	6(31)
1/32	8(31)	7(31)	8(31)	8(31)
1/64	9(33)	8(32)	9(33)	8(32)
1/128	9(33)	9(33)	9(33)	9(32)

The final test involved the capillary number K; see Table 4.6. The solvers were the most robust with respect to this parameter, showing that although it plays a vital part in the evolution of the flow (see [4] for an examination of this) it does not affect the solution method proposed here.

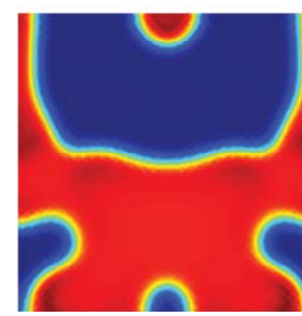
For completeness we conclude this section by giving the following figures showing a full example of Rayleigh–Taylor instability with $Pe = 1000$, $\epsilon = 0.02$, $\nu = 0.01$, $\delta = 0.99$, $K = 0.001$. These figures show the concentration field and streamlines of the velocity at the given times. The maximum velocity at each time is also shown to demonstrate how the velocity field evolves.



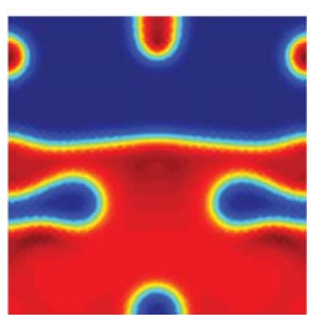
$t = 1$



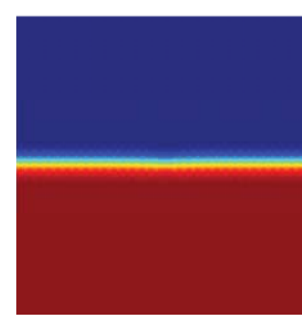
$t = 1.5$



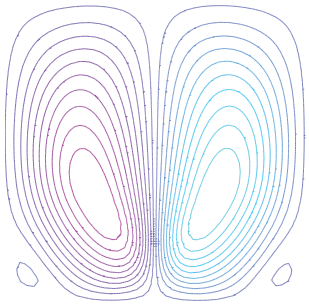
$t = 2$



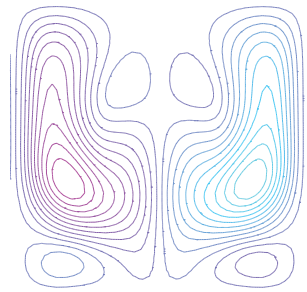
$t = 2.5$



$t = 25$



$t = 1$
$\|\mathbf{u}\|_\infty = 4.36$



$t = 1.5$
$\|\mathbf{u}\|_\infty = 1.35$

$t = 2$
$\|\mathbf{u}\|_\infty = 0.82$

$t = 2.5$
$\|\mathbf{u}\|_\infty = 0.66$

$t = 25$
$\|\mathbf{u}\|_\infty = 0.14$

4.3. Spinodal decomposition. For the second experiment we perform spinodal decomposition under a gravitational field. The initial data for c is again given by a random perturbation of magnitude 0.1 about a mean composition of 0 with $\mathbf{u}^0 = \mathbf{0}$; see Figure 4.1. For this problem we took $\epsilon = 0.02$, $\nu = 0.01$, $Pe = 1000$, $K = 0.001$, $\delta = 0.99$.

Following [18, 21] we set the time step to be $\tau = 0.001t^{1/4}$ to see how the solver can cope with variable time-step size. This scaling of the time-step size is based on the late stage growth law $l(t) \sim t^{1/4}$ that is followed by the evolution of surfaces undergoing motion by surface diffusion (Laplacian of the curvature). Motion by surface diffusion is the sharp-interface limit (that is, $\epsilon \to 0$) of the Cahn–Hilliard equation [7]. In the early stages of spinodal decomposition the solution rapidly evolves into regions where $c \approx \pm 1$. Then over much slower time scales, these "bulk" regions evolve and conglomerate into larger regions.

The iteration counts obtained (see Figure 4.5) show that as in [18] the Cahn–Hilliard part counts remain fixed. However the GMRES iteration counts grow as $t^{1/4}$, suggesting a relationship of number of iterations $\sim \tau$. The iteration count reaches a maximum that is dependent on the Reynolds number of the problem. It is known that the number of iterations for the F_p preconditioned GMRES grows with the time-step size (see [12]) so this behavior is expected.

The evolution of this problem is presented following Figure 4.5. We note that in the fourth image ($t = 9.5$) there is a small bubble of the lighter fluid trapped in the bottom right-hand corner of the domain. Such bubbles evolve over extremely long time scales [1], and therefore numerically we expect this to remain for all sensible computational times.

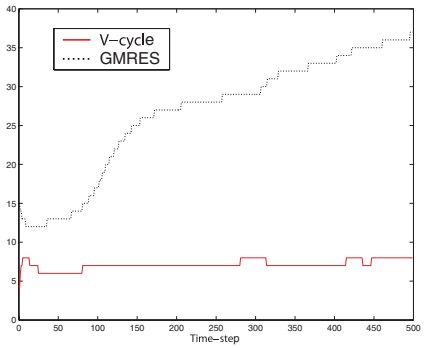
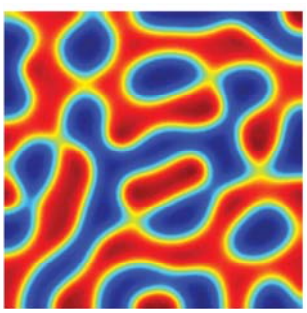
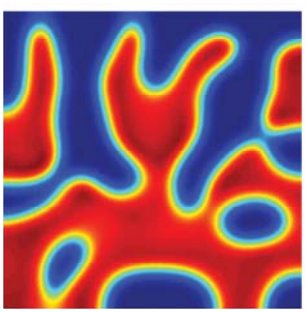


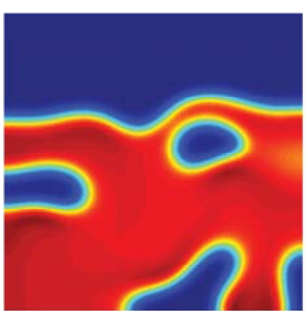
FIG. 4.5. *Iteration counts for variable time-step spinodal decomposition.*



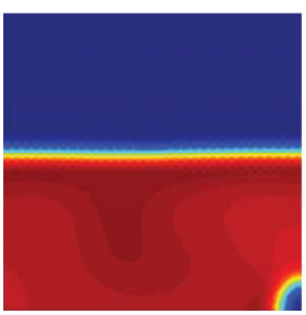
$t = 0.17$



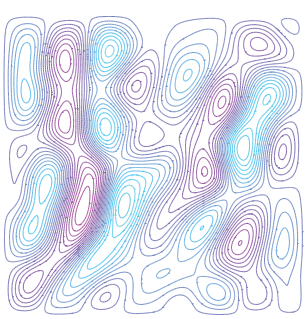
$t = 0.59$



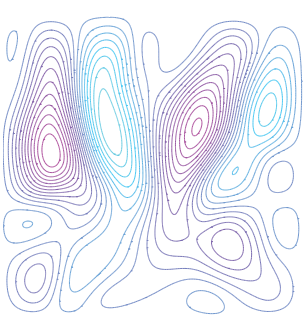
$t = 1.15$



$t = 9.5$

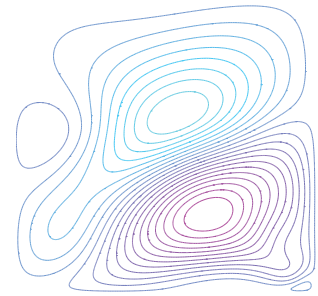


$t = 0.17$
$\|\mathbf{u}\|_\infty = 0.55$



$t = 0.59$
$\|\mathbf{u}\|_\infty = 1.37$

$t = 1.15$
$\|\mathbf{u}\|_\infty = 0.96$



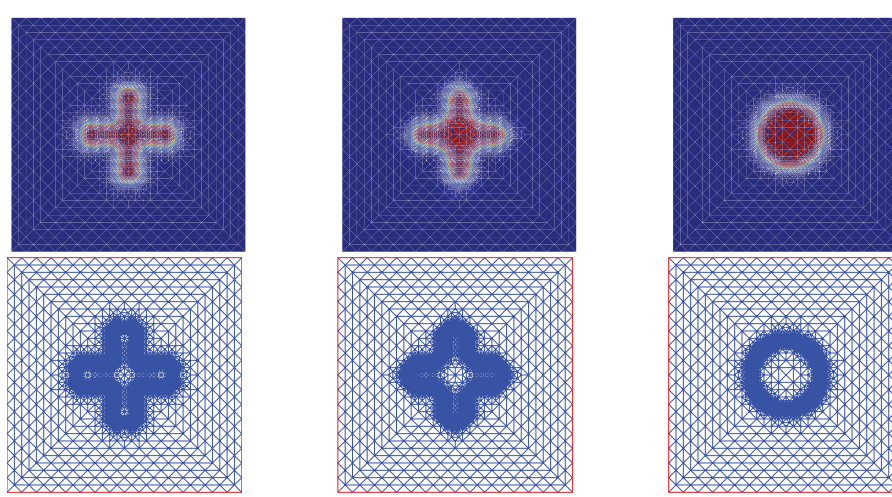
$t = 9.5$
$\|\mathbf{u}\|_\infty = 0.15$

4.4. Scheme $(\widetilde{\mathbf{P}}^{h,\tau})$. In this section we return to the Rayleigh–Taylor example of section 4.2 but with discretization scheme $(\widetilde{\mathbf{P}}^{h,\tau})$. The results are given in Table 4.7. We can see that the solver performs just as well with this discretization, retaining its optimality with respect to mesh refinement.

TABLE 4.7
$\tau = 0.005$, $Pe = 1000$, $\nu = 0.01$, $K = 0.001$, $\delta = 0.99$.

$h \setminus \epsilon$	0.08	0.04	0.02	0.01
1/16	5(37)	4(38)	4(44)	5(54)
1/32	8(29)	6(31)	6(39)	8(56)
1/64	7(26)	8(29)	7(33)	8(41)
1/128	8(29)	8(30)	8(33)	8(38)

4.5. Adaptivity. The final example is calculated using an adaptive mesh refinement scheme. Coupling the multigrid solver to an adaptive mesh refinement scheme provides us with a very efficient solution method for this problem. The evolution of the concentration and the mesh are shown below. Figure 4.6 shows the numbers of degrees of freedom (NDOFS), divided by 100, and the iteration counts for the Cahn–Hilliard and the Navier–Stokes solvers. Again we see that the solver performs well with this adaptive mesh scheme.



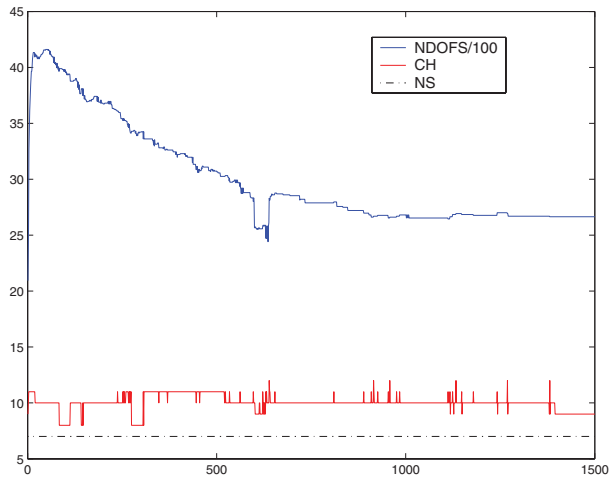


FIG. 4.6. *Number of degrees of freedom and iteration counts for the evolution of a cross.*

5. Conclusions. A new multigrid finite element solver for density-dependent Cahn–Hilliard fluids has been presented that exhibits iteration counts that are independent of mesh size h. Two discretizations of the system have been given, and the solver has been shown via numerical examples to be robust and efficient with respect to the problem parameters. In particular we note that the solver can be used in conjunction with an adaptive mesh refinement scheme to provide a greater increase in efficiency of the scheme.

Acknowledgment. The authors would like to thank the referees for their comments and suggestions.

REFERENCES

[1] N. D. ALIKAKOS AND G. FUSCO, *Slow dynamics for the Cahn-Hilliard equation in higher space dimensions: The motion of bubbles*, Arch. Rational Mech. Anal., 141 (1998), pp. 1–61.
[2] V. E. BADALASSI, H. D. CENICEROS, AND S. BANERJEE, *Computation of multiphase systems with phase field models*, J. Comput. Phys., 190 (2003), pp. 371–397.
[3] F. BOYER, *Nonhomogeneous Cahn-Hilliard fluids*, Ann. Inst. H. Poincaré Anal. Non Linéaire, 18 (2001), pp. 225–259.
[4] F. BOYER, *A theoretical and numerical model for the study of incompressible mixture flows*, Comput. & Fluids, 31 (2002), pp. 41–68.
[5] F. BOYER, L. CHUPIN, AND B. A. FRANCK, *Numerical study of viscoelastic mixtures through a Cahn-Hilliard fluid*, Eur. J. Mech. B Fluids, 23 (2004), pp. 759–780.
[6] J. W. CAHN, *On spinodal decomposition*, Acta Metall. Mater., 9 (1961), pp. 795–801.
[7] J. W. CAHN, C. M. ELLIOTT, AND A. NOVICK-COHEN, *The Cahn-Hilliard equation with a concentration dependent mobility: Motion by minus the Laplacian of the mean curvature*, European J. Appl. Math., 7 (1996), pp. 287–301.
[8] J. W. CAHN AND J. E. HILLIARD, *Free energy of a nonuniform system.* I. *Interfacial free energy*, J. Chem. Phys., 28 (1958), pp. 258–267.
[9] P. CIARLET, JR., AND V. GIRAULT, *Condition inf-sup pour l'élément fini de Taylor-Hood P_2-iso-P_1, 3-D; application aux équations de Maxwell*, C. R. Math. Acad. Sci. Paris, 335 (2002), pp. 827–832.
[10] C. M. ELLIOTT, *The Cahn-Hilliard model for the kinetics of phase separation*, in Mathematical Models for Phase Change Problems, Internat. Ser. Numer. Math. 88, Birkhäuser-Verlag, Basel, Switzerland, 1989, pp. 35–73.
[11] C. M. ELLIOTT AND A. M. STUART, *The global dynamics of discrete semilinear parabolic equations*, SIAM J. Numer. Anal., 30 (1993), pp. 1622–1663.

[12] H. Elman, D. Kay, D. Silvester, and A. Wathen, *Efficient preconditioning of the linearized Navier-Stokes equations for incompressible flow*, J. Comput. Appl. Math., 128 (2001), pp. 261–279.

[13] X. Feng, *Fully discrete finite element approximations of the Navier–Stokes–Cahn–Hilliard diffuse interface model for two-phase fluid flows*, SIAM J. Numer. Anal., 44 (2006), pp. 1049–1072.

[14] B. Fischer, A. Ramage, D. J. Silvester, and A. J. Wathen, *On parameter choice and iterative convergence for stabilised discretisations of advection-diffusion problems*, Comput. Methods Appl. Mech. Engrg., 179 (1999), pp. 179–195.

[15] D. Jacqmin, *Calculations of two-phase Navier-Stokes flows using phase-field modelling*, J. Comput. Phys., 155 (1999), pp. 96–127.

[16] C. Johnson, U. Nävert, and U. Pitkäranta, *Finite element methods for linear hyperbolic problems*, Comput. Methods Appl. Mech. Engrg., 45 (1984), pp. 285–312.

[17] D. Kay, D. Loghin, and A. Wathen, *A preconditioner for the steady-state Navier–Stokes equations*, SIAM J. Sci. Comput., 24 (2002), pp. 237–256.

[18] D. Kay and R. Welford, *A multigrid finite element solver for the Cahn-Hilliard equation*, J. Comput. Phys., 212 (2006), pp. 288–304.

[19] J. Kim, *A diffuse-interface model for axisymmetric immiscible two-phase flow*, Appl. Math. Comput., 160 (2005), pp. 589–606.

[20] J. Kim, K. Kang, and J. Lowengrub, *Conservative multigrid methods for Cahn-Hilliard fluids*, J. Comput. Phys., 193 (2004), pp. 357–379.

[21] R. V. Kohn and F. Otto, *Upper bounds on coarsening rates*, Comm. Math. Phys., 229 (2002), pp. 375–395.

[22] J. Lowengrub and L. Truskinovsky, *Quasi-incompressible Cahn-Hilliard fluids and topological transitions*, R. Soc. Lond. Proc. Ser. A Math. Phys. Eng. Sci., 454 (1998), pp. 2617–2654.

[23] M. F. Murphy, G. H. Golub, and A. J. Wathen, *A note on preconditioning for indefinite linear systems*, SIAM J. Sci. Comput., 21 (2000), pp. 1969–1972.

[24] J. F. Rodrigues, ed., *Mathematical Models for Phase Change Problems*, Internat. Ser. Numer. Math. 88, Birkhäuser-Verlag, Basel, Switzerland, 1989.

[25] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[26] A. J. Wathen, D. Loghin, D. Kay, H. C. Elman, and D. J. Silvester, *A new preconditioner for the Oseen equations*, in Proceedings of the European Conference on Numerical Mathematics and Advanced Applications, Springer-Verlag, Berlin, 2001.

[27] P. Wesseling, *An Introduction to Multigrid Methods*, John Wiley and Sons, Chichester, England, 1992.

SIAM J. SCI. COMPUT.
Vol. 29, No. 6, pp. 2258–2282
© 2007 Society for Industrial and Applied Mathematics

A DISCONTINUOUS GALERKIN MOVING MESH METHOD FOR HAMILTON–JACOBI EQUATIONS*

J. A. MACKENZIE† AND A. NICOLA‡

Abstract. In this paper we consider the numerical solution of first-order Hamilton–Jacobi equations using the combination of a discontinuous Galerkin finite element method and an adaptive r-refinement (mesh movement) strategy. Particular attention is given to the choice of an appropriate adaptivity criterion when the solution becomes discontinuous. Numerical examples in one and two dimensions are presented to demonstrate the effectiveness of the adaptive procedure.

Key words. adaptivity, moving meshes, discontinuous Galerkin finite elements, Hamilton–Jacobi equations

AMS subject classifications. 35F25, 35L45, 35L65, 65M50, 65M60

DOI. 10.1137/060656243

1. Introduction.

In this paper we consider the adaptive numerical solution of Hamilton–Jacobi (HJ) equations

$$\phi_t + H(\phi_{x_1}, \ldots, \phi_{x_d}) = 0, \quad \phi(\boldsymbol{x}, 0) = \phi_0(\boldsymbol{x}), \tag{1.1}$$

where $\boldsymbol{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$, $t > 0$. HJ equations arise in many practical areas such as differential games, mathematical finance, image enhancement, and front propagation. It is well known that solutions of (1.1) are Lipschitz continuous, but derivatives can become discontinuous even if the initial data is smooth. Since generalized solutions are not unique, a selection principle is required to pick out the physically relevant solution. For HJ equations the most commonly used condition is the vanishing viscosity condition, which requires that the correct solution should be the vanishing viscosity limit of smooth solutions of corresponding viscous problems. The notion of viscosity solutions was introduced by Crandall and Lions [8], where the questions of existence, uniqueness, and stability of solutions were addressed.

Crandall and Lions were also the first to study numerical approximations of (1.1) and introduced the important class of monotone finite difference methods which were shown to converge to the viscosity solution [7]. However, monotonic schemes are well known to be at most first-order accurate.

There is a close relation between HJ equations and hyperbolic conservation laws (see section 2.1 below). With this in mind, it is not surprising to find that many of the numerical methods used to solve HJ equations are motivated by conservative finite difference or finite volume methods for conservation laws. Methods that have been proposed include high-order essentially nonoscillatory (ENO) schemes [22], [23], weighted ENO schemes [15], and high-resolution central schemes [21].

An increasingly popular approach to solving hyperbolic conservation laws is the discontinuous Galerkin (DG) finite element method [5], [6]. Recently, Hu and Shu

*Received by the editors April 4, 2006; accepted for publication (in revised form) February 20, 2007; published electronically October 5, 2007.
http://www.siam.org/journals/sisc/29-6/65624.html

†Department of Mathematics, University of Strathclyde, Livingstone Tower, 26 Richmond Street, Glasgow, G1 1XH, UK (jam@maths.strath.ac.uk).

‡Faculty of Mathematics and Computer Science, University of Ovidius, Blvd. Mamaia 124, 900527, Constanta, Romania (anicola@univ-ovidius.ro).

[11] proposed a DG method to solve HJ equations by first rewriting (1.1) as a system of conservation laws

$$(w_i)_t + (H(\boldsymbol{w}))_{x_i} = 0, \quad i = 1, \ldots, d, \quad \boldsymbol{w}(\boldsymbol{x}, 0) = \nabla\phi_0(\boldsymbol{x}), \tag{1.2}$$

where $\boldsymbol{w} = \nabla\phi$. The usual DG formulation would be obtained if $\boldsymbol{w}$ belonged to a space of piecewise polynomials. However, we note that w_i, $i = 1, \ldots, d$, are not independent due to the restriction that $\boldsymbol{w} = \nabla\phi$. In [11] a least squares procedure was used to enforce this condition. More recently it was shown that it is possible to enforce the gradient condition using a smaller solution space [19]. Theoretical analysis of the accuracy and stability of the method was performed in [18].

One of the often cited advantages of DG methods is that since the numerical solution is not continuous across interelement boundaries, this, in theory, makes solution adaptive strategies much easier to implement. This has led to the development of a number of adaptive methods based on *hp*-refinement strategies for hyperbolic conservation laws [4], [10].

An alternative adaptive strategy that has worked well for time-dependent problems is to use moving meshes. A useful way to construct an adaptive moving mesh is to regard it as the image of a uniform mesh covering a computational domain under a time-dependent transformation that clusters mesh elements towards areas where improved spatial resolution is required. The transformation is often found through a variational formulation where the mapping is the minimizer of a functional involving properties of the mesh and the solution (see, e.g., [17], [29]). To improve the stability and smooth the evolution of the moving mesh, Russell, Huang, and coworkers [14], [12] found it useful to obtain the mapping as the solution of parabolic moving mesh PDEs (MMPDEs) which are modified gradient flow equations for the minimization of a suitable mesh functional.

For problems where solution discontinuities exist, the correct choice of an adaptivity criterion, or monitor function, is problematic [30]. It is not unusual to find in the literature that many grid adaptation criteria are singular at solution discontinuities. To prevent singularities from producing degenerate meshes, either some form of smoothing procedure is employed or a regularized functional is used in the variational formulation [1].

There are two main ways to solve PDEs using moving meshes. The first is to reformulate the governing equations with respect to the moving reference frame. If conservation is important, then this reformulation must be done carefully. Examples of this approach are arbitrary Lagrangian–Eulerian (ALE) methods which are commonly used for problem with moving domains. Alternatively one may evolve the numerical solution over one time step using a stationary mesh. Thereafter, the mesh is moved, and some form of interpolation procedure is used to transfer the solution values from one mesh to another. This approach has been used successfully in the work of Tang and coworkers [20], [31], [27] to solve hyperbolic problems where a suitable conservative form of interpolation was proposed.

The aim of this paper is to consider the use of the DG method of Hu and Shu [11] to solve HJ equations using a moving mesh method based on the solution of MMPDEs. The governing equation is transformed to include the effect of the movement of the mesh and this is done in such a way that the conservation properties of the original equation are not lost. The adaptive mesh is driven by a monitor function which is shown to be nonsingular in the presence of solution discontinuities. To produce an acceptable mesh we smooth the monitor function before it is used to drive the adaptive procedure.

The layout of the rest of this paper is as follows: in the next section we present the governing HJ equations and their discretization using a moving mesh and a DG formulation. In section 3 we present the MMPDEs and their discretization and we discuss the choice of the monitor function. Finally, in section 4 we conduct a number of numerical experiments in one and two dimensions.

2. DG discretization of HJ equations.

2.1. HJ equation in one dimension. Let us consider the HJ equation in one dimension:

$$\begin{cases} \phi_t + H(\phi_x) = 0, \quad (x,t) \in (a,b) \times (0,T], \\ \phi(x,0) = \phi_0(x), \end{cases} \tag{2.1}$$

with appropriate boundary conditions. If $u = \phi_x$ and we differentiate (2.1) with respect to x, then u satisfies the hyperbolic conservation law

$$\begin{cases} u_t + H(u)_x = 0, \quad (x,t) \in (a,b) \times (0,T], \\ u(x,0) = u_0(x). \end{cases} \tag{2.2}$$

We assume that the physical domain $\Omega_p = [a,b]$ is the image of a computational domain $\Omega_c = [0,1]$ which is obtained via the time-dependent mapping $x = x(\xi,t)$. If (2.2) is considered with respect to the moving coordinate frame, then

$$\dot{u} - \dot{x} u_x + H(u)_x = 0, \quad (x,t) \in (a,b) \times (0,T], \tag{2.3}$$

where $\dot{(\cdot)}$ represents differentiation with respect to time keeping ξ fixed. Note that the transformed equation is not in conservation form. Therefore, a discretization of (2.3) will not be conservative in general, even if the $H(u)_x$ term is treated conservatively, and this will lead to problems in convergence towards discontinuous solutions of (2.2) (see [24]). Therefore, instead we will consider a discretization of the conservative form in computational coordinates:

$$\begin{cases} \dot{(x_\xi} u) + (H(u) - \dot{x}\, u)_\xi = 0, \quad (\xi,t) \in (0,1) \times (0,T], \\ u(\xi,0) = u_0(\xi). \end{cases} \tag{2.4}$$

The aim is to discretize (2.4) in space using a DG method.

2.2. The DG discretization. For each partition $\{\xi_{j+\frac{1}{2}}\}_{j=0}^{N}$ of Ω_c, we denote

$$I_j = \left(\xi_{j-\frac{1}{2}}, \xi_{j+\frac{1}{2}}\right), \quad \Delta_j = \xi_{j+\frac{1}{2}} - \xi_{j-\frac{1}{2}}, \quad j = 1, \ldots, N.$$

We use a uniform mesh to cover the computational domain such that

$$h = \xi_{j+\frac{1}{2}} - \xi_{j-\frac{1}{2}} = \frac{1}{N}, \quad \xi_j = \frac{1}{2}\left(\xi_{j+\frac{1}{2}} + \xi_{j-\frac{1}{2}}\right), \quad j = 1, \ldots, N. \tag{2.5}$$

If we multiply (2.4) by an arbitrary smooth function v, then integrating by parts over the interval I_j we obtain

$$\int_{I_j} (\dot{x_\xi} u)\, v \, d\xi - \int_{I_j} \widetilde{H}(u, \dot{x})\, v_\xi \, d\xi + \widetilde{H}(u, \dot{x})\, v \Big|_{I_j} = 0, \tag{2.6}$$

where

$$\widetilde{H}(u,\dot{x}) = H(u) - \dot{x}\,u, \tag{2.7}$$

and

$$\widetilde{H}(u,\dot{x})v\Big|_{I_j} = \widetilde{H}(u(\xi_{j+\frac{1}{2}},t),\dot{x}(\xi_{j+\frac{1}{2}}))\,v(\xi_{j+\frac{1}{2}}) - \widetilde{H}(u(\xi_{j-\frac{1}{2}},t),\dot{x}(\xi_{j-\frac{1}{2}}))\,v(\xi_{j-\frac{1}{2}}). \tag{2.8}$$

In this paper we will approximate the exact solution u of (2.6) and the smooth function v locally by linear polynomials. The numerical approximations u_h and v_h belong to the finite dimensional space

$$V_h^1 = \left\{v : v|_{I_j} \in P^1(I_j), \quad j = 1,\dots,N\right\}, \tag{2.9}$$

where $P^1(I_j)$ denotes the space of polynomials on I_j of degree at most one. In the discontinuous Galerkin numerical method u_h and v_h are discontinuous at the points $\xi_{j+\frac{1}{2}}$, $j = 0,\dots,N$, and the question is how to evaluate their values in (2.8). We set

$$v_{h_{j+\frac{1}{2}}}^- = \lim_{\xi\to\xi_{j+\frac{1}{2}}^-} v_h(\xi) \quad \text{and} \quad v_{h_{j-\frac{1}{2}}}^+ = \lim_{\xi\to\xi_{j-\frac{1}{2}}^+} v_h(\xi). \tag{2.10}$$

Furthermore, we replace the nonlinear flux function $\widetilde{H}(u_h,\dot{x})$ defined in (2.7) by a numerical flux function that depends on the values of u_h from the left and right at the point $\left(\xi_{j+\frac{1}{2}},t\right)$, that is,

$$\widetilde{H}(u_h,\dot{x})_{\xi_{j+\frac{1}{2}}}(t) \approx \overline{H}_{j+\frac{1}{2}}\left(u_h\left(\xi_{j+\frac{1}{2}}^-,t\right),u_h\left(\xi_{j+\frac{1}{2}}^+,t\right),\dot{x}_{\xi_{j+\frac{1}{2}}}(t)\right). \tag{2.11}$$

In this paper we have used the local Lax–Friedrichs flux

$$\overline{H}(p,q,\dot{x}) = \frac{1}{2}\left(\widetilde{H}(p,\dot{x}) + \widetilde{H}(q,\dot{x}) - c(q-p)\right), \tag{2.12}$$

where

$$c = \max_{\min(p,q)\le u\le\max(p,q)}\left|\frac{\partial\widetilde{H}}{\partial u}\right|.$$

It is well known that some form of slope or flux limiting procedure is necessary to prevent oscillations at solution discontinuities when high-order methods are used to solve nonlinear hyperbolic conservation laws [25]. For example, if u_j^0 denotes the cell average of u_h in I_j, then we have

$$u_h(\xi) = u_j^0 + u_j^1(\xi - \xi_j), \quad j = 1,\dots,N, \tag{2.13}$$

and

$$\frac{du_h}{d\xi} = u_j^1, \quad j = 1,\dots,N,$$

is the slope of the numerical solution. One formula for limiting the slope in the numerical approximation is the minmod function

$$m(q,r,s) = \begin{cases} \operatorname{sgn}(q)\,\min(|q|,|r|,|s|) & \text{if } \operatorname{sgn}(q) = \operatorname{sgn}(r) = \operatorname{sgn}(s), \\ 0 & \text{otherwise,} \end{cases} \tag{2.14}$$

where sgn is the usual sign function. The initial linear expression for u_h is then replaced by

$$u_h(\xi) = u_j^0 + (u_j^1)^{\mathrm{mod}}(\xi - \xi_j), \quad j = 1, \ldots, N, \tag{2.15}$$

where $(u_j^1)^{\mathrm{mod}}$ is the limited slope computed as

$$(u_j^1)^{\mathrm{mod}} = m(u_j^1,\, \Delta_+ u_j^0/\Delta_j,\, \Delta_- u_j^0/\Delta_j),$$

in which Δ_+ and Δ_- are the usual forward and backward difference operators.

The discontinuous Galerkin space approximation is given as the solution of the weak formulation

$$\int_{I_j} (x_\xi \dot{u}_h) v_h \, d\xi - \int_{I_j} \widetilde{H}(u_h, \dot{x})(v_h)_\xi \, d\xi + \overline{H}_{j+\frac{1}{2}} v^-_{h_{j+\frac{1}{2}}} - \overline{H}_{j-\frac{1}{2}} v^+_{h_{j-\frac{1}{2}}} = 0, \tag{2.16}$$

and the initial condition is given by the projection

$$\int_{I_j} u_h(\xi, 0) v_h(\xi) \, d\xi = \int_{I_j} u_0(\xi) v_h(\xi) \, d\xi, \quad j = 1, \ldots, N, \tag{2.17}$$

for all $v_h \in V_h^1$.

It remains to describe how the gradient of the mesh mapping x_ξ is approximated. We will assume that the map $x = x(\xi, t)$ is piecewise linear so that

$$x(\xi, t) = x_{j-\frac{1}{2}}(t) + N\,(\xi - \xi_{j-\frac{1}{2}})\left(x_{j+\frac{1}{2}}(t) - x_{j-\frac{1}{2}}(t)\right) \tag{2.18}$$

for

$$\xi_{j-\frac{1}{2}} \le \xi \le \xi_{j+\frac{1}{2}}, \quad j = 1, \ldots, N.$$

It then follows that x_ξ used in the weak formulation (2.16) is given by

$$x_\xi(\xi, t) = N(x_{j+\frac{1}{2}}(t) - x_{j-\frac{1}{2}}(t)), \quad j = 1, \ldots, N. \tag{2.19}$$

2.2.1. Temporal integration. The discontinuous Galerkin discretization gives rise to a system of ODEs which can be written as

$$\begin{cases} \dfrac{d}{dt}(x_\xi \mathbf{u}_h) = L_h(\dot{x}, x_\xi, \mathbf{u}_h) \quad \text{in } (0, T], \\ \mathbf{u}_h(0) = \mathbf{u}_{0h}, \end{cases} \tag{2.20}$$

where $\mathbf{u}_{0h}$ is the initial approximation for the vector of the degrees of freedom $\mathbf{u}_h$. To integrate (2.20) numerically we use the second-order TVD Runge–Kutta method

$$\begin{cases} x_\xi^{n+1}\mathbf{u}_h^{(1)} = x_\xi^n \mathbf{u}_h^n + \Delta t\, L(\dot{x}^n, x_\xi^n, \mathbf{u}_h^n), \\ x_\xi^{n+1}\mathbf{u}_h^{n+1} = \dfrac{1}{2} x_\xi^n \mathbf{u}_h^n + \dfrac{1}{2} x_\xi^{n+1}\mathbf{u}_h^{(1)} + \dfrac{1}{2}\Delta t\, L(\dot{x}^{n+1}, x_\xi^{n+1}, \mathbf{u}_h^{(1)}). \end{cases} \tag{2.21}$$

For the temporal approximation of $x(\xi, t)$ we will assume that we have a set of grid points at time level t^n and at t^{n+1} which have been obtained using the MMPDE

described in section 3. The map will be assumed to be piecewise linear in time so that

$$x_{j+\frac{1}{2}}(t) = x^n_{j+\frac{1}{2}} + (t - t^n)\left(\frac{x^{n+1}_{j+\frac{1}{2}} - x^n_{j+\frac{1}{2}}}{t^{n+1} - t^n}\right), \quad t^n \le t \le t^{n+1}, \tag{2.22}$$

and hence

$$\dot{x}_{j+\frac{1}{2}}(t) = \frac{x^{n+1}_{j+\frac{1}{2}} - x^n_{j+\frac{1}{2}}}{t^{n+1} - t^n}, \quad j = 0, \ldots, N. \tag{2.23}$$

Therefore, $\dot{x}(\xi, t)$ will be piecewise linear in space and piecewise constant in time.

As the time integrator is explicit, the time step Δt has to be chosen to satisfy an appropriate CFL condition $\nu \le 1$, where the CFL number is

$$\nu = \Delta t \max_j \left\{ \frac{|(H_u)_{j+\frac{1}{2}} - \dot{x}_{j+\frac{1}{2}}|}{x_{j+1} - x_j} \right\}. \tag{2.24}$$

Note that the time step restriction on a moving mesh is different from that arising on a fixed nonuniform mesh due to the use of a modified wave speed. If the mesh velocity is close to the same speed as a solution discontinuity, then the shifted wave speed will be small where the mesh spacing is small, leading to global relaxation of the CFL condition. On the other hand, if the mesh is stationary and is highly graded towards large solution gradients, then stability requires the use of a globally small time step. The use of local time-stepping procedures can mitigate against this problem, but this certainly makes the overall algorithm more complicated (see, e.g., [26]).

2.2.2. Recovery of ϕ_h. If $(\phi_h)_x = u_h$, then ϕ_h will be determined on each interval I_j up to a constant. Following [11], the constant can be retrieved in two ways:

1. Require that

$$\int_{I_j} (\dot{\phi}_h + \tilde{H}(u_h, \dot{x}))\, \mathrm{d}\xi = 0, \quad j = 1, \ldots, N. \tag{2.25}$$

   The ODE (2.25) is again solved using the second-order TVD Runge–Kutta method (2.21).
2. Use (2.25) to update only the leftmost element I_1, and then since $(\phi_h)_x = u_h$ we have

$$\phi_h(x_j, t) = \phi_h(x_1, t) + \int_{x_1}^{x_j} u_h(x, t)\, \mathrm{d}x. \tag{2.26}$$

In section 4 we compare the performance of both recovery procedures.

2.3. HJ equations in two dimensions. In two dimensions we have

$$\phi_t + H(\phi_x, \phi_y) = 0, \quad (x, y) \in \Omega_p, \quad t > 0, \tag{2.27}$$

where $\Omega_p \subseteq \mathbb{R}^2$ is the physical domain. To generate an adaptive mesh we have seen earlier that it is useful to regard the physical domain Ω_p as the image of a computational domain Ω_c under the invertible maps

$$x = x(\xi, \eta, t), \quad y = y(\xi, \eta, t), \quad \text{and} \quad \xi = \xi(x, y, t), \quad \eta = \eta(x, y, t), \tag{2.28}$$

where $\boldsymbol{x} = (x, y)^T$ and $\boldsymbol{\xi} = (\xi, \eta)^T$ are the physical and computational coordinates, respectively. A mesh covering Ω_p is obtained by applying the mapping given by (2.28) to a partitioning of Ω_c.

Given that the mesh can move, it is necessary to express the Eulerian ($\boldsymbol{x}$-fixed) temporal derivative in (2.27) in terms of the Lagrangian derivative along the trajectory of the moving mesh. If a dot denotes differentiation with respect to time with (ξ, η) fixed, then (2.27) becomes

$$\dot{\phi} - \dot{x}\phi_x - \dot{y}\phi_y + H(\phi_x, \phi_y) = 0, \tag{2.29}$$

where $(\dot{x}, \dot{y})$ represents the mesh velocity. If we let $u = \phi_\xi$ and $v = \phi_\eta$, then by differentiating (2.29) with respect to ξ and η we arrive at the system

$$\dot{u} + \bar{H}_\xi(u, v, \dot{x}, \dot{y}) = 0, \tag{2.30}$$

$$\dot{v} + \bar{H}_\eta(u, v, \dot{x}, \dot{y}) = 0, \tag{2.31}$$

where

$$\bar{H}(u, v, \dot{x}, \dot{y}) = H(\xi_x u + \eta_x v, \xi_y u + \eta_y v) - \dot{x}(\xi_x u + \eta_x v) - \dot{y}(\xi_y u + \eta_y v). \tag{2.32}$$

Using the vectorial notation

$$\boldsymbol{u} = \begin{pmatrix} u \\ v \end{pmatrix}, \quad \boldsymbol{F}_1 = \begin{pmatrix} \bar{H} \\ 0 \end{pmatrix}, \quad \boldsymbol{F}_2 = \begin{pmatrix} 0 \\ \bar{H} \end{pmatrix},$$

we can rewrite (2.30) and (2.31) as the system of conservation laws

$$\dot{\boldsymbol{u}} + [\boldsymbol{F}_1(\boldsymbol{u})]_\xi + [\boldsymbol{F}_2(\boldsymbol{u})]_\eta = 0. \tag{2.33}$$

We apply a discontinuous Galerkin method to solve this coupled system (2.33) for $(u, v) = \nabla_{\boldsymbol{\xi}}\phi$, where the discretization takes place in the computational domain Ω_c. A recovery procedure will then be used to obtain an approximation of ϕ.

2.4. The DG discretization in two dimensions. Let $\mathcal{T}_h$ denote a partition of the computational domain Ω_c and K an arbitrary element in this partition. We consider the space of all polynomials of degree k restricted on the element K to be

$$V_h(K) = \{p \in P^k(K) : p \text{ is a polynomial of degree at most } k \text{ on } K\}. \tag{2.34}$$

To obtain a weak formulation we take the inner product of (2.33) with a test function $\boldsymbol{v}_h \in V_h$, integrate over $K \in \mathcal{T}_h$, and replace $\boldsymbol{u}$ by its approximation $\boldsymbol{u}_h \in V_h$. That is,

$$\frac{d}{dt}\int_K \boldsymbol{u}_h(\boldsymbol{\xi}, t) \cdot \boldsymbol{v}_h(\boldsymbol{\xi})\, d\boldsymbol{\xi} + \int_K \left[(\boldsymbol{F}_1)_\xi \cdot \boldsymbol{v}_h + (\boldsymbol{F}_2)_\eta \cdot \boldsymbol{v}_h\right]\, d\boldsymbol{\xi} = 0, \tag{2.35}$$

and using integration by parts we obtain

$$\begin{aligned} &\frac{d}{dt}\int_K \boldsymbol{u}_h(\boldsymbol{\xi}, t) \cdot \boldsymbol{v}_h(\boldsymbol{\xi})\, d\boldsymbol{\xi} \\ &+ \sum_{e \in \partial K} \int_e \left\{(n_{e,K})_1 \boldsymbol{F}_1 \cdot \boldsymbol{v}_h(\boldsymbol{\xi}^{\text{int(K)}}) + (n_{e,K})_2 \boldsymbol{F}_2 \cdot \boldsymbol{v}_h(\boldsymbol{\xi}^{\text{int(K)}})\right\} ds \\ &- \int_K \left[\boldsymbol{F}_1 \cdot (\boldsymbol{v}_h)_\xi + \boldsymbol{F}_2 \cdot (\boldsymbol{v}_h)_\eta\right]\, d\boldsymbol{\xi} = 0, \end{aligned} \tag{2.36}$$

where $n_{e,K} = [(n_{e,K})_1, (n_{e,K})_2]$ is the unit normal to the edge e of the element K, $\mathrm{int}(K)$ denotes the value taken from the interior of the element K, and $\mathrm{ext}(K)$ denotes the value taken from the exterior of the element K. Next we define the normal flux

$$\boldsymbol{F}_{e,K}(\boldsymbol{u}, \dot{x}, \dot{y}) = (n_{e,K})_1 \boldsymbol{F}_1 + (n_{e,K})_2 \boldsymbol{F}_2 = \begin{bmatrix} (n_{e,K})_1 \bar{H} \\ (n_{e,K})_2 \bar{H} \end{bmatrix}. \tag{2.37}$$

Note that $\boldsymbol{F}_{e,K}$ is not well defined on the element edge as $\boldsymbol{u}_h$ is discontinuous there. Therefore, we replace it by the numerical flux function

$$\widehat{\boldsymbol{F}}_{e,K}\left(\boldsymbol{u}_h(\boldsymbol{\xi}^{\mathrm{int(K)}}), \boldsymbol{u}_h(\boldsymbol{\xi}^{\mathrm{ext(K)}}), \dot{x}, \dot{y}\right). \tag{2.38}$$

Again we will use the Lax–Friedrichs flux

$$\widehat{\boldsymbol{F}}_{e,K}(\boldsymbol{a}, \boldsymbol{b}, \dot{x}, \dot{y}) = \frac{1}{2}\left[\boldsymbol{F}_{e,K}(\boldsymbol{a}) + \boldsymbol{F}_{e,K}(\boldsymbol{b}) + \alpha_{e,K}(\boldsymbol{a} - \boldsymbol{b})\right], \tag{2.39}$$

where

$$\alpha_{e,K} = \rho\left((n_{e,K})_1 \frac{\partial \boldsymbol{F}_1}{\partial \boldsymbol{u}} + (n_{e,K})_2 \frac{\partial \boldsymbol{F}_2}{\partial \boldsymbol{u}}\right) \tag{2.40}$$

and $\rho(\cdot)$ denotes the spectral radius of $(\cdot)$.

In this paper we will consider only the case that $\boldsymbol{u}_h \in (P^1)^2$. In particular we now discuss the basis used when $\boldsymbol{u}_h$ is a discontinuous bilinear function defined on a topologically rectangular mesh covering Ω_c. We will assume that $\Omega_c = [0,1]^2$ is covered by a uniform $N \times N$ element mesh. Following Hu and Shu [11], we assume that the approximate solution of the HJ equation ϕ_h is piecewise discontinuous and quadratic. Therefore, in the cell $K_{ij} = (\xi_{i-1/2}, \xi_{i+1/2}) \times (\eta_{j-1/2}, \eta_{j+1/2})$ we assume

$$\begin{aligned} \phi_h(t) = \overline{\phi}(t) &+ \phi_\xi(t)\mu_i + \phi_\eta(t)\nu_j + \phi_{\xi\eta}(t)\mu_i\nu_j \\ &+ \phi_{\xi\xi}(t)\left(\mu_i^2 - \frac{1}{3}\right) + \phi_{\eta\eta}(t)\left(\nu_j^2 - \frac{1}{3}\right), \end{aligned} \tag{2.41}$$

where

$$\mu_i(\xi) = \frac{2(\xi - \xi_i)}{\Delta\xi} \quad \text{and} \quad \nu_j(\eta) = \frac{2(\eta - \eta_j)}{\Delta\eta}.$$

The gradient in computational space $\nabla_{\boldsymbol{\xi}}\phi_h$ therefore takes the form

$$\nabla_{\boldsymbol{\xi}}\phi_h = \begin{bmatrix} \frac{2}{\Delta\xi}\phi_\xi(t) + \frac{2}{\Delta\xi}\phi_{\xi\eta}(t)\nu_j + \frac{4}{\Delta\xi}\phi_{\xi\xi}(t)\mu_i \\ \frac{2}{\Delta\eta}\phi_\eta(t) + \frac{2}{\Delta\eta}\phi_{\xi\eta}(t)\mu_i + \frac{4}{\Delta\eta}\phi_{\eta\eta}(t)\nu_j \end{bmatrix}, \tag{2.42}$$

and hence there are five unknowns that determine $\nabla_{\boldsymbol{\xi}}\phi_h$. A suitable basis for

$$V_2^1 = \{(v_1, v_2, v_3, v_4, v_5) : \boldsymbol{v}|_K = \nabla_{\boldsymbol{\xi}}\phi, \quad \phi \in P^2(K) \quad \forall K \in \mathcal{T}_h\}$$

is given by

$$\boldsymbol{b}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \boldsymbol{b}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \boldsymbol{b}_3 = \begin{bmatrix} \mu_i \\ 0 \end{bmatrix}, \quad \boldsymbol{b}_4 = \begin{bmatrix} 0 \\ \nu_j \end{bmatrix}, \quad \boldsymbol{b}_5 = \begin{bmatrix} \frac{\nu_j}{\Delta\xi} \\ \frac{\mu_i}{\Delta\eta} \end{bmatrix}. \tag{2.43}$$

The local basis functions (2.43) can easily be shown to be orthogonal, and hence the mass matrix is diagonal.

The evaluation of the edge and element integrals in (2.36) is achieved using quadrature. For the edges,

$$\int_e \widehat{\boldsymbol{F}}_{e,K} \cdot \boldsymbol{v}_h(\boldsymbol{\xi}^{\mathrm{int}(K)}) \approx \sum_{l=1}^{q} \omega_l \widehat{\boldsymbol{F}}_{e,K} \left(\boldsymbol{u}_h(\boldsymbol{\xi}_l^{\mathrm{int(K)}}), \boldsymbol{u}_h(\boldsymbol{\xi}_l^{\mathrm{ext(K)}}), \dot{x}_l, \dot{y}_l \right) \cdot \boldsymbol{v}_h(\boldsymbol{\xi}_l^{\mathrm{int(K)}}) |e|,$$

where ω_l are the weights and $\boldsymbol{\xi}_l$ are the quadrature points along the edge. To estimate these integrals we use the two-point Gauss quadrature rule. Similarly, for the element integrals appearing in (2.36) we use the 2×2 Gauss product rule.

The mesh mapping $\boldsymbol{x}(\boldsymbol{\xi}, t)$ is assumed to vary bilinearly in space based on the position of the four grid nodes defining each element.

After carrying out the integrations and inverting the mass matrix, we finally arrive at the system of ODEs

$$\dot{\boldsymbol{u}} = L(\dot{\boldsymbol{x}}, \boldsymbol{u}), \quad \boldsymbol{u} = (u_1, u_2, u_3, u_4, u_5)^T, \tag{2.44}$$

which will be solved using the explicit two-stage explicit TVD Runge–Kutta scheme (2.21) to evolve $\boldsymbol{u}$, where $\dot{\boldsymbol{x}}$ is assumed to be piecewise constant in time.

The solution of (2.44) allows us to evolve $\nabla_{\boldsymbol{\xi}} \phi_h$. To calculate ϕ_h, we note from (2.41) that we require a procedure to calculate the cell average ϕ_{av}. Following Hu and Shu [11], we project the original HJ equation onto piecewise constants. That is, for any element K we set

$$\int_K (\dot{\phi}_h + \bar{H}(u_h, v_h, \dot{x}, \dot{y}))\, d\xi d\eta = 0. \tag{2.45}$$

We note from (2.41) that

$$\phi_{av}(t) = \frac{1}{|K|} \int_K \phi_h \, d\xi \, d\eta,$$

and hence

$$\dot{\phi}_{av} = -\frac{1}{|K|} \int_K \bar{H}(u_h, v_h, \dot{x}, \dot{y})\, d\xi d\eta. \tag{2.46}$$

We approximate the integral on the right-hand side of (2.46) using a 2×2 Gauss quadrature rule so that

$$\dot{\phi}_{av} = -\frac{1}{|K|} \sum_{l=1}^{4} \frac{\omega_l}{4} \bar{H}((u_h)_l, (v_h)_l, \dot{x}_l, \dot{y}_l)\, |K|. \tag{2.47}$$

The ODE (2.47) is again solved using the second-order TVD Runge–Kutta method (2.21). One possibility is to solve (2.47) for each element of the mesh. However, numerical experience reported in [11] indicates that this approach does not work well when there are singularities in the derivatives and the integral path in time passes

through the singularities at earlier times. Alternatively, one can solve (2.47) for only one element, where there are no derivative singularities, and use the formula

$$\phi(B,t) = \phi(A,t) + \int_A^B (\phi_x\, dx + \phi_y\, dy) \tag{2.48}$$

to update all of the remaining elements.

3. Moving mesh equations.

3.1. MMPDEs in one and two dimensions. Let us consider $\boldsymbol{x} = (x_1, x_2)^T$ to be a point in the physical domain Ω_p and $\boldsymbol{\xi} = (\xi_1, \xi_2)^T$ to be a point in the computational domain Ω_c. The mapping from the computational domain to the physical domain is assumed to minimize the energy functional

$$I[\xi_1, \xi_2] = \frac{1}{2} \int_{\Omega_p} \left((\nabla \xi_1)^T G^{-1} \nabla \xi_1 + (\nabla \xi_2)^T G^{-1} \nabla \xi_2 \right) dx\, dy, \tag{3.1}$$

where G is a symmetric positive-definite monitor matrix which involves various properties of the mesh and physical solution, and ∇ is the gradient operator with respect to $\boldsymbol{x} = (x_1, x_2)^T$.

A two-dimensional MMPDE is given by the gradient flow equations

$$\frac{\partial \boldsymbol{\xi}}{\partial t} = -\frac{P}{\tau} \frac{\delta I}{\delta \boldsymbol{\xi}} = \frac{P}{\tau} \nabla \cdot \left(G^{-1} \nabla \boldsymbol{\xi} \right), \tag{3.2}$$

where $\tau > 0$ is a temporal smoothing parameter and P is an operator with positive spectrum used to regularize the evolution of the mesh. The parameter τ determines how quickly the mesh adapts in time in an attempt to minimize (3.1). The value of τ should be chosen to be small in relation to the time scale of the physical PDE to ensure that the mesh adapts quickly enough to follow important solution features. On the other hand, it is important to make sure that τ is not too small; otherwise the mesh can evolve in an erratic manner, and this can have an adverse effect on accuracy. Further discussion on the choice of τ can be found in [13].

In practice we work directly with the map $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{\xi}, t)$ because it defines explicitly the location of the physical mesh points. Following [13] and [12], we switch the roles of the dependent and independent variables in (3.2) to obtain

$$\tau \frac{\partial \boldsymbol{x}}{\partial t} = P \left(\sum_{i,j=1}^{2} (\boldsymbol{a}^i \cdot G^{-1} \boldsymbol{a}^j) \frac{\partial^2 \boldsymbol{x}}{\partial \xi_i \partial \xi_j} - \sum_{i,j=1}^{2} \left(\boldsymbol{a}^i \cdot \frac{\partial G^{-1}}{\partial \xi_j} \boldsymbol{a}^j \right) \frac{\partial \boldsymbol{x}}{\partial \xi_i} \right), \tag{3.3}$$

where $\boldsymbol{a}^i = \nabla \xi_i$. In [12], Huang proposed choosing P in order to limit the variation over the domain of the coefficients in (3.3). To do this we choose

$$P = \left(\sum_{i=1}^{2} a_{i,i}^2 + b_i^2 \right)^{-\frac{1}{2}}, \tag{3.4}$$

where

$$a_{i,j} = \boldsymbol{a}^i \cdot G^{-1} \boldsymbol{a}^j, \quad b_i = -\sum_{i,j=1}^{2} \left(\boldsymbol{a}^i \cdot \frac{\partial G^{-1}}{\partial \xi_j} \boldsymbol{a}^j \right). \tag{3.5}$$

Dirichlet boundary conditions for the above system are obtained by solving a one-dimensional MMPDE. If $\partial_p \in \partial\Omega_p$ and $\partial_c \in \partial\Omega_c$ denote the physical and computational boundary segments with arc-lengths l and l_c, respectively, then the mesh on ∂_p is the solution of

$$\tau \frac{\partial s}{\partial t} = \frac{1}{\sqrt{(M^2 + (M_\zeta)^2}} \frac{\partial}{\partial \zeta}\left(M \frac{\partial s}{\partial \zeta}\right), \qquad \zeta \in (0, l_c), \tag{3.6}$$

with $s(0) = 0$ and $s(l_c) = l$. Here M is the one-dimensional projection of the two-dimensional monitor function along the boundary. That is, if $\boldsymbol{t}$ is a unit tangent vector along the boundary, then $M(s,t) = \boldsymbol{t}^T G \boldsymbol{t}$.

3.2. Discretization and boundary conditions. The temporal discretization of (3.3) is achieved using a semi-implicit method where

$$(\boldsymbol{x}^{n+1} - \boldsymbol{x}^n) = \frac{\Delta t P^n}{\tau}(a_{11}^n \boldsymbol{x}_{\xi\xi}^{n+1} + 2a_{12}^n \boldsymbol{x}_{\xi\eta}^{n+1} + a_{22}^n \boldsymbol{x}_{\eta\eta}^{n+1} + b_1^n \boldsymbol{x}_{\xi}^{n+1} + b_2^n \boldsymbol{x}_{\eta}^{n+1}). \tag{3.7}$$

Freezing the coefficients a_{ij}, b_i, and P has two very important effects. The first is that it linearizes the equations that define the mesh $\boldsymbol{x}^{n+1}$. Moreover, the system of PDEs represented by (3.7) for $x^{n+1}(\xi,\eta)$ and $y^{n+1}(\xi,\eta)$ decouples into two scalar equations which have the same spatial derivatives. For simplicity, the spatial discretization of (3.7) is performed using second-order central finite differences on an $N \times N$ uniform partition of $\Omega_c = (0,1) \times (0,1)$. Since the spatial derivatives are identical, the same coefficient matrix needs to be inverted to find the x and y coordinates. We therefore solve first (3.7) for the x coordinates using an ILU-preconditioned BiCGStab method until the mean-squared-root residual is less than 10^{-8}. The same preconditioner is then used to solve (3.7) for the y coordinates. In general, we have found that convergence of the iterative solver is achieved in around 3–4 iterations per time step. Additional details about the spatial discretization of the MMPDEs and the performance of the ILU-preconditioned BiCGStab routine can be found in [3].

3.3. Choice of monitor function. The choice of a suitable adaptivity criterion for problems which develop shocks is highly nontrivial. It is important that the mesh points move smoothly towards regions where discontinuities exist so that the $O(1)$ error at the shock is localized within one or two mesh elements. For robustness, it is also important that the rate of convergence of the method is not severely impaired by the nonuniformity of the moving mesh.

A popular choice of monitor function is the scaled solution arc-length

$$M = \sqrt{1 + \frac{1}{\alpha}|u_x|^2}, \tag{3.8}$$

where $\alpha > 0$ is a user-chosen parameter. The role of this parameter when $\alpha > 1$ is to reduce the local value of the monitor function when $|u_x|$ is large. This then leaves the issue of the proper choice of α. The simplest approach is to set $\alpha =$ constant. However, this procedure is unsatisfactory as the ideal value of α will be problem dependent and any a priori choice of α will never prevent $M \to \infty$ as $|u_x| \to \infty$.

A more sophisticated approach is to make α solution dependent. For example, in [9], $\alpha \approx \max |u|^2$. However, for problems where u becomes discontinuous, this choice of α will be ineffective in again stopping M from becoming unbounded. Another popular choice is

$$M(x,t) = \left(1 + \frac{|u_x|^2}{\alpha}\right)^{1/2}, \qquad \alpha = \frac{1}{\beta(b-a)} \int_a^b |u_x|^2 dx. \tag{3.9}$$

If $\beta = 1$, then (3.9) is exactly a regularization of the monitor function used in [24]. For this monitor function the local derivative term is scaled by its average value over the domain. It is not obvious that this monitor function is bounded above in the presence of a solution discontinuity. Let us assume an idealized situation where

$$|u_x| \simeq \mathcal{O}\left(\frac{1}{\Delta x_{\min}}\right) \tag{3.10}$$

in the cell where $|u_x|$ is large and is constant $|u_x| \simeq C$ everywhere else. If the mesh is obtained from an equidistribution principle, then

$$\int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} M(x,t)\,dx = \frac{1}{N}\int_a^b M(x,t)\,dx, \quad j = 1,\dots,N. \tag{3.11}$$

If we expand the integral in (3.9), we get

$$\alpha = \frac{1}{\beta(b-a)}\left(\int_a^{x^*-\frac{1}{2}\Delta x_{\min}} C^2\,dx + \int_{x^*-\frac{1}{2}\Delta x_{\min}}^{x^*+\frac{1}{2}\Delta x_{\min}} \frac{1}{\Delta x_{\min}^2}\,dx + \int_{x^*+\frac{1}{2}\Delta x_{\min}}^{b} C^2\,dx\right), \tag{3.12}$$

and hence

$$\alpha \simeq \frac{1}{\beta}\left((1-\Delta x_{\min})\,C^2 + \frac{1}{\Delta x_{\min}(b-a)}\right) \approx (\Delta x_{\min})^{-1}. \tag{3.13}$$

It is clear from (3.13) that $\alpha \to \infty$ as $\Delta x_{\min} \to 0$. If we examine the behavior of M, we observe that the monitor function is approximately 1 away from the shock as $|u_x|$ is bounded and α tends to infinity. Close to the shock,

$$M \simeq \mathcal{O}\left(1 + \frac{\frac{1}{\Delta x_{\min}^2}}{\frac{1}{\Delta x_{\min}}}\right)^{\frac{1}{2}} \simeq \mathcal{O}\left(\sqrt{\frac{1}{\Delta x_{\min}}}\right),$$

which is not bounded as $\Delta x_{\min}$ tends to zero. The problem with this monitor function is that the maximum value of $|u_x|^2$ and its average value do not scale in the same way as N increases.

To obtain a bounded monitor function we consider

$$M(x,t) = \left(1 + \frac{|u_x|^2}{\alpha}\right)^{\frac{1}{2}}, \qquad \alpha = \frac{1}{\beta^2}\max_x |u_x|^2, \tag{3.14}$$

and β is a user-chosen parameter. It is clear that this monitor function satisfies the bounds

$$1 \le M(x,t) \le \sqrt{1+\beta^2}.$$

If we have a location x^* where $|u_x|$ becomes large (such as at a shock) and the solution elsewhere is smooth, then the monitor function will have a localized peak at x^*. It is clear that the transition from the smooth to the nonsmooth region of the domain will be almost discontinuous. This leads to an extremely rapid change in the mesh size which might be far from ideal in practice and can lead to a loss in accuracy. Again if

the mesh is found by equidistribution of the monitor function, then if $M \simeq 1$ except in the cell containing x^*, we have

$$\sqrt{1+\beta^2}\Delta x_{\min} \simeq \frac{1}{N}\left((b-a) + \sqrt{1+\beta^2}\,\Delta x_{\min}\right). \tag{3.15}$$

Therefore, the minimum mesh spacing is

$$\Delta x_{\min} \simeq \frac{(b-a)}{(N-1)\sqrt{1+\beta^2}}, \tag{3.16}$$

which is approximately a factor of $\sqrt{1+\beta^2}$ smaller than the spacing that occurs when using a uniform mesh. This is the monitor function that we will use in the following one-dimensional examples.

For two-dimensional problems the monitor function used is of Winslow type, where

$$G = \begin{bmatrix} w & 0 \\ 0 & w \end{bmatrix}$$

and

$$w = \sqrt{1 + \frac{|\nabla u|^2}{\alpha}}, \quad \text{where} \quad \alpha = \frac{1}{\beta^2}\max_{\Omega_p} |\nabla u|^2.$$

This monitor function behaves in a similar fashion to the one-dimensional monitor in the presence of solution discontinuities since the maximum of the solution gradient is used rather than its average.

3.3.1. Smoothing of the monitor function. If the underlying problem involves large solution variations, then the monitor function will be nonsmooth in space, and this will affect the smoothness of the coordinate transformation. In practice, the monitor function is smoothed before it is used in the numerical approximation of the MMPDE.

A smooth transformation can be achieved using a boundary value problem (see [13]) which involves an artificial diffusion term for smoothing the monitor function. The smoothed monitor function $\widetilde{M}$ satisfies the boundary value problem

$$\widetilde{M} - \frac{1}{\lambda^2}\widetilde{M}_{\xi\xi} = M, \tag{3.17}$$

with the boundary conditions

$$\widetilde{M}_\xi(0,t) = \widetilde{M}_\xi(1,t) = 0, \tag{3.18}$$

where λ is a positive parameter. Equation (3.17) is discretized using central differences as

$$\widetilde{M}^n_{j+\frac{1}{2}} - \frac{1}{\lambda^2 h^2}\left(\widetilde{M}_{j+\frac{3}{2}} - 2\widetilde{M}_{j+\frac{1}{2}} + \widetilde{M}_{j-\frac{1}{2}}\right) = M_{j+\frac{1}{2}}, \quad j = 1,\ldots,N-1, \tag{3.19}$$

where $h = 1/N$. The boundary conditions are discretized as

$$\widetilde{M}_{-\frac{1}{2}} = \widetilde{M}_{\frac{1}{2}}, \quad \widetilde{M}_{N+\frac{1}{2}} = \widetilde{M}_{N-\frac{1}{2}}. \tag{3.20}$$

It can be shown that the smoothed monitor function so obtained is bounded and equidistribution leads to a locally quasi-uniform mesh [13]. In particular it can be shown that

$$\nu < \frac{x_{j+\frac{3}{2}}(t) - x_{j+\frac{1}{2}}(t)}{x_{j+\frac{1}{2}}(t) - x_{j-\frac{1}{2}}(t)} < \nu^{-1}, \qquad j = 1, \ldots, N-1,$$

where

$$\nu = \frac{\sqrt{1+4\gamma}-1}{\sqrt{1+4\gamma}+1} \quad \text{and} \quad \gamma = \frac{1}{\lambda h^2}.$$

We can therefore see that the mesh becomes less smooth as we increase the parameter λ. A similar procedure is used to smooth the monitor function in two dimensions.

3.3.2. Complete algorithm. If we assume that at time $t = t^n$ we have an approximation of the physical solution u_h^n and a mesh $\boldsymbol{x}^n$, then we integrate forward in time using the algorithm below.

1. Compute the monitor matrix $G^n(\boldsymbol{x}) = G(\boldsymbol{x}^n, t^n)$ using u_h^n and $\boldsymbol{x}^n$.
2. Integrate the discretized MMPDE to get the mesh $\boldsymbol{x}^{n+1}$.
3. Integrate the physical PDEs to obtain u_h^{n+1} using the meshes $\boldsymbol{x}^n$ and $\boldsymbol{x}^{n+1}$ to form $\dot{\boldsymbol{x}}$.
4. Goto the next time step.

In theory this algorithm could be modified to return to step 1 after u_h^{n+1} has been obtained in step 3 and the coefficients a_{ij}, b_i in (3.5) have been evaluated at t^{n+1} and $\boldsymbol{x}^{n+1}$. A new estimate of the mesh $\boldsymbol{x}^{n+1}$ could then be found by resolving the MMPDEs. This process could be repeated a fixed number of times or until some measure of grid convergence is reached. A similar approach was investigated in [2] and [3], where it was found that larger time steps could be used without affecting accuracy and that the gain in efficiency outweighed the additional cost of performing the additional steps. In the numerical experiments in the next section we have, however, considered only the use of one pass of the algorithm above.

4. Numerical examples.

4.1. One-dimensional problems.

4.1.1. Example 1. The first example considered is the Burgers equation

$$\begin{cases} \phi_t + \dfrac{(\phi_x + 1)^2}{2} = 0, & -1 < x < 1,\ t > 0, \\ \phi(x, 0) = -\cos(\pi(x - 0.85)) \end{cases} \tag{4.1}$$

with periodic boundary conditions. This problem has also been considered in [28] and [15]. As the solution evolves from the smooth initial condition, a discontinuity in the derivative appears and propagates across the domain.

We first test the rate of convergence of the moving mesh DG method when the solution is still smooth. In all of the following results we set CFL = 0.1, $\beta = 10$, $\lambda = 25$, and $\tau = 0.001$. Table 1 shows that the DG solution maintains its second-order accuracy. Note that slope limiting was not needed for this example.

Figure 1 shows the computed DG solutions on a fixed and moving mesh at $t = 0.99/\pi^2$, which is just before the solution becomes discontinuous. We can see that the moving mesh results are a considerable improvement over the fixed mesh results

TABLE 1
Convergence of moving mesh DG solution at $t = 0.5/\pi^2$.

N	$\|\|e_u\|\|_{L_\infty}$	Rate	$\|\|e_\phi\|\|_{L_\infty}$	Rate	$\|\|e_u\|\|_{L_1}$	Rate	$\|\|e_\phi\|\|_{L_1}$	Rate
20	1.92E − 01	–	4.26E − 03	–	4.13E − 02	–	3.21E − 03	–
40	4.24E − 02	2.18	9.75E − 04	2.13	8.53E − 03	2.28	8.03E − 04	2.00
80	1.10E − 02	1.95	2.61E − 04	1.90	2.04E − 03	2.07	2.17E − 04	1.89
160	2.78E − 03	1.98	6.71E − 05	1.96	5.05E − 04	2.01	5.75E − 05	1.92
320	7.02E − 04	1.99	1.71E − 05	1.97	1.26E − 04	2.00	1.48E − 05	1.96

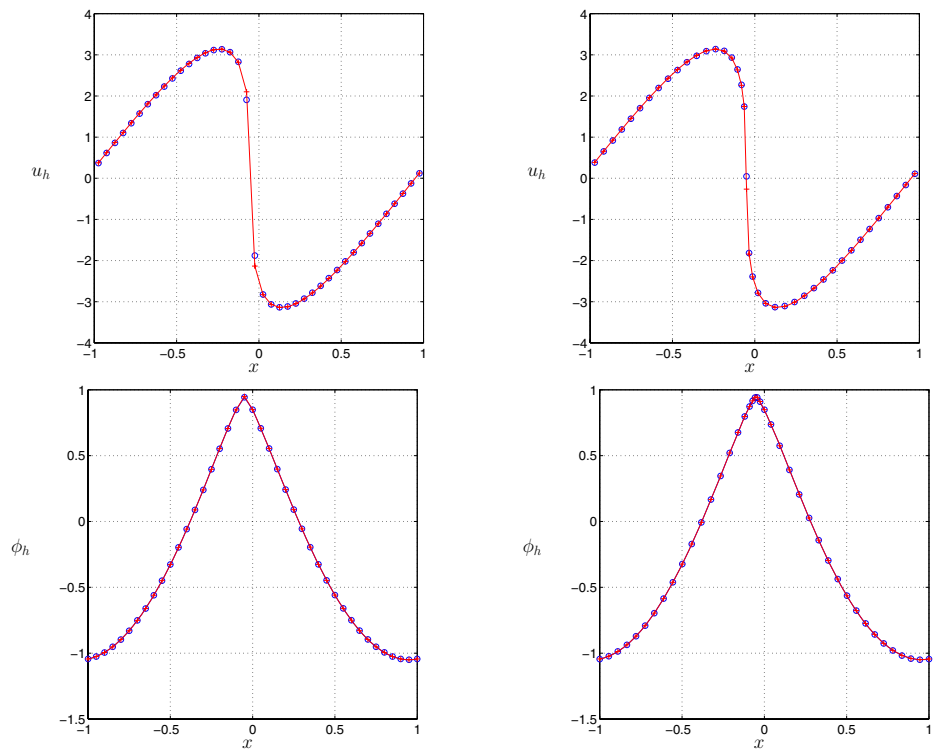


FIG. 1. *DG numerical solution (∘) and exact solution (+) obtained on a uniform stationary mesh (left) and on a moving mesh (right) at time* $t = 0.99/\pi^2$.

as the shock in u and the corner in ϕ are much better resolved. Figure 2 shows the solutions at $t = 3/\pi^2$, where we can see that the shock in u is resolved over one or two cell widths that are much smaller around the shock than those used with the uniform mesh.

The computed mesh trajectories shown in Figure 3 are quite smooth, which is the result of the smoothing of the monitor function. We observe that once the discontinuity forms, the mesh points cluster around it to resolve it more accurately. Numerical experience shows that as long as the numerical solution is smooth, both recovery procedures (2.25) and (2.26) give similar results. In Figure 3 we also compare the two recovery procedures at $t = 1.5/\pi^2$ when the solution is nonsmooth. We can see that the recovery procedure (2.25) does not perform well, whereas (2.26) appears to perform much better and hence is the procedure we use for all further examples.

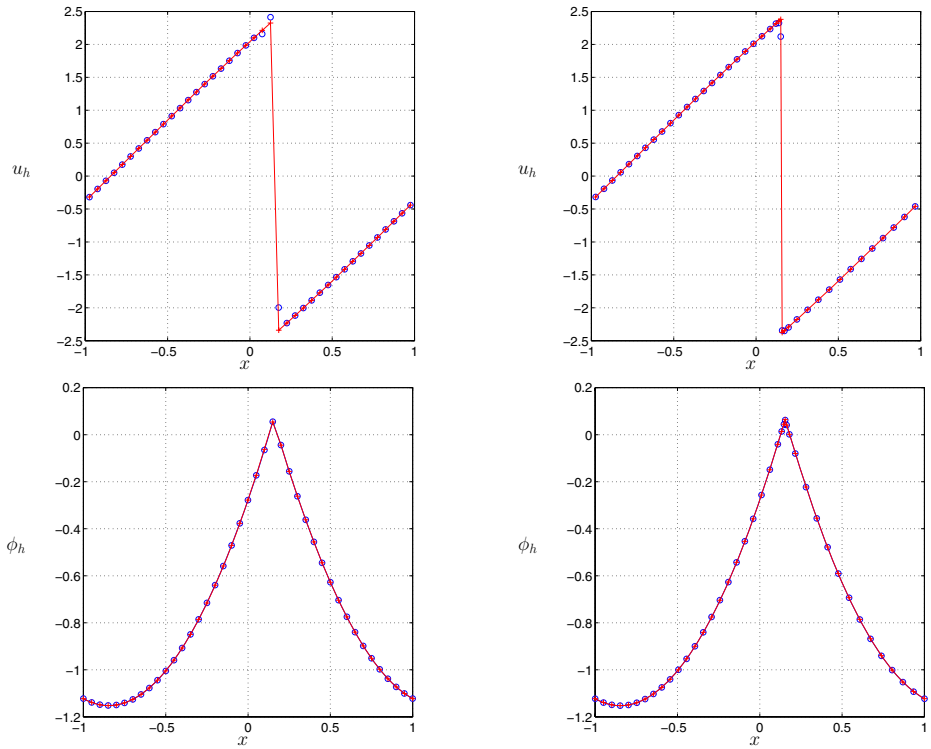


FIG. 2. *DG solution (∘) and exact solution (+) obtained on a uniform stationary mesh (left) and on a moving mesh (right) at time* $t = 3/\pi^2$.

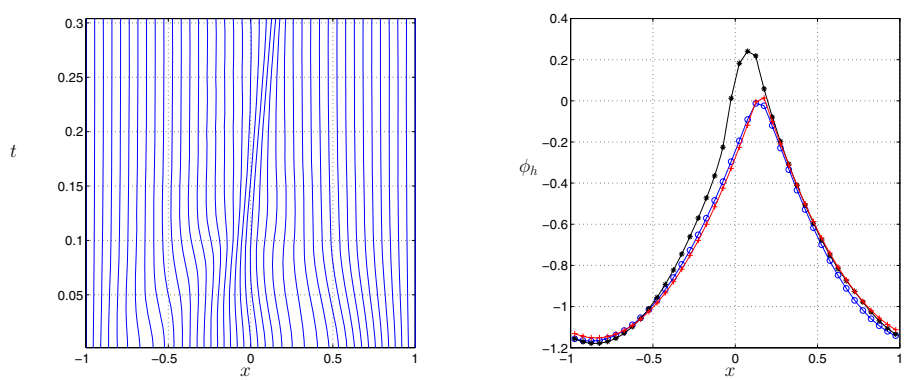


FIG. 3. *Mesh trajectories and comparison between recovery procedures at* $t = 1.5/\pi^2$, (2.25) *(*),* (2.26) *(∘), and exact solution (+), with* $N = 40$.

In Figure 4 we plot the evolution of the time step using a uniform stationary mesh and the adaptive moving mesh. We can see that the time step used with the adaptive mesh is not considerably smaller than that used with the uniform mesh even though the mesh cells close to the shock region are much smaller than those occurring with the uniform mesh.

4.1.2. Example 2. The second example considered is the following Riemann problem with a nonconvex flux:

$$\begin{cases} \phi_t + \frac{1}{4}(\phi_x^2 - 1)(\phi_x^2 - 4) = 0, \quad -1 < x < 1,\ t > 0, \\ \phi(x,0) = -2|x|. \end{cases} \tag{4.2}$$

This problem has been considered in [11] and [28]. This test problem has an initial discontinuity in the derivative of the solution u, and in time this develops into two discontinuities moving in opposite directions. In [11] the authors point out that the DG method fails to "open up" the initial single discontinuity in the derivative to generate the correct entropy solution. Therefore, for this example we used the slope limiting procedure as outlined earlier.

A comparison of the DG solutions using a uniform fixed mesh and a moving mesh is shown in Figure 5. Again there is considerable improvement in the resolution of u_h and ϕ_h using the moving mesh. In Figure 6 we show the computed mesh trajectories and we can see that the monitor function is able to both track discontinuities and concentrate points in regions where more accuracy is needed.

4.1.3. Example 3. The third example considered is

$$\begin{cases} \phi_t - \left(1 + \phi_x^2\right)^{1/2} = 0, \quad -1 < x < 1,\ t > 0, \\ \phi(x,0) = \dfrac{1}{2}\cos(6\pi x), \end{cases} \tag{4.3}$$

with periodic boundary conditions. This is an example of front propagation in one dimension and has also been considered in [22]. As the numerical solution evolves in time, the solution develops corners since the graph of the initial solution moves in its normal direction with unit speed. For this example we found it necessary to use the slope limiting procedure.

In Figure 7 we show a comparison between ϕ_h obtained with DG using a stationary mesh and a moving mesh. The differences between the numerical solutions can be seen more clearly in Figure 8. It can be observed that the adaptive mesh does a much better job of resolving the corner which has developed in the numerical solution ϕ_h. Furthermore, the smooth part of the solution also appears to be much better resolved using the moving mesh. In Figure 9 we plot mesh trajectories of the DG moving mesh solution with $N = 81$ mesh points. We can see that the monitor function drives mesh points towards the discontinuities in the first derivative of the solution. In Figure 9 we also plot the evolution of the numerical solution ϕ_h at time intervals of $\Delta t = 0.01$, up to the final time $t = 0.1$. We can see clearly the initial function evolving with unit speed in its normal direction and that the moving mesh DG method resolves well the evolution of the corner singularities.

4.2. Two-dimensional problems.

4.2.1. Example 4. We consider the two-dimensional Burgers equation

$$\phi_t + \frac{1}{2}(1 + \phi_x + \phi_y)^2 = 0, \quad (x,y) \in (-2,2)^2, \tag{4.4}$$

$$\phi(x,y,0) = -\cos(\pi(x+y)/2), \tag{4.5}$$

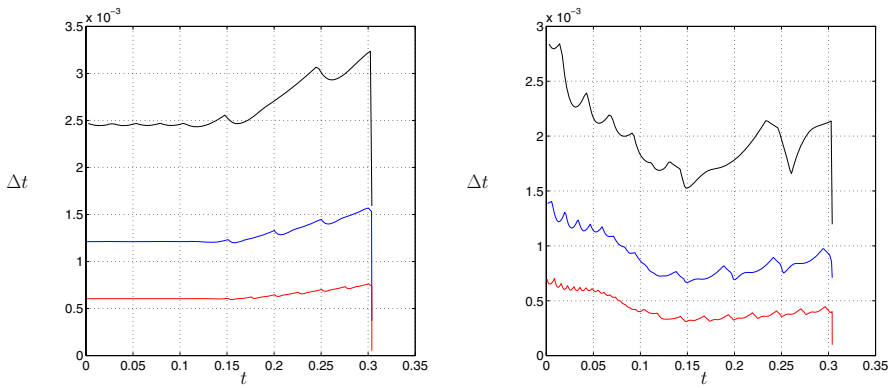


FIG. 4. *Time step evolution for Example* 1: *Uniform mesh (left), moving mesh (right),* $N = 20$ *(top line),* $N = 40$ *(middle line),* $N = 80$ *(bottom line).*

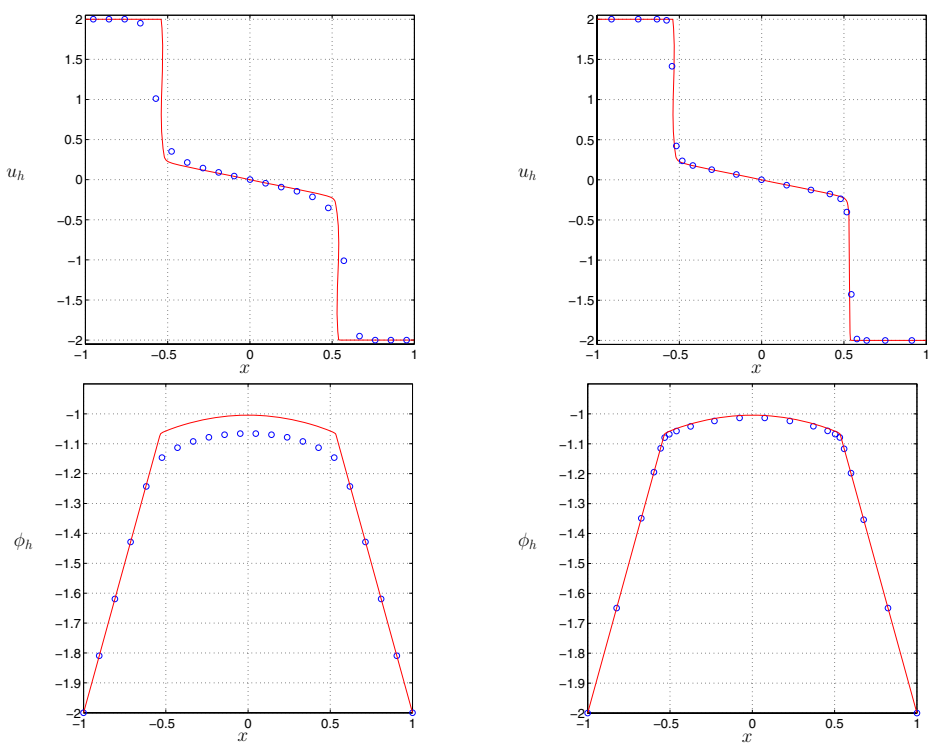


FIG. 5. *Comparison of results using DG uniform mesh (left), DG moving mesh method (right) at* $t = 1$ *with* $N = 21$, $CFL = 0.33$, $\beta = 10$, $\tau = 0.001$ *for Example* 2.

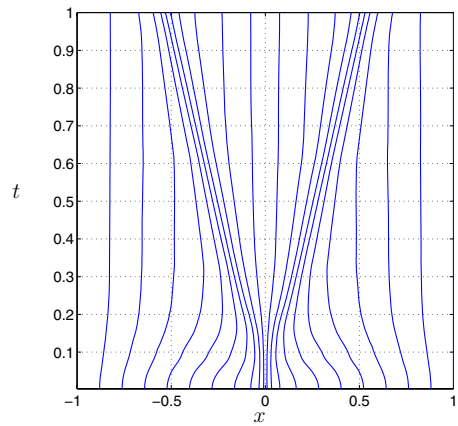


FIG. 6. *Mesh trajectories with* $N = 21$ *for Example* 2.

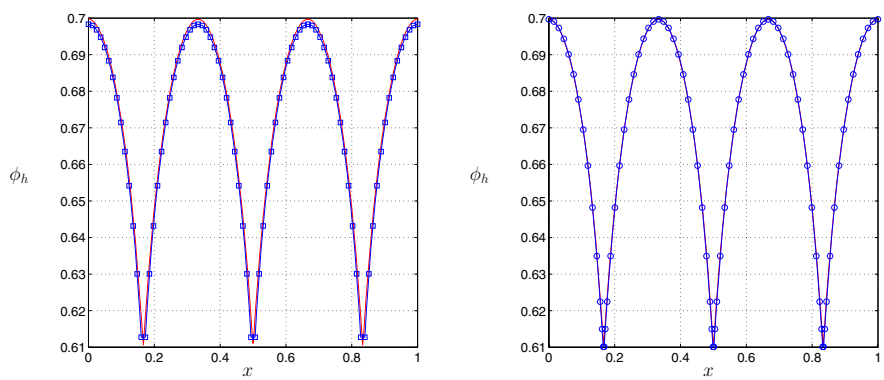


FIG. 7. *Numerical solution* ϕ_h *obtained with DG on a stationary mesh (* $\square$*, left), on a moving mesh (*$\circ$*, right), and the reference solution (*$-$*) with* $N = 81$ *mesh points at* $t = 0.2$.

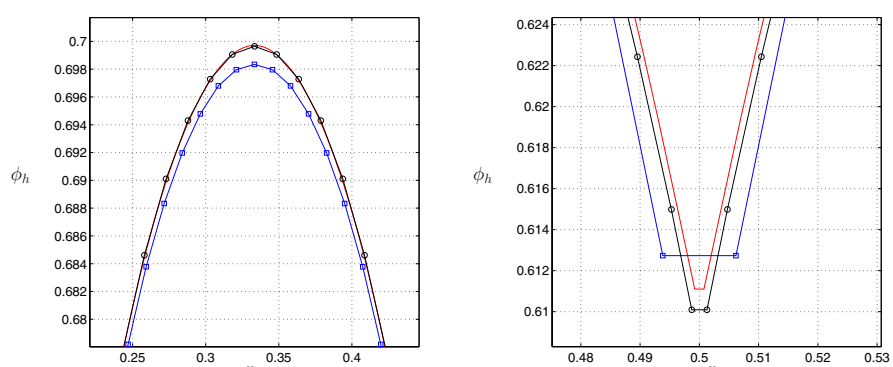


FIG. 8. *Zoom of the numerical solution* ϕ_h *obtained with DG on a stationary mesh (* $\square$*, left), on a moving mesh (*$\circ$*, right), and the reference solution (*$-$*) with* $N = 81$ *at* $t = 0.2$*, in the smooth region (left), and nonsmooth region (right).*

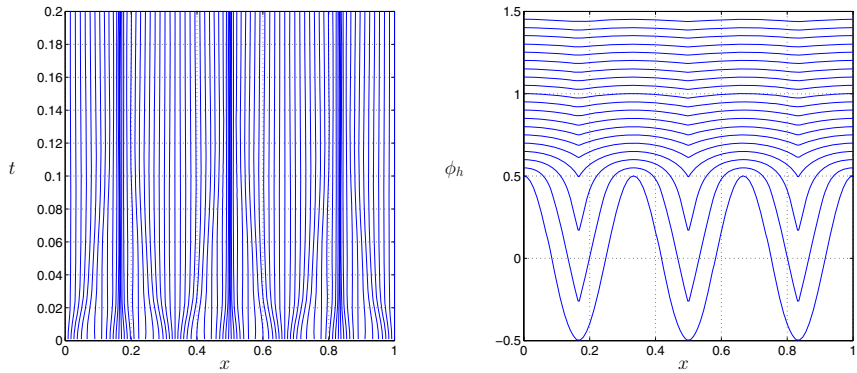


FIG. 9. *Mesh trajectories and evolution of the numerical solution at time intervals* $\Delta t = 0.01$, *up to* $t = 1$ *for Example* 3.

with periodic boundary conditions. This problem was proposed in [23] and has also been used as a test case in [28], [11], and [16]. From the smooth initial condition the solution develops a discontinuity in $\nabla\phi$ which propagates across the domain from the bottom left to the top right. Figure 10 shows the results calculated at $t = 1.5/\pi^2$ when $\nabla\phi$ has become discontinuous. Note that flux limiting was not necessary for this example, highlighting the additional stability properties of the DG formulation. We can see that while there seem to be some slight oscillations in the moving mesh calculations of ϕ_x, these seem to be damped out by the recovery procedure used to calculate ϕ_h. The computed adaptive meshes at $t = 0.5/\pi^2$ and $t = 1.5/\pi^2$ are shown in Figure 11, where we can see that the meshes have indeed clustered towards the areas where the gradient is largest. Finally, Figure 12 shows a comparison of the moving and fixed mesh results compared with a fine grid reference solution. We can see that the adaptive mesh results are a considerable improvement on the fixed grid results both in the corner region of the solution and in regions where the solution is smooth.

4.2.2. Example 5. We next consider a problem with a nonconvex flux

$$\phi_t - \cos(1 + \phi_x + \phi_y) = 0, \quad (x, y) \in (-2, 2)^2, \tag{4.6}$$

$$\phi(x, y, 0) = -\cos(\pi(x + y)/2), \tag{4.7}$$

with periodic boundary conditions. The numerical results for this case are shown in Figures 13 and 14. We again can see that the adaptive moving mesh algorithm does a good job of clustering elements towards the discontinuities in the solution gradient. The numerical results are comparable to the adaptive calculations presented in [28].

4.2.3. Example 6. Our final example is a prototype model in geometric optics [16], [22]. The governing equation is

$$\phi_t + \sqrt{1 + \phi_x^2 + \phi_y^2} = 0, \quad (x, y) \in (0, 1)^2, \tag{4.8}$$

$$\phi(x, y, 0) = 0.25(\cos(2\pi x) - 1)(\cos(2\pi y) - 1) - 1, \tag{4.9}$$

with periodic boundary conditions. The viscosity solution for this problem is a surface with its maximum focusing into a peak which opens up. The surface moves downward

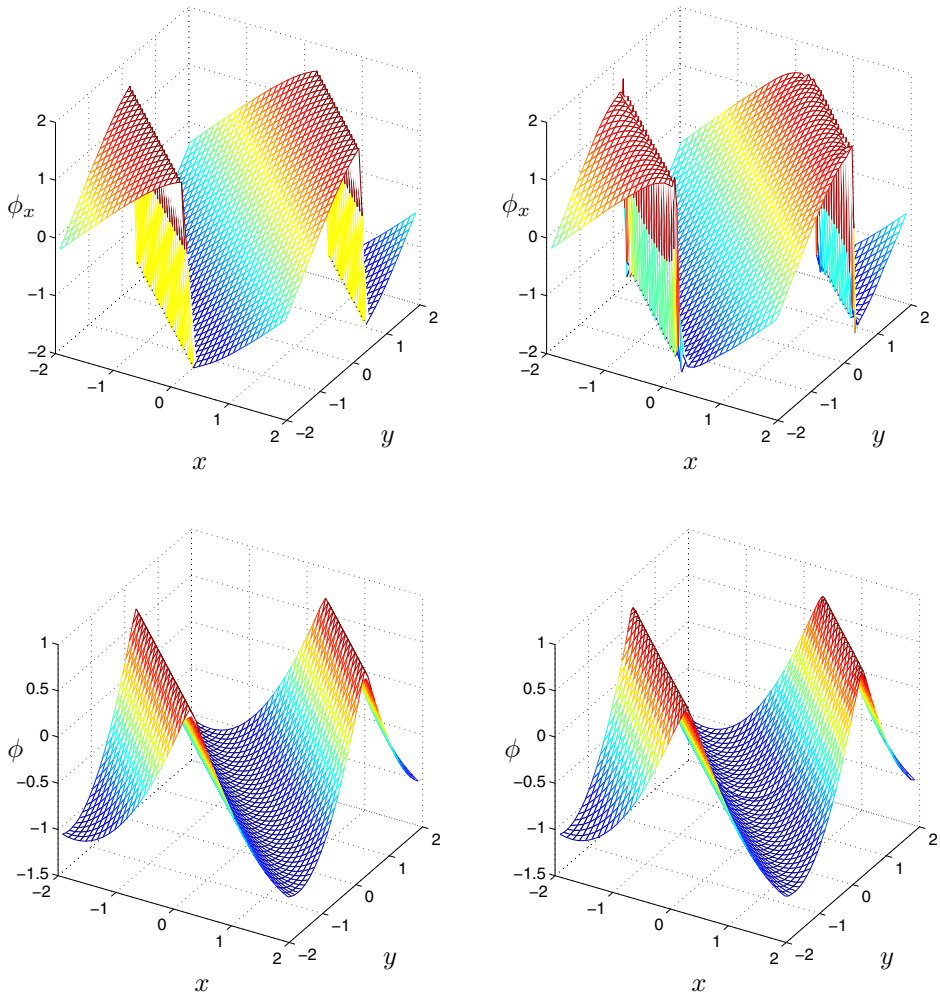


FIG. 10. *Approximations of ϕ_x (top) and ϕ (bottom) at $t = 1.5/\pi^2$ using uniform mesh (left) and moving mesh (right), with $N = 40$ for Example 4.*

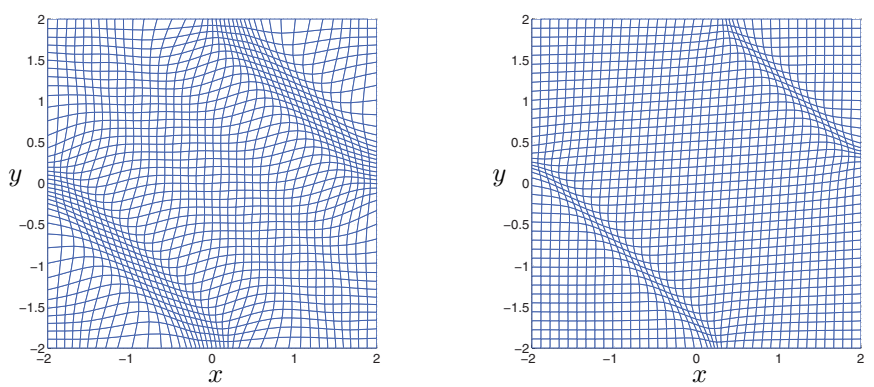


FIG. 11. *Adapted meshes at $t = 0.5/\pi^2$ (left) and $t = 1.5/\pi^2$ (right) obtained with $N = 40$ for Example 4.*

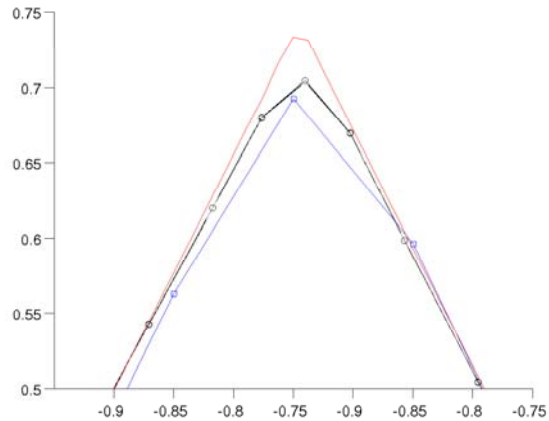


FIG. 12. *Comparison of numerical approximation of ϕ obtained using a uniform mesh with $N = 40$ ($\square$), moving mesh with $N = 40$ ($\circ$), and uniform mesh $N = 320$ (—) for Example* 4.

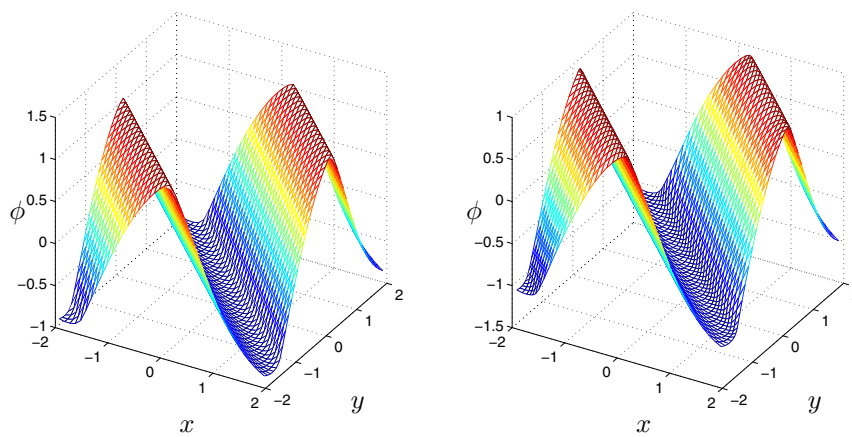


FIG. 13. *Approximations of ϕ at $t = 1.5/\pi^2$ using a uniform mesh (left) and adaptive moving mesh (right), where $CFL = 0.05$ and $N = 40$, for Example* 5.

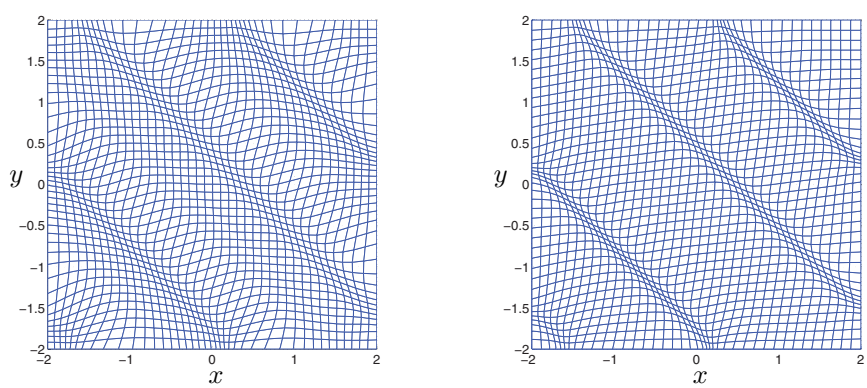


FIG. 14. *40×40 adaptive meshes at $t = 0.5/\pi^2$ (left) and $t = 1.5/\pi^2$ (right) for Example* 5.

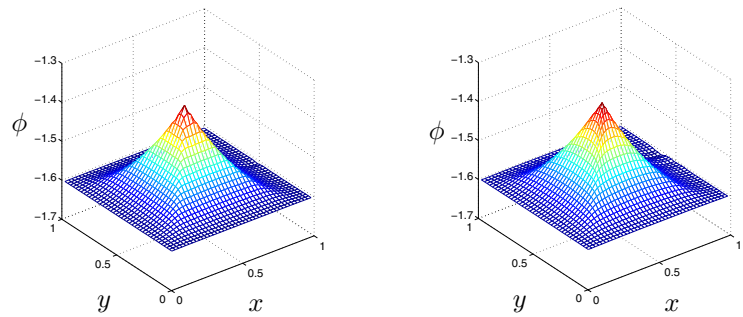


FIG. 15. *Approximations of* ϕ *at* $t = 0.6$ *using a uniform mesh (left) and moving mesh (right) for Example* 6.

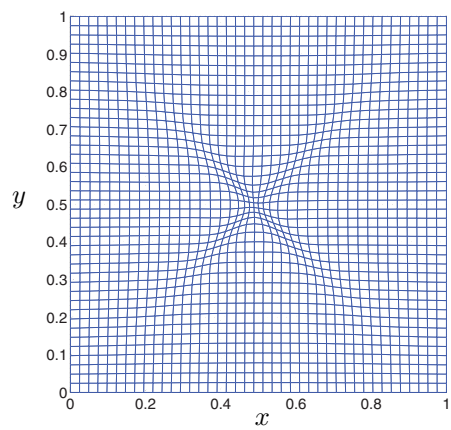


FIG. 16. 40×40 *adapted mesh at* $t = 0.6$ *for Example* 6.

with a unit speed, asymptotically approaching a flat sheet. Figure 15 shows the computed approximations of ϕ using a uniform and adaptive mesh with $N = 40$. We can see that the adaptive moving mesh does a better job of resolving the focusing of the solution. The mesh at the final time is shown in Figure 16, where we can see that the mesh has been concentrated towards the focusing of the solution at the center of the domain.

5. Conclusions. We have presented a DG method for the solution of nonlinear HJ equations using an adaptive moving mesh refinement strategy. A suitable mesh refinement criterion was used which does not become singular when the solution become nonsmooth. Numerical experiments in one and two dimensions show that this method works well and efficiently delivers high-resolution solutions using relatively coarse meshes.

Although we have focused here on HJ equations, the moving mesh DG method can also be applied to other hyperbolic problems such as the compressible Euler equations, and work is progressing in this direction. Finally, solution adaptivity was achieved in this paper using mesh movement. Future work will include a study of the combination of mesh movement along with additional refinement strategies such as

h and p refinement. The DG framework should hopefully make this combination of strategies easier to implement.

REFERENCES

[1] B. N. Azarenok, *Variational barrier method of adaptive grid generation in hyperbolic problems of gas dynamics*. SIAM J. Numer. Anal., 40 (2002), pp. 651–682.

[2] G. Beckett, J. A. Mackenzie, A. Ramage, and D. M. Sloan, *On the numerical solution of one-dimensional PDEs using adaptive methods based on equidistribution*, J. Comput. Phys., 167 (2001), pp. 372–392.

[3] G. Beckett, J. A. Mackenzie, A. Ramage, and D. M. Sloan, *Computational solution of two-dimensional unsteady PDEs using moving mesh methods*, J. Comput. Phys., 182 (2002), pp. 478–495.

[4] K. S. Bey and J. T. Oden, *hp-version discontinuous Galerkin methods for hyperbolic conservation laws*, Comput. Methods Appl. Mech. Engrg., 133 (1996), pp. 259–286.

[5] B. Cockburn, G. E. Karniadakis, and C. W. Shu, *Discontinuous Galerkin Methods: Theory, Computation and Applications*, Springer, Berlin, 2000.

[6] B. Cockburn and C. W. Shu, *The Runge-Kutta discontinuous Galerkin method for conservation laws. V. Multidimensional systems*, J. Comput. Phys., 141 (1998), pp. 199–224.

[7] M. G. Crandall and P. L. Lions, *Two approximations of solutions of Hamilton-Jacobi equations*, Math. Comput., 43 (1984), pp. 1–19.

[8] M. J. Crandall and P. L. Lions, *Viscosity solutions of Hamilton-Jacobi equations*, Trans. Amer. Math. Soc., 277 (1983), pp. 1–42.

[9] E. A. Dorfi and L. O'C. Drury, *Simple adaptive grids for 1-D initial value problems*, J. Comput. Phys., 69 (1987), pp. 175–195.

[10] P. Houston, B. Senior, and E. Suli, *hp-discontinuous Galerkin finite element methods for hyperbolic problems: Analysis and adaptivity*, Internat. J. Numer. Methods Fluids, 40 (2002), pp. 153–169.

[11] C. Hu and C.-W. Shu, *A discontinuous Galerkin finite element method for Hamilton–Jacobi equations*, SIAM J. Sci. Comput., 21 (1999), pp. 666–690.

[12] W. Huang, *Practical aspects of formulation and solution of moving mesh partial differential equations*, J. Comput. Phys., 171 (2001), pp. 753–775.

[13] W. Huang and R. D. Russell, *Analysis of moving mesh partial differential equations with spatial smoothing*, SIAM J. Numer. Anal., 34 (1997), pp. 1106–1126.

[14] W. Huang and R. D. Russell, *Moving mesh strategy based on a gradient flow equation for two-dimensional problems*, SIAM J. Sci. Comput., 20 (1999), pp. 998–1015.

[15] G.-S. Jiang and D. Peng, *Weighted ENO schemes for Hamilton–Jacobi equations*, SIAM J. Sci. Comput., 21 (2000), pp. 2126–2143.

[16] S. Jin and Z. Xin, *Numerical passage from systems of conservation laws to Hamilton–Jacobi equations, and relaxation schemes*, SIAM J. Numer. Anal., 35 (1998), pp. 2385–2404.

[17] P. Knupp and S. Steinberg, *Fundamentals of Grid Generation*, CRC Press, Boca Raton, FL, 1994.

[18] O. Lepsky, C. Hu, and C. W. Shu, *Analysis of the discontinuous Galerkin method for Hamilton-Jacobi equations*, Appl. Numer. Math., 33 (2000), pp. 423–434.

[19] F. Li and C. W. Shu, *Reinterpretation and simplified implementation of a discontinuous Galerkin method for Hamilton-Jacobi equations*, Appl. Math. Lett., 18 (2005), pp. 1204–1209.

[20] R. Li and T. Tang, *Moving mesh discontinuous Galerkin method for hyperbolic conservation laws*, J. Sci. Comp., 27 (2006), pp. 347–363.

[21] C.-T. Lin and E. Tadmor, *High-resolution nonoscillatory central schemes for Hamilton–Jacobi equations*, SIAM J. Sci. Comput., 21 (2000), pp. 2163–2186.

[22] S. Osher and J. A. Sethian, *Fronts propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations*, J. Comput. Phys., 79 (1988), pp. 12–49.

[23] S. Osher and C.-W. Shu, *High-order essentially nonoscillatory schemes for Hamilton–Jacobi equations*, SIAM J. Numer. Anal., 28 (1991), pp. 907–922.

[24] J. M. Stockie, J. A. Mackenzie, and R. D. Russell, *A moving mesh method for one-dimensional hyperbolic conservation laws*, SIAM J. Sci. Comput., 22 (2001), pp. 1791–1813.

[25] P. K. Sweby, *High resolution schemes using flux limiters for hyperbolic conservation laws*, SIAM J. Numer. Anal., 21 (1984), pp. 995–1011.

[26] Z. Tan, Z. Zhang, Y. Huang, and T. Tang, *Moving mesh methods with locally varying time steps*, J. Comput. Phys., 200 (2004), pp. 347–367.

[27] H. Tang and T. Tang, *Moving mesh methods for one- and two-dimensional hyperbolic conservation laws*, SIAM J. Numer. Anal., 41 (2003), pp. 487–515.

[28] H.-Z. Tang, T. Tang, and P.-W. Zhang, *An adaptive mesh redistribution method for nonlinear Hamilton-Jacobi equations in two- and three-dimensions*, J. Comput. Phys., 188 (2003), pp. 543–572.

[29] J. F. Thompson, Z. A. Warsi, and C. W. Mastin, *Numerical Grid Generation: Foundations and Applications*, North–Holland, New York, 1985.

[30] N. K. Yamaleev and M. H. Carpenter, *On accuracy of adaptive grid methods for captured shocks*, J. Comput. Phys., 181 (2002), pp. 280–316.

[31] P. A. Zegeling, W. D. de Boer, and H. Tang, *Robust and efficient adaptive moving mesh solution of the* 2*-D Euler equations*, in Recent Advances in Adaptive Computation, Contemp. Math. 383, AMS, Providence, RI, 2005, pp. 375–386.

SIAM J. SCI. COMPUT.
Vol. 29, No. 6, pp. 2283–2304
© 2007 Society for Industrial and Applied Mathematics

SIMULTANEOUS HIGHER-ORDER OPTICAL FLOW ESTIMATION AND DECOMPOSITION*

JING YUAN†, CHRISTOPH SCHÖRR†, AND GABRIELE STEIDL‡

Abstract. We study the estimation and decomposition of optical flows from highly nonrigid motions. To this end, recent methods from image decomposition into structural and textural parts are combined with variational optical flow estimation. The approaches we suggest amount to minimizing discrete convex functionals using second-order cone programming. Higher-order regularization is necessary in order to accurately recover important flow structure like vortices, and to incorporate key physical properties such as vanishing divergence. For proper discretization, we apply the finite mimetic difference method, which preserves the identities fulfilled by the continuous differential operators. Numerical examples demonstrate the feasibility of the complex approaches.

Key words. optical flow estimation, optical flow decomposition, divergence-free flows, higher-order regularization, mimetic finite difference method, convex optimization, dual optimization techniques, second-order cone programming

AMS subject classifications. 68Y10, 65K10, 90C25, 76-04

DOI. 10.1137/060660709

1. Introduction.

The representation, estimation, and analysis of nonrigid motion is relevant to many scenarios in computer vision, medical imaging, remote sensing, and experimental fluid dynamics. Especially in fluid dynamics, sophisticated measurement techniques including pulsed laser light sheets and modern CCD cameras, as well as dedicated hardware, enable the recording of high-resolution image sequences that reveal the evolution of spatio-temporal structures of unsteady flows [25]. In this connection, an important research problem is to develop variational approaches that render flow estimation from image sequences into a well-posed and numerically stable problem while preserving small-scale flow structures that are important for empirical investigations of turbulent phenomena.

To this end, we investigate a novel class of variational flow estimation schemes by combining higher-order flow regularization with recent techniques developed for nonsmooth image decomposition–see below for a more detailed exposition. As a result, we obtain variational approaches that not only allow for estimating fluid flow from image sequences but simultaneously yield a decomposition of the flow into coherent spatio-temporal flow patterns and small-scale structures.

In the following, we briefly describe the respective basic ideas in a continuous setting. In the remainder of the paper, we will derive and investigate discrete approaches using the so-called mimetic finite difference method developed by Hyman and Shashkov [17], which preserves the integral identities fulfilled by the corresponding continuous integral operators.

Image decomposition. In image denoising one is typically interested in removing noise without destroying important structures such as edges. This goal cannot be

*Received by the editors May 24, 2006; accepted for publication (in revised form) March 14, 2007; published electronically October 5, 2007.

http://www.siam.org/journals/sisc/29-6/66070.html

†CVGPR-Group, University of Mannheim, 68131 Mannheim, Germany (yuanjing@uni-mannheim.break de).

‡Institute of Mathematics, University of Mannheim, 68131 Mannheim, Germany (steidl@math.uni-mannheim.de).

achieved with linear filters, e.g., by minimizing

$$J(f) := \|g - f\|^2_{L_2(\Omega)} + \lambda \int_\Omega |\nabla f|^2 \,\mathrm{d}x\mathrm{d}y \tag{1.1}$$

for a given noisy image $g(x, y)$ in $\Omega \subset \mathbb{R}^2$. The regularizer incorporates the quadratic function $\Phi(s) = s^2$ with $s = |\nabla f|$. Via the Euler–Lagrange equation this variational approach can be related to a linear diffusion equation. As a consequence, the solution f smooths g in a completely homogeneous way and therefore blurs semantically important signal structures. To overcome this drawback a variety of nonlinear methods have been proposed. One of the frequently applied approaches replaces the function in the regularization term by $\Phi(s) = s$ and thus penalizes larger deviations of $|\nabla f|$ not as hard as the quadratic function:

$$J(f) := \frac{1}{2} \|g - f\|^2_{L_2(\Omega)} + \lambda \int_\Omega |\nabla f| \,\mathrm{d}x\mathrm{d}y. \tag{1.2}$$

This functional was first considered by Rudin, Osher, and Fatemi [26]. For f having partial derivatives in L_1 the regularizer coincides with the TV seminorm of f,

$$\|f\|_{\mathrm{TV}} := \int |\nabla f| \,\mathrm{d}x\mathrm{d}y.$$

In contrast to the *linear approach* (1.1) we will refer to this method as the *TV approach*. Based on the dual TV norm, the so-called G norm, this denoising model was enlarged for the decomposition of given images g into a structural (cartoon) part f^s and a textural part f^t as

$$\begin{aligned} J(f^s, f^t) &= \|g - (f^s + f^t)\|^2_{L_2(\Omega)} + \lambda \int_\Omega |\nabla f^s| \,\mathrm{d}x\mathrm{d}y, \\ &\text{s.t.} \quad \|f^t\|_{\mathrm{G}} \le \delta. \end{aligned} \tag{1.3}$$

For a more sophisticated treatment of the TV and G norms, we refer to [11, 21]. Meanwhile there exist various numerical realizations of (1.3), e.g., [3, 24, 32, 34]. Although not considered in this paper, we note that there exist other models including other than Gaussian white noise by using the L_1 norm in the data fitting term [5, 23], noise as a third decomposition component by applying Besov norms [4], or frame and TV/frame approaches [10, 28] from harmonic analysis. Moreover, second-order derivatives were incorporated into the regularizer to avoid, for example, staircasing effects; see, e.g., [8, 29].

In this paper, we are interested in the decomposition of vector fields rather than scalar images. Specifically, we want to deal with optical flow fields arising, e.g., in experimental fluid dynamics.

Optical flow estimation. Instead of a single image let us consider an image sequence $\{g(x, y, t) : t \in [0, T]\}$ with a time parameter t. A common assumption is that image intensities are preserved over time:

$$g\big(x(t_1), y(t_1), t_1\big) = g\big(x(t_2), y(t_2), t_2\big), \quad t_2 > t_1. \tag{1.4}$$

Generalizations to other constraints exist but are not relevant for our present investigation. The linearized version of the gray value constancy assumption (1.4) yields the *optical flow constraint*

$$g_x u_1 + g_y u_2 + g_t = \nabla g \cdot u + g_t = 0, \tag{1.5}$$

where the optical flow field $u = (u_1, u_2)^\top := (\dot{x}, \dot{y})^\top$ denotes the instantaneous velocities of image elements.

Obviously, (1.5) cannot be solved pointwise because at each location and time point it consists of solving a single scalar equation for two scalar unknowns. To overcome this so-called *aperture problem* additional requirements have to be imposed. Instead of (1.5) we consider the least squares approximation

$$F(u) := \|\nabla g \cdot u + g_t\|^2_{L_2(\Omega)}. \tag{1.6}$$

In 1981, Horn and Schunck [16] pioneered the field of regularization methods for optical flow computation by suggesting minimization of the functional

$$J(u) = F(u) + \lambda \int_\Omega |\nabla u_1|^2 + |\nabla u_2|^2 \,\mathrm{d}x\mathrm{d}y. \tag{1.7}$$

We will refer to this widely used approach as the *Horn–Schunck model*. Obviously, the Horn–Schunck model suffers from the same drawbacks as the linear image filtering model (1.1). The solution creates blurry optic flow fields where the blur appears also across important flow discontinuities. One way to overcome this limitation consists again if using another function in the regularizer, e.g.,

$$J(u) = \frac{1}{2} F(u) + \lambda \int_\Omega (|\nabla u_1|^2 + |\nabla u_2|^2)^{\frac{1}{2}} \,\mathrm{d}x\mathrm{d}y, \tag{1.8}$$

as counterpart to the TV model (1.2); cf. [2, 13, 15].

In this paper we focus on nonrigid motion analysis. In particular, we are interested in the representation of motion by components that capture different physical aspects, e.g., solenoidal (divergence-free) flows. Referring again to experimental fluid dynamics, for example, the extraction of coherent flow structures which are immersed into additional motion components at different spatial scales [19] poses a challenge for image sequence analysis. However, this goal cannot be achieved using first-order derivatives in the regularizer. Rather, based on early work on second-order regularizers constraining the gradients of the flows' divergence and curl [1, 9, 14, 30], instead of (1.7) we deal with the functional

$$J(u) = F(u) + \lambda_d \int_\Omega |\nabla \mathrm{div}\, u|^2 \,\mathrm{d}x\mathrm{d}y + \lambda_c \int_\Omega |\nabla \mathrm{curl}\, u|^2 \,\mathrm{d}x\mathrm{d}y + \gamma \int_{\partial\Omega} (\partial_{\mathrm{n}} u)^2 \,\mathrm{d}s, \tag{1.9}$$

where the boundary is incorporated for stability reasons; cf. [35, 37]. A key property of this approach is that due to the second order of the flow derivatives involved, the regularizing terms do *not* penalize the magnitude of the basic flow components involving first-order derivatives, i.e., divergence and curl. To further motivate this approach consider Figure 1.1, which shows the estimation of a solenoidal flow field by first- and second-order approaches. The figure clearly demonstrates the superiority of the div-curl model.

Finally, related to the TV model (1.8), we deal with

$$J(u) = \frac{1}{2} F(u) + \lambda_d \int_\Omega |\nabla \mathrm{div}\, u| \,\mathrm{d}x\mathrm{d}y + \lambda_c \int_\Omega |\nabla \mathrm{curl}\, u| \,\mathrm{d}x\mathrm{d}y + \frac{\gamma}{2} \int_{\partial\Omega} (\partial_{\mathrm{n}} u)^2 \,\mathrm{d}s, \tag{1.10}$$

involving regularizing terms that are of second order, as in (9), and that are additionally suited to preserve jumps of the divergence and the curl of a flow field,

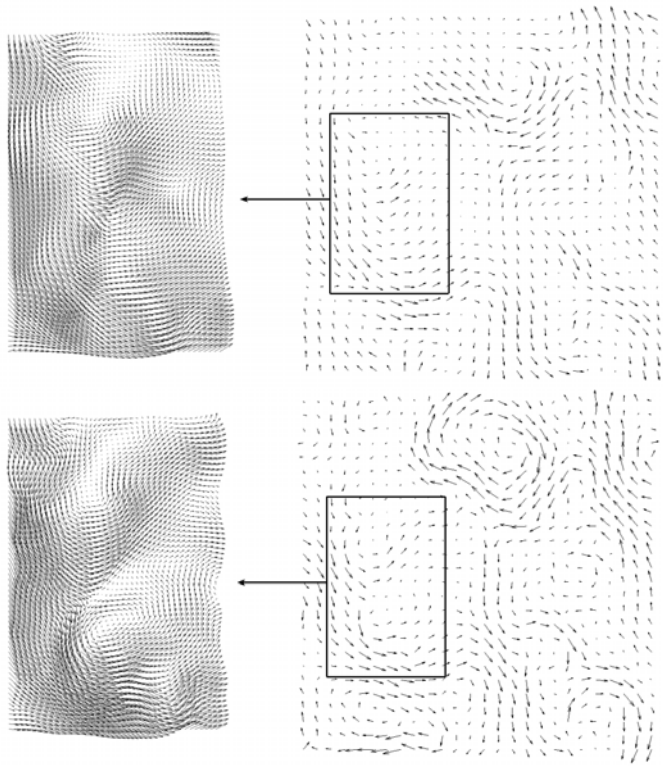


FIG. 1.1. *Estimation of a typical solenoidal flow field u. Top: Restored flow based on the Horn–Schunck model* (1.7). *Bottom: Restored flow based on the second-order model* (1.9). *Vortex structures are better recovered by the div-curl approach (bottom); cf.* [35].

respectively, by utilizing the TV-norm. As in image decomposition we will extend this estimation approach to the decomposition of motion vector fields into physically relevant components at different scales by using a discrete equivalent of the G norm. Moreover, we will study the feasibility of an extension to the *simultaneous* estimation and decomposition of optical flows.

Organization of the paper. We start by introducing our discrete setting based on the mimetic finite difference method in section 2. We define scalar and vector fields on primal and dual grids together with appropriate norms. Then we introduce the first-order operators, where we focus on their matrix representation using tensor product notation. We show how integral identities such as the Gaussian integral identity and the Helmholtz decomposition carry over to the discrete situation. Finally, we define discrete TV and G norms with respect to the primal and dual grid. Based on these definitions and results we consider the flow estimation task in section 3. Here we are interested in the linear versus the TV approach. While the linear method was already treated in [35], the higher-order TV approach is novel. In section 4, we deal with flow decomposition by the TV-G norm model. To this end we have to introduce a decomposition of the optical flow into components with constant divergence and curl and variable components which can be further decomposed into a "structural" part and a "textural" part. In fact, these parts comprise flow patterns at different scales. In section 5, we combine the flow estimation and the decomposition model in

order to solve both tasks simultaneously. In sections 3, 4, and 5 we especially focus on the practically relevant case of solenoidal flows. Some parts of sections 4 and 5 were previously announced at a conference [36]. Section 6 provides the numerical algorithms to solve the convex optimization problems posed in sections 3, 4, and 5. We use second-order cone programming (SOCP) [12, 20]. Finally, we demonstrate the feasibility of our approaches by numerical examples in section 7.

2. Discretization. For our discrete versions of the functionals we need a careful discretization which preserves the identities satisfied by the continuous operators as, e.g., $\operatorname{div} = -\nabla^*$, $\operatorname{curl}\nabla = 0$, the Gaussian integral identity, and the Helmholtz decomposition. For discretizing the relevant spaces and differential operators we apply the *mimetic finite difference method* introduced by Hyman and Shashkov in [17]. This method defines the corresponding discrete operators of ∇, div, and curl simultaneously on two grids. These grids, which we call the *primal* and *dual grid*, are shifted with respect to each other by half a pixel.

Primal and dual scalar/vector fields. Corresponding to the grids, we define the following scalar fields and vector fields:

H_P: space of scalar fields on vertices,
H_S: space of vector fields defined normal to sides,
H_V: space of scalar field on cells,
H_E: space of vector fields defined tangential to sides.

By H_P^o, H_S^o, and H_E^o we denote the corresponding spaces with zero boundary elements. Finally, let $H_{V+\partial V}$ and $H_{E+\partial E}$ be the extended version of H_V and H_E, respectively, to the boundary. If our primal grid consists of $m \times n$ vertices, then $\dim H_P = mn$, $\dim H_V = (m-1)(n-1)$, $\dim H_S = m(n-1)+n(m-1)$, and $\dim H_E = (m-1)(n-2)+(n-1)(m-2)$. The fields are illustrated in Figure 2.1.

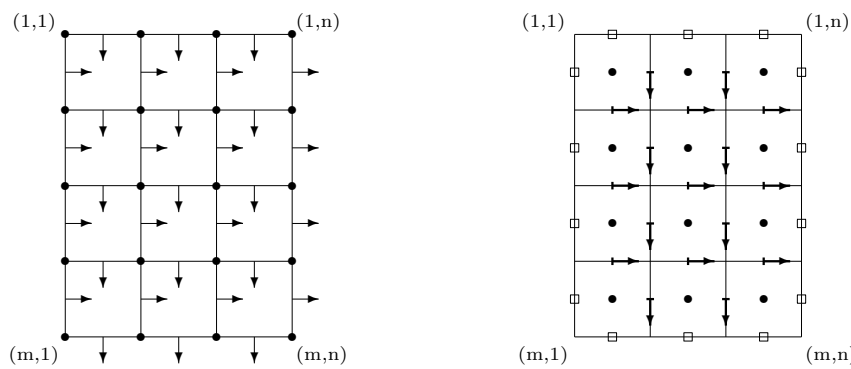


FIG. 2.1. *Primal and dual grid. Left: Spaces H_P (dots) and H_S (arrows). Right: Spaces H_V (dots), $H_{\partial V}$ (squares), and H_E (arrows) for $m = 5$ and $n = 4$.*

While H_P and H_V are equipped with the usual ℓ_2 inner product, the inner products on H_S and H_E are defined with respect to the cells. To this end, we consider the elements of H_S associated with the elements of H_V by

$$u_{V_{i+\frac{1}{2},j+\frac{1}{2}}} := \frac{1}{\sqrt{2}} \left(u_{i,j+\frac{1}{2}}, u_{i+1,j+\frac{1}{2}}, u_{i+\frac{1}{2},j}, u_{i+\frac{1}{2},j+1}\right)^{\mathrm{T}}$$

for $i = 1, \ldots, m-1, j = 1, \ldots, n-1$. Then we define a dotproduct $\cdot : H_S \times H_S \to H_V$

by

$$u \cdot v := \Big((u \cdot v)_{i+\frac{1}{2},j+\frac{1}{2}}\Big)_{i,j=1}^{m-1,n-1} = \left(\left\langle u_{V_{i+\frac{1}{2},j+\frac{1}{2}}}, v_{V_{i+\frac{1}{2},j+\frac{1}{2}}} \right\rangle\right)_{i,j=1}^{m-1,n-1},$$

where $\langle \cdot, \cdot \rangle$ denotes the usual inner product on ℓ_2. Now the inner product on the vector field H_S is given by

$$\langle u, v \rangle_{H_S} := \sum_{i=1}^{m-1} \sum_{j=1}^{n-1} (u \cdot v)_{i+\frac{1}{2},j+\frac{1}{2}}$$

and the corresponding norm by

$$\begin{aligned} \|u\|_{H_S}^2 &:= \left\| \big(\|u_{V_{i+\frac{1}{2},j+\frac{1}{2}}}\|_2 \big)_{i,j=1}^{m-1,n-1} \right\|_2^2 \\ &= \frac{1}{2} \sum_{i=1}^{m-1} \sum_{j=1}^{n-1} (u_{i,j+\frac{1}{2}}^2 + u_{i+1,j+\frac{1}{2}}^2 + u_{i+\frac{1}{2},j}^2 + u_{i+\frac{1}{2},j+1}^2). \end{aligned}$$

Related to the ℓ_1 norm we define

$$\begin{aligned} |u|_{H_S} &:= \left\| \big(\|u_{V_{i+\frac{1}{2},j+\frac{1}{2}}}\|_2 \big)_{i,j=1}^{m-1,n-1} \right\|_1 \\ &= \frac{1}{\sqrt{2}} \sum_{i=1}^{m-1} \sum_{j=1}^{n-1} \sqrt{u_{i,j+\frac{1}{2}}^2 + u_{i+1,j+\frac{1}{2}}^2 + u_{i+\frac{1}{2},j}^2 + u_{i+\frac{1}{2},j+1}^2}. \end{aligned}$$

Similarly, we associate elements of $u \in H_E$ with elements of H_P^o by

$$u_{P_{i,j}} := \frac{1}{\sqrt{2}} \left(u_{i,j-\frac{1}{2}}, u_{i,j+\frac{1}{2}}, u_{i-\frac{1}{2},j}, u_{i+\frac{1}{2},j}\right)^{\mathrm{T}}$$

for $i = 2, \ldots, m-1, j = 2, \ldots, n-1$, and set

$$\|u\|_{H_E}^2 := \left\| \big(\|u_{P_{i,j}}\|_2 \big)_{i,j=2}^{m-1,n-1} \right\|_2^2, \qquad |u|_{H_E} := \left\| \big(\|u_{P_{i,j}}\|_2 \big)_{i,j=2}^{m-1,n-1} \right\|_1.$$

Primal and dual operators. We use the following discrete versions of the first-order operators ∇, div, and curl with respect to the primal and dual grid:

$$\mathbb{G}rad : H_P \to H_{E+\partial E}, \qquad \mathbb{G}rad^{\perp} : H_P \to H_S$$

and

$$\begin{array}{lll} \mathbb{G} : H_P^o \to H_E^o, & \mathbb{D}iv : H_S \to H_V, & \mathbb{G}^{\perp} : H_P^o \to H_S^o, \\ \overline{\mathbb{G}} : H_V \to H_S^o, & \overline{\mathbb{D}iv} : H_E^o \to H_P^o, & \overline{\mathbb{C}url} : H_S^o \to H_P^o. \end{array}$$

Reshaping the scalar/vector fields columnwise into vectors of appropriate lengths, these operators act on the corresponding vector spaces as matrices which can be described by using the first-order forward difference matrix

$$D_m := \begin{pmatrix} -1 & 1 & 0 & \ldots & 0 & 0 & 0 \\ 0 & -1 & 1 & \ldots & 0 & 0 & 0 \\ & & \ddots & \ddots & \ddots & & \\ 0 & 0 & 0 & \ldots & -1 & 1 & 0 \\ 0 & 0 & 0 & \ldots & 0 & -1 & 1 \end{pmatrix} \in \mathbb{R}^{m-1,m}$$

and the tensor product $\otimes$ of matrices. Then we have

$$\mathbb{G}rad = \begin{pmatrix} I_n \otimes D_m \\ D_n \otimes I_m \end{pmatrix}, \quad \mathbb{G}rad^{\perp} = \begin{pmatrix} -D_n \otimes I_m \\ I_n \otimes D_m \end{pmatrix},$$

which explains the notation $^{\perp}$. The operator $\mathbb{G}^{\perp}$ is just the restriction of $\mathbb{G}rad^{\perp}$ to H_P^o. Although $\mathbb{G}$ is not the restriction of $\mathbb{G}rad$ to H_P^o, we will again use the notation $^{\perp}$. More precisely, we define

$$\mathbb{G} = \begin{pmatrix} I_{n-2} \otimes D_{m-2} \\ D_{n-2} \otimes I_{m-2} \end{pmatrix}, \quad \mathbb{G}^{\perp} = \begin{pmatrix} D_{n-1}^{\mathrm{T}} \otimes I_{m-2} \\ -I_{n-2} \otimes D_{m-1}^{\mathrm{T}} \end{pmatrix},$$

$$\mathbb{D}iv \ = \begin{pmatrix} I_{n-1} \otimes D_m, D_n \otimes I_{m-1} \end{pmatrix},$$

where I_n denotes the $n \times n$ identity matrix and the matrices are considered without the zero rows/columns due to embedding. Then we have on the dual grid

$$\overline{\mathbb{G}} = -(\mathbb{D}iv \,_{|_{H_S^o}})^*, \quad \overline{\mathbb{D}iv} = -\mathbb{G}^*, \quad \overline{\mathbb{C}url} = (\mathbb{G}^{\perp})^*, \tag{2.1}$$

where the adjoint operator corresponds to the transposed matrices. Using properties of the tensor product of matrices, we obtain that $\operatorname{curl} \nabla = 0$ reads as

$$\overline{\mathbb{C}url}\ \overline{\mathbb{G}} = 0, \quad \mathbb{D}iv\ \mathbb{G}^{\perp} = -\overline{\mathbb{G}}^{\,*}\, \overline{\mathbb{C}url}^{\,*} = 0. \tag{2.2}$$

We extend the operator $\overline{\mathbb{G}}$ to $H_{V+\partial V}$ by incorporating the boundary elements $H_{\partial V}$ into the forward differences. Since the distance of a boundary point to its neighboring inner point is only $\frac{1}{2}$, we have to multiply the difference filter $(-1, 1)$ at the boundary by 2. The matrix of the resulting operator,

$$\overline{\mathbb{G}}_1 : \ H_{V+\partial V} \to H_S,$$

is given in the appendix. Finally, we define another extension of $\overline{\mathbb{G}}$ in one of the following ways:

- apply a linear extrapolation operator from H_V to $H_{V+\partial V}$ and then $\overline{\mathbb{G}}_1$,
- apply $\overline{\mathbb{G}}$ and then a constant extrapolation from H_S^o to H_S.

Both procedures result in the same operator,

$$\overline{\mathbb{G}}_2 : \ H_V \to H_S,$$

given in matrix form in the appendix.

Discrete integral identities. Next we are interested in a discrete version of the *Gaussian integral identity* $\int_\Omega \operatorname{div} u \, \mathrm{d}x = \int_{\partial\Omega} n \cdot u \, \mathrm{d}s$. To this end, we introduce the boundary operator

$$\mathbb{B} := \begin{pmatrix} I_{n-1} \otimes B_m & 0 \\ 0 & B_n \otimes I_{m-1} \end{pmatrix},$$

where 0 are zero matrices of appropriate sizes and

$$B_m := \begin{pmatrix} -1 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 1 \end{pmatrix} \in \mathbb{R}^{2,m}.$$

Then the *mimetic Gaussian integral identity* becomes

$$1^{\mathrm{T}}_{\dim H_V}\,\mathbb{D}iv\;u = 1^{\mathrm{T}}_{\dim \partial H_V}\,\mathbb{B}u. \tag{2.3}$$

We will apply the following discrete version of the Helmholtz decomposition.

PROPOSITION 2.1 (mimetic Helmholtz decomposition). *Any $u \in H_S$ can be decomposed into an irrotational part u_{irr} and a soleniodal part u_{sol} by*

$$u = u_{\mathrm{irr}} + u_{\mathrm{sol}} = \overline{\mathbb{G}}_1\psi + \mathbb{G}^{\perp}\varphi, \tag{2.4}$$

where $\varphi \in H^o_P$ is uniquely determined and $\psi \in H_{V+\partial V}$ is uniquely determined up to an additive constant. Moreover, the decomposition is orthogonal in the sense that $\langle \overline{\mathbb{G}}_1\psi, \mathbb{G}^{\perp}\varphi\rangle_{H_S} = 0$.

Proof. By (2.2) and since $\overline{\mathbb{G}}_1$ considered as mapping onto H^o_S equals $\overline{\mathbb{G}}$, we see that

$$\overline{\mathbb{C}url}\,u_{\mathrm{irr}} = \overline{\mathbb{C}url}\;\overline{\mathbb{G}}_1\,\psi = 0, \quad \mathbb{D}iv\;u_{\mathrm{sol}} = \mathbb{D}iv\;\mathbb{G}^{\perp}\varphi = 0.$$

Therefore we have indeed an irrotational and a solenoidal part.

By definition of the matrices it is easy to check that $\dim\mathcal{R}(\overline{\mathbb{G}}_1) = \dim H_{V+\partial V} - 1 = mn + m + n - 4$, and $\dim\mathcal{R}(\mathbb{G}^{\perp}) = \dim H^o_P = (m-2)(n-2)$ so that $\dim\mathcal{R}(\overline{\mathbb{G}}_1) + \dim\mathcal{R}(\mathbb{G}^{\perp}) = \dim H_S$. Here $\mathcal{R}$ denotes the range of the operator. By (2.1) and (2.2) we obtain that

$$\langle \overline{\mathbb{G}}_1\psi, \mathbb{G}^{\perp}\varphi\rangle_{H_S} = \langle \overline{\mathbb{C}url}\;\overline{\mathbb{G}}_1\psi, \varphi\rangle_{H^o_P} = 0,$$

and consequently $H_S = \mathcal{R}(\overline{\mathbb{G}}_1) \oplus \mathcal{R}(\mathbb{G}^{\perp})$ with the orthogonal sum $\oplus$.

Since $\mathbb{G}^{\perp}$ has full rank, the potential φ is uniquely determined by u_{sol}. Since the kernel of $\overline{\mathbb{G}}_1$ is given by $\mathcal{N}(\overline{\mathbb{G}}_1) = \{c1_{\dim H_{V+\partial V}} : c \in \mathbb{R}\}$ we have that ψ is uniquely determined by u_{irr} up to an element from the kernel. This completes the proof. ∎

Conversely, assume for $u \in H_S$ that

$$\begin{aligned} \rho = \mathbb{D}iv\;u &= \triangle_d\,\psi, \\ \omega = \overline{\mathbb{C}url}\,u &= \triangle_c\,\varphi, \end{aligned}$$

where

$$\triangle_d := \mathbb{D}iv\;\overline{\mathbb{G}}_1, \quad \triangle_c := \overline{\mathbb{C}url}\,\mathbb{G}^{\perp}$$

are given. Since $\triangle_c : H^o_P \to H^o_P$ is invertible, the potential φ is uniquely determined by ω, while ψ (and u) are given only up to a laminar flow u_{lam} with vanishing $\mathbb{D}iv$ and $\overline{\mathbb{C}url}$. The laminar flow can be determined by boundary conditions where we have to take the mimetic Gaussian integral identity (2.3) into account. More precisely, we have the following proposition; cf. [36].

PROPOSITION 2.2. *The operator*

$$\mathcal{H} := \begin{pmatrix} \mathbb{D}iv \\ \hline \overline{\mathbb{C}url} \\ \hline \mathbb{B} \end{pmatrix} \in \mathbb{R}^{\dim H_S + 1, \dim H_S}$$

provides a bijection between the spaces H_S and

$$V_S := \{(\rho, \omega, \nu)^T : 1^T_{\dim H_V}\,\rho = 1^T_{\dim H_{\partial V}}\,\nu\}.$$

More precisely, if $(\rho,\omega,\nu)^T = \mathcal{H}u$, *then* $u = \mathcal{H}^\dagger(\rho,\omega,\nu)^T$; *i.e., the inverse mapping on* V_S *is given by the pseudoinverse* $\mathcal{H}^\dagger := (\mathcal{H}^T\mathcal{H})^{-1}\mathcal{H}^T$ *of* $\mathcal{H}$.

In case of a solenoidal flow field from $\{u \in H_S : \mathbb{D}iv\, u = 0\}$, we have by Proposition 2.1 that $\mathbb{D}iv\, \overline{\mathbb{G}}_1\psi = 0$ and $\overline{\mathbb{C}url}\, \overline{\mathbb{G}}_1\psi = 0$. Hence $\overline{\mathbb{G}}_1\psi = u_{\text{lam}}$ is a laminar flow and

$$u = u_{\text{lam}} + u_{\text{sol}} = \overline{\mathbb{G}}_1\psi + \mathbb{G}^\perp\varphi, \quad \mathcal{H}u = (0,0,\nu)^{\text{T}} + (0,\omega,0)^{\text{T}}. \tag{2.5}$$

Further we have the following proposition.

PROPOSITION 2.3. *Every soleniodal flow field* $u \in H_S$ *can be written as*

$$u = \mathbb{G}rad^\perp\phi \tag{2.6}$$

for some $\phi \in H_P$.

Proof. By properties of the tensor product it is easy to check that $\mathbb{D}iv\, \mathbb{G}rad^\perp = 0$ so that the right-hand side of (2.6) is indeed solenoidal.

By Proposition 2.2 we obtain that $\dim(\{u \in H_S : \mathbb{D}iv\, u = 0\}) = mn - 1$. On the other hand, we have $\dim(\mathcal{R}(\mathbb{G}rad^\perp)) = \dim H_P - 1 = mn - 1$. ∎

Primal and dual TV/G norms. For our optical flow decomposition and estimation we need discrete versions of the TV norm $\|f\|_{\text{TV}} := \int |\nabla f|\, dxdy$ and the G norm $\|f\|_{\text{G}} := \inf_{f=\text{div}p} \| |p| \|_{L_\infty}$, where $|p| := (p_1^2 + p_2^2)^{1/2}$ for a vector field $p = (p_1, p_2)$, $p_1, p_2 \in L_\infty$. For a more sophisticated treatment of these norms in the continuous setting, see, e.g., [11, 21].

For $\omega \in H_P^o$ we set

$$\|\omega\|_{\text{TV}} := |\mathbb{G}\,\omega|_{H_E^o}.$$

This defines a seminorm on H_P^o. As usual we will skip the prefix "semi" in the following. Let $N := \dim H_P^o$. According to the norm in H_E^o let $S : \mathbb{R}^{\dim H_E^o} \to \mathbb{R}^{4N}$ be the matrix which assigns the four appropriate elements of H_E^o to each point of H_P^o; cf. the appendix. For $w := (w^1, w^2, w^3, w^4)^{\text{T}} \in \mathbb{R}^{4N}$ with $w^r := (w_j^r)_{j=1}^N$, $r = 1, \ldots, 4$, we define the vector

$$|w| := \left(\|(w_j^r)_{r=1}^4\|_2\right)_{j=1}^N. \tag{2.7}$$

Then the TV norm can be rewritten as

$$\|\omega\|_{\text{TV}} = \| \, |S\,\mathbb{G}\,\omega| \, \|_1.$$

Now the G norm of ω is defined by

$$\|\omega\|_{\text{G}} := \min_{\omega = \mathbb{G}^* S^* w} \| \, |w| \, \|_\infty = \min_{\omega = \mathbb{D}iv\ S^* w} \| \, |w| \, \|_\infty.$$

Similarly, we can define the TV and the G norm on H_V. For $\rho \in H_V$ let

$$\|\rho\|_{\overline{\text{TV}}} := \left|\overline{\mathbb{G}}_2\,\rho\right|_{H_S}.$$

We focus on the second way of extending $\overline{\mathbb{G}}$ to $\overline{\mathbb{G}}_2$ and introduce the operator $\overline{S} : \mathbb{R}^{\dim H_S^o} \to \mathbb{R}^{4N}$ with $N := \dim H_V$, which constantly extrapolates from H_S^o to H_S and assigns the elements of H_S to those of H_V. For the matrix structure of $\overline{S}$, see the appendix. Then we obtain that

$$\|\rho\|_{\overline{\text{TV}}} = \| \, |\overline{S}\,\overline{\mathbb{G}}\,\rho| \, \|_1$$

and

$$\|\rho\|_{\overline{\mathrm{G}}} := \min_{\rho=\overline{\mathbb{G}}*\overline{S}*p} \| \, |p| \, \|_\infty = \min_{\rho=\mathbb{D}iv\,\overline{S}*p} \| \, |p| \, \|_\infty. \tag{2.8}$$

The G norm ($\overline{\mathrm{G}}$ norm) can be considered as dual to the TV norm ($\overline{\mathrm{TV}}$ norm) in the following sense.

PROPOSITION 2.4. *For $\tilde{\omega} \in \mathcal{R}(\mathbb{G}^* S^*)$ the following relation holds true:*

$$\sup_{\|\omega\|_{\mathrm{TV}} \neq 0} \frac{|\langle \tilde{\omega}, \omega \rangle|}{\|\omega\|_{\mathrm{TV}}} = \|\tilde{\omega}\|_{\mathrm{G}}. \tag{2.9}$$

Proof. 1. Set

$$L := S\mathbb{G} : \mathbb{R}^N \to \mathbb{R}^{4N}, \quad N := \dim H_E^o.$$

Then, using the notation (2.7), the relation (2.9) can be rewritten as

$$\sup_{\| \, |L\omega| \, \|_1 \neq 0} \frac{|\langle \tilde{\omega}, \omega \rangle|}{\| \, |L\omega| \, \|_1} = \min_{\tilde{\omega}=L^{\mathrm{T}} v} \| \, |v| \, \|_\infty, \qquad \tilde{\omega} \in \mathcal{R}(L^{\mathrm{T}}).$$

Let $\nu(\tilde{\omega}) := \sup_{\| \, |L\omega| \, \|_1 \neq 0} \frac{|\langle \tilde{\omega}, \omega \rangle|}{\| \, |L\omega| \, \|_1}$. By applying the Schwarz inequality to the four parts of the corresponding vectors in $\mathbb{R}^{4N}$, we obtain for all $v \in \mathbb{R}^{4N}$ with $L^{\mathrm{T}} v = \tilde{\omega}$ that

$$\langle \tilde{\omega}, \omega \rangle = \langle v, L\omega \rangle \leq \langle |v|, |L\omega| \rangle \quad \forall \omega \in \mathbb{R}^N,$$

and then obviously

$$\langle |v|, |L\omega| \rangle \leq \| \, |L\omega| \, \|_1 \| \, |v| \, \|_\infty \qquad \forall \omega \in \mathbb{R}^N.$$

Consequently, we have

$$\nu(\tilde{\omega}) \leq \min_{\tilde{\omega}=L^{\mathrm{T}} v} \| \, |v| \, \|_\infty.$$

To show the reverse direction we consider the subspace $\mathcal{B} := \mathcal{R}(L)$ of $\mathbb{R}^{4N}$ equipped with the norm $\| \, |\cdot| \, \|_1$. Let $\tilde{w}$ with $L^{\mathrm{T}} \tilde{w} = \tilde{\omega}$ be fixed. Then the mapping $l_{\tilde{w}}(L\omega) := \langle \tilde{w}, L\omega \rangle$ is a linear functional on $\mathcal{B}$ which has exactly the norm $\nu(\tilde{\omega})$. By the Hahn–Banach theorem this functional can be extended to a linear functional l on $(\mathbb{R}^{4N}, \| \, |\cdot| \, \|_1)$ with $\|l\| = \|l_{\tilde{w}}\|$. Then there exists $\tilde{v} \in \mathbb{R}^{4N}$ such that $l(v) = \langle \tilde{v}, v \rangle$ for all $v \in \mathbb{R}^{4N}$ and

$$\langle \tilde{v}, L\omega \rangle = \langle \tilde{w}, L\omega \rangle \quad \forall \omega \in \mathbb{R}^N.$$

Since this can be rewritten as

$$\langle L^{\mathrm{T}} \tilde{v}, \omega \rangle = \langle L^{\mathrm{T}} \tilde{w}, \omega \rangle \quad \forall \omega \in \mathbb{R}^N$$

the vector $\tilde{v}$ must fulfil $L^{\mathrm{T}} \tilde{v} = L^{\mathrm{T}} \tilde{w} = \tilde{\omega}$. Then

$$\nu(\tilde{\omega}) = \|l_{\tilde{w}}\| = \|l\| = \| \, |\tilde{v}| \, \|_\infty,$$

where the last equality follows from the fact that $\| \, |\cdot| \, \|_1$ and $\| \, |\cdot| \, \|_\infty$ are dual norms on $\mathbb{R}^{4N}$. Thus, $\nu(\tilde{\omega}) \geq \min_{\tilde{\omega}=L^{\mathrm{T}} v} \| \, |v| \, \|_\infty$, and we obtain the assertion. ∎

3. Flow estimation. Assume now that we are given an image sequence $\{g(\cdot, j) \in H_V : j = 1, \dots, T\}$. For fixed $j \in \mathbb{N}$, we set $g_t = g_t(j) := g(\cdot, j+1) - g(\cdot, j)$. Then the data fitting functional (1.6) can be discretized by

$$F(u) = \|\overline{\mathbb{G}}_2 g \cdot u + g_t\|^2_{H_V}$$

with the dotproduct on H_S and by applying the mimetic Helmholtz decomposition (2.4) by

$$F(\psi, \varphi) = \|\overline{\mathbb{G}}_2 g \cdot \overline{\mathbb{G}}_1 \psi + \overline{\mathbb{G}}_2 g \cdot \mathbb{G}^{\perp} \varphi + g_t\|^2_{H_V}.$$

This further decomposition can be useful from two points of view. First the vectors ψ and φ have only approximately half the lengths of the vector u. Second this approach allows for efficiently handling solenoidal flows, which are very common in computational fluid dynamics.

To discretize the boundary integral $\int_{\partial\Omega} (\partial_{\mathrm{n}} u)^2 \, \mathrm{d}s$ in (1.9) and (1.10) we use the operator

$$\mathbb{B}^{\mathrm{n}} := \begin{pmatrix} I_{n-1} \otimes B^{\mathrm{n}}_m & 0 \\ 0 & B^{\mathrm{n}}_n \otimes I_{m-1} \end{pmatrix}$$

with

$$B^{\mathrm{n}}_m := \begin{pmatrix} -1 & 1 & \dots & 0 & 0 \\ 0 & 0 & \dots & -1 & 1 \end{pmatrix} \in \mathbb{R}^{2,m}.$$

For the smoothing penalizer we distinguish between the linear and the TV approach.

Linear approach. The discrete version of the functional (1.9) becomes

$$J(u) = F(u) + \lambda_d \left\|\overline{\mathbb{G}}_2 \, \mathbb{D}iv \, u\right\|^2_{H_S} + \lambda_c \left\|\mathbb{G} \, \overline{\mathbb{C}url} \, u\right\|^2_{H^o_E} + \gamma \left\|\mathbb{B}^{\mathrm{n}} u\right\|^2_{\partial H_S}. \tag{3.1}$$

The minimizer u of (3.1) can be obtained by solving the corresponding normal equation.

Using the Helmholtz decomposition together with $u_{|_{\partial H_S}} = (\overline{\mathbb{G}}_1 \psi)_{|_{\partial H_S}}$ and (2.2), problem (3.1) can be rewritten as

$$J(\psi, \varphi) = F(\psi, \varphi) + \lambda_d \left\|\overline{\mathbb{G}}_2 \triangle_d \psi\right\|^2_{H_S} + \lambda_c \left\|\mathbb{G} \triangle_c \varphi\right\|^2_{H^o_E} + \gamma \left\|\mathbb{B}^{\mathrm{n}} \overline{G}_1 \psi\right\|^2_{\partial H_S}. \tag{3.2}$$

Since ψ is determined only up to a constant, we add the additional condition

$$1^{\mathrm{T}}_{\dim H_{V+\partial V}} \psi = 0$$

to make the functional strictly convex. This problem can be solved by subspace correction methods [31, 33]. More precisely, we iterate the following process: we consider φ as given and find the solution ψ of (3.2) via the normal equation. Then we fix this solution as ψ and compute φ.

In case of a solenoidal flow field $u \in H_S$ we can apply (2.5), and our variational problem reduces to

$$\begin{aligned} J(\psi, \varphi) &= F(\psi, \varphi) + \lambda_c \left\|\mathbb{G} \triangle_c \varphi\right\|^2_{H^o_E} + \gamma \left\|\mathbb{B}^{\mathrm{n}} \overline{G}_1 \psi\right\|^2_{\partial H_S} \\ \text{s.t.} \quad & \overline{\mathbb{G}}_2 \triangle_d \psi = 0, \quad 1^{\mathrm{T}}_{\dim H_{V+\partial V}} \psi = 0. \end{aligned}$$

The linear case was more extensively treated by some of the authors in [35].

TV approach. Functional (1.10) becomes

$$J(u) = \frac{1}{2}\,F(u) + \lambda_d\,\|\mathbb{D}iv\;u\|_{\overline{\mathrm{TV}}} + \lambda_c\,\left\|\overline{\mathbb{C}url}\;u\right\|_{\mathrm{TV}} + \frac{\gamma}{2}\;\|\mathbb{B}^{\mathrm{n}}u\|^2_{\partial H_S}\,, \tag{3.3}$$

and by applying the Helmholtz decomposition,

$$J(\psi,\varphi) = \frac{1}{2}\,F(\psi,\varphi) + \lambda_d\,\|\triangle_d\,\psi\|_{\overline{\mathrm{TV}}} + \lambda_c\,\|\triangle_c\,\varphi\|_{\mathrm{TV}} + \frac{\gamma}{2}\;\left\|\mathbb{B}^{\mathrm{n}}\overline{G}_1\psi\right\|^2_{\partial H_S}\,, \tag{3.4}$$

where $1^{\mathrm{T}}_{\dim H_{V+\partial V}}\psi = 0$. In case of solenoidal flows the functional further reduces as in the linear case.

To solve (3.3) we propose using SOCP; cf. section 6.

Alternatively, we can again apply a splitting algorithm in (3.4) which iteratively fixes φ as previously computed and finds the minimizer ψ and conversely. Then any of the subproblems with fixed φ or fixed ψ can be written in the form

$$\frac{1}{2}\|Ax-b\|_2^2 + \lambda\|\,|Lx|\,\|_1, \tag{3.5}$$

where $A : \mathbb{R}^{\dim H_{V+\partial V}} \to \mathbb{R}^{\dim H_V}, L : \mathbb{R}^{\dim H_{V+\partial V}} \to \mathbb{R}^{4\dim H_V}$ in case of fixed φ and $A : \mathbb{R}^{\dim H_P^o} \to \mathbb{R}^{\dim H_V},\ L : \mathbb{R}^{\dim H_P^o} \to \mathbb{R}^{4\dim H_P^o}$ in case of fixed ψ. The minimizer of (3.5) can be found by applying dual optimization techniques as proposed by Chambolle and coworkers [3, 7, 6].

4. Flow decomposition. In this section, we want to decompose a given flow vector $u \in H_S$ in a meaningful way. To this end we have to compute some basic decomposition of u first. We apply Proposition 2.2 and consider $\mathcal{H}u = (\rho,\omega,\nu)^{\mathrm{T}}$. Let

$$c_\rho := \frac{1^{\mathrm{T}}_{\dim H_V}\,\rho}{\dim H_V}\,, \quad c_\omega := \frac{1^{\mathrm{T}}_{\dim H_P^o}\,\omega}{\dim H_P^o}$$

be the discrete versions of $|\Omega|^{-1}\int_\Omega \operatorname{div} u\,\mathrm{d}x\mathrm{d}y$ and $|\Omega|^{-1}\int_\Omega \operatorname{curl} u\,\mathrm{d}x\mathrm{d}y$ and

$$\rho_{\mathrm{const}} := c_\rho\,1_{\dim H_V}\,, \quad \omega_{\mathrm{const}} := c_\omega\,1_{\dim H_P^o}. \tag{4.1}$$

Then we can decompose $(\rho,\omega,\nu)^{\mathrm{T}} \in V_S$ as

$$(\rho,\omega,\nu) = (\rho_{\mathrm{const}},\omega_{\mathrm{const}},\nu) + (\rho_{\mathrm{var}},\omega_{\mathrm{var}},0), \tag{4.2}$$

where $1^{\mathrm{T}}_{\dim H_V}\rho_{\mathrm{var}} = 1^{\mathrm{T}}_{\dim H_P^o}\omega_{\mathrm{var}} = 0$. Obviously, both summands are in V_S again, so that

$$u = u_{\mathrm{const}} + u_{\mathrm{var}}$$

is the corresponding basic decomposition of $u \in H_S$, where

$$u_{\mathrm{const}} = \mathcal{H}^\dagger((\rho_{\mathrm{const}},\omega_{\mathrm{const}},\nu)^{\mathrm{T}}, \quad u_{\mathrm{var}} = \mathcal{H}^\dagger(\rho_{\mathrm{var}},\omega_{\mathrm{var}},0)^{\mathrm{T}}.$$

We call vector u_{const} (resp., $(\rho_{\mathrm{const}},\omega_{\mathrm{const}},\nu)$) the basic pattern of the nonrigid flow and its boundary behavior, while u_{var} (resp., $(\rho_{\mathrm{var}},\omega_{\mathrm{var}},0)$) is related to the variable (oscillating) flow pattern.

We are interested in further decomposing the intrinsic flow variation u_{var} into a structural part u^s and a texture part u^t, i.e.,

$$u_{\mathrm{var}} = u^s + u^t.$$

This can be done in two ways. The *first approach* uses Proposition 2.2. Given $u \in H_S$, we compute $(\rho, \omega, \nu)^{\mathrm{T}} = \mathcal{H}u$ and then $\rho_{\mathrm{var}} = \rho - \rho_{\mathrm{const}}$ and $\omega_{\mathrm{var}} = \omega - \omega_{\mathrm{const}}$ by (4.1). Next we decompose $\rho_{\mathrm{var}} = \rho^s + \rho^t$ and $\omega_{\mathrm{var}} = \omega^s + \omega^t$ by minimizing separately

$$J(\rho^s, \rho^t) = \frac{1}{2}\,\|\rho_{\mathrm{var}} - \rho^s - \rho^t\|_2^2 \;+\; \lambda_d \|\rho^s\|_{\overline{\mathrm{TV}}} \;+\; \mu_d \|\rho^t\|_{\overline{\mathrm{G}}},$$
$$J(\omega^s, \omega^t) = \frac{1}{2}\,\|\omega_{\mathrm{var}} - \omega^s - \omega^t\|_2^2 \;+\; \lambda_c \|\omega^s\|_{\mathrm{TV}} \;+\; \mu_c \|\omega^t\|_{\mathrm{G}}.$$

The minimizers can be computed as proposed in [3]. By (2.8) we see that $\rho^t \in \mathcal{R}(\overline{\mathbb{G}}^*) = \mathcal{R}(\mathbb{D}iv_{|_{H_S^o}})$, and since $1^{\mathrm{T}}\mathbb{D}iv_{|_{H_S^o}} = 0$ we have that $(\rho^t, \omega^t, 0)^{\mathrm{T}} \in V_S$ and further by (4.2) that $(\rho^s, \omega^s, 0)^{\mathrm{T}} \in V_S$. Thus we can finally compute u^s and u^t by

$$u^s = \mathcal{H}^\dagger(\rho^s, \omega^s, 0)^{\mathrm{T}}, \quad u^t = \mathcal{H}^\dagger(\rho^t, \omega^t, 0)^{\mathrm{T}}.$$

In this paper, we prefer a *second approach* that computes the components of u directly. This variational approach extends Meyer's model for the decomposition of scalar-valued functions to the *simultaneous* decomposition of vector fields into basic and variable (structural and textural) flow patterns. Moreover, it also fits into our flow estimation-decomposition model in the next section. For $u \in H_S$ we propose to to find $u_{\mathrm{const}} \in H_S$ and $u^s, u^t \in H_S^o$ by minimizing

$$\begin{aligned} J(u_{\mathrm{const}}, u^s, u^t) &= \lambda_d \,\|\mathbb{D}iv\; u^s\|_{\overline{\mathrm{TV}}} + \lambda_c \left\|\overline{\mathbb{C}url}\; u^s\right\|_{\mathrm{TV}} \\ \text{s.t.}\quad & \|\mathbb{D}iv\; u^t\|_{\overline{\mathrm{G}}} \le \delta_d\,, \quad \|\overline{\mathbb{C}url}\; u^t\|_{\mathrm{G}} \le \delta_c, \\ & u_{\mathrm{const}} + u^s + u^t = u, \\ & \overline{\mathbb{G}}\mathbb{D}iv\; u_{\mathrm{const}} = 0, \quad \mathbb{G}\overline{\mathbb{C}url}\; u_{\mathrm{const}} = 0, \\ & 1^{\mathrm{T}}_{\dim H_P^o}\overline{\mathbb{C}url}\; u^s = 0. \end{aligned} \tag{4.3}$$

Concerning the last three constraints we note the following: to obtain the desired decomposition we have to ensure that $\mathbb{D}iv\; u_{\mathrm{const}}$ and $\overline{\mathbb{C}url}\; u_{\mathrm{const}}$ are constant vectors $c\,1$ and that $1^{\mathrm{T}}_{\dim H_V}\mathbb{D}iv\;(u^s + u^t) = 0$, $1^{\mathrm{T}}_{\dim H_P^o}\overline{\mathbb{C}url}\,(u^s + u^t) = 0$. The first two conditions are fulfilled by the fourth and fifth constraint. The third condition is fulfilled by the mimetic Gaussian integral identity and since u^s, $u^t \in H_S^o$. The last condition follows by the last constraint and since $\overline{\mathbb{C}url}\; u^t \in \mathcal{R}(\overline{\mathbb{D}iv}\,)$ and $1^{\mathrm{T}}\overline{\mathbb{D}iv} = 0$.

The solution of (4.3) can be found by SOCP, as shown in section 6.

5. Flow estimation and decomposition. In this section, we combine optical flow estimation with structure-texture flow decomposition. Given image sequence $\{g(\cdot, j) \in H_V : j = 1, \ldots, T\}$, we want to compute the components u_{const} with constant divergence and curl, the large scale patterns $u^s \in H_S^o$ of divergence and curl with bounded TV norms, and the small scale patterns $u^t \in H_S^o$ of divergence and curl with bounded G norms. To this end, we introduce the fitting functional

$$F(u_{\mathrm{const}}, u^s, u^t) = \|\overline{\mathbb{G}}_2 g \cdot (u_{\mathrm{const}} + u^s + u^t) + g_t\|_2^2.$$

Then, with respect to the previous section, one can consider minimizing

$$\begin{aligned} J(u_{\mathrm{const}}, u^s, u^t) = \frac{1}{2}\,F(u_{\mathrm{const}}, u^s, u^t) &+ \lambda_d\|\mathbb{D}iv\; u^s\|_{\overline{\mathrm{TV}}} + \lambda_c\|\overline{\mathbb{C}url}\; u^s\|_{\mathrm{TV}} + \frac{\gamma}{2}\|\mathbb{B}^n u\|^2_{\partial H_S} \\ \text{s.t.}\quad & \|\mathbb{D}iv\; u^t\|_{\overline{\mathrm{G}}} \le \delta_d\,, \quad \|\overline{\mathbb{C}url}\; u^t\|_{\mathrm{G}} \le \delta_c, \\ & \overline{\mathbb{G}}\mathbb{D}iv\; u_{\mathrm{const}} = 0, \quad \mathbb{G}\overline{\mathbb{C}url}\; u_{\mathrm{const}} = 0, \\ & 1^{\mathrm{T}}_{\dim H_P^o}\overline{\mathbb{C}url}\; u^s = 0. \end{aligned}$$

Unfortunately, this approach is not well posed. For the image areas where $\nabla g = 0$, the data term disappears, and the local constraints on the G norms will lead to unbounded solutions. Therefore, we propose to replace the G norm by the L_2 norm, which leads to

$$\begin{aligned} J(u_{\text{const}}, u^s, u^t) = \frac{1}{2} F(u_{\text{const}}, u^s, u^t) + \lambda_d \|\mathbb{D}iv\ u^s\|_{\overline{\text{TV}}} + \lambda_c \|\overline{\mathbb{C}url}\ u^s\|_{\text{TV}} \\ + \mu_d \|\mathbb{D}iv\ u^t\|_2^2 + \mu_c \|\overline{\mathbb{C}url}\ u^t\|_2^2 + \frac{\gamma}{2} \|\mathbb{B}^n u\|_{\partial H_S}^2, \\ \text{s.t.} \quad \overline{\mathbb{G}}\mathbb{D}iv\ u_{\text{const}} = 0, \quad \mathbb{G}\overline{\mathbb{C}url}\ u_{\text{const}} = 0, \\ 1_{\dim H_P^o}^{\text{T}} (\overline{\mathbb{C}url}\ u^s + \overline{\mathbb{C}url}\ u^t) = 0. \end{aligned} \tag{5.1}$$

We will see in our experiments that this approach works well, although the superiority of the G norm over the L_2 norm in capturing (scalar) oscillating patterns was experimentally shown in [4].

Finally, we are interested in solenoidal flows $u \in H_S$. Coupling the mimetic Helmholtz decomposition and the basic decomposition, we obtain with the laminar flow $\overline{\mathbb{G}}_1 \psi$ and $\varphi = \varphi_{\text{const}} + \varphi^s + \varphi^t$ that

$$u = u_{\text{const}} + u^s + u^t = (\overline{\mathbb{G}}_1 \psi + \mathbb{G}^{\perp} \varphi_{\text{const}}) + \mathbb{G}^{\perp} \varphi^s + \mathbb{G}^{\perp} \varphi^t.$$

By Proposition 2.3 and since $\mathbb{G}rad^{\perp}_{|H_P^o} = \mathbb{G}^{\perp}$ this can be rewritten as

$$u = \mathbb{G}rad^{\perp}(\phi_{\text{const}} + \varphi^s + \varphi^t), \qquad \phi \in H_P, \varphi^s, \varphi^t \in H_P^o.$$

Then the fitting term reads

$$F(\phi_{\text{const}}, \varphi^s, \varphi^t) = \|\overline{\mathbb{G}}_2 g \cdot \mathbb{G}rad^{\perp}(\phi_{\text{const}} + \varphi^s + \varphi^t) + g_t\|_2^2,$$

and (5.1) can be rewritten as

$$\begin{aligned} J(\phi_{\text{const}}, \varphi^s, \varphi^t) = &\frac{1}{2} F(\phi_{\text{const}}, \varphi^s, \varphi^t) + \lambda_c \|\triangle_c \varphi^s\|_{\text{TV}} + \mu_c \|\triangle_c \varphi^t\|_2^2 \\ &+ \frac{\gamma}{2} \|\mathbb{B}^n \mathbb{G}rad^{\perp} \phi_{\text{const}}\|_{\partial H_S}^2 \\ \text{s.t.} \quad &\mathbb{G}\,\overline{\mathbb{C}url}\ \mathbb{G}rad^{\perp} \phi_{\text{const}} = 0,\ 1_{\dim H_P^o}^{\text{T}} \triangle_c (\varphi^s + \varphi^t) = 0, \\ &1_{\dim H_P}^{\text{T}} \phi_{\text{const}} = 0, \end{aligned} \tag{5.2}$$

where the last constraint appears since ϕ_{const} is determined only up to an additive element from the kernel of $\mathbb{G}rad^{\perp}$, which consists of constant vector fields.

6. Algorithmic aspect—SOCP. Our computational approach to flow estimation via (4.3) and to simultaneous flow estimation and decomposition via (5.2) is based on SOCP [20]. This amounts to minimizing a linear objective function subject to the constraints that several affine functions of the variables have to lie in a *second-order cone* $\mathcal{L}^{n+1} \subset \mathbb{R}^{n+1}$ defined by the convex set

$$\mathcal{L}^{n+1} = \left\{ \begin{pmatrix} x \\ t \end{pmatrix} = (x_1, \ldots, x_n, t)^{\top} : \|x\|_2 \le t \right\}.$$

With this notation, the general form of an SOCP is given by

$$\inf_{x \in \mathbb{R}^n} f^{\top} x, \quad \text{s.t.} \begin{pmatrix} A_i x + b_i \\ c_i^T x + d_i \end{pmatrix} \in \mathcal{L}^{n+1}, \quad i = 1, \ldots, r. \tag{6.1}$$

Problem (6.1) is a convex program for which efficient large-scale solvers are available [22]. In connection with TV-based image decomposition the application of SOCPs was recently suggested in [12, 34].

We reformulate the variational approach (4.3) as the SOCP

$$
\begin{aligned}
&J(u_{\text{const}}, u^s, u^t) = \lambda_d 1^{\mathrm{T}}_{\dim H_V} v + \lambda_c 1^{\mathrm{T}}_{\dim H_P^o} w,\\
&\text{s.t. } u_{\text{const}} + u^s + u^t = u\,, \quad \overline{\mathbb{G}}\mathbb{D}iv\, u_{\text{const}} = 0\,, \quad \mathbb{G}\overline{\mathbb{C}url}\, u_{\text{const}} = 0,\\
&1^{\mathrm{T}}_{\dim H_P^o} \overline{\mathbb{C}url}\, u^s = 0\,, \quad \mathbb{D}iv\, u^t = \mathbb{D}iv\, p_d\,, \qquad \overline{\mathbb{C}url}\, u^t = \mathbb{D}iv\, p_c,\\
&\begin{pmatrix} (\overline{\mathbb{G}}\mathbb{D}iv\, u^s)_{V_{i+\frac12,j+\frac12}} \\ v_{V_{i+\frac12,j+\frac12}} \end{pmatrix} \in \mathcal{L}^5\,, \qquad \begin{pmatrix} (\overline{\mathbb{G}\mathbb{C}url}\, u^s)_{P_{i,j}} \\ w_{P_{i,j}} \end{pmatrix} \in \mathcal{L}^5,\\
&\begin{pmatrix} (p_d)_{V_{i+\frac12,j+\frac12}} \\ \delta_d \end{pmatrix} \in \mathcal{L}^5\,, \qquad \begin{pmatrix} (p_c)_{P_{i,j}} \\ \delta_c \end{pmatrix} \in \mathcal{L}^5.
\end{aligned}
$$

In order to incorporate the quadratic terms of the variational approaches to optical flow estimation, we use the following rotated version of the standard cone:

$$
\mathcal{R}^{n+2} := \left\{ \left(x, x_{n+1}, x_{n+2}\right)^\top \in \mathbb{R}^{n+2} : \frac{1}{2}\|x\|_2^2 \le x_{n+1}x_{n+2},\, x_{n+1}, x_{n+2} \ge 0 \right\}.
$$

Fixing $x_{n+2} = 1/2$, we have $\|x\|_2^2 \le x_{n+1}$. Now we can rewrite (5.2) as follows:

$$
\begin{aligned}
&J(\phi_{\text{const}}, \varphi^s, \varphi^t) = v + \mu_c t + \lambda_c 1^{\mathrm{T}}_{\dim H_P^o} w,\\
&\text{s.t.} \quad \mathbb{G}\overline{\mathbb{C}url}\, \mathbb{G}rad^{\perp}\phi_{\text{const}} = 0\,, \qquad 1^{\mathrm{T}}_{\dim H_P}\phi_{\text{const}} = 0,\\
&1^{\mathrm{T}}_{\dim H_P^o} \triangle_c(\varphi^s + \varphi^t) = 0,\\
&\begin{pmatrix} (\overline{\mathbb{G}}\triangle_c\varphi^s)_{V_{i+\frac12,j+\frac12}} \\ w_{V_{i,j}} \end{pmatrix} \in \mathcal{L}^5, \quad \begin{pmatrix} \triangle_c\varphi^t \\ t \\ 1/2 \end{pmatrix} \in \mathcal{R}^{\dim H_P+2},\\
&\begin{pmatrix} \overline{\mathbb{G}}_2 g \cdot \mathbb{G}rad^{\perp}(\phi_{\text{const}} + \varphi^s + \varphi^t) + g_t \\ v \\ 1/2 \end{pmatrix} \in \mathcal{R}^{\dim H_V+2}.
\end{aligned}
$$

7. Numerical examples. In this section we verify our approaches by numerical examples. The programs were written in MATLAB and used the software package SeDuMi for SOCP [27]. Unfortunately, we do not have an automatized choice of parameter values. Parameter values were chosen by hand following two general rules: (i) choose the weights as small as possible in order to not smooth out turbulent motion, (ii) for TV terms, smaller values than in the linear case (e.g., (21)) are appropriate, because these regularizers return larger values than their linear counterparts (assuming that image data are scaled to the range $[0, 1]$). Rule (ii) leads to parameter values of the order $\propto e^{-4}$. Thanks to the mimetic discretization, this suffices for numerical stability

Flow estimation. We start by comparing flow estimations obtained by the linear approach (3.2) and the TV approach (3.4). We consider the artificial example in Figure 7.1 with the groundtruth on top of Figure 7.2. Figure 7.2 (middle) was obtained by solving the linear systems of equations resulting from (3.2) with parameters

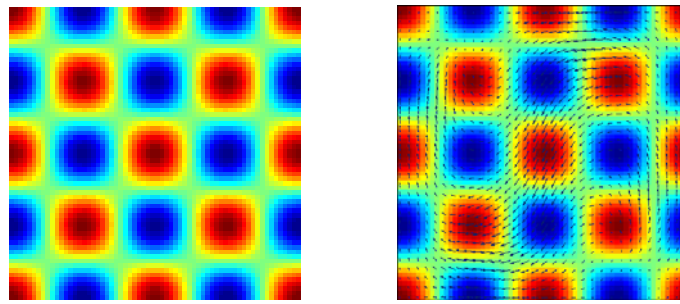


FIG. 7.1. *Frame g (left) and its warping with the groundtruth flow (right) for flow estimation by the linear approach and the TV approach.*

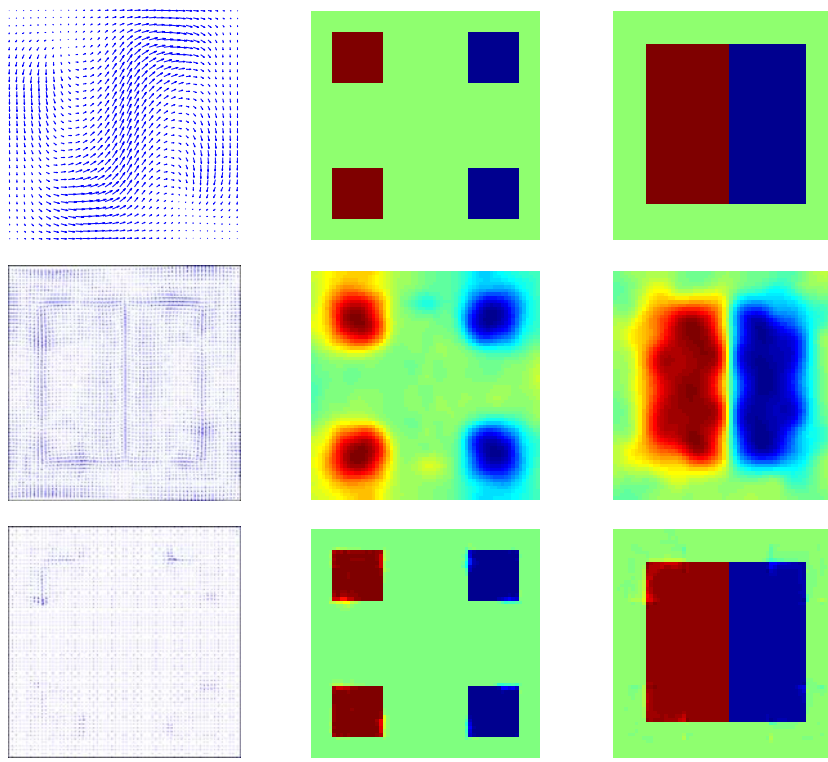


FIG. 7.2. *Top: The groundtruth flow u, its div and curl (left to right). Middle: Linear reconstruction by* (3.1)*, difference between u and its estimation, reconstructed div and curl (left to right). Bottom: TV reconstruction by* (3.3)*, difference between u and its estimation, reconstructed div and curl (left to right).*

$\lambda_d = 0.06$, $\lambda_c = 0.048$. As the boundary parameter we have used $\gamma = 0.04$. The result shows the typical blurring effects at the edges. If we decrease the parameter values, the rectangular shape of div and curl becomes easier to see, but the artifacts always visible in Figure 7.2 (middle) increase too. The bottom of Figure 7.2 contains the solution of (3.3) with the parameters $\lambda_d = e^{-6}$, $\lambda_c = e^{-6}$, and $\gamma = e^{-5}$ by SOCP and 17 iterations. As expected for this example, the TV approach gives very good results by preserving discontinuities of the flow derivatives (div, curl).

Flow decomposition. Figure 7.3 shows a turbulent flow field u as groundtruth along with its divergence ρ and curl ω. Applying the variational method (4.3) with $\lambda_d = 0.2\,,\ \lambda_c = 0.5$, and $\delta_d = 0.05\,,\ \delta_c = 0.1$, we obtain the decompositions depicted in Figures 7.4 and 7.5. The structural and textural components recovered the interesting motion patterns at different scales, which are not easily visible in the flow u itself.

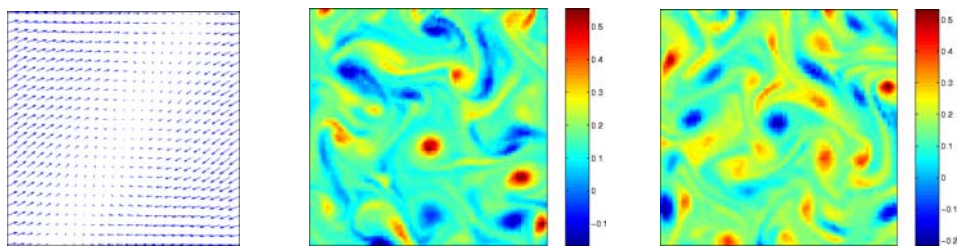


FIG. 7.3. *Groundtruth data to be decomposed: flow field u (left), its divergence field ρ (middle), and its curl field ω (right).*

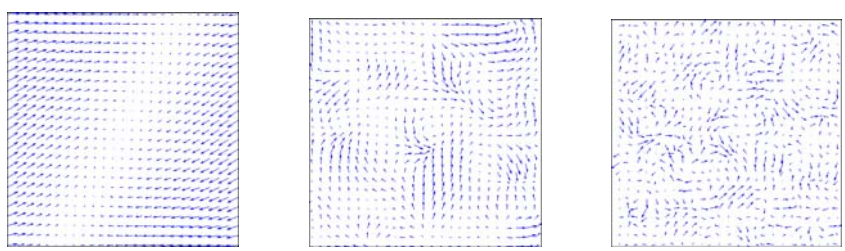


FIG. 7.4. *The components of the flow u from Figure* 7.3*: u_{const} (left), u^s (middle), and u^t (right). The vectors of u^s, u^t are scaled up for better visibility. Note that despite $|u| \approx |u_{\text{const}}|$, the structural and texture parts u^s and u^t are recovered well.*

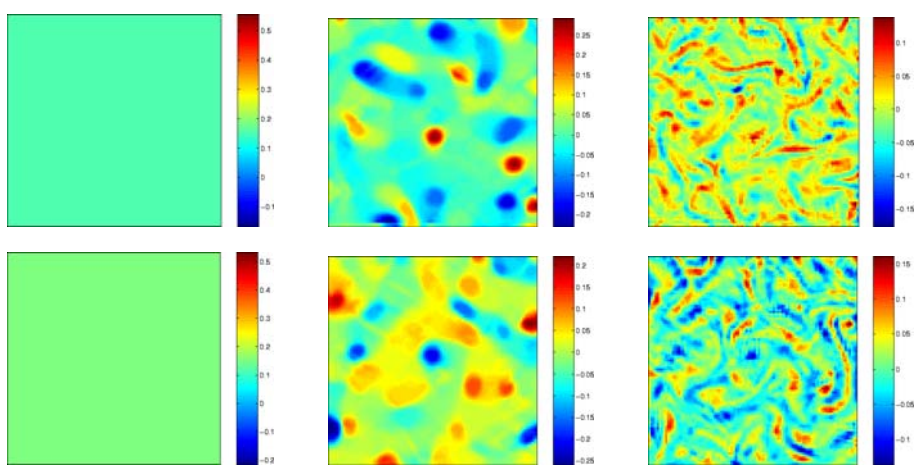


FIG. 7.5. *Decomposition of u from Figure* 7.3 *with the approach* (4.3)*. Top: $\rho_{\text{const}}, \rho^s, \rho^t$ (left to right). Bottom: $\omega_{\text{const}}, \omega^s, \omega^t$ (left to right). The structure and texture components reveal turbulent flow patterns at different scales, which are not easily visible in the flow u itself.*

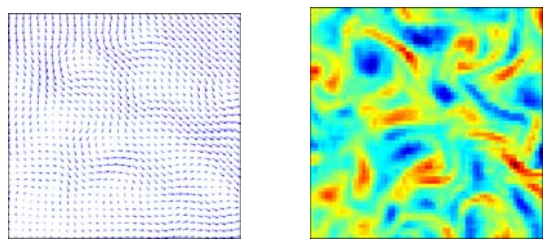


FIG. 7.6. *Groundtruth data u (left) and its curl (right) to be estimated from a corresponding artificially created image sequence. Here u is a superposition of a laminar flow and turbulent vortices.*

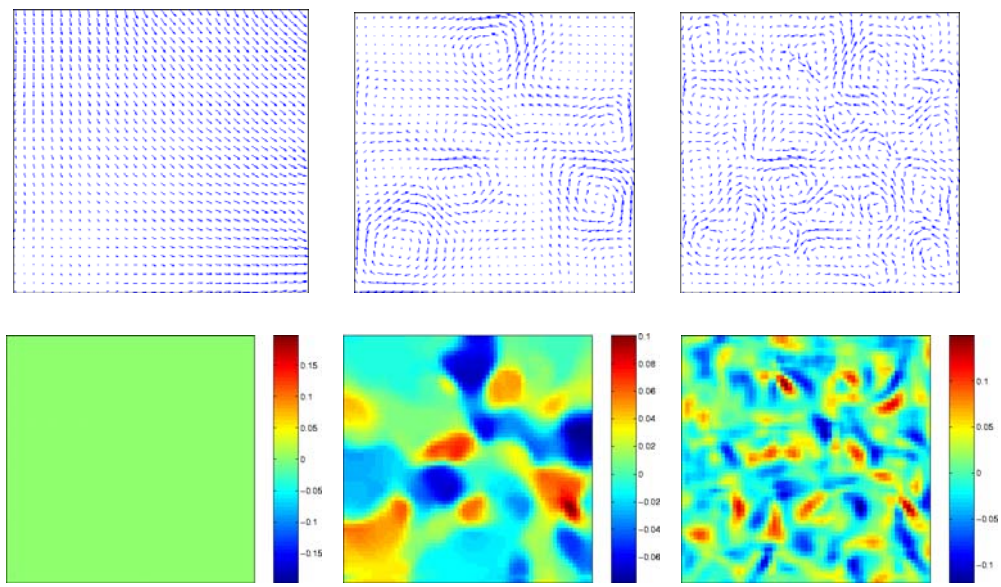


FIG. 7.7. *Estimated and decomposed flow corresponding to Figure* 7.6 *using the $TV - L_2$ approach* (5.2). *Top: u_{const}, u^s, and u^t (left to right). Bottom: ω_{const}, ω^s, and ω^t (left to right).*

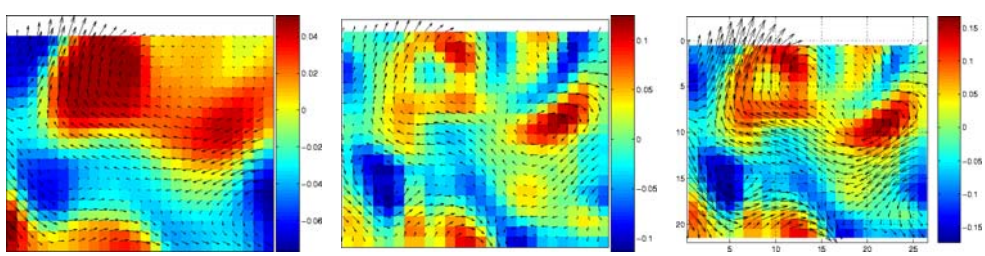


FIG. 7.8. *Close-up view of a section of Figure* 7.7. *From left to right: $\omega^s, \omega^t, \omega^s + \omega^t$ with the corresponding flows as overlays.*

Flow estimation and decomposition. In this section we will validate the flow estimation-decomposition model (5.2). First we create a divergence-free groundtruth flow field u by superimposing a dominant laminar flow with some turbulent vortex structures; see Figure 7.6. Using this flow, an artificial image sequence $\{g\}$ was created.

Figures 7.7 and 7.8 show the decomposition-based optical flow estimates, where

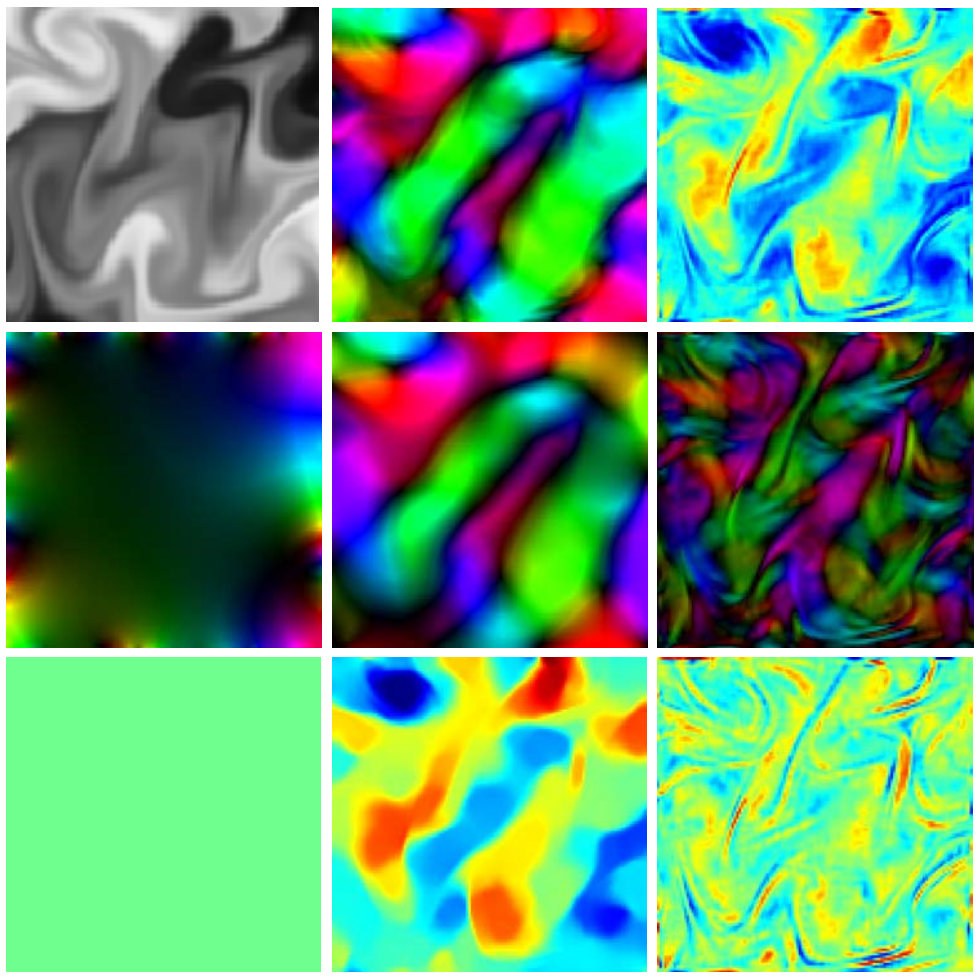


FIG. 7.9. *Top-left: frame of a real image sequence depicting the mixture of fluorescein and water* [18]. *Top-center: the turbulent solenoidal flow estimated by minimizing* (5.2). *Flow vectors are color-coded for better visibility (color* $\simeq$ *direction, magnitude* $\simeq$ *brightness). Top-right: the curl field comprising large-scale patterns immersed in turbulent oscillations. Middle, from left to right:* $u_{\text{const}}, u^s, u^t$. *Note that* u_{const} *contains the (nonvanishing) boundary values, and that* u^s *and* u^t *separate coherent motion patterns and turbulent fluctuations, respectively. Bottom, from left to right: the curl fields* ω_{const}, ω^s, *and* ω^t.

we have used $\lambda_c = 6e^{-5}$ and $\mu_c = 3e^{-4}$. The boundary parameter was chosen slightly smaller than λ_d. The u_{const} component nicely recovers the laminar flow, whereas the structural and textural components reveal the turbulent curl field. Finally, Figure 7.8 gives a close-up view of a section of Figure 7.7.

A real-world example. Figure 7.9 (top-left) shows a sample image of the experimental evaluation of the spreading of a low-diffusivity dye in a two-dimensional turbulent flow, forced at a large scale. The passive scalar is a mixture of fluorescein and water. The divergence of the corresponding flow vanishes. For more details about the experimental setup, we refer to [18].

Figure 7.9 shows the components of the flow and curl field estimated in terms of $\phi_{\text{const}}, \varphi^s, \varphi^t$ by minimizing (5.2) (parameter values: $\lambda_c = 1e^{-4}$, $\mu_c = 3e^{-4}$, $\gamma = 5e^{-4}$). This result clearly demonstrates how the convex constrained optimization

approach allows for estimating complex flows while simultaneously separating large-scale coherent motion patterns from turbulent fluctuations.

Appendix. Additional matrix notation. The extensions $\overline{\mathbb{G}}_1$, $\overline{\mathbb{G}}_2$ of $\overline{\mathbb{G}} = -(\mathbb{D}iv\,_{|H_S^o})^* = \left(\begin{smallmatrix} I_{n-1}\otimes D_{m-1} \\ D_{n-1}\otimes I_{m-1}\end{smallmatrix}\right)$ are given as follows. Let

$$D_m^1 := \begin{pmatrix} -2 & 2 & 0 & \dots & 0 & 0 & 0 \\ 0 & -1 & 1 & \dots & 0 & 0 & 0 \\ & & \ddots & \ddots & \ddots & & \\ 0 & 0 & 0 & \dots & -1 & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & -2 & 2 \end{pmatrix} \in \mathbb{R}^{m,m+1}$$

and

$$D_m^2 := \begin{pmatrix} -1 & 1 & 0 & \dots & 0 & 0 & 0 \\ -1 & 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & -1 & 1 & \dots & 0 & 0 & 0 \\ & & \ddots & \ddots & \ddots & & \\ 0 & 0 & 0 & \dots & -1 & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & -1 & 1 \\ 0 & 0 & 0 & \dots & 0 & -1 & 1 \end{pmatrix} \in \mathbb{R}^{m+1,m},$$

and let the diagonal matrices Z_1 and Z_2 be defined by

$$Z_1 := \left(0_{(n-1)m,m-1}, I_{(n-1)(m+1)}, 0_{(n-1)m,m-1}\right),$$

$$Z_2 := \begin{pmatrix} I_{m-1} & 0_{m-1,(n-1)(m+1)} & 0_{m-1,m-1} \\ 0_{(n-1)(m-1),m-1} & I_{n-1}\otimes(0_{m-1,1}, I_{m-1}, 0_{m-1,1}) & 0_{(n-1)(m-1),m-1} \\ 0_{m-1,m-1} & 0_{m-1,(n-1)(m+1)} & I_{m-1} \end{pmatrix},$$

where $0_{m,n}$ is the matrix consisting of $m \times n$ zeros. Then

$$\overline{\mathbb{G}}_1 := \begin{pmatrix} (I_{n-1}\otimes D_m^1)Z_1 \\ (D_n^1\otimes I_{m-1})Z_2 \end{pmatrix}, \quad \overline{\mathbb{G}}_2 := \begin{pmatrix} I_{n-1}\otimes D_{m-1}^2 \\ D_{n-1}^2\otimes I_{m-1} \end{pmatrix}.$$

The matrices S and $\overline{S}$ for the computation of the TV and G norm can be defined using

$$L_m(a) := \begin{pmatrix} \frac{1}{\sqrt{2}}a & 0_{1,m-1} \\ \frac{1}{\sqrt{2}}I_m \end{pmatrix}, \quad R_m(a) := \begin{pmatrix} \frac{1}{\sqrt{2}}I_m \\ 0_{1,m-1} & \frac{1}{\sqrt{2}}a \end{pmatrix}$$

as

$$S := \begin{pmatrix} I_{n-2}\otimes L_{m-3}(0) & 0 \\ I_{n-2}\otimes R_{m-3}(0) & 0 \\ 0 & L_{n-3}(0)\otimes I_{m-2} \\ 0 & R_{n-3}(0)\otimes I_{m-2} \end{pmatrix},$$

$$\overline{S} := \begin{pmatrix} I_{n-1}\otimes L_{m-2}(1) & 0 \\ I_{n-1}\otimes R_{m-2}(1) & 0 \\ 0 & L_{n-2}(1)\otimes I_{m-1} \\ 0 & R_{n-2}(1)\otimes I_{m-1} \end{pmatrix}.$$

REFERENCES

[1] L. Amodei and M. N. Benbourhim, *A vector spline approximation*, J. Approx. Theory, 67 (1991), pp. 51–79.

[2] G. Aubert, R. Deriche, and P. Kornprobst, *Computing optical flow via variational techniques*, SIAM J. Appl. Math., 60 (1999), pp. 156–182.

[3] J.-F. Aujol, G. Aubert, L. Blanc-Féraud, and A. Chambolle, *Image decomposition application to SAR images*, in Scale-Space Methods in Computer Vision, L. D. Griffin and M. Lilholm, eds. Lecture Notes in Comput. Sci. 2695, Springer, Berlin, 2003, pp. 297–312.

[4] J. F. Aujol and A. Chambolle, *Dual norms and image decomposition models*, Int. J. Computer Vision, 63 (2005), pp. 85–104.

[5] J. F. Aujol, G. Gilboa, T. Chan, and S. Osher *Structure-texture image decomposition—Modeling, algorithms, and parameter selection*, Internat. J. Computer Vision, 67 (2001), pp. 111–136.

[6] J. Bect, L. Blanc-Féraud, G. Aubert, and A. Chambolle, *A l_1-unified variational framework for image restoration*, in ECCV 2004, T. Pajdla and J. Matas, eds., Lecture Notes in Comput. Sci. 3024, Springer, Berlin-Heidelberg, 2004, pp. 1–13.

[7] A. Chambolle, *An algorithm for total variation minimization and applications*, J. Math. Imaging and Vision, 20 (2004), pp. 89–97.

[8] T. Chan, A. Marquina, and P. Mulet, *High-order total variation-based image restoration*, SIAM J. Sci. Comput., 22 (2000), pp. 503–516.

[9] T. Corpetti, E. Mémin, and P. Pérez, *Dense estimation of fluid flows*, IEEE Trans. Pattern Anal. and Machine Intell., 24 (2002), pp. 365–380.

[10] I. Daubechies and G. Teschke, *Variational image restoration by means of wavelets: Simultaneous decomposition, deblurring and denoising*, Appl. Comput. Harmonic Anal., 19 (2005), pp. 1–16.

[11] L. C. Evans and R. F. Gariepy, *Measure Theory and Fine Properties of Functions*, CRC Press, Boca Raton, FL, 1992.

[12] D. Goldfarb and W. Yin, *Second-order cone programming methods for total variation-based image restoration*, SIAM J. Sci. Comput., 27 (2005), pp. 622–645.

[13] F. Guichard and L. Rudin, *Accurate estimation of discontinuous optical flow by minimizing divergence related functionals*, in Proceedings of the IEEE International Conference on Image Processing (ICIP), Lausanne, Switzerland, 1996, volume I, pp. 497–500.

[14] S. Gupta and J. Prince, *Stochastic models for div-curl optical flow methods*, Signal Process. Lett., 3 (1996), pp. 32–34.

[15] W. Hinterberger, O. Scherzer, C. Schnörr, and J. Weickert, *Analysis of optical flow models in the framework of calculus of variations*, Numer. Funct. Anal. Optim., 23 (2002), pp. 69–89.

[16] B. Horn and B. Schunck, *Determining optical flow*, Artificial Intell., 17 (1981), pp. 185–203.

[17] J. M. Hyman and M. J. Shashkov, *Natural discretizations for the divergence, gradient, and curl on logically rectangular grids*, Comput. Math. Appl., 33 (1997), pp. 81–104.

[18] M. C. Jullien, P. Castiglione, and P. Tabeling, *Intermittency of a passive tracer in the inverse energy cascade*, Phys. Rev. E, 64 (2001), paper 035301.

[19] J. Lewalle, J. Delville, and J.-P. Bonnet, *Decomposition of mixed layer turbulence into coherent structures and background fluctations*, Appl. Sci. Res., 64 (2000), pp. 301–328.

[20] S. B. M. S. Lobo, L. Vandenberghe, and H. Lebret, *Applications of second order cone programming*, Linear Algebra Appl., 284 (1998), pp. 193–228.

[21] Y. Meyer, *Oscillating Patterns in Image Processing and Nonlinear Evolution Equations*, University Lecture Series 22, AMS, Providence, RI, 2001.

[22] H. Mittelmann, *An independent benchmarking of SDP and SOCP solvers*, Math. Program. Ser. B, 95 (2003), pp. 407–430.

[23] M. Nikolova, *Minimizers of cost-functions involving nonsmooth data-fidelity terms. Application to the processing of outliers*, SIAM J. Numer. Anal., 40 (2002), pp. 965–994.

[24] S. Osher, A. Sole, and L. A. Vese, *Image decomposition and restoration using total variation minimization and the H^{-1} norm*, Multiscale Model. Simul., 1 (2003), pp. 349–370.

[25] M. Raffel, C. Willert, and J. Kompenhans, *Particle Image Velocimetry*, 2nd ed., Springer, New York, 2001.

[26] L. I. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Phys. D, 60 (1992), pp. 259–268.

[27] *SeDuMi, Optimization software*, http://sedumi.mcmaster.ca/.

[28] J.-L. Starck, M. Elad, and D. L. Donoho, *Image decomposition via the combination of sparse representations and a variational approach*, IEEE Trans. Image Process., 14 (2005), pp. 1570–1582.

[29] G. Steidl, *A note on the dual treatment of higher order regularization functionals*, Computing, 76 (2005), pp. 135–148.
[30] D. Suter, *Motion estimation and vector splines*, in Proceedings of the Conference on Computer Vision and Pattern Recognition, Los Alamos, CA, 1994, pp. 939–942.
[31] X.-C. Tai and J. Xu, *Global and uniform convergence of subspace correction methods for some convex optimization problems*, Math. Comput., 71 (2002), pp. 105–124.
[32] L. Vese and S. Osher, *Modeling textures with total variation minimization and oscillating patterns in image processing*, J. Sci. Comput., 19 (2003), pp. 553–572.
[33] J. Xu, *Iterative methods by space decomposition and subspace correction*, SIAM Rev., 34 (1992), pp. 581–613.
[34] W. Yin, D. Goldfarb, and S. Osher, *Total Variation Based Image Cartoon-Texture Decomposition*, Technical Report CORC TR-2005-01, Columbia University, New York, 2005.
[35] J. Yuan, R. Ruhnau, E. Mémin, and C. Schnörr, *Discrete orthogonal decomposition and variational fluid flow estimation*, in Scale Space and PDE Methods in Computer Vision, Lecture Notes in Comput. Sci. 3459, Springer, New York, pp. 267–278.
[36] J. Yuan, C. Schnörr, G. Steidl, and F. Becker, *A study of non-smooth convex flow decomposition*, in Proceedings of the Variational, Geometric and Level Set Methods in Computer Vision, Lecture Notes in Comput. Sci. 3752, Springer, New York, 2005, pp. 1–12.
[37] J. Yuan, C. Schnörr, and E. Mémin, *Discrete orthogonal decomposition and variational fluid flow estimation*, J. Math. Imag. Vision, to appear.

SIAM J. SCI. COMPUT.
Vol. 29, No. 6, pp. 2305–2328
© 2007 Society for Industrial and Applied Mathematics

A SPARSE DISCRETIZATION FOR INTEGRAL EQUATION FORMULATIONS OF HIGH FREQUENCY SCATTERING PROBLEMS *

DAAN HUYBRECHS† AND STEFAN VANDEWALLE†

Abstract. We consider two-dimensional scattering problems, formulated as an integral equation defined on the boundary of the scattering obstacle. The oscillatory nature of high-frequency scattering problems necessitates a large number of unknowns in classical boundary element methods. In addition, the corresponding discretization matrix of the integral equation is dense. We formulate a boundary element method with basis functions that incorporate the asymptotic behavior of the solution at high frequencies. The method exhibits the effectiveness of asymptotic methods at high frequencies with only few unknowns, but retains accuracy for lower frequencies. New in our approach is that we combine this hybrid method with very effective quadrature rules for oscillatory integrals. As a result, we obtain a sparse discretization matrix for the oscillatory problem. Moreover, numerical experiments indicate that the accuracy of the solution actually increases with increasing frequency. The sparse discretization applies to problems where the phase of the solution can be predicted a priori, for example in the case of smooth and convex scatterers.

Key words. integral equations, high-frequency scattering, oscillatory integrals, steepest descent

AMS subject classifications. 45B05, 65D30, 41A60

DOI. 10.1137/060651525

1. Introduction.

The accurate numerical modeling of physical problems involving strong oscillations is a challenging problem. Scattering problems in unbounded domains are often modeled by an integral equation defined on the boundary of the scattering obstacle. As such, the problem on an unbounded domain is reduced to a lower-dimensional problem on a bounded domain. Numerically, these are two important advantages. Still, the method has its drawbacks, and difficulties arise as the frequency of the problem increases. Contrary to the case for partial differential equations, the discretization matrix of an integral equation is dense. Furthermore, in order to represent an oscillatory solution, the number of unknowns in a boundary element approach has to be large. Typically, one chooses a fixed number of unknowns per wavelength per dimension. This results in a linear system with a very large and dense discretization matrix. Hence, classical solution methods for scattering problems rapidly become prohibitively expensive [4].

Substantial efforts have been made over the last two decades to overcome these difficulties. One direction has been to improve on the solution time required to solve the dense linear system. The fast multipole method achieves a fast matrix-vector product in $O(N \log N)$ operations, when the number of unknowns N increases at least linearly with the wavenumber [30]. The fast matrix-vector product can be combined with a preconditioned iterative technique for an efficient overall solution algorithm [10]. Another direction is given by a number of asymptotic methods, such as geometrical optics, physical optics, and the geometrical theory of diffraction [17, 23]. A common

*Received by the editors February 4, 2006; accepted for publication (in revised form) March 15, 2007; published electronically October 5, 2007.
http://www.siam.org/journals/sisc/29-6/65152.html
†Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, B-3001 Leuven, Belgium (daan.huybrechs@cs.kuleuven.be, stefan.vandewalle@cs.kuleuven.be). The first author was supported by the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

characteristic of asymptotic methods is that they have an error of the order $O(k^{-n})$, where k is the wavenumber and where the exponent n is typically equal to 1 or 2. This means that the accuracy of such methods improves with increasing frequencies. Asymptotic methods break down for low to moderate frequencies, however.

A more recent trend is the combination of finite element methods with asymptotic methods. This is achieved by considering basis functions that are, e.g., piecewise polynomial, multiplied by the asymptotic form of the solution at large frequencies. The asymptotic behavior of the solution to the problem of scattering by smooth convex obstacles was analyzed in [27]. Motivated by these results, a hybrid scheme was considered in [1]. The authors report an overall solution method that requires $O(k^{1/3})$ operations as a function of the wavenumber, a huge improvement over the linear dependence on k. The basis functions are piecewise polynomials, multiplied by plane waves in a number of directions. Similar hybrid methods with even better results are proposed in [2, 7, 25, 24, 15, 12]. A number of operations that is independent of the wavenumber, for a fixed error, is achieved by Bruno et al. in [7] for the scattering by smooth convex obstacles, and by Langdon and Chandler-Wilde in [25] for scattering on a half-plane. In the present paper, we combine a similar approach with recent insights into the behavior of oscillatory integrals [21, 19]. As a result, we obtain a small and highly sparse discretization matrix. In addition, the accuracy of the solution actually increases with increasing frequency.

We start the paper in section 2 with a brief review of a suitable quadrature rule for the evaluation of oscillatory integrals that have the general form

$$I[f] := \int_a^b f(x) e^{ikg(x)} \, \mathrm{d}x, \tag{1.1}$$

where both f and g are smooth functions and where the wavenumber k determines the frequency of the oscillations. The review is based on results by Iserles and Nørsett in [21] and by the authors in [19]. We show how Filon-type quadrature rules using derivatives can be constructed that have the form

$$I[f] \approx Q[f] := \sum_{l=0}^{n} \sum_{j=1}^{d_l} w_{l,j} f^{(j)}(x_l) \tag{1.2}$$

such that the accuracy of the rule improves with increasing frequency. This is quite contrary to the rapid deterioration of classical quadrature rules, based on polynomial interpolation, for increasingly oscillatory integrals. The asymptotic order of accuracy as a function of the frequency is $O(k^{-s-1})$, where the value of s depends on the number of derivatives used. This is much like the behavior of truncated asymptotic expansions. However, due to the nature of the constructed rules, the result is exact for an arbitrarily large family of functions regardless of the frequency. The main difference with asymptotic expansions therefore is that there is no breakdown at low frequencies. In general, one can also expect a smaller error for moderate frequencies.

The results of [19] are generalized in section 3 to an appropriately chosen model form

$$I_H[f] := \int_a^b f(x) H_\nu^{(1)}(kg_1(x)) e^{ikg_2(x)} \, \mathrm{d}x,$$

where f, g_1, and g_2 are assumed to be smooth functions. The function $H_\nu^{(1)}(z)$ is the Hankel function of the first kind and order ν; it is an oscillatory function for large

arguments. The model form is chosen to represent the integrals that will appear later in the solution of scattering problems. The resulting quadrature rule has the same form as (1.2). The scattering problem is introduced in section 4. The quadrature rules are subsequently used in the discretization of the oscillatory integral equation in section 5. We identify a setting in which each row of the discretization matrix corresponds to the discretization of a specific one-dimensional oscillatory integral with a known phase. Owing to the small number of quadrature points required for the evaluation of such integrals, the discretization matrix is sparse. The accuracy of the solution improves because the quadrature rules themselves improve for increasing frequencies.

We illustrate the method with numerical results in section 6. We consider the scattering of a plane wave and of a circular wave, by a circle and by an ellipse, respectively. We end the paper with some concluding remarks in section 7.

2. The efficient evaluation of oscillatory integrals. The classical approach for the evaluation of oscillatory integrals is to use a fixed number of quadrature points per wavelength. This automatically leads to a number of operations that scales linearly with the frequency. However, the asymptotic expansion of an oscillatory integral for large frequencies reveals that the value of the integral is actually determined by the behavior of the integrand near a small set of special points [31, 5]. These are the boundary points of the interval, and the so-called *stationary points*.

Consider the oscillatory integral (1.1), where both f and g are smooth functions. We call f the *amplitude*, and g the *oscillator*, of the integral. The stationary points of (1.1) are all solutions ξ to the equation $g'(\xi) = 0$, $\xi \in [a, b]$. A stationary point is said to have *order* r if $g^{(j)}(\xi) = 0$, $j = 1, \ldots, r$, but $g^{(r+1)}(\xi) \neq 0$. The importance of such points lies in the fact that, locally, the integrand does not oscillate near a stationary point. Away from all stationary points and the boundary points a and b, the oscillations of the integrand increasingly cancel out. Hence, such regions do not contribute much to the value of the integral.

Mathematically, this property is reflected in the asymptotic expansion of (1.1). It can be shown that integral (1.1) with one interior stationary point of order r has an asymptotic expansion of the form (see [31])

$$I[f] \sim \sum_{j=0}^{\infty} \frac{c_j[f]}{k^{(j+1)/(r+1)}}, \qquad k \to \infty. \tag{2.1}$$

It is proved in [21] that the first few coefficients $c_j[f]$ of the asymptotic expansion depend only on the first few derivatives of f and g, evaluated at the boundary points and at the stationary points. A number of recent methods exploit this behavior in order to obtain an approximation of $I[f]$ that improves with increasing k [21, 19, 26, 29]. We recall one particularly useful approach, which leads to a quadrature rule with a classical form. We refer the reader to [20] for a more general overview.

2.1. A numerical steepest descent method. A well-known technique for obtaining the asymptotic expansion (2.1) is the method of steepest descent [5, 33]. Justified by Cauchy's integral theorem for analytic functions, the integration path is deformed into a complex integration path that has more desirable properties. Specifically, the new path is chosen such that the integrand is not oscillatory along the path and has exponential decay. The asymptotic expansion can then be derived from the resulting line integrals in the complex plane. Asymptotic expansions have the disadvantage, however, that the error is not controllable for low values of k. Rather than

forming the asymptotic expansion, the line integrals can also be evaluated numerically. This numerical approach leads to a method that improves with increasing k, but that retains accuracy for small values of k [19].

Assume that both f and g are analytic functions. Subdivide the integration interval $[a,b]$ into subintervals $[a_l,b_l]$, $l=1,\dots,L$, such that $g'(x)\neq 0$, $x\in(a_l,b_l)$, i.e., the oscillator g is a monotonic function on $[a_l,b_l]$. Then the inverse of g exists uniquely on $[a_l,b_l]$, and it is also analytic. We denote it by g_l^{-1}. The integral over $[a_l,b_l]$ is evaluated by deforming the integration path into the complex plane. Define the path $h_{x,l}(p)$ by

$$h_{x,l}(p)=g_l^{-1}(g(x)+ip),\qquad x\in[a_l,b_l]. \tag{2.2}$$

This path is called the *path of steepest descent*. It follows immediately from the definition that $e^{ikg(h_{x,l}(p))}=e^{-kp}e^{ikg(x)}$. Hence, the line integral along the new integration path, originating in the point x and terminating in the point $h_{x,l}(P)$, can be written as

$$S_l[f;x]=e^{ikg(x)}\int_0^P f(h_{x,l}(p))h'_{x,l}(p)e^{-kp}\,\mathrm{d}p,\qquad P>0. \tag{2.3}$$

The integrand of (2.3) is not oscillatory and decays exponentially fast. The integral over $[a_l,b_l]$ can now be approximated by $S_l[f;a_l]-S_l[f;b_l]$. In general, the value of $I[f]$ can be approximated by a sum of nonoscillatory line integrals with exponentially small error,

$$I[f]=\sum_{l=1}^{L}\left(S_l[f;a_l]-S_l[f;b_l]\right)+O(e^{-kP}), \tag{2.4}$$

with $P>0$. The larger P, the better the approximation, but for all finite P the error decays exponentially fast as the frequency parameter k increases. In many cases of practical interest, the limit case $P\to\infty$ is possible, and the error $O(e^{-kP})$ vanishes. In that case, decomposition (2.4) is exact.

2.2. A quadrature rule for oscillatory integrals. The decomposition (2.4) can be used to obtain a quadrature rule for $I[f]$ with a classical form. Approximating f by its truncated Taylor series at each point a_l and b_l, we find a quadrature rule using derivatives of the form

$$Q[f]:=\sum_{l=0}^{L}\sum_{j=0}^{d_l}w_{l,j}f^{(j)}(x_l), \tag{2.5}$$

with $x_0=a$, $x_l=b_l$, $l=1,\dots,L$, and with the weights given by

$$\begin{aligned}
w_{0,j}&=S_1\left[\frac{(x-a)^j}{j!};a\right],\\
w_{l,j}&=-S_l\left[\frac{(x-x_l)^j}{j!};x_l\right]+S_{l+1}\left[\frac{(x-x_l)^j}{j!};x_l\right],\quad l=1,\dots,L-1,\\
w_{l,j}&=-S_l\left[\frac{(x-b)^j}{j!};b\right].
\end{aligned}$$

Note that this rule uses only local information of f: its values and derivatives at the endpoints a and b of the integration region and at all the stationary points in

between. Yet the approximation $I[f] \approx Q[f]$ is very accurate and becomes increasingly accurate with increasing values of the frequency parameter k. It can be shown that the first few terms of the asymptotic expansion of the error $I[f] - Q[f]$ vanish. This is due to the fact that these terms depend only on the derivatives of f at exactly the points we consider in the quadrature rule. Hence, the approximation has a high asymptotic accuracy. The exact order depends on the number of derivatives d_l that are used and on the order of the stationary points involved [19].

3. Specialized quadrature rules. The quadrature rule of the previous section applies only to the model integral (1.1). Intuitively, however, one sees that the ideas can be readily generalized to any oscillatory integral. The value of an oscillatory integral is determined by the behavior of the integrand near the endpoints of the integration interval, and near the points where the integrand locally does not oscillate. In order to construct similar quadrature rules, one requires knowledge of the phase of the integral. In this section, we will construct such rules for a family of integrals that will arise in the scattering problem discussed later. In particular, the integrand involves an oscillatory Hankel function.

3.1. A generalized model form. Consider the oscillatory integral

$$I_H[f] = \int_a^b f(x) H_\nu^{(1)}(k g_1(x)) e^{ikg_2(x)} \, \mathrm{d}x, \tag{3.1}$$

where f, g_1, and g_2 are smooth functions and $H_\nu^{(1)}(z)$ is the Hankel function of the first kind of order ν. The Hankel function of order zero, $H_0^{(1)}(z)$, has a logarithmic singularity at $z = 0$. Hankel functions of higher order have algebraic singularities of the form $1/z^\nu$, $z \to 0$ [3].

For large arguments, the Hankel functions behave like an oscillatory complex exponential with a decaying amplitude,

$$H_\nu^{(1)}(z) \sim \sqrt{\frac{2}{\pi z}} e^{i(z - \frac{1}{2}\nu\pi - 1/4\pi)}, \qquad -\pi < \arg z < \pi, \quad |z| \to \infty. \tag{3.2}$$

Hence, the oscillator of the integrand of (3.1) is approximately given by

$$g(x) = g_1(x) + g_2(x), \tag{3.3}$$

up to the addition of a constant. The Hankel function decays exponentially fast for complex arguments with a positive imaginary part, as can be seen from the asymptotic behavior (3.2). This means that the approach of section 2.2 using the path of steepest descent is applicable. Hence, we conjecture that a quadrature rule exists of the form

$$I_H[f] \approx Q_H[f] := \sum_{l=0}^{L} \sum_{j=0}^{d_l} w_{l,j}^H f^{(j)}(x_l). \tag{3.4}$$

In the remainder of the section, we will prove this conjecture, determine the quadrature abscissae x_l, and show how the weights can be computed efficiently.

3.2. Construction of the quadrature rule. We start by stating some assumptions on the functions f and g, which are needed to guarantee the integrability and analyticity of the integrand in (3.1). First, we assume that f is analytic in an open complex neighborhood D of $[a, b]$, so that $[a, b] \subset \operatorname{int} D$. Likewise, we assume

that g_1 and g_2 are nonsingular and analytic in D, except possibly along a branch cut that extends from a or b to the boundary of the region D; i.e., a and b may be branch points but not singular points. We assume furthermore that $g(x)$, defined by (3.3), is strictly monotonic on the open interval (a, b) and hence invertible, but possibly $g'(a) = 0$ or $g'(b) = 0$. Also, we assume that $g_1(x) \neq 0$, $x \in (a, b)$. Finally, if $\nu > 0$ and $g_1(\xi) = 0$, we assume that f behaves like

$$f(x) \sim (x - \xi)^{\nu - 1 + \epsilon}, \quad x \to \xi, \quad \text{with} \quad \epsilon > 0. \tag{3.5}$$

Condition (3.5) guarantees that the integrand of $I_H[f]$ is integrable. Subject only to condition (3.5) and the analyticity requirements, the integration interval of (3.1) can always be split into a number of subintervals that satisfy the conditions. The assumptions guarantee that the integrand of $I_H[f]$ is analytic on $[a, b]$ except possibly in the points a and b. In particular, this will allow us to apply Cauchy's integral theorem to select the integration path of (3.1).

THEOREM 3.1. *Under the assumptions stated above, the integral $I_H[f]$ can be approximated by a sum of contributions*

$$I_H[f] = S^H[f; a] - S^H[f; b] + O(e^{-kP}), \tag{3.6}$$

with $P > 0$, and with the contributions given by the integrals

$$S^H[f; x] = \int_0^P f(h_x(p)) H_\nu^{(1)}(k g_1(h_x(p))) e^{ikg_2(h_x(p))} h_x'(p) \, \mathrm{d}p, \tag{3.7}$$

where $h_x(p)$ satisfies

$$g(h_x(p)) = g(x) + ip. \tag{3.8}$$

The proof is almost identical to the proof of Lemma 4.1 in [18] and is omitted; it differs mainly in the special treatment of the Hankel function based on the asymptotic expression (3.2).

We note from the asymptotic behavior (3.2) that the integrand of the line integral $S^H[f; x]$ in (3.7) is nonoscillatory and exponentially decaying in the integration variable p,

$$H_\nu^{(1)}(k g_1(h_x(p))) e^{ikg_2(h_x(p))} \sim \sqrt{\frac{2}{\pi k g_1(h_x(p))}} e^{ikg(x)} e^{i(-\frac{1}{2}\nu\pi - 1/4\pi)} e^{-kp}, \qquad k \to \infty.$$

The size of the constant P is related to the size of the region of analyticity of f and g [18]. In the numerical examples of the scattering problem, given in section 6, we can in fact consider the limit case $P = \infty$. The error of decomposition (3.6) then vanishes even at low frequencies.

We proceed in a way similar to that in section 2.2. Since f is analytic in D, it has an absolutely convergent Taylor series. By the linearity of S^H, we may write

$$S^H[f; x_0] = \sum_{j=0}^{\infty} f^{(j)}(x_0) S^H \left[\frac{(x - x_0)^j}{j!}; x_0 \right].$$

Now, consider a subdivision of $[a, b]$ into subintervals $[a_l, b_l]$, $l = 1, \ldots, L$, such that on each subinterval the conditions of Theorem 3.1 are satisfied. Truncating the Taylor

series of f at each special point a_l and b_l after a finite number of terms, we arrive at a quadrature rule $Q_H[f]$ of the form (3.4), with weights given by

$$w_{0,j}^H = S_1^H \left[\frac{(x-a)^j}{j!}; a \right], \tag{3.9}$$

$$w_{l,j}^H = -S_l^H \left[\frac{(x-x_l)^j}{j!}; x_l \right] + S_{l+1}^H \left[\frac{(x-x_l)^j}{j!}; x_l \right], \quad l = 1, \ldots, L-1, \tag{3.10}$$

$$w_{l,j}^H = -S_l^H \left[\frac{(x-b)^j}{j!}; b \right]. \tag{3.11}$$

The weights can be explicitly computed very efficiently, by using Gauss–Laguerre quadrature or similar techniques [11]. The accuracy of these methods improves rapidly as a function of k, due to the faster decay of the integrands as k increases. For the purposes of our application, this advantageous characteristic is not needed. It suffices already that the number of operations for a fixed accuracy is bounded with respect to k. We therefore choose to focus on the convergence properties of the quadrature rule itself, rather than on the convergence of methods to compute the weights.

3.3. Convergence properties of the quadrature rule. We discuss the properties of the quadrature rule $Q_H[f]$, with weights given by (3.9)–(3.11), as a function of k. The rule is exact by construction for polynomials of degree less than or equal to

$$p = \min_l d_l. \tag{3.12}$$

For more general functions, the accuracy as a function of k is determined by the asymptotic size of the weights. We will show that the size of the weights decreases both with increasing frequency and with increasing order of the corresponding derivative. The order of accuracy of the quadrature rule is therefore equal to the asymptotic size of the first weight that is discarded by truncation. In order to quantify this size, we require a few technical lemmas.

LEMMA 3.2. *Assume that x_0 is a stationary point that has order r. The parameterization of the path* (3.8) *behaves as*

$$h_{x_0}(p) = x_0 + O(p^{1/(r+1)}), \qquad p \to 0, \tag{3.13}$$

$$h'_{x_0}(p) = O(p^{1/(r+1)-1}), \qquad p \to 0. \tag{3.14}$$

Proof. Since $g^{(j)}(x_0) = 0$, $j = 1, \ldots, r$, we can write the Taylor series of g as

$$g(x) = g(x_0) + g^{(r+1)}(x_0) \frac{(x-x_0)^{r+1}}{(r+1)!} + O((x-x_0)^{r+2}).$$

The path $h_{x_0}(p) = g^{-1}(g(x_0) + ip)$ solves $g(h_{x_0}(p)) = g(x_0) + ip$, and hence

$$h_{x_0}(p) \sim x_0 + \sqrt[r+1]{\frac{ip(r+1)!}{g^{(r+1)}(x_0)}}, \qquad p \to 0. \tag{3.15}$$

The second result follows by differentiation. Note that the complex root is multivalued: the correct root is selected by using the analytic continuation of the inverse g_i^{-1} that satisfies $g_i^{-1}(g(x)) = x$ on $[a_i, b_i]$ in expression (3.8). ∎

The size of the weights follows from the size of the line integrals $S^H[\frac{(x-x_0)^j}{j!}; x_0]$. Recall that the integral may be singular if $g_1(x) = 0$. We will assume for the sake

of brevity that, in that case, $g_1'(x) \neq 0$. This condition is always satisfied by the applications in section 6.

LEMMA 3.3. *Let $S^H, [f;x]$ be defined by* (3.7) *with $P = \infty$, and g defined by* (3.3). *Assume that $g'(x_0) \neq 0$; i.e., x_0 is not a stationary point. If $g_1(x_0) \neq 0$, we have*

$$\left|S^H[(x-x_0)^j; x_0]\right| = O(k^{-j-3/2}), \qquad k \to \infty.$$

If $g_1(x_0) = 0$ and $g_1'(x_0) \neq 0$, the integral is singular and we have

$$\left|S^H[(x-x_0)^j; x_0]\right| = O(k^{-j-1}), \qquad j \geq \nu, \quad k \to \infty.$$

Proof. We write the integral $S^H[(x-x_0)^j; x_0]$ as

$$S^H[(x-x_0)^j; x_0] = \int_0^\infty u(p) e^{-kp}\, \mathrm{d}p = \frac{1}{k} \int_0^\infty u(q/k) e^{-q}\, \mathrm{d}q, \tag{3.16}$$

with

$$u(p) = (h_{x_0}(p) - x_0)^j h_{x_0}'(p) H_\nu^{(1)}(k g_1(h_{x_0}(p))) e^{ikg_2(h_{x_0}(p))} e^{kp}. \tag{3.17}$$

It is a consequence of Watson's lemma that the asymptotic expansion of the integral can be obtained by integrating the asymptotic expansion of $\frac{1}{k}u(q/k)$ as $k \to \infty$, term by term in (3.16) [5, 32]. Generalizing Watson's lemma, this remains true for integrals of the form $\int_0^\infty u(p)h(kp)dp$, where $h(z) \sim \log(z)^n z^s e^{-z}$, $n \geq 0$, $s \in \mathbb{Z}$, $z \to 0$, if the integrand is integrable [6]. This means that the singularity of the Hankel function has no influence on the asymptotic expansion.

First, consider the case $g_1(x_0) \neq 0$. Then, combining the asymptotic behavior of the Hankel function for large arguments, (3.2), with the results (3.13)–(3.14) of Lemma 3.2 for $r = 0$, we have $u(q/k) \sim k^{-j-1/2}$. From (3.16) we can conclude $|S^H[(x-x_0)^j; x_0]| = O(k^{-j-3/2})$.

Next, consider the case $g_1(x_0) = 0$. If $g_1'(x_0) \neq 0$, then we have $g_1(h_{x_0}(p)) \sim p^{1/(r+1)} = p$. It follows that $H_\nu^{(1)}(k g_1(h_{x_0}(q/k))) = O(1)$, $k \to \infty$. Hence, by the generalization of Watson's lemma, we may conclude $|S^H[(x-x_0)^j; x_0]| = O(k^{-j-1})$, $j \geq \nu$. ∎

The corresponding lemma for stationary points is very similar. The difference is due to the different behavior of the parameterization as described by Lemma 3.2.

LEMMA 3.4. *Let $S^H[f;x]$ be defined by* (3.7) *with $P = \infty$, and g defined by* (3.3). *Assume that x_0 is a stationary point of order r. If $g_1(x_0) \neq 0$, then we have*

$$x\left|S^H[(x-x_0)^j; x_0]\right| = O(k^{-(j+1)/(r+1)-1/2}), \qquad k \to \infty.$$

If $g_1(x_0) = 0$ and $g_1'(x_0) \neq 0$, then we have

$$\left|S^H[(x-x_0)^j; x_0]\right| = O(k^{-(j+1+r/2)/(r+1)}), \qquad j \geq \nu, \quad k \to \infty.$$

Proof. Consider again the function $u(p)$, given by (3.17). Assume first that $g_1(x_0) \neq 0$. We have $(h_{x_0}(q/k) - x_0)^j \sim k^{-j/(r+1)}$ and $h_{x_0}'(q/k) \sim k^{r/(r+1)}$. Since $k g_1(h_{x_0}(q/k)) \sim k g_1(h_{x_0}(0)) = k g_1(x_0)$, we also have $H_\nu^{(1)}(k g_1(h_{x_0}(q/k))) \sim k^{-1/2}$. Combined in (3.16), and by applying Watson's lemma, this yields the first result.

The case where $g_1(x_0) = 0$ is slightly different. Since $g_1'(x_0) \neq 0$, we have $g_1(h_{x_0}(q/k)) \sim k^{-1/(r+1)}$ and hence $k g_1(h_{x_0}(q/k)) \sim k^{r/(r+1)}$. The Hankel function therefore yields the factor $k^{-(r/2)/(r+1)}$ instead of $k^{-1/2}$ as in the first case. ∎

TABLE 3.1
Absolute error of the approximation of $I_H[f]$ by $Q_H[f]$, with $f(x) = \cos(x-1)$, $g_1(x) = x$, and $g_2(x) = x^2 + x^3 - x$. The last row shows the value of $\log_2(e_{400}/e_{800})$: this value should approximate $d_0/2 + 5/4$ (shown in parentheses).

$k \setminus d_0$	0	1	2	3
100	$1.2E-3$	$2.8E-5$	$1.3E-6$	$2.6E-8$
200	$5.1E-4$	$8.6E-6$	$2.9E-7$	$4.1E-9$
400	$2.2E-4$	$2.6E-6$	$6.4E-8$	$6.2E-10$
800	$9.3E-5$	$7.8.1E-7$	$1.4E-8$	$9.7E-11$
rate	1.23 (1.25)	1.73 (1.75)	2.20 (2.25)	2.68 (2.75)

The convergence of the quadrature rule (3.4) as a function of k can now be established. Note that the results of Lemma 3.3 agree with those of Lemma 3.4 if we take the order of a regular point to be $r = 0$. Hence, we need not distinguish between stationary points and regular (end)points.

LEMMA 3.5. *The error of the approximation of $S_l^H[f; x_0]$, $x_0 \in [a_l, b_l]$, is*

(3.18)
$$\left|S_l^H[f;x_0] - Q_l^S[f;x_0]\right| := \left|S_l^H[f;x_0] - \sum_{j=0}^{d_l} w_{l,j}^H f^{(j)}(x_0)\right| = O(k^{-\alpha_l}), \qquad k \to \infty.$$

If $g_1(x_0) \neq 0$, then $\alpha_l := (d_l + 2)/(r+1) - 1/2$. If $g_1(x_0) = 0$ and $g_1'(x_0) \neq 0$, then $\alpha_l := (d_l + 2 + r/2)/(r+1)$.

Proof. Since the weights decay as a function of k, and as a function of the order of derivative j, the error of the quadrature scheme is asymptotically determined by the size of the first discarded weight. The result follows from Lemmas 3.3 and 3.4 by setting $j = d_l + 1$. ∎

The theorem that characterizes the accuracy of the complete quadrature rule follows immediately.

THEOREM 3.6. *Consider the approximation of $I_H[f]$ by $Q_H[f]$. The error has asymptotic order $\alpha = \min_l \alpha_l$, where α_l is specified in Lemma 3.5.*

As an example, we consider the integral $\int_0^1 \cos(x-1) H_0^{(1)}(kx) e^{ik(x^2+x^3-x)}\,\mathrm{d}x$. The total oscillator for this integral is $g(x) = x^2 + x^3$. There are two quadrature points: there is a singularity and a stationary point of order 1 at $x = 0$, and a regular endpoint at $x = 1$. The weights $w_{0,j}^H$ and $w_{1,j}^H$ are given by (3.9) and (3.11), respectively. From Lemma 3.4 we have $|w_{0,j}^H| = O(k^{-(j+1)/2-1/4})$, and from Lemma 3.3 we have $|w_{1,j}^H| = O(k^{-j-3/2})$. Using d_0 and d_1 derivatives, the error has order $\min\{O(k^{-(d_0+2)/2-1/4}), O(k^{-(d_1+1)-3/2})\}$ by Theorem 3.6. We choose $d_1 = \max\{0, \lceil (2d_0 - 5)/4 \rceil\}$ to match the errors. Table 3.1 shows the convergence of the quadrature rule $Q_H[f]$ as a function of k and d_0.

4. High-frequency scattering problems.

4.1. Problem statement. We are interested in the solution of the two-dimensional Helmholtz equation

$$\Delta u + k^2 u = 0 \tag{4.1}$$

in the domain $\Omega^+ = \mathbb{R}^2 \setminus \overline{\Omega}$ that is exterior to the open domain Ω, subject to a Dirichlet boundary condition $u = 0$ on the boundary $\Gamma = \partial\Omega$. We assume that a periodic parameterization for Γ is given by

$$\kappa(\tau) : [0,1] \to \Gamma \subset \mathbb{R}^2, \tag{4.2}$$

with $|\nabla\kappa(\tau)| > 0$. The Helmholtz equation models the scattering of time-harmonic waves of the form $u(x)e^{-i\omega t}$. In the absence of damping we have $k = \omega/c$, with c the speed of propagation. For the scattering of an incoming acoustic or electromagnetic wave $u^i(x)$ by the obstacle Ω, the solution can be written as $u = u^i + u^s$. The function $u^s(x)$ represents the scattered wave; it satisfies the Helmholtz equation with the Dirichlet boundary condition $u^s(x) = -u^i(x)$ on Γ.

The scattered wave can be represented in terms of the *single-layer potential*,

$$(Sq)(x) = \int_\Gamma K(x,y)q(y)\,\mathrm{d}s_y, \qquad \text{with} \quad K(x,y) = \frac{i}{4}H_0^{(1)}(k|x-y|), \tag{4.3}$$

where $q(x)$ is the *density function* defined on the boundary Γ, and where $K(x,y)$ is the Green's function of the two-dimensional Helmholtz equation. The scattered wave due to an incoming wave u^i is given by $u^s = -Sq$ if the density function $q(x)$ is found as the solution to the combined potential integral equation

$$\frac{q(x)}{2} + \int_\Gamma \left(\frac{\partial K}{\partial n_x}(x,y) + i\eta K(x,y)\right) q(y)\,\mathrm{d}s_y = \frac{\partial u^i}{\partial n}(x) + i\eta u^i(x), \quad x \in \Gamma, \tag{4.4}$$

where n_x is used to denote the normal with respect to the variable x. Equation (4.4) is uniquely solvable for all values of the wavenumber k [8].

An important observation is that $q(x) = \frac{\partial u}{\partial n}(x)$; i.e., the density function is exactly the (exterior) normal derivative of the solution to the Helmholtz problem. This means that the solution to (4.4) is directly related to a physical property of the problem. For example, in electromagnetics, the normal derivative of the electric field is proportional to the induced current on the surface of the conducting obstacle [28].

4.2. High-frequency integral equation formulation. The density function $q(x)$ is highly oscillatory for large values of the wavenumber k. The solution of (4.4) therefore generally requires a large number of unknowns. In some cases, however, one has a priori information about the phase of the solution. For example, if the obstacle is convex, and if the incoming wave is a plane wave, then the phase of the solution q is approximately the same as the phase of the incoming wave. Assume that the incoming wave is given by $u^i(x) = u_s^i(x)e^{ikg^i(x)}$. Then we can write

$$q(\kappa(t)) = q_s(t)e^{ikg^i(\kappa(t))}, \qquad t \in [0,1], \tag{4.5}$$

where $q_s(t)$ is a function that is less oscillatory than $q(\kappa(t))$, in a sense that will be made more precise in section 4.3. In physical terms, the oscillations of the induced current on a perfectly conducting surface tend to follow the oscillations of the incoming electromagnetic wave. This is the main reason why the problem is formulated such that the solution $q(x)$ corresponds to a physical variable—only in that case is the phase known in the form of (4.5). This was noted in [7]; the integral equation formulation of this section follows the same pattern as in [7].

Substituting the ansatz (4.5) into (4.4) leads to the integral equation of the second kind

$$\frac{1}{2}q_s(t)e^{ikg^i(\kappa(t))} + \int_0^1 G(t,\tau)q_s(\kappa(\tau))e^{ikg^i(\kappa(\tau))}\,\mathrm{d}\tau = v^i(\kappa(t)), \tag{4.6}$$

with

$$G(t,\tau) := \left(\frac{\partial K}{\partial n_x}(\kappa(t),\kappa(\tau)) + i\eta K(\kappa(t),\kappa(\tau))\right)|\nabla\kappa(\tau)|,$$

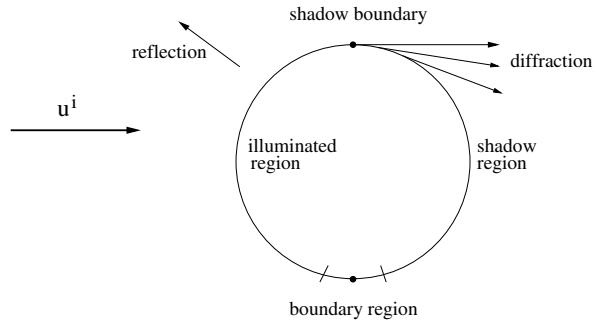


FIG. 4.1. *Reflection and diffraction effects in the scattering of an incoming wave u^i by a smooth and convex obstacle.*

and with the right-hand side given by

$$v^i(x) := \frac{\partial u^i}{\partial n}(x) + i\eta u^i(x).$$

The unknown in (4.6) is the function $q_s(t)$. Since $q_s(t)$ is less oscillatory than the original density function $q(\kappa(t))$, one can solve (4.6) using a coarser discretization. In the next section, we discuss the remaining oscillatory behavior of $q_s(t)$.

4.3. Asymptotic behavior of the solution. In the past decades, a lot of effort has been invested in studying the asymptotic behavior of the solution $q(x)$ to (4.4) as a function of the wavenumber, concentrating mainly on the scattering of a plane wave (see, e.g.,[23, 27] and references therein). For smooth and convex obstacles, there are three important regions with different properties, illustrated in Figure 4.1: the illuminated region, the shadow region, and the transitional shadow boundary region. In the illuminated region, the scattered wave is described asymptotically by geometrical optics: a wave is reflected such that the angle of incidence and the angle of reflection are identical. A wave tangential to the shadow boundary is diffracted. The density function decays rapidly away from the shadow boundary into the shadow region, due to the continuous emission of diffracted waves. In the shadow region the function $q(x)$ approaches zero as k increases.

The asymptotic expansion of $q_s(t)$ for large k reflects these three regions. Assume an incoming plane wave in the direction α of the form $u^i(x) = e^{ik\alpha\cdot x}$, with $|\alpha| = 1$. It was proved in [27] that $q_s(t)$ has a uniform asymptotic expansion of the form

$$q_s(t) \sim \sum_{m,n\geq 0} k^{2/3-n-2m/3} b_{m,n}(\alpha,t)\Psi^{(n)}(k^{1/3}Z(\alpha,t)), \tag{4.7}$$

which remains valid as t crosses the shadow boundaries. The shadow boundary points can be characterized here by $\alpha\cdot\nu = 0$, with ν the exterior normal to Ω. For a thorough discussion and analysis of this expansion, we refer the reader to [12]. Here, we only recall the main characteristics of the expansion, as a means to motivate the choices that are made later on.

The function $Z \in C^\infty$ is infinitely smooth and has a simple root at the two shadow boundary points. The function Z is positive when $\alpha\cdot\nu < 0$, i.e., in the illuminated region, and it is negative in the shadow region. The function $\Psi(z)$ is smooth for positive arguments, with

$$\Psi(z) \sim z, \qquad z \to \infty, \tag{4.8}$$

and it is decaying rapidly but in an oscillatory manner for large negative arguments. It follows that the function $q_s(t)$ is smooth in the illuminated region but may be oscillatory in the shadow region, corresponding to the behavior of $\Psi(z)$ for large positive arguments and large negative arguments, respectively. One can also deduce the following asymptotic properties from the leading order of (4.7):

$$|q_s(t)| = \begin{cases} O(k), & \text{illuminated region,} \\ O(k^{2/3}), & \text{shadow boundary.} \end{cases} \tag{4.9}$$

There is a k-dependent transition region near the shadow boundary. Motivated by (4.7), we introduce a transition region of size $O(k^{-1/3})$ around the shadow boundary points, and define the shadow boundary regions as

$$T_{B1}(k) = [t_{sb1} - D_1 k^{-1/3}, t_{sb1} + C_1 k^{-1/3}], \tag{4.10}$$

$$T_{B2}(k) = [t_{sb2} - C_2 k^{-1/3}, t_{sb2} + D_2 k^{-1/3}], \tag{4.11}$$

with constants $C_1, C_2, D_1, D_2 > 0$ independent of k, but small enough such that $T_{B1}(k)$ and $T_{B2}(k)$ are nonoverlapping, and with t_{sb1} and t_{sb2} the locations of the two shadow boundary points in the parameter domain $[0, 1]$. The illuminated region is defined as

$$T_I(k) = (t_{sb1} + C_1 k^{-1/3}, t_{sb2} - C_2 k^{-1/3}). \tag{4.12}$$

The shadow region is the remaining part of the interval $[0, 1]$.

The size of the transition region is related to the behavior of the argument $k^{1/3} Z(\alpha, t)$ of the function $\Psi(z)$ in (4.7). Because $Z(\alpha, t)$ has a simple zero at t_{sb1}, we have

$$Z(\alpha, t) \approx Z'(\alpha, t_{sb1})(t - t_{sb1}), \qquad t \to t_{sb1}.$$

Hence, there exists a constant c such that for sufficiently large k we have

$$|k^{1/3} Z(\alpha, t)| \leq c \qquad \text{if } t \in T_{B1}(k). \tag{4.13}$$

A similar bound holds for the region $T_{B2}(k)$. Since the oscillations of $q_s(t)$ originate in the oscillatory behavior of $\Psi(z)$ for negative arguments, the meaning of the bound (4.13) is that the number of oscillations of $q_s(t)$ in the shadow boundary region $T_{B1}(k)$ is bounded in k.

Remark 4.1. The remaining oscillatory behavior of $q_c(t)$ in the shadow boundary region indicates that the oscillations of the incoming wave do not completely describe the oscillatory behavior of $q(x)$. Still, we will see that the ansatz (4.5) is suitable for further computations.

5. A hybrid high-frequency boundary element method. The collocation of integral equation (4.6) in a point x_n leads to a one-dimensional and oscillatory integral in the integration variable τ. In this section, we show how an efficient quadrature rule can be used for the discretization of that collocation integral. First, we discuss the classical boundary element approach in section 5.1. The application of specialized quadrature rules, leading to a highly sparse discretization matrix, is described in section 5.2. We comment on the location of the quadrature points section 5.3 and on the limitations of the approach in section 5.4. It turns out that the quadrature rule can not be applied everywhere. Finally, we arrive at a method combining a sparse discretization with a dense part in section 5.5.

5.1. Collocation approach for the discretization. Consider a collocation scheme for integral equation (4.6), with a set of N distinct collocation points $x_n = \kappa(t_n)$, $t_n \in [0,1]$, $n = 1,\dots,N$. The classical way to proceed is to look for an approximation q_c to solution q_s in the form

$$q_c(t) = \sum_{m=1}^{N} c_m \phi_m(t), \tag{5.1}$$

where the ϕ_m functions are a set of linearly independent basis functions with support $\Omega_m := \mathrm{supp}(\phi_m)$. The number of basis functions may be small, since the exact solution q_s is not very oscillatory. Collocating (4.6) in the points t_n, with q_s replaced by q_c, leads to the equations

$$\frac{1}{2} q_c(t_n) e^{ikg^i(x_n)} + \int_0^1 G(t_n,\tau) q_c(\tau) e^{ikg^i(\kappa(\tau))}\,\mathrm{d}\tau = v^i(x_n), \qquad n = 1,\dots,N. \tag{5.2}$$

The collocation approach therefore leads to a linear system $Ac = b$ of size $N \times N$, where the elements of the discretization matrix A are given by

$$A_{n,m} = \frac{1}{2}\phi_m(t_n) e^{ikg^i(\kappa(t_n))} + \int_{\Omega_m} G(t_n,\tau) e^{ikg^i(\kappa(\tau))} \phi_m(\tau)\,\mathrm{d}\tau, \tag{5.3}$$

and the right-hand side by $b_n = v^i(x_n)$. The discretization matrix A is dense, but small compared to the classical boundary element discretization for the original equation. Hence, this is a big improvement over the direct discretization of (4.4). Since the elements (5.3) are given by oscillatory integrals, they can be computed efficiently using the numerical steepest descent technique described in section 2.1. This would yield an efficient total solution method, which remains efficient when k increases, but that is also accurate for small values of k.

However, there are still some issues associated with this approach. Since the matrix is dense, the method requires the evaluation of N^2 integrals. Although N may be rather small, the computational cost can still be high. Interestingly, it was observed in [16] that many of the elements are small and can in fact be discarded, reducing the computation time. A second, more important issue is that the results of [16] indicate that the error of the scheme increases with increasing wavenumber. Here, we examine a different discretization of (5.2) that aims to address these issues, based on the quadrature rule developed in section 3 and motivated by the accuracy of this rule for high wavenumbers. Owing to the small number of required quadrature points, the resulting discretization matrix will be highly sparse.

5.2. The application of specialized quadrature rules. Based on the collocation integral (5.2), we define the following general oscillatory integral:

$$I_c[f;t_n] := \int_0^1 f(\tau) G(t_n,\tau) e^{ikg^i(\kappa(\tau))}\,\mathrm{d}\tau. \tag{5.4}$$

The oscillatory function $G(t,\tau)$ in the integrand is given explicitly by

$$\begin{aligned} G(t,\tau) = &-\frac{i}{4} k \frac{\nabla\kappa(\tau)\cdot(\kappa(t)-\kappa(\tau))}{|\kappa(t)-\kappa(\tau)|} |\nabla\kappa(\tau)| H_1^{(1)}(k|\kappa(t)-\kappa(\tau)|) \\ &-\frac{\eta}{4} |\nabla\kappa(\tau)| H_0^{(1)}(k|\kappa(t)-\kappa(\tau)|). \end{aligned}$$

Choosing $\eta = k$, this function can further be written as

$$G(t,\tau) = a(\tau)kH_0^{(1)}(k|\kappa(t)-\kappa(\tau)|) + b(t,\tau)kH_1^{(1)}(k|\kappa(t)-\kappa(\tau)|), \tag{5.5}$$

with functions $a(\tau)$ and $b(t,\tau)$ that are independent of k. Hence, integral (5.4) can be written as a sum of two model integrals of the form $I_H[f]$, involving Hankel functions, multiplied by k. Both model integrals have the same oscillator,

$$g(\tau;t_n) := |\kappa(t_n)-\kappa(\tau)| + g^i(\kappa(\tau)). \tag{5.6}$$

In particular, this means that one can find a quadrature rule such that

$$I_c[f;t_n] \approx Q_c[f;t_n] := \sum_{l=0}^{L_n}\sum_{j=0}^{d_{n,l}} w^c_{n,l,j} f^{(j)}(\tau_{n,l}). \tag{5.7}$$

The rule $Q_c[f;t_n]$ is computed as follows:

- The weights $w^c_{n,l,j}$ are given by the expressions (3.9)–(3.11) for the weights $w^H_{l,j}$ of the model integral, with the line integrals $S^H[f;x]$ replaced by

$$S^c[f;x] := \int_0^P f(h_x(p))G(t_n,h_x(p))e^{ikg^i(\kappa(h_x(p)))}h_x'(p)\,\mathrm{d}p, \tag{5.8}$$

  where the integration path is the path of steepest descent corresponding to the oscillator (5.6). Compared to $S^H[f;x]$, we have included the additional factors $ka(\tau)$ and $kb(t,\tau)$ from (5.5) into the weight function of the integral. We note that it implies the assumption that $\kappa(\tau)$ is analytic. The scaling by a factor k reduces the asymptotic order, based on the absolute error, by 1.
- The quadrature points $\tau_{n,l}$ are found from the oscillator (5.6). They are the points where the integrand becomes singular (and hence nonanalytic) and the stationary points of the oscillator $g(\tau;t_n)$. These points are derived by a straightforward but technical analysis of $g(\tau;t_n)$. There are no contributing endpoints, as the integrand is periodic on the closed curve Γ.

We now describe how the quadrature rule (5.7) can be used in the discretization. The derivatives of q_c can be written in terms of the basis functions ϕ_m,

$$q_c^{(j)}(\tau) = \sum_{m=1}^{N} c_m \phi_m^{(j)}(\tau). \tag{5.9}$$

Hence, applying the quadrature rule to q_c yields a matrix B with entries

$$B_{n,m} = \begin{cases} \sum_{l:\tau_{n,l}\in\Omega_m}\sum_{j=0}^{d_{n,l}} w^c_{n,l,j}\phi_m^{(j)}(\tau_{n,l}), & \exists l\in[0,L_n]:\tau_{n,l}\in\Omega_m, \\ 0 & \text{otherwise.} \end{cases} \tag{5.10}$$

The entry $B_{n,m}$ is nonzero only if at least one quadrature point $\tau_{n,l}$ exists that lies in the support of the basis function ϕ_m. The number of nonzero points therefore depends on the size of the supports of the basis functions. If all basis functions are local, then the matrix B is highly sparse.

Remark 5.1. The asymptotic order of quadrature rule (5.7) is limited by the number of derivatives $d_{n,l}$ that are used, as described by Theorem 3.6. In turn, note from (5.9) that $d_{n,l}$ is limited by the number of derivatives that exist for the chosen basis functions.

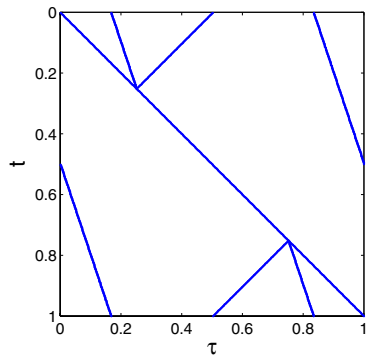


FIG. 5.1. *The location of the contributing points of the collocation integral for the scattering of a plane wave by a circular obstacle. Each row corresponds to a fixed value* $t \in [0, 1]$*, with the shadow boundary points at* 0.25 *and* 0.75*, and the illuminated region in between. The singular points are located along the diagonal. The off-diagonal points correspond to stationary points.*

5.3. Location of the quadrature points. The location of the quadrature points is illustrated in Figure 5.1 for the scattering of a plane wave by a circular obstacle. There is one stationary point if t_n lies in the illuminated region, and there are three stationary points if t_n lies in the shadow region. Two of these points coalesce into one stationary point of order $r = 2$ exactly at the shadow boundary. Using local basis functions, the structure of the discretization matrix B closely resembles that of Figure 5.1.

The coalescence of two stationary points at the shadow boundary is a general property for convex obstacles [22]. The quadrature points on the diagonal reflect the general principle of localization of high-frequency scattering, which states that the scattering of a ray is asymptotically determined by properties local to the point of incidence [13]. As such, Figure 5.1 is illustrative for more general convex shapes.

Remark 5.2. Figure 5.1 shows the existence of a stationary point in the shadow region when $t \in T_I(k)$ is a point in the illuminated region. This seems to contradict the localization principle. Indeed, this stationary point has no physical relevance for the scattering problem. Following Remark 4.1, it is an artifact of the incorrect ansatz in the shadow region.

5.4. Limitations of the sparse discretization. The convergence of quadrature rule $Q_c[f; t_n]$ as a function of k is described by the corresponding results for $Q_H[f]$ discussed in section 3.3. In particular, the results imply that the accuracy of the approximation increases with increasing k, at a rate that depends on the number of derivatives that are used in the quadrature rule. For example, for $t \in T_I(k)$, we have

$$|I_c[f; t_n] - Q_c[f; t_n]| = O\left(k^{-\min\{d_{n,0}+1,(d_{n,1}+2)/2-3/2\}}\right), \tag{5.11}$$

where $\tau_{n,0} = t_n$ and $\tau_{n,1}$ is the stationary point of order $r = 1$ in the shadow region. Hence, the sparse discretization matrix B is very well suited for computing the application of the integral operator under consideration to a function of the form $f(t)e^{ikg^i(\kappa(t))}$.

The convergence results in section 3.3 were derived, however, with the assumption that the function f is independent of k. No statements can therefore be made

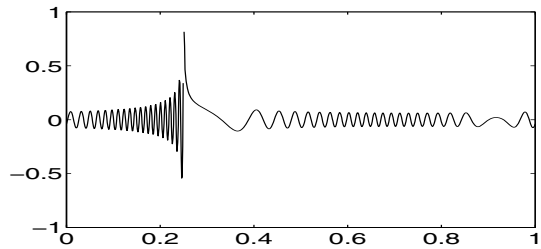


FIG. 5.2. *The real part of the integrand for $t_n = t_{sb1}$ for a circular obstacle and $k = 100$.*

regarding the computed solution of an integral equation using the sparse discretization matrix B, because the solution generally depends on k. Moreover, it was shown that the assumed oscillatory behavior $e^{ikg^i(\kappa(t))}$ in the ansatz (4.5) does not fully describe the oscillatory behavior of the exact solution $q(x)$. Therefore, $Q_c[q_s; t_n]$ might not be a good approximation to $I_c[q_s; t_n]$. It is important that the quadrature rule be accurate when applied to the exact solution q_s. Otherwise, one cannot expect that solving the system with the sparse representation matrix B can yield a good approximation of the exact solution.

It can be verified that $Q_c[q_s; t_n]$ is accurate when $t_n \in T_I(k)$, i.e., when t_n lies in the illuminated region. The cases $t \in T_{sb1}(k)$ and $t \in T_{sb2}(k)$ are more problematic. Figure 5.2 shows the real part of the integrand of the collocation integral $I_c[1; t_{sb1}]$, where $t_n = t_{sb1}$ coincides with the first shadow boundary. This integral has a stationary point of order $r = 2$ at t_{sb1}, but only to the right of the singularity. The integral is highly oscillatory to the left of the shadow boundary, and not oscillatory to the right. It turns out that a quadrature rule similar to $Q_c[f; t_n]$ can be used on the left interval $[0, t_{sb1}]$ and in the illuminated region $T_I(k)$. It is not suited for the intermediate interval $[t_{sb1}, t_{sb1} + C_1 k^{-1/3}]$. Due to the stationary point, however, the integrand is not oscillatory in that interval. In the following section, we will construct a sparse approximation matrix that is based on the use of specialized quadrature rules in the regions where they apply. They are combined with a classical dense discretization in the remaining part of the integration domain.

Remark 5.3. The failure of $Q_c[q_s; t_{sb1}]$ for the exact solution q_s can be seen as follows: the asymptotic decay of the quadrature weights $w_{n,l,j}$ for increasing k is cancelled exactly by the growth of the derivatives $q_s^{(j)}(t_{sb1})$ for increasing k. Hence, the rule does not converge.

5.5. A sparse discretization for the scattering problem.

Basis functions and collocation points. The basis functions for the discretization are chosen corresponding to the behavior of the solution in the three different regions identified in section 4.3. Recall that the solution is smooth in the illuminated region $T_I(k)$, and oscillatory but rapidly decaying in the shadow region. First, following [7], we approximate the solution by zero in the shadow region. We choose a fixed number of basis functions in the illuminated region $T_I(k)$. Finally, we also choose a fixed number of basis functions in the transitional shadow boundary regions $T_{B1}(k)$ and $T_{B2}(k)$, independently of k. This corresponds to using a fixed number of basis functions per oscillation of the solution in these regions.

The collocation points V are chosen equidistantly in the regions $T_{B1}(k)$, $T_{B2}(k)$,

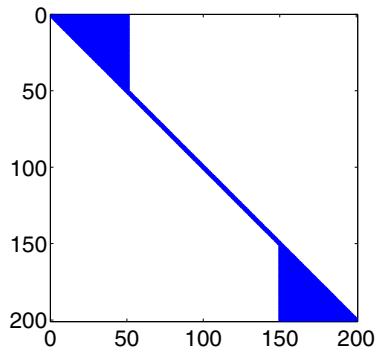


FIG. 5.3. *Illustration of the sparse discretization matrix using cubic B-splines for the scattering of a plane wave by a circular obstacle. The middle part of the matrix is tridiagonal.*

and $T_I(k)$. We have $V = V_1 \cup V_2 \cup V_3$, where

$$V_1 := \{t_{sb1} - D_1 k^{-1/3} + j h_1\}_{j=0}^{N_1}, \quad \text{with } h_1 := \frac{(C_1 + D_1)k^{-1/3}}{N_1},$$

$$V_2 := \{t_{sb1} + C_1 k^{-1/3} + j h_2\}_{j=1}^{N_2-2}, \quad \text{with } h_2 := \frac{(t_{sb2} - t_{sb1} - (C_1 + C_2)k^{-1/3})}{N_2 - 1},$$

$$V_3 := \{t_{sb2} - C_2 k^{-1/3} + j h_3\}_{j=0}^{N_3}, \quad \text{with } h_3 := \frac{(C_2 + D_2)k^{-1/3}}{N_3}.$$

These are $N = N_1 + N_2 + N_3$ points. As basis functions, we choose B-splines of odd degree s with knots U given by

$$U := V \cup \{t_{sb1} - D_1 k^{-1/3} - j h_1\}_{j=1}^{(s+1)/2} \cup \{t_{sb2} + D_1 k^{-1/3} + j h_3\}_{j=1}^{(s+1)/2}. \tag{5.12}$$

These $N + s + 1$ points uniquely determine N distinct B-splines. The basis functions are $s-1$ times continuously differentiable. The nodes in the shadow region are added in order to ensure a smooth transition into the shadow region where the solution is approximated by zero.

Discretization. For the discretization, we propose the following scheme. If $t_n \in T_I(k)$, a collocation point in the illuminated region, then

- the quadrature rule $Q_c[f; t_n]$ is applied for the singular point t_n,
- the stationary points lie in the shadow region, and they are discarded.

For splines of degree s, the singular point t_n lies in the support of only s separate basis functions. The corresponding matrix entries are given by (5.10). The contributions of the stationary points are discarded because the solution is approximated by zero in the shadow region. Recall that in the context of the scattering problem, following Remark 5.2, these are only spurious stationary points.

If $t_n \in T_{B1}(k)$, corresponding to the first shadow boundary region, then

- a specialized quadrature rule for oscillatory integrals is applied on the interval $[0, t_n]$,
- a classical dense discretization is used on the interval $[t_n, t_{sb1} + C_1 k^{-1/3}]$,
- a specialized quadrature rule for oscillatory integrals is again applied on the interval $[t_{sb1} + C_1 k^{-1/3}, 1]$.

The quadrature rule on $[0, t_n]$ reduces to the contribution of the singular point t_n. The corresponding weights have the form of (3.11) with S_l^H replaced by S_l^c,

$$w_{n,l_n,j}^c = -S_{l_n}^c \left[\frac{(x-t_n)^j}{j!}; t_n \right],$$

where l_n is the index such that $t_n = \tau_{n,l_n}$. The quadrature rule on $[t_{sb1} + C_1 k^{-1/3}, 1]$ consists of the contributions of the stationary points $\tau_{n,l}$ outside the shadow boundary region, and of the endpoint $t_r := t_{sb1} + C_1 k^{-1/3}$. The weights corresponding to that endpoint have the form of (3.9),

$$w_{n,l_r,j}^c = S_{l_r+1}^c \left[\frac{(x-t_r)^j}{j!}; t_r \right],$$

where l_r is the index such that $t_r \in [\tau_{n,l_r}, \tau_{n,l_r+1}]$. Finally, the dense discretization in the interval $[t_n, t_r]$ leads to elements of the form

$$\sigma_{n,m} = \frac{1}{2}\phi_m(t_n) e^{ikg^i(x_n)} + \int_{\Omega_m \cap [t_n, t_r]} G(t_n, \tau) e^{ikg^i(\kappa(\tau))} \phi_m(\tau)\, \mathrm{d}\tau. \tag{5.13}$$

The only difference compared to (5.3) is that the integration domain may be cut at the boundaries of $[t_n, t_r]$. Summarizing, the elements of the discretization matrix for $t_n \in T_{B1}(k)$ can be written as

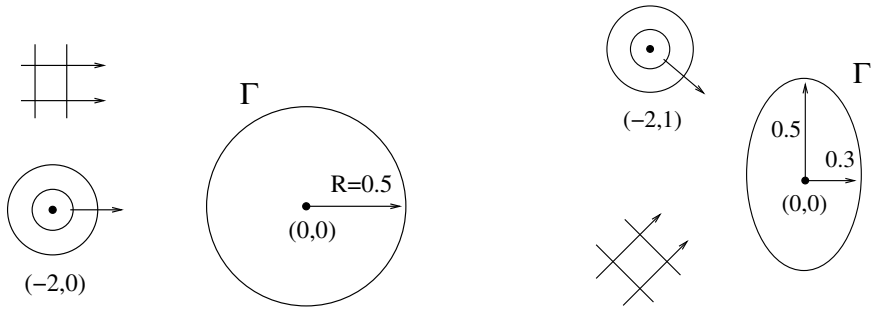
$$C_{n,m} = \begin{cases} \sigma_{n,m} & \text{if } \Omega_m \cap [t_n, t_r] \neq \varnothing \\ \quad + \sum_{j=0}^{d_{n,l_n}} w_{n,l_n,j}^c \phi_m^{(j)}(t_n) & \text{if } t_n \in \Omega_m \\ \quad + \sum_{j=0}^{d_{n,l_r}} w_{n,l_r,j}^c \phi_m^{(j)}(t_r) & \text{if } t_r \in \Omega_m \\ \quad + \sum_{l: t_r < \tau_{n,l} \in \Omega_m} \sum_{j=0}^{d_{n,l}} w_{n,l,j}^c \phi_m^{(j)}(\tau_{n,l}), & \exists l \in [0, L_n] : t_r < \tau_{n,l} \in \Omega_m, \\ \sum_{l: t_r < \tau_{n,l} \in \Omega_m} \sum_{j=0}^{d_{n,l}} w_{n,l,j}^c \phi_m^{(j)}(\tau_{n,l}), & \exists l \in [0, L_n] : t_r < \tau_{n,l} \in \Omega_m, \\ 0 & \text{otherwise.} \end{cases}$$

The case $t_n \in T_{B2}(k)$ can be treated similarly. The structure of the sparse matrix C is illustrated in Figure 5.3. The two small dense parts correspond to the dense discretization in the intervals $[t_n, t_{sb1} + C_1 k^{-1/3}]$ and $[t_{sb2} - C_2 k^{-1/3}, t_{sb2}]$. For simplicity, we have chosen the constant C_1 large enough such that, for $t_n \in T_{B1}$, all stationary points $\tau_{n,l} \in T_{B1}(k)$ also lie in the shadow boundary region. The constant C_2 was chosen similarly.

One can verify that the required integrals of the form (5.13) are not oscillatory. Due to the stationary point of order $r = 2$, the integrand behaves as $e^{ikc(\tau - t_{sb1})^3}$ near the shadow boundary. The argument of the exponential is bounded in k, since by construction we have

$$|\tau - t_{sb1}| \leq \max\{C_1, D_1\} k^{-1/3}.$$

Hence, there is only a bounded number of oscillations in the integrals for increasing k. It is therefore reasonable to expect that the integrals can be evaluated with a number of operations that is independent of k. Since the weights of the quadrature rule can be evaluated efficiently as well, and because the number of unknowns is fixed, the matrix in Figure 5.3 can be computed with a total number of operations that is independent of k.



(a) Circle (b) Ellipse

FIG. 6.1. *Illustration of two smooth convex scattering obstacles. The boundary conditions are plane waves, or circular waves originating from a point source.*

6. Numerical results. We consider the scattering by two convex obstacles, a circle and an ellipse, shown in Figure 6.1. We use two types of boundary conditions: a plane wave, modeled in the form $u^i(x) = e^{ik\alpha\cdot x}$, and a point source, modeled by $u^i(x) = H_0^{(1)}(|x - x_0|)$, with x_0 a point in the exterior Ω^+ of the obstacle. The circle and ellipse are parameterized by

$$\kappa(t) = \begin{cases} R\cos(2\pi t), \\ R\sin(2\pi t), \end{cases} \quad \text{and} \quad \kappa(t) = \begin{cases} R_1\cos(2\pi t), \\ R_2\sin(2\pi t), \end{cases}$$

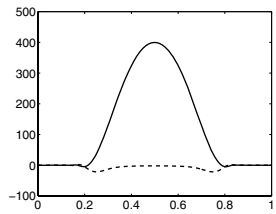
respectively.

In all examples, we have chosen the constants C_1 and C_2 large enough (w.r.t. D_1 and D_2) such that the sparse matrix has the form shown in Figure 5.3. The highest order of the derivative used in the specialized quadrature rule for oscillatory integrals is kept fixed, $d_{n,l} = d$, $\forall n, l$, with $d \leq s - 1$, where s is the degree of the piecewise polynomial spline basis. The integrals (5.13) for the elements of the dense parts that are singular were evaluated using Cubpack [9].

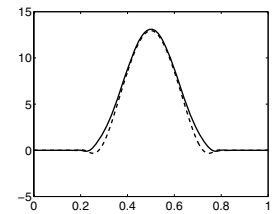
6.1. Total solution time. The smooth function $q_c(t)$ is illustrated in Figure 6.2 as it was computed for the different scattering problems. The mild oscillatory behavior of the function near the shadow boundary is illustrated in the left panel of Figure 6.3, showing only the real part of the solution. Two spikes are present near the shadow boundary, with a peak value that scales as $O(k^{2/3})$, as predicted by the estimate (4.9). The dashed line shows the effect of doubling k. The $O(k)$ behavior in the illuminated region is clear from the imaginary part illustrated in the right panel of Figure 6.3.

Table 6.1 shows the timings for an implementation of the algorithm of section 5.5 in Matlab. In all the examples considered, the time actually decreases with increasing wavenumber k. This is due to the fact that, at larger frequencies, the weights of the specialized quadrature rule $Q_c[f; t_n]$ are easier to compute. Also, the integrals (5.13) that determine the elements of the dense parts do not become more oscillatory. In a classical boundary element method, and using 10 unknowns per wavelength, the case $k = 10^5$ would correspond to a dense matrix with approximately $N = 0.5 \times 10^6$ unknowns.

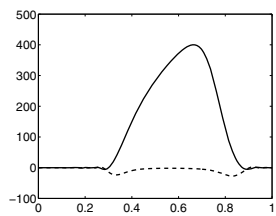
6.2. Error and convergence. In this section, we present numerical results evaluating the error of the proposed scheme. First, we investigate the absolute and



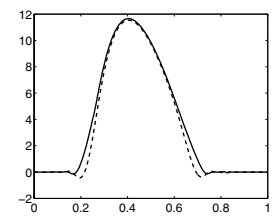
(a) Circle, plane wave



(b) Circle, point source

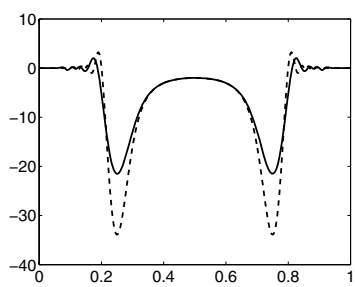


(c) Ellipse, plane wave



(d) Ellipse, point source

FIG. 6.2. *The real part (dashed lines) and imaginary part (solid lines) of the computed smooth function* $q_c(t)$. *The parameters correspond to the case* $k = 200$ *in Table* 6.1.



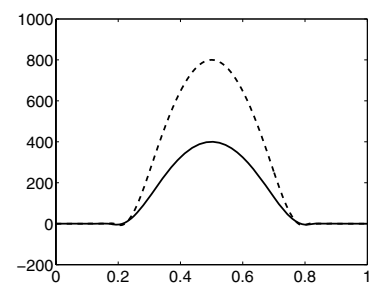


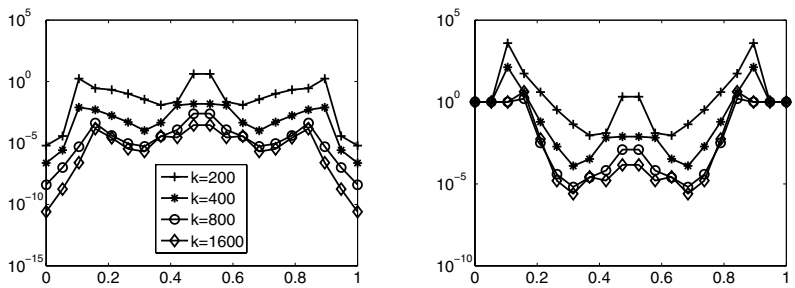
FIG. 6.3. *The real part (left) and imaginary part (right) of* $q_c(t)$ *for the scattering of a plane wave by a circle. The solid lines correspond to* $k = 200$, *the dashed lines to* $k = 400$.

relative error for increasing values of k. Next, we show that the accuracy is improved by increasing the number of derivatives d that are used in the specialized quadrature rules. We then illustrate the influence of the constants C_1, C_2, D_1, and D_2 that determine the size of the shadow boundary region. Finally, we illustrate the influence of the parameters N_1, N_2, and N_3 that determine the total degrees of freedom.

In applications, one is usually interested in the quantity q_s/k. For example, in electromagnetics, this quantity is proportional to the induced current on the surface of the obstacle with a proportionality constant that is independent of k. One can see from (4.9) that q_s/k corresponds to a normalization of q_s for different k. For this

TABLE 6.1
Total solution time in seconds for the different scattering problems. All parameters are kept fixed, except the wavenumber k. We used $d = 2$ derivatives in the quadrature rules.

	Circle		Ellipse	
	Plane wave	Point source	Plane wave	Point source
(C_1, C_2)	$(1.3, 1.3)$	$(1, 0.1.0)$	$(1.5, 1.3)$	$(1.3, 1.3)$
(D_1, D_2)	$(1.0, 1.0)$	$(0.8, 0.8)$	$(1.3, 1.0)$	$(1.0, 1.0)$
(N_1, N_2, N_3)	$(30, 30, 30)$	$(30, 30, 30)$	$(60, 30, 60)$	$(60, 30, 60)$
$k = 200$	$287s$	$312s$	$463s$	$512s$
$k = 400$	$283s$	$308s$	$450s$	$496s$
$k = 800$	$281s$	$306s$	$443s$	$491s$
$k = 1600$	$279s$	$302s$	$438s$	$484s$
$k = 10000$	$273s$	$294s$	$422s$	$470s$
$k = 100000$	$269s$	$289s$	$416s$	$459s$



(a) Absolute error E_a (b) Relative error E_r

FIG. 6.4. *Absolute and relative error for the scattering of a plane wave by a circle for different values of k. We have used $d = 1$ derivative in the specialized quadrature rules. The other parameters correspond to those in Table* 6.1.

reason, in all following results we evaluate the absolute error in a point t as

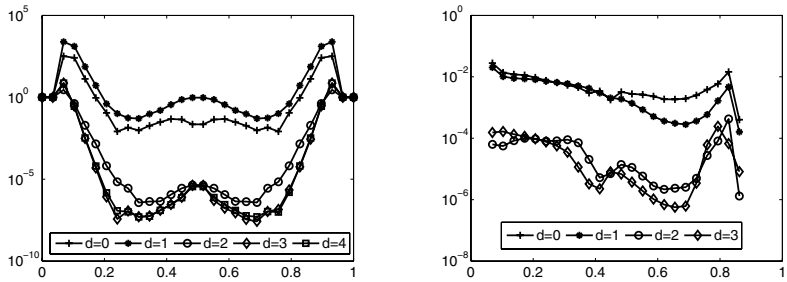
$$E_a(t) = \frac{|q_s(t) - q_c(t)|}{k}, \tag{6.1}$$

and the relative error as

$$E_r(t) = \frac{|q_s(t) - q_c(t)|}{|q_s(t)|}. \tag{6.2}$$

The exact solution q_s of the scattering problem is known analytically only for the case of scattering of a plane wave by a circle. For the other cases, the error is estimated by comparing computed solutions to each other.

The relative and absolute error for the scattering of a plane wave by the circle is shown in Figure 6.4 for increasing values of k. We have chosen to use cubic spline basis functions and derivatives up to order $d = 1$ in each specialized quadrature rule. The figures show that both the absolute error and the relative error decrease with increasing k. This is due to the higher accuracy of the quadrature rule $Q_c[f; t]$ at larger frequencies. The relative error tends to 100% in the deep shadow region because we have approximated the solution by 0 in that region. One can verify from



(a) Relative error for the case of a circle

(b) Absolute error for the case of an ellipse

FIG. 6.5. *Comparison of the absolute and relative error at $k = 200$ for different values of d, the number of derivatives used in the specialized quadrature rules. The left panel corresponds to the case of column 1 in 6.1, the right panel corresponds to the case of column 4.*

TABLE 6.2

Maximal absolute error as a function of the size of the shadow boundary region, and pointwise absolute and relative error in a fixed point $T = 0.13$ in the shadow boundary region. The constants $C_j = C$ and $D_j = D$ are chosen pairwise equal ($k = 200$, $d = 2$, $N_j = 50$).

(C, D)	$(1.0, 0.7)$	$(1.1, 0.8)$	$(1.2, 0.9)$	$(1.3, 1.0)$
$\|E_a\|_\infty$	$1.0E-3$	$4.6E-4$	$2.4E-4$	$1.8E-4$
$E_a(T)$	$4.4E-4$	$7.9E-5$	$3.6E-5$	$5.1E-6$
$E_r(T)$	$3.2E-1$	$5.6E-2$	$2.6E-2$	$3.7E-3$

the figures that the absolute error in that region is still quite small compared to the average value of the function $q_s(t)$. At $k = 1600$, a relative error of order 10^{-5} is achieved in the illuminated region and near the shadow boundary.

The accuracy shown in Figure 6.4 for the smaller value $k = 200$ is quite poor, however. The accuracy is much improved by increasing the number of derivatives used in the quadrature rules. This is illustrated in Figure 6.5(a) for the same scattering example with $k = 200$. The number of derivatives d ranges from 0 to 4. We used cubic spline basis functions for $d = 0, 1, 2$ and splines of degree 5 for $d = 3, 4$. A relative error of order 10^{-8} is achieved in the shadow boundary region, and a relative error of order 10^{-5} in the illuminated region. A similar experiment is illustrated in Figure 6.5(b), corresponding to an incident circular wave on an ellipse-shaped boundary. The error for $d = 0, \ldots, 3$ in this case was estimated by comparing to the results of using $d = 4$.

The constants C_1, C_2, D_1, and D_2 determine the size of the transitional shadow boundary regions. Since the solution is approximated by zero in the shadow region outside these intervals, it can be expected that increasing the value of these parameters results in smaller errors. This is confirmed by the results in Table 6.2, which show the maximum of the absolute error E_a. The maximum is reached in the points $t_{sb1} - D_1 k^{-1/3}$ and $t_{sb2} + D_2 k^{-1/3}$, i.e., on the border of the shadow region. It is also shown that the accuracy of a point in the shadow boundary region increases if the size of that region is increased. The accuracy of the points in the illuminated region is unaffected by this change.

Finally, Table 6.3 shows the result of increasing the total number of degrees of

TABLE 6.3
Maximal absolute error as a function of the number of degrees of freedom.

$N_1 = N_2 = N_3$	20	30	40	50	60
$\|E_a\|_\infty$	$1.5E-2$	$2.4E-3$	$4.9E-4$	$2.2E-4$	$1.9E-4$

freedom for the same scattering example. The accuracy initially increases rapidly, but remains stable as N increases further.

In conclusion, the accuracy of the proposed method is influenced by a number of factors: the value of k, the accuracy of the specialized quadrature rules, the size of the shadow boundary region, and the number of degrees of freedom. The results show that the error is quite small and can be improved by changing the parameters. However, the experiments do not indicate that arbitrary accuracy can be achieved for a given value of k. For achieving high accuracy at very low values of k, the use of specialized quadrature rules is not recommended; one can instead employ the dense discretization described in section 5.1.

7. Concluding remarks. The method presented in this paper achieves a sparse discretization matrix for an integral equation, which is made possible by the presence of strong oscillations. The method is based on a classical boundary element technique near the shadow boundary, and uses a new quadrature formula that is very effective for oscillatory integrals in the illuminated region. The a priori knowledge of the phase of the solution allows a discretization with a fixed small number of unknowns. Since the new quadrature rule requires only few quadrature points, a sparse discretization matrix is obtained. Moreover, numerical results indicate that the accuracy of the solution increases with increasing frequency.

The method can still be improved in a number of ways. For example, we have only considered a localized Filon-type method, while numerical experiments indicate that regular Filon-type methods are typically more accurate for the same frequency. One could also add additional quadrature points besides the singular and stationary points to improve the accuracy. The small densely discretized part near the shadow boundary may possibly be avoided by using a more elaborate ansatz for the asymptotic behavior of the solution, such as those described by the geometric theory of diffraction [17, 23, 16].

The method is limited to smooth convex obstacles. The approach of [7] for smooth convex obstacles can be extended to multiple scattering configurations using an iterative approach [14]. We expect the same will hold for our approach. The extension to three-dimensional problems is the subject of future research. This might be based on suitable cubature rules for multivariate highly oscillatory integrals which have recently been constructed [18].

REFERENCES

[1] T. ABBOUD, J. NÉDÉLEC, AND B. ZHOU, *Méthode des équations intégrales pour les hautes fréquences*, C. R. Acad. Sci. Paris, 318 (1994), pp. 165–170.

[2] K. R. ABEREGG AND A. F. PETERSON, *Application of the integral equation-asymptotic phase method to two-dimensional scattering*, IEEE Trans. Antennas Propagat., 43 (1995), pp. 534–537.

[3] M. ABRAMOWITZ AND I. A. STEGUN, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, Dover Publications, New York, 1965.

[4] P. BETTESS, *Short wave scattering, problems and techniques*, Phil. Trans. R. Soc. Lond. A, 362 (2004), pp. 421–443.

[5] N. Bleistein and R. Handelsman, *Asymptotic Expansions of Integrals*, Holt, Rinehart and Winston, New York, 1975.
[6] N. Bleistein, *Asymptotic expansions of integral transforms of functions with logarithmic singularities*, SIAM J. Math. Anal., 8 (1977), pp. 655–672.
[7] O. P. Bruno, C. A. Geuzaine, J. A. Monro, and F. Reitich, *Prescribed error tolerances within fixed computational times for scattering problems of arbitrarily high frequency: The convex case*, Phil. Trans. R. Soc. Lond. A, 362 (2004), pp. 629–645.
[8] D. Colton and R. Kress, *Integral Equation Methods in Scattering Theory*, Wiley, New York, 1983.
[9] R. Cools and A. Haegemans, *Algorithm* 824*: CUBPACK: A package for automatic cubature; Framework description*, ACM Trans. Math. Softw. 29 (2003), pp. 287–296.
[10] E. Darve, *The fast multipole method: Numerical implementation*, J. Comput. Phys., 160 (2000), pp. 195–240.
[11] P. J. Davis and P. Rabinowitz, *Methods of Numerical Integration*, Comput. Sci. Appl. Math., Academic Press, New York, 1984.
[12] V. Domínguez, I. G. Graham, and V. P. Smyshlyaev, *A hybrid numerical-asymptotic boundary integral method for high-frequency acoustic scattering*, Numer. Math., 106 (2006), pp. 471–510.
[13] V. A. Fock, *Electromagnetic Diffraction and Propagation Problems*, Pergamon Press, New York, 1965.
[14] C. Geuzaine, O. Bruno, and F. Reitich, *On the* $O(1)$ *solution of multiple-scattering problems*, IEEE Trans. Magn., 41 (2005), pp. 1488–1491.
[15] E. Giladi and J. B. Keller, *A hybrid numerical asymptotic method for scattering problems*, J. Comput. Phys., 174 (2001), pp. 226–247.
[16] E. Giladi, *Asymptotically derived boundary elements for the Helmholtz equation in high frequencies*, J. Comput. Appl. Math., 198 (2007), pp. 52–74.
[17] R. F. Harrington, *Time-Harmonic Electromagnetic Fields*, McGraw–Hill, New York, 1961.
[18] D. Huybrechs and S. Vandewalle, *The construction of cubature rules for multivariate highly oscillatory integrals*, Math. Comp., (2007), to appear.
[19] D. Huybrechs and S. Vandewalle, *On the evaluation of highly oscillatory integrals by analytic continuation*, SIAM J. Numer. Anal., 44 (2006), pp. 1026–1048.
[20] A. Iserles, S. P. Nørsett, and S. Olver, *Highly oscillatory quadrature: The story so far*, in Proceedings of ENuMath 2005, Santiago de Compostela, Berlin, 2006, Springer-Verlag, New York, 2006, pp. 97–118.
[21] A. Iserles and S. P. Nørsett, *Efficient quadrature of highly oscillatory integrals using derivatives*, Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci., 461 (2005), pp. 1383–1399.
[22] D. S. Jones, *High-frequency refraction and diffraction in general media*, Phil. Trans. R. Soc. Lond. A, 255 (1963), pp. 341–387.
[23] J. B. Keller, *Geometrical theory of diffraction*, J. Opt. Soc. Amer., 52 (1962), pp. 116–130.
[24] S. Langdon and S. N. Chandler-Wilde, *Implementation of a boundary element method for high frequency scattering by convex polygons*, in Proceedings of the 5th U.K. Conference on Boundary Integral Methods, K. Chen, ed., 2005, University of Liverpool, Liverpool, UK, 2005, pp. 2–11.
[25] S. Langdon and S. N. Chandler-Wilde, *A wavenumber independent boundary element method for an acoustic scattering problem*, SIAM J. Numer. Anal., 43 (2006), pp. 2450–2477.
[26] D. Levin, *Fast integration of rapidly oscillatory functions*, J. Comput. Appl. Math., 67 (1996), pp. 95–101.
[27] R. B. Melrose and M. E. Taylor, *Near peak scattering and the corrected Kirchhoff approximation for a convex obstacle*, Adv. in Math., 55 (1985), pp. 242–315.
[28] J.-C. Nédélec, *Acoustic and Electromagnetic Equations*, Appl. Math. Sci. 144, Springer, Berlin, 2001.
[29] S. Olver, *Moment-free numerical integration of highly oscillatory functions*, IMA J. Numer. Anal., 26 (2006), pp. 213–227.
[30] V. Rokhlin, *Diagonal forms of translation operators for the Helmholtz equation in three dimensions*, Appl. Comput. Harmon. Anal., 1 (1993), pp. 82–93.
[31] E. M. Stein, *Harmonic Analysis: Real-Variable Methods, Orthogonality and Oscillatory Integrals*, Princeton University Press, Princeton, NJ, 1993.
[32] G. N. Watson, *Harmonic functions associated with the parabolic cylinder*, Proc. London Math. Soc. Ser. 2, 17 (1918), pp. 116–148.
[33] R. Wong, *Asymptotic Approximation of Integrals*, Classics in Appl. Math. 34, SIAM, Philadelphia, 2001.

SIAM J. SCI. COMPUT.
Vol. 29, No. 6, pp. 2329–2351
© 2007 Society for Industrial and Applied Mathematics

ON THE PERFORMANCE OF TENSOR METHODS FOR SOLVING ILL-CONDITIONED PROBLEMS*

BRETT W. BADER† AND ROBERT B. SCHNABEL‡

Abstract. This paper investigates the performance of tensor methods for solving small- to large-scale systems of nonlinear equations where the Jacobian matrix at the root is ill-conditioned or singular. This condition occurs on many classes of problems, such as identifying or approaching turning points in path-following problems. The singular case has been studied more than the highly ill-conditioned case, for both Newton and tensor methods. It is known that Newton-based methods do not work well with singular problems because they converge linearly to the solution and, in some cases, with poor accuracy. On the other hand, direct tensor methods have performed well on singular problems and have superlinear convergence on such problems under certain conditions. This behavior originates from the use of a special, restricted form of the second-order term included in the local tensor model that provides information lacking in a (nearly) singular Jacobian. With several implementations available for large-scale problems, tensor methods now are capable of solving larger problems. We compare the performance of tensor methods and Newton-based methods for small- to large-scale problems over a range of conditionings, from well-conditioned to ill-conditioned to singular. Previous studies with tensor methods concerned only the ends of this spectrum. Our results show that tensor methods are increasingly superior to Newton-based methods as the problem grows more ill-conditioned.

Key words. nonlinear equations, tensor methods, Newton's method, ill-conditioned problems, singular problems

AMS subject classification. 65H10

DOI. 10.1137/040607745

1. Introduction.

This paper examines two classes of methods for solving the following nonlinear equations problem:

$$\text{given } F : \mathbb{R}^n \to \mathbb{R}^n, \quad \text{find } x_* \in \mathbb{R}^n \text{ such that } F(x_*) = 0, \tag{1.1}$$

where it is assumed that $F(x)$ is at least once continuously differentiable. General systems of nonlinear equations defined by (1.1) arise in many practical situations, including systems produced by finite-difference or finite-element discretizations of boundary value problems for ordinary differential equations (ODEs) and partial differential equations (PDEs).

As a subset of the general nonlinear equations problem (1.1), there is a class of important problems where $F'(x_*)$ is singular or, at least, very ill-conditioned. Such examples arise in bifurcation tracking and path-following problems where the goal is to locate turning points, such as the ignition and extinction points in chemical combustion. Resolving these features is important to engineers, who, for instance, may

*Received by the editors May 4, 2004; accepted for publication (in revised form) March 15, 2007; published electronically October 5, 2007. This work was supported in part by Air Force Office of Scientific Research grant F49620-00-1-0162 and Army Research Office grant DAAG55-98-1-0176. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights.

http://www.siam.org/journals/sisc/29-6/60774.html

†Computational Sciences Department, Sandia National Laboratories, Albuquerque, NM 87185-1318 (bwbader@sandia.gov).

‡Department of Computer Science, University of Colorado, Boulder, CO 80309. Current address: School of Informatics, Indiana University, Bloomington, IN 47405 (schnabel@indiana.edu).

be designing control systems for such applications and may need to know important operating boundaries.

Standard methods for solving (1.1), such as Newton's method, base each iteration upon a local, linear model $M(x_k + d)$ of the function $F(x)$ around the current iterate $x_k \in \mathbb{R}^n$. Standard methods work well for problems where the Jacobian at the solution, $F'(x_*)$, is well-conditioned, but they face difficulties when the Jacobian is singular, or even nearly singular, at the solution. Many authors have analyzed the behavior of Newton's method on singular problems and have proposed acceleration techniques as remedies (see, e.g., Decker, Keller, and Kelley [9]; Decker and Kelley [10, 11, 12]; Griewank [19]; Griewank and Osborne [20]; Kelley and Suresh [22]; and Reddien [28]). Their collective analysis shows that, from many starting points, Newton's method is locally q-linearly convergent with constant converging to $\frac{1}{2}$ on singular problems where the second-order term $F''(x_k)$ contains appropriate null space information. Acceleration techniques can improve this behavior; however, they require a priori knowledge that the problem is singular, which is not practical for general problem solving.

Tensor methods, however, do not require a priori knowledge of whether the problem is singular or not. These methods were introduced by Schnabel and Frank in [34] and base each iteration on a simplified quadratic model of $F(x)$ such that the quadratic term is a low-rank secant approximation that augments the standard linear model. Tensor methods also have been extended by other authors to utilize iterative solvers, making the methods appropriate for solving large-scale problems (see [1], [4], and [17]).

The analysis in [16] proves that direct tensor methods have quadratic convergence on nonsingular problems and a superlinear convergence rate on problems where the Jacobian matrix at the solution is singular. Specifically, when the rank of the Jacobian at the root is $n-1$, "practical" tensor methods (i.e., those using secant approximations for the tensor term) have three-step superlinear convergence behavior with q-order $\frac{3}{2}$. In practice, one-step superlinear convergence frequently is observed on these problems, which makes the method even more attractive. The second-order term provides higher order information in recent step directions, which aids in cases where the Jacobian is (nearly) singular at the solution. As the iterates approach the solution, the Jacobian lacks information in the null space direction, and the second-order term supplies useful information for a better quality step. Computational evidence in [34] on small problems shows that tensor methods show 21–23% average improvement (in terms of nonlinear iterations and function evaluations) over standard methods on nonsingular problems and 40–43% improvement on singular problems with $\text{rank}(F'(x_*)) = n - 1$.

While tensor methods have encouraging theoretical and computational results on singular problems, less is known about their performance relative to Newton's method on ill-conditioned problems. Do tensor methods outperform Newton's method due to the close relationship of ill-conditioned matrices with singular matrices? Or do tensor methods exhibit superior behavior only when the problem is truly singular? Does the computational performance of Newton's method degrade gradually as the problem becomes more singular, or sharply at the singularity? The performance comparison over a spectrum of ill-conditioned problems was previously unknown. Thus, this paper examines the performance of tensor methods versus standard methods as the problem becomes more ill-conditioned. We consider tensor methods using direct factorizations of the Jacobian matrix for small-scale problems in addition to Krylov-based iterative tensor methods for large-scale problems.

The organization of this paper is as follows. Because this research involves meth-

ods for solving small- to large-scale problems, this paper includes background for both types in section 2. Specifically, we review direct methods for solving small-scale problems, and we review Krylov-based iterative methods for solving large-scale problems, including the relevant algorithms from [1] and [17]. Section 3 presents numerical results on several small- and large-scale ill-conditioned test problems to examine the performance of tensor methods on problems over a range of conditionings, from well-conditioned to singular. Finally, section 4 summarizes the numerical results and provides some concluding remarks.

Throughout this paper, a subscript k refers to the current iterate of a nonlinear solver. We denote the Jacobian $F'(x)$ by $J(x)$ and abbreviate $J(x_k)$ as J_k. Similarly, $F(x_k)$ is abbreviated often as F_k. When the context is clear, we may drop the subscript k while still referring to the "current" values at an iteration.

2. Algorithms. In this section, we introduce the relevant methods for solving systems of nonlinear equations. We start with methods that use a direct factorization of the Jacobian matrix, and then we discuss inexact methods that use Krylov subspace projection techniques. General references for these topics in nonlinear solvers include [14], [21], [26], and [30].

2.1. Standard methods. In this paper, we denote by standard methods the class of methods for solving (1.1) that uses a linear approximation to $F(x)$ at each iterate around the current iterate $x_k \in \mathbb{R}^n$. Most notable among these methods is Newton's method, which uses the linear local model

$$M_N(x_k + d) = F_k + J_k d, \tag{2.1}$$

where $d \in \mathbb{R}^n$ is the step and $J_k \in \mathbb{R}^{n\times n}$ is either the current Jacobian matrix or an approximation to it. A root of this local model provides the Newton step

$$d_N = -J_k^{-1} F_k,$$

which is used to reach the next trial point. Thus, Newton's method is defined when J_k is nonsingular and consists of updating the current point with the Newton step,

$$x_+ = x_k + d_N. \tag{2.2}$$

If the Jacobian $J(x_k)$ is Lipschitz continuous in a neighborhood containing the root x_* and $J(x_*)$ is nonsingular, then the sequence of iterates produced by (2.2) converges locally and q-quadratically to x_*. That is, there exist constants $\delta > 0$ and $c \geq 0$ such that the sequence of iterates x_k produced by Newton's method obeys

$$\|x_{k+1} - x_*\| \leq c \|x_k - x_*\|^2$$

if $\|x_0 - x_*\| \leq \delta$.

When these standard approaches use direct factorizations of the Jacobian matrix, we will refer to these methods as direct methods. Due to the storage and linear algebra costs, direct methods are practical only for solving small, dense problems. If the matrix is sparse, then sparse linear algebra techniques may be used to solve larger problems but are still limited by the size of the problem.

2.2. Tensor methods. Tensor methods [34] solve (1.1) by including a second-order term in the local model at each iteration. The local tensor model has the general form

$$M_T(x_k + d) = F_k + J_k d + \tfrac{1}{2} T_k dd, \tag{2.3}$$

where $T_k \in \mathbb{R}^{n\times n\times n}$ is the tensor term at x_k and is selected so that the model interpolates a small number p of function values in the recent history of iterates. By choosing the smallest T_k in the Frobenius norm, T_k has rank p and $T_k dd$ is both simple in form and inexpensive to find. Because (2.3) may not have a root, one solves the minimization subproblem

$$\min_{d\in\mathbb{R}^n} \|M_T(x_k + d)\|_2 , \tag{2.4}$$

and a root or minimizer of the model is the tensor step.

The additional cost of forming, storing, or solving the model is minor compared to Newton's method. Specifically, the additional cost of a direct tensor method is about n^2p multiplications (QR implementation) and $2p$ vectors of length n in storage. For our numerical experiments, we will consider only the simplest case of $p = 1$. Computational evidence in [34] suggests that additional past iterates add little benefit to the computational performance of the tensor method.

2.3. Newton–Krylov methods. Up to this point, we have discussed direct methods for the solution of small, dense problems, such that the local model is solved with direct factorizations of the Jacobian matrix. Standard direct methods, such as Newton's method, are impractical on large-scale problems because of their high linear algebra costs and large memory requirements. Thus, most large systems often are solved successfully using a class of "inexact" Newton methods [13]:

$$x_{k+1} = x_k + d_k, \quad \text{where} \quad F'(x_k)d_k = -F(x_k) + r_k, \quad \|r_k\| \le \eta_k \|F(x_k)\| , \tag{2.5}$$

such that the local model typically is solved only approximately at each step using a less expensive approach. These "inexact" steps then locate the next trial point. The convergence rate of inexact Newton methods depends on the forcing sequence η_k [13]. If η_k is constant, then convergence is linear. Successively better approximations to the linear model at each iteration (i.e., η_k getting smaller as k increases) preserve the rapid convergence behavior of Newton's method when nearing the solution. The computational savings reflected in this less expensive inner iteration is usually partially offset with more outer iterations, but the overall savings still is quite significant on large-scale problems by avoiding the direct methods that solve the local model exactly.

The most common methods for approximately solving the local Newton model are Krylov-based methods, which iteratively solve the linear system projected onto the Krylov subspace $\mathcal{K}$. A linear Krylov subspace method is a projection method that seeks an approximate solution x_m to the linear system $Ax = b$ from an m-dimensional affine subspace $x_0 + \mathcal{K}_m$. Here $\mathcal{K}_m$ is the subspace

$$\mathcal{K}_m(A, r_0) = span\{r_0, Ar_0, A^2r_0, \ldots, A^{m-1}r_0\},$$

where $r_0 = b - Ax_0$ is the residual at an initial guess x_0. A popular Krylov subspace method is the generalized minimum residual method (GMRES), which computes a solution $x_m \in x_0 + \mathcal{K}_m$ such that the residual norm over all vectors in $x_0 + \mathcal{K}_m$ is minimized. That is, at the mth step, GMRES finds x_m such that $\|b - Ax_m\|_2$ is minimized for all $x_m \in x_0 + \mathcal{K}_m$.

Newton-GMRES is one specific method in the class of Newton–Krylov methods. Here, the linear system is the Newton equation $J_k d = -F_k$, and the system is solved via GMRES according to the tolerance η_k in (2.5). Krylov subspace methods have

the appeal of not requiring an explicit Jacobian due to their exclusive use of Jacobian-vector products, which may be calculated by a finite-difference directional derivative. For this reason and others, Newton-GMRES is a popular algorithm for solving large-scale problems, and it will be the standard large-scale Newton-based algorithm for comparisons in our numerical experiments.

Newton–Krylov methods have been considered by many authors, including Brown and Saad [6, 7], Chan and Jackson [8], and Brown and Hindmarsh [5]. Their computational results show that Newton–Krylov methods can be quite effective for many classes of problems in the context of systems of PDEs and ODEs.

2.4. Tensor-Krylov methods. To accommodate large problems and avoid the expensive direct solution of the local model, the three tensor-Krylov methods described in [1] combine the concepts from direct tensor methods with concepts from inexact Newton methods [13] using Krylov-based linear solvers. The tensor-Krylov methods calculate an inexact tensor step from a specially chosen Krylov subspace that facilitates the solution of a minimization subproblem at each iteration. Here, we give a very brief overview of these methods.

All large-scale tensor methods in this paper consider only a rank-one tensor model, which only interpolates the function value at the previous iterate. Thus, the rank-p model of (2.3) reduces to

$$M_T(x_k + d) = F_k + J_k d + \tfrac{1}{2} a_k (s_k^T d)^2, \tag{2.6}$$

such that

$$a_k \in \mathbb{R}^n = \frac{2(F_{k-1} - F_k - J_k s_k)}{(s_k^T s_k)^2}, \tag{2.7}$$

$$s_k \in \mathbb{R}^n = x_{k-1} - x_k. \tag{2.8}$$

In each of the tensor-Krylov methods, the tensor step is found by approximately solving the minimization subproblem

$$\min_{d \in \mathcal{K}_m} \left\| F_k + J_k d + \tfrac{1}{2} a_k (s_k^T d)^2 \right\|_2, \tag{2.9}$$

where $\mathcal{K}_m$ is a specially chosen Krylov subspace that facilitates the solution of the quadratic model. The three methods described in [1] differ in their choice of $\mathcal{K}_m$, and they are identified by this characteristic difference. TK2 and TK2+ use a Krylov-based local solver that starts with an initial block of two vectors (TK2+ also augments the Krylov subspace in a special way). Similarly, TK3 uses an initial block of size three. The three methods share the ability to calculate an approximate tensor step that satisfies the tensor model to within a specified tolerance. Their cost per nonlinear iteration exceeds that of Newton-GMRES by at most $10n+4mn+6m^2$ multiplications (in comparison, GMRES costs $\mathcal{O}(nm^2)$ multiplications). The methods can be readily combined with either left or right preconditioning. More details of these Krylov-subspace methods for solving the tensor model may be found in [1].

2.5. Tensor-GMRES method. Another large-scale tensor method is that of Feng and Pulliam [17], which uses Krylov subspace projection techniques for solving the local tensor model. In particular, it uses GMRES to first find the approximate Newton step $d_N = d_0 + V_m y_m$. The columns of V_m form an orthonormal basis for the Krylov subspace $\mathcal{K}_m$ generated by the corresponding Arnoldi process, and the Hessenberg matrix H_m is also generated from the Arnoldi process. Given these key

matrices, their tensor-GMRES algorithm proceeds to solve a projected version of the tensor model (2.6) along a subspace that spans the Newton step direction and the Krylov subspace from the Newton step solution. (That is, the approximate tensor step is in the span of the Krylov subspace $\mathcal{K}_m^N$ and d_0, or, equivalently, the span of the matrix $[V_m, d_0]$.) Thus, their algorithm solves the least-squares problem

$$\min_{d \in \{d_0\} \cup \mathcal{K}_m^N} \left\| F_k + J_k d + \tfrac{1}{2} P a (s^T d)^2 \right\|, \tag{2.10}$$

where P is the projection matrix

$$P = Y(Y^T Y)^{-1} Y^T, \quad \text{where } Y = J_k[V_m, d_0]. \tag{2.11}$$

The analysis in [17] shows that the same superlinear convergence properties for the unprojected tensor model considered in [16] also hold for the projected tensor model (2.10). The complete tensor-GMRES algorithm for solving (2.10) at the kth nonlinear iteration is listed in [17].

Despite the algorithm's difficult algebra, the design actually is rather straightforward. The algorithm may be viewed as an extension of Newton-GMRES, where the inexact Newton step is calculated via GMRES in the standard way. The tensor step is calculated subsequently using the Krylov subspace information generated for the Newton step. In this way, the method also is consistent with preconditioning techniques and a matrix-free implementation, which makes it appealing for general use.

The extra work and storage beyond GMRES for computing the tensor step are quite small. The extra work is at most $4mn + 5n + 2m^2 + \mathcal{O}(m)$ multiplications plus a single Jacobian-vector product for evaluating the tensor term a_k. The extra storage amounts to two extra n-vectors for a and s plus a few smaller working vectors of length m.

The results in [17] show the superlinear convergence behavior of tensor-GMRES on three singular and nearly singular problems, where the Newton-GMRES method exhibits linear convergence due to a lack of sufficient first-order information. The margin of improvement (in terms of reduction of nonlinear iterations over Newton's method) varied from 20% to 55% on the simpler problems and from 32% to 60% on the more difficult Euler problem. Running times and the total number of Jacobian-vector products for each method were not reported in [17], but from our own experience with the algorithm, we assume that these performance metrics are correlated with the number of nonlinear iterations.

3. Numerical experiments on ill-conditioned problems.

3.1. Small problems solved with direct methods. This section investigates the performance of direct tensor methods as well as Newton's method on a set of small problems that include a parameter for adjusting the ill-conditioning of the Jacobian matrix at the root. The results show that Newton's method requires increasingly more iterations as the problem ill-conditioning grows, whereas the direct tensor methods are only mildly affected.

Following the approach in [34], we created ill-conditioned problems by modifying nonsingular test problems to be of the form

$$\hat{F}(x, \lambda) = F(x) - \lambda F'(x_*) A (A^T A)^{-1} A^T (x - x_*), \tag{3.1}$$

where $F(x)$ is the standard nonsingular test function, x_* is its root, $A \in \mathbb{R}^{n \times j}$ is an arbitrary matrix that has full column rank with $1 \leq j \leq n$, and $\lambda \in [0, 1]$ is a

parameter for ill-conditioning. We denote $\hat{J}(x,\lambda)$ as the Jacobian matrix of $\hat{F}(x,\lambda)$ with respect to x:

$$\hat{J}(x,\lambda) \equiv \frac{\partial}{\partial x}\hat{F}(x,\lambda).$$

These new test problems are similar to problems in continuation or homotopy methods, except that $\hat{F}(x,\lambda)$ has the same root as $F(x)$ (i.e., x_*) for all values of λ. While the problem becomes harder to solve as λ approaches 1, the idea is not to follow the path over a sequence of values of λ. Rather, we solve the modified problem for each value of λ from the same starting point across all tests and record the number of iterations required to reach the solution.

One special quality of this modified problem is that if $\lambda = 1$ and $F'(x_*)$ has full rank, then the rank of $\hat{J}(x_*,\lambda)$ equals $\operatorname{rank}(F'(x_*)) - \operatorname{rank}(A) = n - \operatorname{rank}(A)$. Thus, as λ approaches 1, the rank deficiency of $\hat{J}(x_*,\lambda)$ approaches the rank of A. Stated another way, a set of the smallest singular values of $\hat{J}(x_*,\lambda)$ equal to the rank of A will approach 0 as λ approaches 1.

Different sets of singular and ill-conditioned problems may be created using the matrix A in (3.1); adding more independent columns to A serves to decrease the rank of $\hat{J}(x_*,1)$. We routinely used the matrices

$$A \in \mathbb{R}^{n\times 1}, \quad A^T = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \end{pmatrix},$$

$$A \in \mathbb{R}^{n\times 2}, \quad A^T = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & -1 & 1 & \dots & \pm 1 \end{pmatrix},$$

and

$$A \in \mathbb{R}^{n\times 3}, \quad A^T = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & -1 & 1 & \dots & \pm 1 \\ 1 & 1 & -1 & \dots & \pm 1, \end{pmatrix}$$

because they provide "balanced" problems by acting equally on the whole Jacobian $F'(x_*)$. Letting A equal the unit vector e_1, for example, would operate only on the $(1,1)$ element of $F'(x)$.

We chose six problems for testing from the standard small-dimensional test set of Moré, Garbow, and Hillstrom [25] and modified them according to (3.1). Table 3.1 lists the problems used in our numerical tests, along with the corresponding dimensions. For all problems, the starting vector x_0 was the standard starting point for each problem published in [25]. Except for the cases mentioned below, these starting points did not require a global strategy to reach the solution (i.e., the full step was accepted at all iterations).

We chose the parameter λ in (3.1) to asymptotically approach $\lambda = 1$ by using the values $\lambda = 1 - 10^{-j}$, $j = 0, 1, 2, \dots$. Thus, for $j = 0$ the problem is the original, unmodified problem, and each subsequent value of j makes the problem more ill-conditioned. At some point, round-off errors in the evaluation of $\hat{F}(x,\lambda)$ as well as the numerical precision of the root x_* make the numerical solution indistinguishable from the case of $\lambda = 1$. All subsequent values of λ would produce the same result, so we collected only the results up to these points and indicated the case of $\lambda = 1$ at the rightmost extent of our plots. Due to the numerical precision of the solutions, the Jacobian at x_* may not be fully singular.

TABLE 3.1
Ill-conditioned test problems.

Problem	Size
Broyden tridiagonal	50
Broyden banded	50
Discrete integral equation	50
Discrete boundary value function	50
Rosenbrock's function	2
Brown almost linear	10

We used the following two stopping conditions in all these tests: $\|F(x_k)\|_\infty \leq 10^{-12}$ and $\|x_k - x_{k-1}\|_\infty \leq 10^{-12}$, which are the same conditions Eisenstat and Walker used when analyzing inexact Newton methods in [15]. In many practical applications, less stringent convergence tolerances are commonly used, but these tight tolerances were used in this experiment (and later experiments) to differentiate results at higher condition numbers and to allow asymptotic convergence behavior to become evident. To help gauge what can be expected with looser stopping tolerances, we have included convergence histories for a few problems (see, e.g., Figure 3.1) where the performance difference is still present but less striking.

As a prelude to these results and to help explain what is happening in these experiments, we first provide a graphical description. Figure 3.1 shows the typical iteration profiles for different values of λ that we observed. This figure graphs the function value at each iteration for the tensor and Newton methods on a typical problem approaching rank $n-1$ with various values of $\lambda = 0, 0.9, 0.99, 0.999, \ldots, 1$. All of the tensor method profiles are bunched together on the left, requiring few iterations even as λ nears 1, whereas the profiles of Newton's method are spread out and require increasingly more iterations for convergence as the problem becomes more ill-conditioned. This plot of iteration profiles is typical for all problems. Thus, while all of the tensor runs display superlinear convergence throughout the iterations, it is evident that Newton's method converges linearly for a number of iterations before accelerating to quadratic convergence. With increasing ill-conditioning, the region of quadratic convergence for Newton's method shrinks in size, acting as if the problem were singular outside of this region.

Figure 3.2 shows the relative performance of Newton's method versus a rank-one tensor method on the six problems, in the case where each Jacobian at the solution approaches rank $n-1$. The condition number $\hat{J}(x_*, \lambda)$ is plotted on the abscissa, while the number of iterations for both methods is plotted on the ordinate axis. One tensor method iteration is just slightly more expensive than a Newton iteration. There were no linesearches in all cases except on Rosenbrock's function at $\lambda = 0$, which explains the unusual spike on the left in that plot. Also, both methods arrived at different solutions in the Brown almost linear problem at $\lambda = 0$, and thus these points were not included in that plot. For all other points, both methods arrived at the same x_*.

The six plots of Figure 3.2 show that as the ill-conditioning of a particular problem grows, Newton's method requires more iterations, whereas the tensor method is very mildly affected. This is a key result. Previously, it was suggested that tensor methods behave reasonably well on ill-conditioned problems, but there was no numerical testing of this conjecture.

However, because the condition number is just a single value representing the ratio of the largest to the smallest magnitude singular values, it does not capture the singular value spectrum of the Jacobian. Say, for instance, that there are multiple

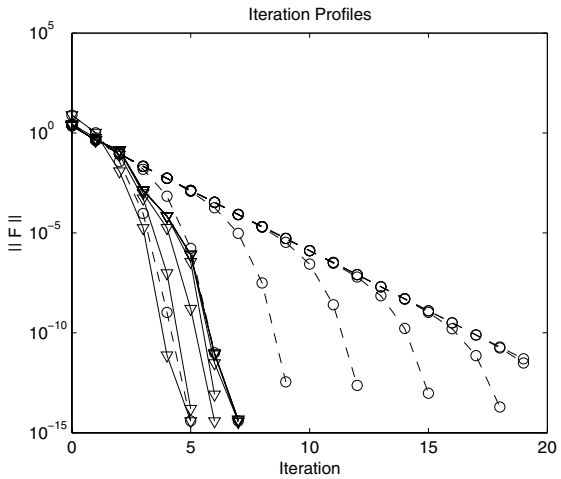


FIG. 3.1. *Superimposed iteration profiles of the tensor method (∇, solid lines) and Newton's method (∘, dashed lines) when solving the modified Broyden tridiagonal function as it approaches rank $n-1$. For both methods and their corresponding set of iteration profiles, problem difficulty (λ) increases from left to right, making the problem more ill-conditioned and requiring more iterations to solve.*

singular values approaching zero, creating a very ill-conditioned matrix. One might speculate that a tensor method using a rank-one tensor term would be less advantageous in this case because it could "handle" only one direction, perhaps the direction associated with the smallest singular value, leaving other (near) singular directions to impede convergence. Figures 3.3 and 3.4 consider exactly these cases where the Jacobians approach rank $n-2$ and $n-3$, respectively, while still using a rank-one tensor model. Two problems were not included in these figures for different reasons. The modified Rosenbrock function had excessive linesearches that obscured the results in the rank $n-2$ problems, and the rank $n-3$ problem is not possible with the Rosenbrock function due to its small dimension of $n=2$. For the Brown almost linear problem, each method arrived at a solution that was different from the root x_* used in (3.1). For these reasons, these two problems were not included in Figures 3.3 and 3.4.

It is evident that the results for higher rank deficiencies are not as striking as in Figure 3.2, but the general trend still remains—tensor methods perform better than Newton's method on ill-conditioned problems, even when the Jacobian approaches a rank deficiency greater than the rank of the tensor term. A few peculiarities exist in the results that warrant explanation. We attribute the "humps" in both curves to the random paths each method takes to arrive at the solution, and we believe this shape is coincidental. The "spikes" in both methods for the two discrete problems in Figure 3.4 are due to linesearches occurring at a specific value of λ. The linesearches force a few more outer iterations to eventually solve the problems.

3.2. Medium-scale problems solved with inexact methods. This subsection investigates the performance of the Newton-GMRES, tensor-GMRES, and tensor-Krylov methods on three ill-conditioned problems of moderate size and complexity: the Bratu problem, a modified discrete boundary value function, and a modified Broyden tridiagonal problem. The tests on the ill-conditioned problems have a design

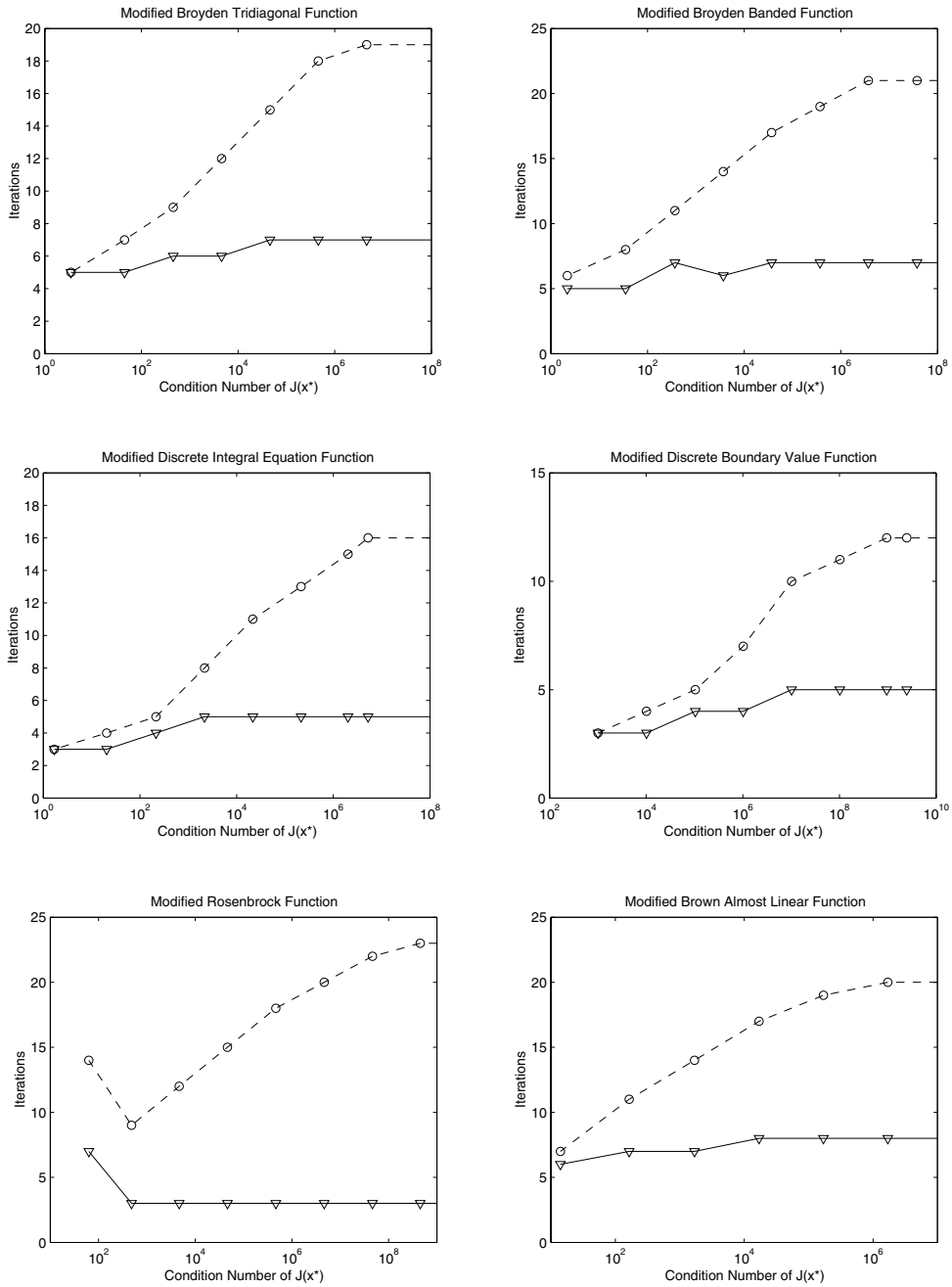


FIG. 3.2. *Comparison of a rank-one tensor method (∇, solid line) with Newton's method (○, dashed line) on variably ill-conditioned test problems that approach rank* $n-1$.

similar to that of the study performed in section 3.1, and we investigate to what extent the benefits of using direct tensor methods for solving small ill-conditioned/singular problems that were shown in section 3.1 extend to inexact versions of tensor methods for larger problems.

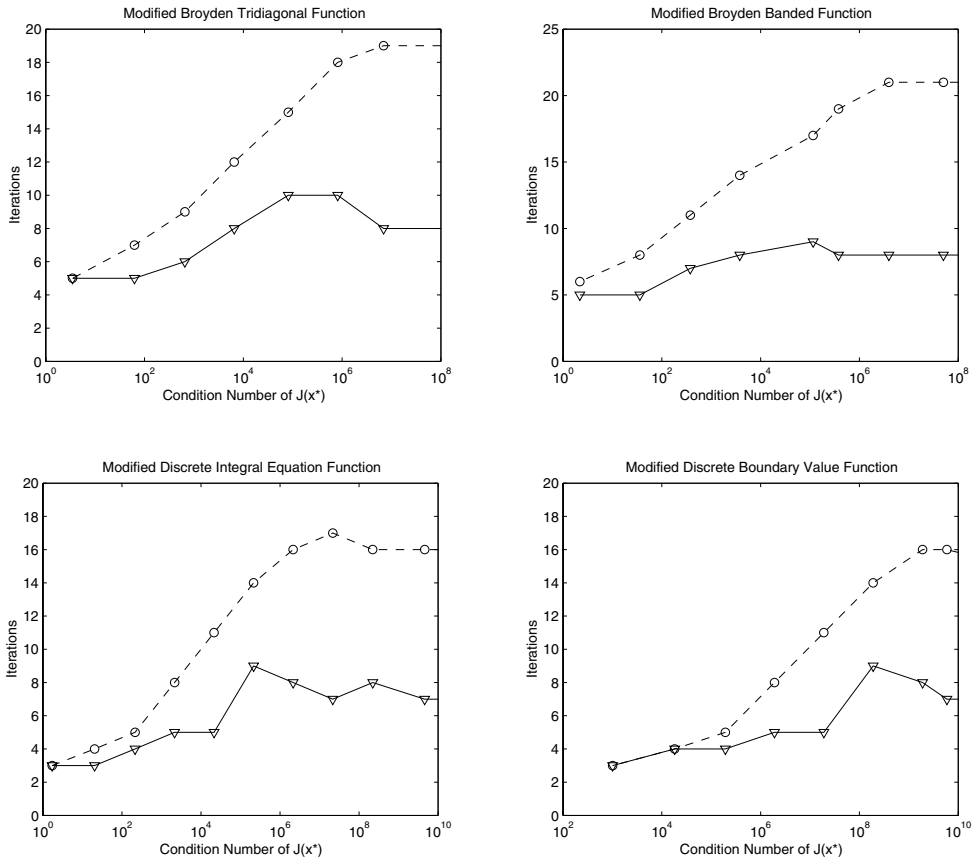


FIG. 3.3. *Comparison of a rank-one tensor method (∇, solid line) with Newton's method (○, dashed line) on variably ill-conditioned test problems that approach rank* $n-2$.

We implemented all of the methods in MATLAB using double precision arithmetic. For objective comparisons, we used the same level of basic linear algebra routines in MATLAB in all of our tests. Thus, the results in this section do not reflect the most efficient implementations that are available, but the statistics that we collected (nonlinear iterations, function evaluations) are invariant to optimal implementations.

For the numerical tests, we used the same stopping conditions and parameters as in section 3.1. For the inner method, we solved the local model to a constant relative tolerance of 10^{-4}, which requires uniformly close approximations to Newton and/or tensor steps at each iterate and results in fast local q-linear convergence. Using a constant relative tolerance may not be ideal in practice (see [15] for alternatives), but it provides for more unbiased comparisons among the methods. Changing this tolerance affects the fastest eventual convergence rate that may be achieved (linear convergence for inexact Newton methods [13]). Specifically, a tighter (or looser) tolerance leads to a faster (slower) convergence rate and does not change the relative performance of these methods with respect to each other. The effect may be seen in the steepest slope that the convergence curve may achieve as the nonlinear iterates near the solution.

In practice, the Krylov subspace dimension should be kept small to reduce arithmetic and storage costs, which may be accomplished via restarted Krylov methods

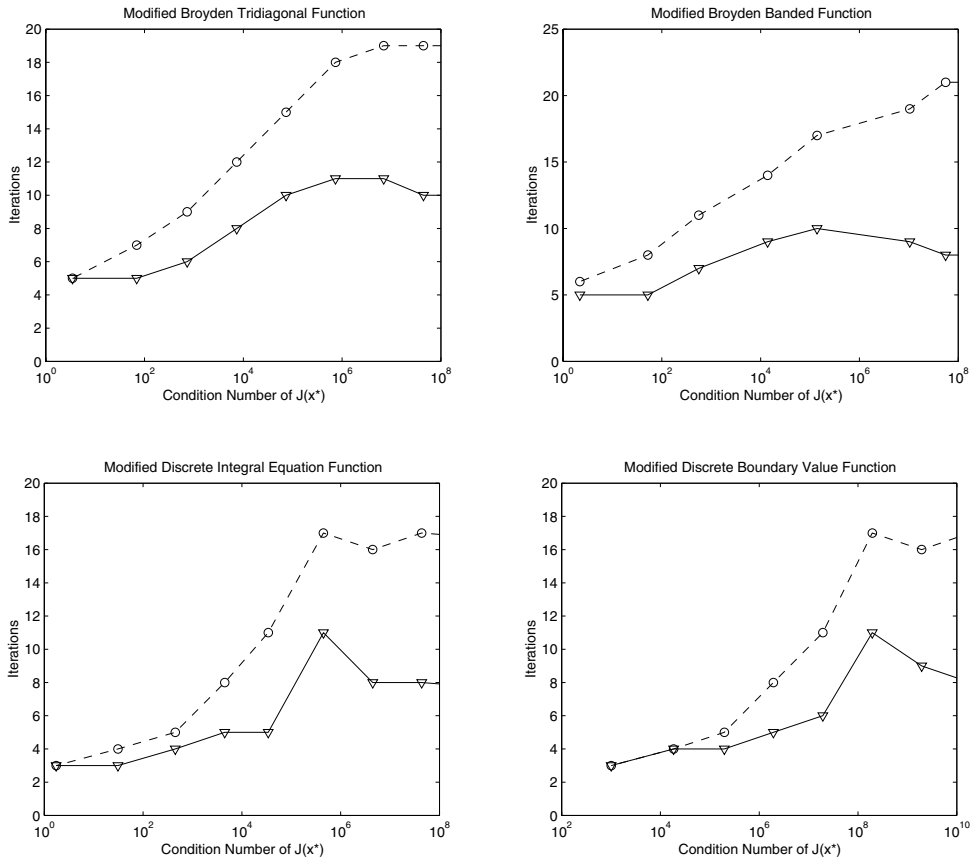


FIG. 3.4. *Comparison of a rank-one tensor method (∇, solid line) with Newton's method (○, dashed line) on variably ill-conditioned test problems that approach rank* $n-3$.

and preconditioning. However, to eliminate any ill effects of small Krylov subspaces preventing convergence in the local model and affecting the outer iterations, the maximum Krylov subspace was set to the problem dimension, $m_{max} = n$, and the solver was not restarted. We used preconditioners that are appropriate for the problem, and they will be discussed with each problem.

In the next three subsections, we present numerical results on the ill-conditioned problems. The results of the tensor-Krylov method TK2 are virtually identical to those of TK2+, so we do not include them here.

3.2.1. Bratu problem. The Bratu problem is a simplified model for nonlinear diffusion phenomena occurring, for example, in semiconductors and combustion, where the source term is related to the Arrhenius law for modeling exothermic reactions. The following version is taken from the set of nonlinear model problems collected by Moré [24]. The problem is the nonlinear PDE in u,

$$-\nabla^2 u = \lambda e^u \text{ in } \Omega, \quad u = 0 \text{ on } \partial\Omega, \tag{3.2}$$

where $\nabla^2 = \sum_{i=1}^{n} \partial^2/\partial x_i^2$ is the Laplace operator, $\lambda \in \mathbb{R}$ is a parameter, Ω is the bounded domain $(0,1) \times (0,1)$, and $\partial\Omega$ is the boundary of Ω.

Problem (3.2) has a unique solution for $\lambda \leq 0$, but for $\lambda > 0$, there may be zero, one, or two solutions (cf. [18]). The critical value $\lambda^* = 6.80812$ is a limit point such that for $0 < \lambda < \lambda^*$, problem (3.2) has two solutions and for $\lambda > \lambda^*$, it has no solutions. Also, the problem at the limit point is singular with a rank of $n - 1$, and as λ approaches the limit point, the discretized problem becomes harder to solve. To investigate the effects of ill-conditioning on the inexact algorithms, we increased λ over the range $\lambda \in [5, 6.806652]$, which increased the condition number of $J(x_*)$ from about 10^3 to 10^6.

When testing the Bratu problem, the initial approximate solution was zero on a uniform grid of size 31×31. The Laplace operator was discretized using centered differences (5-point stencil), and our preconditioner was also the centered differences discretization of the Laplace operator. Note that the preconditioner is not exact due to omitting the forcing term in (3.2). We computed Jacobian-vector products using first-order forward differences. Hence, the number of function evaluations is the sum of the total number of Arnoldi iterations, the number of linesearch backtracks (if any), and the total number of nonlinear iterations. Thus, the number of function evaluations provides a relative measure of overall work for each algorithm.

Figure 3.5 presents the results of these tests, comparing the number of function evaluations computed for each method using the three choices of preconditioning (none, left, and right preconditioning). If one considers the condition number of the preconditioned Jacobian at the root (i.e., $M^{-1}J(x_*)$ or $J(x_*)M^{-1}$), then the condition numbers in the bottom two plots span the range of 3×10^0 to about 10^4 instead of 10^3 to 10^6 for the Jacobian itself. The results in the figure are similar to those in figures in section 3, where Newton-GMRES requires increasingly more function evaluations as the problem becomes more ill-conditioned and difficult. The tensor-Krylov methods required from four to seven outer iterations, whereas Newton-GMRES required from four to 13 outer iterations. That is, Newton-GMRES required almost double the number of tensor-Krylov iterations on the most ill-conditioned problem. The three plots in Figure 3.5 uphold the prior conclusion that increased ill-conditioning only mildly increases the number of outer iterations for tensor methods.

In all cases, when the problem is not overly difficult (e.g., the experiments at a condition number of 10^3 in Figure 3.5), Newton-GMRES is more efficient in terms of nonlinear iterations than the tensor methods. Here, the number of iterations for all of the methods is equal, but GMRES is more efficient at solving the local Newton model, which results in fewer total function evaluations. When the problem is more difficult, however, the tensor methods are superior to Newton-GMRES by a factor of up to 1.9. Approaching the limit point even more closely would have given even more of an advantage to the tensor-Krylov methods.

When comparing only the tensor methods, the tensor-GMRES method of Feng and Pulliam [17] is more efficient than the TK3 and TK2+ methods. It appears that the step produced from the projected tensor model of the Feng–Pulliam method is nearly the same as the steps calculated from the more precise local models of the tensor-Krylov methods, and it is less expensive to compute. This behavior may be understood more clearly by investigating the iteration history of all of the methods, which we describe next.

To investigate the step quality of each tensor-Krylov method, we tested the Bratu problem with $\lambda = 6.806652$, which has a condition number of roughly 1.5×10^6. Using right preconditioning, the results in the first plot of Figure 3.6 show the faster outer (nonlinear) iterations of the three tensor methods, which display very similar performance. All methods start by exhibiting linear convergence until the respective

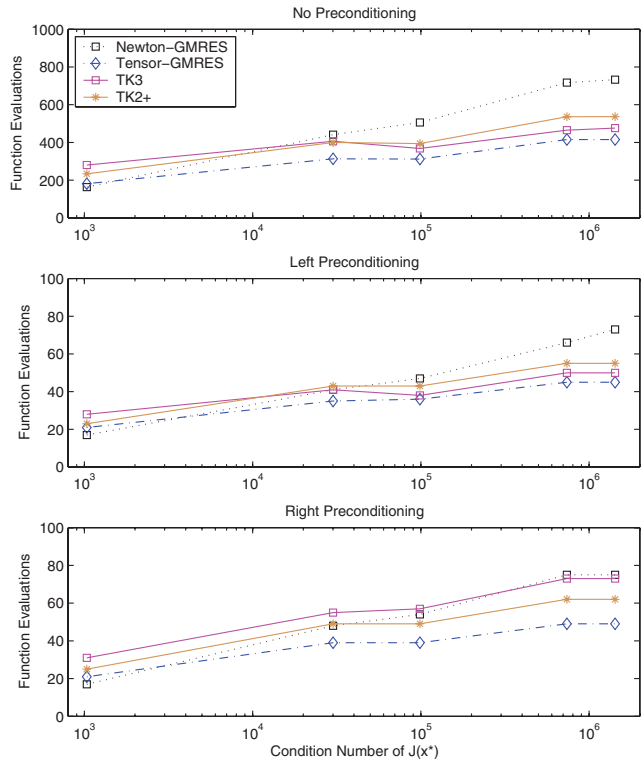


FIG. 3.5. *Effects of ill-conditioning on the inexact algorithms as seen in the Bratu problem.*

method can overcome the near singularity and accelerate convergence. The tensor methods accelerate convergence sooner than Newton-GMRES (iteration 2 versus iteration 5), and this is typical behavior for ill-conditioned problems—Newton-type methods branch into superlinear convergence later and later as the ill-conditioning grows. Because the forcing term for the inner iterative method is constant (instead of decreasing each iteration), all methods exhibit asymptotic linear convergence, as evidenced by their straight trajectories near the solution.

When we consider function evaluations in the bottom plot, the tensor-Krylov methods separate from the Feng–Pulliam method. The block size of the method is a clear indicator of the relative efficiency of the method. Specifically, tensor-GMRES, which uses the scalar (block size one) implementation of GMRES, is more efficient than TK2+ (block size two) and TK3 (block size three). The Bratu problem is unique among our tests in that the steps computed from a projected local tensor model are of roughly the same quality as the steps from TK2+ and TK3. Therefore, the number of outer iterations is the same, but because tensor-GMRES is more efficient in solving the local model, tensor-GMRES has the advantage.

3.2.2. Modified discrete boundary value problem. The discrete boundary value problem is a simple test problem from [25]. The standard discrete boundary value problem is

$$f_i(x) = 2x_i - x_{i-1} - x_{i+1} + \tfrac{1}{2}h^2(x_i + t_i + 1)^3, \quad 1 \le i \le n, \tag{3.3}$$

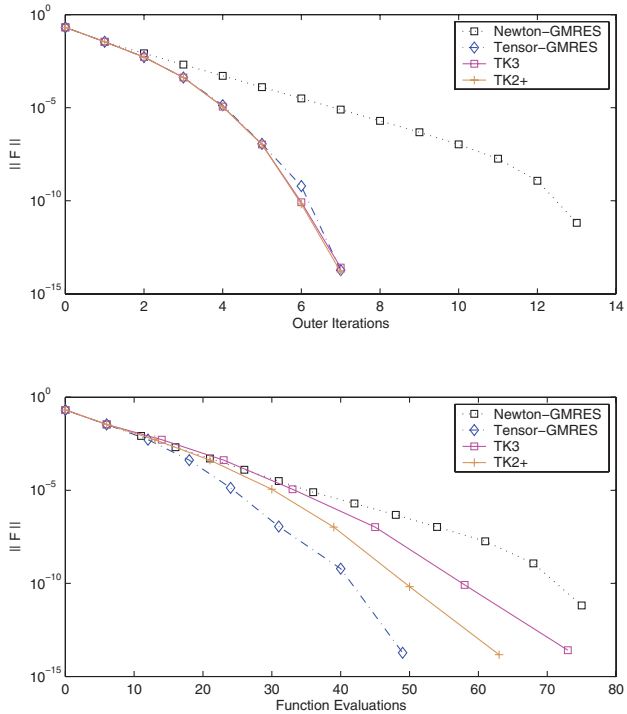


FIG. 3.6. *Example iteration history on the Bratu problem at* $\lambda = 6.806652$.

where $h = \frac{1}{n+1}$, $t_i = ih$, and $x_0 = x_{n+1} = 0$. The initial approximate solution is zero on a problem size of $n = 100$ equations. For our tests, we have modified (3.3) in accordance with (3.1) of section 3.1.

When testing this problem, we used a preconditioner that corresponds to the Jacobian of (3.3) but without the term t_i. We computed Jacobian-vector products using an analytic evaluation of the Jacobian, and we tallied the number of "function evaluation equivalents," which is the sum of function evaluations and Jacobian-vector products. If these products were approximated by first-order forward differences, as is generally the case with complex problems, then "function evaluation equivalents" would be equal to the number of function evaluations. This number also provides a relative measure of overall work for each algorithm.

Figure 3.7 presents the results of these tests, comparing the number of function evaluation equivalents computed for each method using the three choices of preconditioning (none, left, and right preconditioning) for the various values of $\lambda = 1 - 10^{-j}$, $j = 0, 1, 2, \ldots$, in (3.1). If one considers the condition number of the preconditioned Jacobian at the root (i.e., $M^{-1}J(x_*)$ or $J(x_*)M^{-1}$), then the condition numbers in the bottom two plots span the range of 1×10^0 to about 2×10^6 (left preconditioning) or 2×10^8 (right preconditioning) instead of 4×10^3 to 2×10^9 for the Jacobian, not counting the singular case.

The results for the modified discrete boundary value problem are similar to those of the Bratu problem in Figure 3.5: Newton-GMRES requires increasingly more function evaluations as the problem becomes more ill-conditioned, whereas the number of function evaluations required by the tensor methods increases to a lesser extent. It

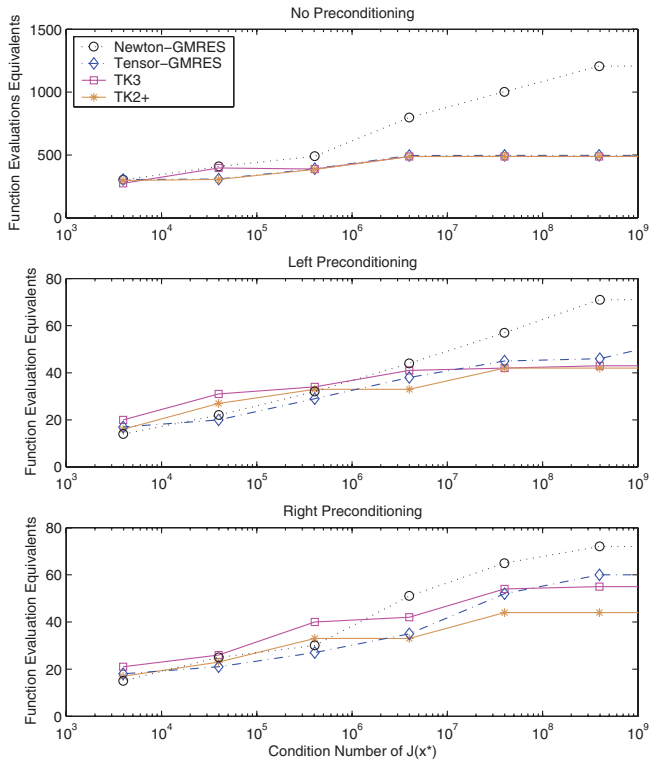


FIG. 3.7. *Effects of ill-conditioning on the inexact algorithms as seen in the discrete boundary value problem.*

is interesting to note that the tensor methods are virtually identical in performance without preconditioning and resemble the results in section 3.1. However, once preconditioning is used, the tensor methods are affected by the ill-conditioning of the problem, yet still to a lesser extent than Newton-GMRES.

When comparing only the tensor methods while using left and right preconditioning, the tensor-GMRES method of Feng and Pulliam is more efficient than the TK3 and TK2+ methods on the easier problems but less efficient on the hardest problems, which is different from the experience with the Bratu problem above. We explain this relative difference by noting that tensor-GMRES required more nonlinear iterations as the problem grew more ill-conditioned. Because the tensor-Krylov methods generally require more Arnoldi (inner) iterations to solve the local tensor model due to the less efficient block-Arnoldi process, any computational savings must come from fewer nonlinear (outer) iterations.

To better illustrate this behavior, Figure 3.8 presents the iteration history of all methods for one test case in Figure 3.7. We present the results of the modified problem using right preconditioning with $\lambda = 1$, which corresponds to the rightmost set of points in the bottom plot of Figure 3.7. Here, the top plot of Figure 3.8 shows that Newton-GMRES has linear convergence, and after one good step on the second iteration, tensor-GMRES appears to have linear convergence that is slightly faster than that of Newton-GMRES. The TK2+ and TK3 methods appear to have superlinear convergence with nearly the same steps. This profile indicates that the

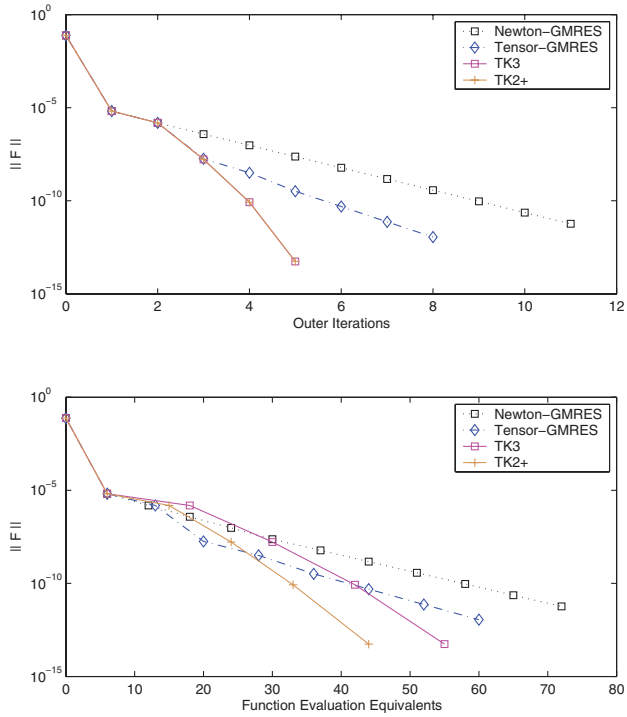


FIG. 3.8. *Example iteration history on the discrete boundary value problem at* $\lambda = 1$.

tensor step found from a projected tensor model is not as good for this problem as the tensor-Krylov steps which use the full tensor model. That is, the projection of the tensor model loses some information that is important for achieving superlinear convergence.

When we consider function evaluations in the bottom plot of Figure 3.8, two features are evident. First, the slopes of the Newton-GMRES and tensor-GMRES lines are nearly identical, which indicates that the cost of solving the projected tensor model in terms of function evaluations is roughly the same as the cost of solving the Newton model in Newton-GMRES. Second, the TK2+ and TK3 trajectories separate, which indicates that the block-2+ local solver in TK2+ is more efficient than the block-3 solver in TK3.

3.2.3. Modified Broyden tridiagonal problem. The Broyden tridiagonal problem is another test problem from [25]. The function is defined as

$$f_i(x) = (3 - 2x_i)x_i - x_{i-1} - 2x_{i+1} + 1, \quad 1 \le i \le n, \tag{3.4}$$

where $x_0 = x_{n+1} = 0$. Here again, we have modified the problem to be ill-conditioned, but in this case, we modify the last function to be

$$f_n(x) = (1 - \lambda)\left[(3 - 2x_n)x_n - x_{n-1} + 1\right] + \lambda\left[(3 - 2x_n)x_n - x_{n-1} + 1\right]^2.$$

This modification is similar to the problem studied by Feng and Pulliam in [17] and makes the problem more ill-conditioned as λ asymptotically approaches one, and singular at $\lambda = 1$ with a rank $n - 1$. We used the values $\lambda = 1 - 10^{-j}$, $j = 0, 1, 2, \ldots,$

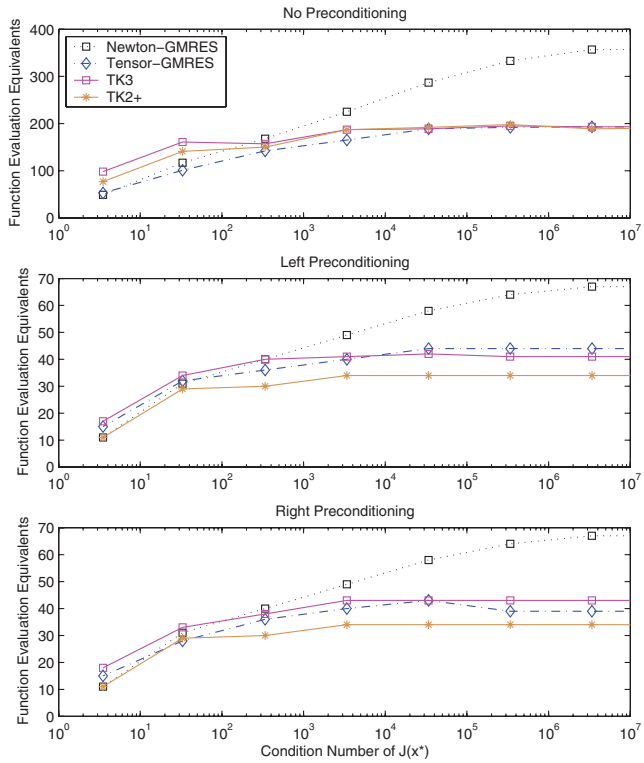


FIG. 3.9. *Effects of ill-conditioning on the inexact algorithms as seen in the modified Broyden tridiagonal problem.*

until the results were indistinguishable from $\lambda = 1$ due to round-off error, at which point we just use $\lambda = 1$. For our tests, we set $n = 100$ and used $(-1, \ldots, -1)^T$ as the starting vector. We used the Jacobian of (3.4) as our preconditioner for all values of λ.

Figure 3.9 presents the results of the ill-conditioning tests over the span of problems using $\lambda \in [0, 1]$, which affects the condition number of $J(x_*)$. The three plots represent the three choices of preconditioning (none, left, and right preconditioning), and each compares the number of function evaluation equivalents computed for the different methods. If one considers the condition number of the preconditioned Jacobian at the root (i.e., $M^{-1}J(x_*)$ or $J(x_*)M^{-1}$), then the condition numbers in the bottom two plots (not counting the singular case) span the range of 1×10^0 to 2×10^6 instead of 3×10^0 to 3×10^6 for the Jacobian.

Here again the results show that Newton-GMRES requires increasingly more function evaluations as the problem becomes more ill-conditioned. The tensor methods are affected somewhat at the low condition numbers, but there is a plateau where the number of function evaluations required by the tensor methods are no longer affected by the ill-conditioning of the problem. Near $\lambda = 1$, Newton-GMRES requires almost twice the number of function evaluations as the tensor methods.

When comparing only the tensor methods while using left and right preconditioning, the tensor-GMRES method of Feng and Pulliam is roughly comparable to TK3. The TK2+ method is more efficient on all problems except one, where tensor-GMRES is the best. In comparison with Newton-GMRES on the easier problems with left or

right preconditioning, TK2+ requires the same number of or fewer function evaluation equivalents.

Figure 3.10 presents the iteration history of all methods for the rightmost test case ($\lambda = 1$) in the right preconditioning plot in Figure 3.9. The top plot of Figure 3.10 shows that Newton-GMRES has linear convergence on this singular problem, and the tensor methods have superlinear convergence. Once again, the two tensor-Krylov methods have the same quality of steps, but the tensor-GMRES method no longer shares the same trajectory as TK2+ and TK3, which indicates that the projected tensor model loses some critical directional information.

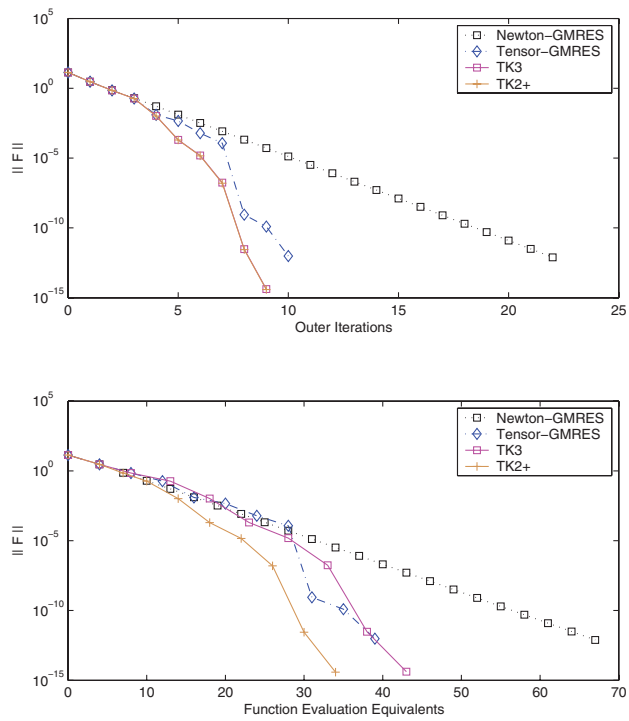


FIG. 3.10. *Example iteration history on the modified Broyden tridiagonal problem at* $\lambda = 1$.

The bottom plot of Figure 3.10 considers the function evaluation equivalents on the x-axis. It shows that tensor-GMRES is generally more efficient than TK3 at solving the local model. Thus, while TK3 is more efficient in total nonlinear iterations, tensor-GMRES is more efficient at solving the local model, which accounts for the difference. Another feature evident in the bottom plot is that the TK2+ and TK3 trajectories separate. As with the discrete boundary value problem above, this indicates that the block-2+ local solver in TK2+ is more efficient than the block-3 solver in TK3.

3.3. Large-scale problem solved with inexact methods. For our largest, most difficult problem, we consider the counterflow jet problem described in [27]. This fluid flow problem offers steady-state multiplicity and exhibits a pitchfork bifurcation, where the Jacobian is singular (rank $n - 1$).

The two-dimensional planar counterflow jet problem consists of a rectangular channel with two opposing inlet flows on the y-axis and two outlet flows on the x-axis. This fluid flow problem is set up using a particular spatial discretization of the continuity and Navier–Stokes equations describing laminar, isothermal flow of an incompressible Newtonian fluid. The particular spatial discretization that we use is from the finite element reacting flow code MPSalsa [33].

The finite-element discretization utilized 6,321 bilinear elements corresponding to 18,963 unknowns and had a fixed aspect ratio of 1.0. For reasons related to coupling external solvers to MPSalsa, the aspect ratio was constrained to the geometry in the original mesh file and could not be scaled with a parameter to achieve different aspect ratios as was done in [27]. Consequently, not all experimental conditions are the same.

As the Reynolds number of the injected fluid changes, multiple steady states may exist. The point at which a single steady state changes to multiple steady states is a bifurcation. We used the Library of Continuation Algorithms (LOCA) [32, 31] to identify a pitchfork bifurcation at $Re^* = 872.20306885$ for our particular geometry and mesh. To investigate the effects of ill-conditioning on the methods, we increased Re from 100 to Re^*, which increased the condition number of $J(x_*)$ from about 10^4 to 10^{15}. Differences in performance among the algorithms were not seen until $Re = 872.19$.

We implemented the algorithms in NOX [23], which is a C++ object-oriented nonlinear solver package developed at Sandia National Laboratories. For objective comparisons, all of the methods, including Newton-GMRES and tensor-GMRES, used the same Arnoldi process (modified Gram–Schmidt) as the tensor-Krylov methods.

We used an explicit Jacobian, which our PDE code computed efficiently by a combination of analytic evaluation and numerical differentiation, and enabled the option for maximum accuracy in the Jacobian. We employed right preconditioning in all cases using an ILUT preconditioner [29].

We used a standard backtracking linesearch procedure for Newton-GMRES and used the complete tensor-GMRES algorithm in [17], including their globalization. For the tensor-Krylov methods, we report results of the standard linesearch, but the curvilinear linesearch [3] provided similar performance. For selecting the linesearch parameter at each trial step, we used interval halving (dividing the search direction in half at each inner iteration).

To get the Jacobian as close to singular as possible, the problem was solved as accurately as possible. We used the following stopping condition: $\|F(x_k)\|_2 \leq 10^{-15} \|F(x_0)\|_2$. In all other respects, we used the same conditions and parameters as before. The initial approximation was the zero vector for all cases.

Figure 3.11 shows that the tensor-Krylov methods require 6–10 iterations to solve the range of problems. On the other hand, Newton-GMRES requires nearly twice the iterations at the pitchfork bifurcation, and tensor-GMRES failed to solve that problem. All methods had identical performance until very near the pitchfork bifurcation as the condition exceeded 10^{10}. TK2+ required four linesearch iterations at 10^{11}, hence the higher number of function evaluations.

Figure 3.12 shows the convergence history for all methods at the pitchfork bifurcation. The tensor-Krylov methods share roughly the same level of performance. Tensor-GMRES, on the other hand, stalls after about 12 orders of magnitude improvement in $\|F\|$. This is not due to excessive linesearches because in most instances a full step is taken; tensor-GMRES just loses directional information here. Newton-GMRES behaves as we have seen previously with smaller problems—roughly linear

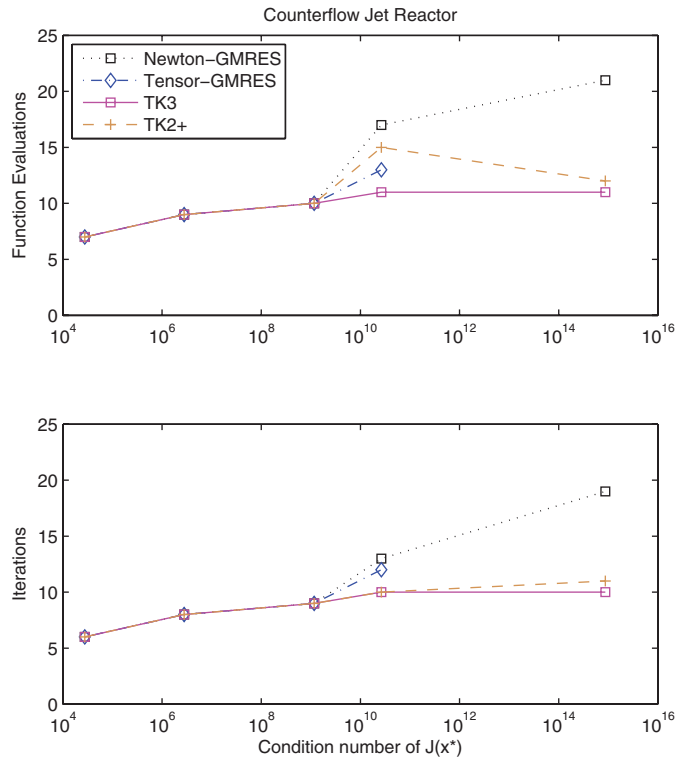


FIG. 3.11. *Effects of ill-conditioning in the counterflow jet reactor problem.*

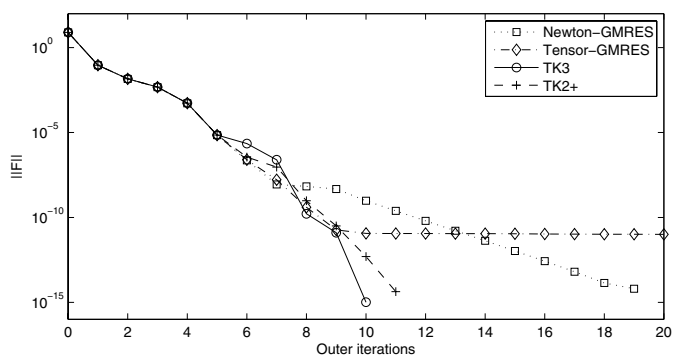


FIG. 3.12. *Iteration history of the counterflow jet reactor problem at the pitchfork bifurcation.*

convergence that is slower than that of the tensor methods (tensor-GMRES notwithstanding).

4. Summary and conclusions. This paper has investigated the performance of tensor methods (both direct and Krylov-based inner solvers) on small- to large-scale problems over a range of conditionings, from well-conditioned to ill-conditioned to singular. Our results showed that tensor methods outperform Newton-based methods as the problems become more ill-conditioned. Prior to this investigation, studies on

direct tensor methods focused only on singular problems or on general problems that are well-conditioned.

Specifically, our results show that, eventual quadratic convergence notwithstanding, the performance of Newton's method will degrade as the ill-conditioning grows, whereas tensor methods appear to be relatively unaffected or only mildly affected (in the case of larger rank deficiencies). Newton-based methods do not handle these singular problems well because they converge linearly to the solution and, in some cases, with poor accuracy.

For the large-scale methods, despite the use of an iterative inner method with an approximate solve, the tensor-Krylov methods appear to retain superlinear convergence properties on ill-conditioned problems. On the other hand, Newton-GMRES is affected by the ill-conditioning and branches into superlinear convergence later and later as the problems become more ill-conditioned. Thus, tensor methods are especially useful for large-scale problems that are highly ill-conditioned or singular, where Newton-based algorithms exhibit very slow convergence.

In terms of the various tensor methods, TK3 and TK2+ generally have the best performance in terms of nonlinear iterations. When considering total function evaluations, then TK2 and TK2+ outperform TK3. Feng and Pulliam's tensor-GMRES method is very competitive but could not solve the large-scale problem at the bifurcation point. A thorough comparison of these methods on other fluid flow problems is in [2].

There are many important and practical problems that have ill-conditioned or singular Jacobian matrices at the solution, such as large PDE problems that exhibit shocks, turning points, and/or pitchfork bifurcations. The conclusions of this research indicate that concepts from tensor methods may benefit algorithms for bifurcation tracking and stability analysis. We intend to investigate these applications in future research.

Acknowledgments. We thank Roger Pawlowski for the counterflowing jet problem in MPSalsa, and we are grateful to Eric Phipps for assistance with finding the pitchfork bifurcation using LOCA. We also thank the two anonymous referees for carefully reading the manuscript and suggesting important improvements.

REFERENCES

[1] B. W. Bader, *Tensor-Krylov Methods for Solving Large-Scale Systems of Nonlinear Equations*, Ph.D. thesis, Department of Computer Science, University of Colorado, Boulder, 2003.

[2] B. W. Bader, *Tensor-Krylov methods for solving large-scale systems of nonlinear equations*, SIAM J. Numer. Anal., 43 (2005), pp. 1321–1347.

[3] B. W. Bader and R. B. Schnabel, *Curvilinear linesearch for tensor methods*, SIAM J. Sci. Comput., 25 (2003), pp. 604–622.

[4] A. Bouaricha, *Solving Large Sparse Systems of Nonlinear Equations and Nonlinear Least Squares Problems Using Tensor Methods on Sequential and Parallel Computers*, Ph.D. thesis, Department of Computer Science, University of Colorado, Boulder, 1992.

[5] P. N. Brown and A. C. Hindmarsh, *Reduced storage methods in stiff ODE systems*, J. Appl. Math. Comput., 31 (1989), pp. 40–91.

[6] P. N. Brown and Y. Saad, *Hybrid Krylov methods for nonlinear systems of equations*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 450–481.

[7] P. N. Brown and Y. Saad, *Convergence theory of nonlinear Newton–Krylov algorithms*, SIAM J. Optim., 4 (1994), pp. 297–330.

[8] T. F. Chan and K. R. Jackson, *The use of iterative linear-equation solvers in codes for large systems of stiff IVPs for ODEs*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 378–417.

[9] D. W. Decker, H. B. Keller, and C. T. Kelley, *Convergence rates for Newton's method at singular points*, SIAM J. Numer. Anal., 20 (1983), pp. 296–314.

[10] D. W. Decker and C. T. Kelley, *Newton's method at singular points.* I, SIAM J. Numer. Anal., 17 (1980), pp. 66–70.
[11] D. W. Decker and C. T. Kelley, *Newton's method at singular points.* II, SIAM J. Numer. Anal., 17 (1980), pp. 465–471.
[12] D. W. Decker and C. T. Kelley, *Convergence acceleration for Newton's method at singular points*, SIAM J. Numer. Anal., 19 (1981), pp. 219–229.
[13] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *Inexact Newton methods*, SIAM J. Numer. Anal., 19 (1982), pp. 400–408.
[14] J. E. Dennis, Jr., and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1983.
[15] S. C. Eisenstat and H. F. Walker, *Choosing the forcing terms in an inexact Newton method*, SIAM J. Sci. Comput., 17 (1996), pp. 16–32.
[16] D. Feng, P. D. Frank, and R. B. Schnabel, *Local convergence analysis of tensor methods for nonlinear equations*, Math. Programming, 62 (1993), pp. 427–459.
[17] D. Feng and T. H. Pulliam, *Tensor-GMRES method for large systems of nonlinear equations*, SIAM J. Optim., 7 (1997), pp. 757–779.
[18] R. Glowinski, H. B. Keller, and L. Reinhart, *Continuation-conjugate gradient methods for the least squares solution of nonlinear boundary value problems*, SIAM J. Sci. Statist. Comput., 6 (1985), pp. 793–832.
[19] A. Griewank, *On solving nonlinear equations with simple singularities or nearly singular solutions*, SIAM Review, 27 (1985), pp. 537–563.
[20] A. Griewank and M. R. Osborne, *Analysis of Newton's method at irregular singularities*, SIAM J. Numer. Anal., 20 (1983), pp. 747–773.
[21] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, SIAM, Philadelphia, 1995.
[22] C. T. Kelley and R. Suresh, *A new acceleration method for Newton's method at singular points*, SIAM J. Numer. Anal., 20 (1983), pp. 1001–1009.
[23] T. Kolda and R. Pawlowski, *NOX: An Object-Oriented Nonlinear Solver Package*, http://trilinos.sandia.gov/packages/nox/ (24 May 2007).
[24] J. J. Moré, *A collection of nonlinear model problems*, in Computational Solution of Nonlinear Systems of Equations, Lectures in Appl. Math. 26, AMS, Providence, RI, 1990, pp. 723–762.
[25] J. J. Moré, B. S. Garbow, and K. E. Hillstrom, *Testing unconstrained optimization software*, ACM Trans. Math. Software, 7 (1981), pp. 17–41.
[26] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer-Verlag, New York, 1999.
[27] R. P. Pawlowski, A. G. Salinger, J. N. Shadid, and T. J. Mountziaris, *Bifurcation and stability analysis of laminar isothermal counterflowing jets*, J. Fluid Mech., 551 (2006), pp. 117–139.
[28] G. W. Reddien, *On Newton's method for singular problems*, SIAM J. Numer. Anal., 15 (1978), pp. 993–996.
[29] Y. Saad, *ILUT: A dual threshold incomplete ILU preconditioner*, Numer. Linear Algebra Appl., 1 (1994), pp. 387–402.
[30] Y. Saad, *Analysis of augmented Krylov subspace methods*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 435–449.
[31] A. G. Salinger, N. M. Bou-Rabee, R. P. Pawlowski, E. T. Phipps, and L. A. Romero, *Bifurcation tracking algorithms and software for large scale applications*, Internat. J. Bifur. Chaos Appl. Sci. Engrg., 15 (2005), pp. 1015–1032.
[32] A. G. Salinger, N. M. Bou-Rabee, R. P. Pawlowski, E. D. Wilkes, E. A. Burroughs, R. B. Lehoucq, and L. A. Romero, *LOCA* 1.0*: Library of Continuation Algorithms: Theory and Implemenation Manual*, Technical report SAND2002-0396, Sandia National Laboratories, Albuquerque, NM, 2002.
[33] A. G. Salinger, K. D. Devine, G. L. Hennigan, H. K. Moffat, S. A. Hutchinson, and J. N. Shadid, *MPSalsa, A Finite Element Computer Program for Reacting Flow Problems, Part* 2*—User's Guide*, Technical report SAND96-2331, Sandia National Laboratories, Albuquerque, NM, 1996.
[34] R. B. Schnabel and P. D. Frank, *Tensor methods for nonlinear equations*, SIAM J. Numer. Anal., 21 (1984), pp. 815–843.

SIAM J. SCI. COMPUT.
Vol. 29, No. 6, pp. 2352–2380
© 2007 Society for Industrial and Applied Mathematics

NUMERICAL SIMULATION OF ACOUSTIC STREAMING ON SURFACE ACOUSTIC WAVE-DRIVEN BIOCHIPS*

DANIEL KÖSTER†

Abstract. A novel type of microfluidic biochip employs acoustic surface waves as a pumping mechanism for fluids. Piezoelectric substrate materials in the chip efficiently convert an electromagnetic input signal into an elastic wave confined to the surface layer of the substrate, hence the surface acoustic wave. The interaction of these waves with an adjoining fluid volume leads to streaming patterns in the fluid or of the motion of the fluid as a whole. In this work we study this acoustic streaming effect. We present a two-scale mathematical model, some theoretical results on the well-posedness of the model problem, as well as numerical simulations based on the finite element method. The model will include the case of free capillary fluid boundaries.

Key words. scientific computing, modeling, numerical simulation, acoustic streaming, finite elements, numerical analysis, biochips, surface acoustic waves, lab-on-a-chip

AMS subject classifications. 74F10, 65M60, 65Z05

DOI. 10.1137/060676623

1. Introduction.

The past decades have seen unprecedented advances in the production of miniaturized electronic circuits, from the leviathan computers filling entire rooms in the humble beginnings to the powerful and omnipresent microchips of today. A similar revolution under the motto "smaller is better" is currently taking place in the life sciences regarding the manipulation of fluids, with the transition from test tubes and beakers to micropipettes and microtitre plates.

In contrast to the field of microelectronics, where the aim is still to reduce the size of transistors, in microfluidics the main concern nowadays is to manufacture increasingly complex systems of channels with sophisticated pumps, valves, mixers, separators, and filters, all on a scale of micrometers. A popular concept today is known as *lab-on-a-chip*. The goal of a lab-on-a-chip is the integration of a complete analysis and reaction system on a small surface (≈ 1 cm^2 or less). A sample device is shown in Figure 1.1.

There are many reasons for desiring a small scale. Less energy and lesser amounts of expensive reagents are needed, reducing costs. Devices can be used in parallel and with automatic control, useful in applications such as gene sequencing. From the physical point of view we have more efficient heat and material transport, leading to better yields and faster analysis.

Despite all of these promising aspects there are certain technical difficulties arising in the production of microfluidic components. Specifically, the task of performing controlled pumping or mixing of nano- or picoliter quantities of fluids becomes difficult. For instance, the low Reynolds number of typical flows on this scale signifies that flows in this regime are laminar and not turbulent. This is an obstacle in designing a component to mix reagents.

*Received by the editors December 4, 2006; accepted for publication (in revised form) May 14, 2007; published electronically October 5, 2007. This work was partially supported by the German National Science Foundation (DFG) within the Priority Program SPP 1253.
http://www.siam.org/journals/sisc/29-6/67662.html
†Numerical Analysis and Computational Mechanics Group, University of Twente, Faculty EEMCS, Department of Applied Mathematics, P.O. Box 217, 7500 AE Enschede, The Netherlands (kosterdtp@ewi.utwente.nl).

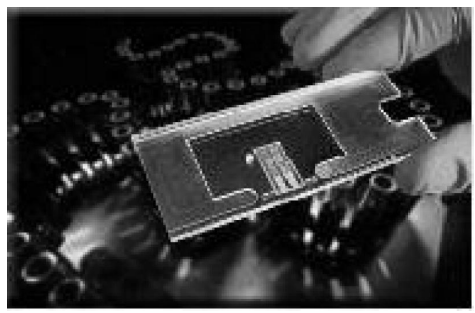


FIG. 1.1. *Example of a biochip.*

A novel type of microfluidic biochip employs surface waves as a driving force. The key of this technology is a pump able to position reagents on the surface of chips or in microfluidic channels without mechanical contact. This is achieved using surface acoustic waves induced using radio frequency electric signals. The waves arise through the use of piezoelectric substrate materials in the chip, e.g., lithium niobate. The electromagnetic signal is efficiently converted into an elastic wave confined to the surface layer of the substrate, the hence surface acoustic wave (SAW).

The interaction of these waves with the fluid leads to streaming patterns in the fluid or of the motion of the fluid as a whole. It is thus possible to mix minute amounts of fluid or conduct chemical reactions through precisely controlled transport of reagents. See, e.g., [48, 21] for further details. Microfluidic pumps based on this technology are already available on the market; see [49].

In this work we will use the compressible Navier–Stokes equations as a starting point to describe fluid flow. This is justified since the typical value of the Knudsen number, defined as the ratio of the characteristic physical length scale to the molecular mean free path length, is much less than 1. Due to the presence of widely different time scales in the physical problem we will derive further model equations to describe the fluid behavior.

The resulting two subproblems describe fluid flow at a microscopic time scale, where acoustic effects are dominant, and at a macroscopic time scale, where the streaming effects are observable. The solution of the microscopic model is necessary to calculate the effective force terms in the macroscopic model. Our model fits into the formal framework of *heterogeneous multiscale methods* as described in [14, 15]. This general approach attempts to formalize the idea of performing microscale simulations to close macroscale models.

Furthermore, we will consider the case of free capillary fluid boundaries. This problem is of interest because some biochip models do not employ closed fluid channels. Instead these chips are constructed with a virtual channel geometry defined by planar regions with different wetting properties on the chip surface. Fluids adhere to the hydrophilic regions by surface tension and are thus contained; see [49]. Another use of free capillary boundaries is for the passive filling of small cavities by surface tension; see [10].

1.1. Prior work and aims of the paper. The mathematical modeling and numerical simulation of the interaction between SAWs and surrounding fluids is a mostly unexplored field. This is mainly due to the novelty of the applied technology. The book [24] may serve as a recent overview of small-scale fluid-streaming phenomena.

Many numerical studies of acoustic streaming effects are already available. However, most of these use the older streaming force approach by Nyborg [32] as a model. As an example we cite [40], where the authors simulated acoustic streaming in microfluidic channels induced by disk actuators.

Another work of interest is [5] since SAWs are also employed there and simulated with true fluid-structure interaction. However, the focus of that work is the simulation of first order acoustic effects and not of acoustic streaming. Hence, the problem of different time scales does not appear there.

The model used in this work was introduced by Bradley [6]. It provides a rigorous treatment of boundary conditions often neglected in other works. As far as we know, the present work contains the first numerical simulation based on this improved model. In this work we have extended this model to include the effect of free capillary fluid boundaries which complicates the numerical algorithm considerably. We note that relatively few works exist which treat the simulation of *compressible* flows with free capillary surfaces.

The main contribution of this paper is a viable numerical algorithm to predict the deformation of free droplet shapes under the influence of acoustic streaming. The numerical results suggest that the method possesses good stability properties, although much numerical effort is currently invested for solving the microscale subproblem. Alternative approaches for the microscale subproblem are current work.

1.2. Outline of this work. We start by presenting the physical fundamentals in section 2. The classical Navier–Stokes equations of hydrodynamics serve as a basis to develop the numerical model. The model consists of two separate systems of partial differential equations to describe the flow variables at pressure p and velocity $\boldsymbol{v}$ on two different time scales.

Section 3 presents an analysis of the central mathematical problems arising in the model. The problem can be characterized as a generalized instationary, compressible Stokes problem, supplied with initial and boundary conditions. We study several different solution concepts: general instationary solutions, periodic solutions, and time-harmonic solutions. We define suitable function spaces as well as the used concept of weak solutions within these spaces.

We will prove the existence and uniqueness of weak solutions, as well as continuous dependence on the problem data. We will additionally introduce a result demonstrating the relationship between the different solution concepts.

Section 4 will introduce the extension of the prior problems to problems incorporating free capillary boundaries. Due to the complexity of the problem we restrict ourselves to the study of appropriate numerical algorithms.

Section 5 presents an overview of the practical algorithms used to solve the discretized problems. We have used standard Lagrange-type conforming triangular and tetrahedral finite elements.

The final section presents results of numerical experiments. Academic test problems are formulated which serve to validate the software implementation. Problems with realistic physical parameters are also defined which demonstrate the applicability to real-life situations.

1.3. Notation and function spaces. In what follows we will denote by Ω a bounded domain in $\mathbb{R}^d$. We will use standard notation from Lebesgue and Sobolev space theory.

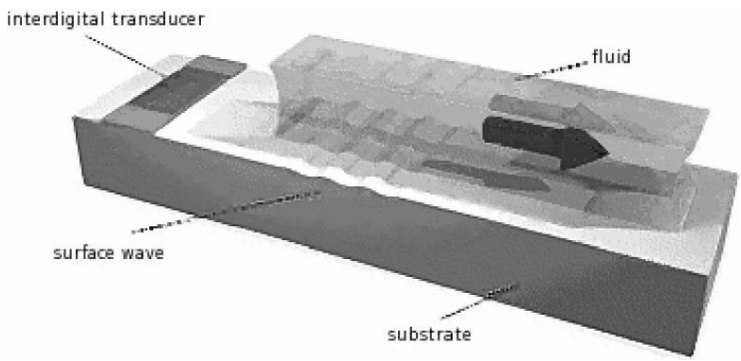


FIG. 2.1. *Working principles of a SAW-driven fluidic device.*

2. Derivation of the model. The biochips treated in this work contain a substrate layer consisting of a piezoelectric material, e.g., lithium niobate. Embedded on the substrate surface are fine electrodes known as *interdigital transducers* (IDTs) due to their comblike structure. When a radio frequency signal (typical frequencies are 100 MHz) is applied to the IDTs, the inverse piezoelectric effect causes a distortion of the substrate layer. Using special crystal cuts one obtains coupled elastic-electromagnetic waves in the substrate.

The chips used for manipulating fluid are designed to favor a special type of wave known as SAWs, also called *Rayleigh waves*. These waves spread out along the surface of the substrate in the fashion of a miniature earthquake. The wavelength is defined by the IDT geometry, and typical amplitudes of displacement are in the nanometer regime. Rayleigh waves are restricted to a thin layer of several wavelengths on the surface of the chip.

If the surface of the substrate is in contact with fluid and if the normal component of displacement is nonzero, part of the energy will be radiated into the fluid. The thus damped SAW is known as a *leaky SAW* (LSAW). The interaction between the LSAW and the fluid causes an internal streaming effect as explained below; see Figure 2.1 for an illustration. This streaming can be used to mix the fluid or as a contact-free pump. It is even possible [49] to define a virtual fluidic network on the chip surface using hydrophobic and/or hydrophilic materials etched onto the substrate using photolithographic techniques. Here we are interested in the effect of the streaming on free capillary boundaries.

Our approach is based on a splitting into two subproblems, with coupling being either neglected or treated empirically. The analysis and numerical simulation of the elastic/electromagnetic subproblem describing the creation of SAWs was treated in [17]. In this work we will focus on the coupling of LSAWs with the fluid. This interaction will be modeled in only one direction. To be precise, we use an empiric LSAW displacement as the input parameter for the streaming effect and neglect the backward coupling of the fluid on the elastic substrate.

2.1. Overview of the fluidics problem. For the modeling of the fluid flow we use the compressible Navier–Stokes equations; see [27, 6]. Assume a bounded fluid domain $\Omega(t)$ with the boundary $\Gamma_d(t) \subseteq \partial\Omega(t)$ in contact with elastic walls partially

traversed by SAWs.

$$
\begin{aligned}
\rho\left(\frac{\partial \boldsymbol{v}}{\partial t} + (\nabla \boldsymbol{v})\boldsymbol{v}\right) &= \nabla \cdot \Sigma && \text{in } \Omega(t),\ t > 0,\\
\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{v}) &= 0 && \text{in } \Omega(t),\ t > 0,\\
\boldsymbol{v}(t, \boldsymbol{x} + \boldsymbol{u}(t, \boldsymbol{x})) &= \frac{\partial \boldsymbol{u}}{\partial t}(t, \boldsymbol{x}) && \text{on } \Gamma_d(t),\ t > 0,\\
\boldsymbol{v} = 0, \quad p = p_0, \quad \Omega(0) &= \Omega_0 && \text{for } t = 0,
\end{aligned}
\tag{2.1}
$$

with the Newtonian stress tensor $\Sigma = \Sigma(\boldsymbol{v}, p)$ defined as

$$
\Sigma_{ij}(\boldsymbol{v}, p) = -p\delta_{ij} + \eta\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) + \left(\zeta - \frac{2}{3}\eta\right)(\nabla \cdot \boldsymbol{v})\delta_{ij}.
$$

The quantities that appear are

$\boldsymbol{v}$	fluid velocity,
p	fluid pressure,
ρ	fluid density,
η, ζ	standard and bulk viscosity coefficients,
$\boldsymbol{u}$	SAW displacement from equilibrium position, typically a harmonic function of time,
$\boldsymbol{n}$	outer unit normal vector on $\partial\Omega$.

The divergence of the stress tensor yields

$$
\nabla \cdot \Sigma = -\nabla p + \eta \Delta \boldsymbol{v} + \left(\zeta + \frac{\eta}{3}\right)\nabla(\nabla \cdot \boldsymbol{v}).
$$

We will need a constitutive equation to close this system. We use an equation of state arising from the assumption of constant entropy in time and space (see [11]). This leads to a simple relation of the form

$$
p - p_0 = a(\rho - \rho_0)^\gamma, \tag{2.2}
$$

where p_0, ρ_0 are equilibrium states and $a, \gamma > 0$. We assume a linear dependence for liquids such as water; thus, $\gamma = 1$.

2.2. Physical origin of acoustic streaming. As discussed above we assume that flow in the fluid region arises exclusively from the harmonic oscillation of the solid boundary. When using standard acoustics theory, where nonlinear effects are neglected and low frequencies and amplitudes are assumed, we expect the propagation of acoustic waves in the fluid. An additional effect observed in real life, however, is that of a *stationary* average flow field, not accounted for by standard acoustic theory.

This effect, known as *acoustic streaming*, was first studied by Lord Rayleigh [35]. Further contributions to the study of this phenomenon were made by Lighthill, Eckart, Nyborg, Riley, and others [36, 37, 31, 32, 28, 38, 16]. Two types of acoustic streaming are generally distinguished today:

- *Rayleigh streaming* is associated with the boundary layers of a fluid near solid surfaces.

- *Quartz wind* is associated with the dissipation of acoustic energy in the main body of the fluid.

Acoustic streaming effects are a result of the nonlinearity of the Navier–Stokes equations (2.1) in combination with the viscosity. The main problem for the numerical treatment of acoustic streaming is that extremely different time scales are involved. Typical frequencies of the harmonic oscillation are 100 MHz, requiring a time discretization on the order of 10^{-8} s. The acoustic streaming relaxation times are of the order 10^{-3}–10^{-1} s.

2.3. Acoustic subproblem. We assume an expansion of the unknowns $\boldsymbol{v}$, p, and ρ of the form

$$\begin{aligned} p &= p^{(0)} + \epsilon p' + \epsilon^2 p'' + \mathcal{O}\big(\epsilon^3\big)\,, \\ \rho &= \rho^{(0)} + \epsilon \rho' + \epsilon^2 \rho'' + \mathcal{O}\big(\epsilon^3\big)\,, \\ \boldsymbol{v} &= 0 + \epsilon \boldsymbol{v}' + \epsilon^2 \boldsymbol{v}'' + \mathcal{O}\big(\epsilon^3\big)\,. \end{aligned} \tag{2.3}$$

The small parameter $0 < \epsilon \ll 1$ is later fixed as the ratio $\boldsymbol{u}_0/L$, with $\boldsymbol{u}_0$ being the maximal SAW displacement of the domain boundary and L being the typical length scale used for a dimensionless formulation.

Set $p^{(1)} := \epsilon p'$, $p^{(2)} := \epsilon^2 p''$, and analogously ρ, $\boldsymbol{v}$, Σ. All terms with index 0 are constant in time and space and represent the known equilibrium state without SAW driving. By gathering all terms of order $\mathcal{O}(\epsilon)$ we arrive at

$$\rho^{(0)} \frac{\partial \boldsymbol{v}^{(1)}}{\partial t} - \nabla \cdot \Sigma^{(1)} = 0 \qquad \text{in } \Omega(t),\ t > 0, \tag{2.4a}$$

$$\frac{\partial \rho^{(1)}}{\partial t} + \rho^{(0)} \nabla \cdot \boldsymbol{v}^{(1)} = 0 \qquad \text{in } \Omega(t)\ t > 0, \tag{2.4b}$$

$$p^{(1)} = c_0^2 \rho^{(1)}, \tag{2.4c}$$

$$\boldsymbol{v}^{(1)}(t, \boldsymbol{x}) = \frac{\partial \boldsymbol{u}}{\partial t}(t, \boldsymbol{x}) \qquad \text{on } \Gamma_d(t),\ t > 0, \tag{2.4d}$$

$$\boldsymbol{v}^{(1)} = 0, \quad p^{(1)} = 0 \qquad \Omega(0) = \Omega_0 \qquad \text{for } t = 0, \tag{2.4e}$$

where c_0 is the small signal sound velocity in the fluid. The boundary condition is derived by observing that $\boldsymbol{u} = \mathcal{O}(\epsilon)$ and performing a Taylor expansion:

$$\boldsymbol{v}(t, \boldsymbol{x} + \boldsymbol{u}) = \boldsymbol{v}(t, \boldsymbol{x}) + (\nabla \boldsymbol{v})\boldsymbol{u} + \mathcal{O}(\|\boldsymbol{u}\|^2) = \frac{\partial \boldsymbol{u}}{\partial t}(t, \boldsymbol{x}).$$

In the future we will neglect the time dependence of Ω arising through the SAW oscillation. This is possible since the SAW displacement $\boldsymbol{u}$ is much smaller than the typical length L, equivalent to the fact that $\epsilon \ll 1$.

This linear system describes the *damped* propagation of acoustic waves. These arise solely as a reaction to the time-harmonic boundary values (2.4d), since the right-hand sides are zero. Due to the parabolic character of the system and the boundedness of Ω we expect the solution to converge into a time-harmonic quasi-equilibrium state for large t. Since the goal is the solution of the system (2.1) on the large time scale, the startup effects are of less interest than this equilibrium state.

2.4. Acoustic streaming subproblem. Assuming that the above first order system (2.4) is solved we proceed to gather terms of second order in ϵ while treating

the solution of (2.4) as given data. The result is

$$\rho^{(0)}\frac{\partial \boldsymbol{v}^{(2)}}{\partial t} - \nabla\cdot\Sigma^{(2)} = -\rho^{(1)}\frac{\partial \boldsymbol{v}^{(1)}}{\partial t} - \rho^{(0)}(\nabla \boldsymbol{v}^{(1)})\boldsymbol{v}^{(1)} \qquad \text{in } \Omega(t),\ t>0, \tag{2.5a}$$

$$\frac{\partial \rho^{(2)}}{\partial t} + \rho^{(0)}\nabla\cdot\boldsymbol{v}^{(2)} = -\nabla\cdot(\rho^{(1)}\boldsymbol{v}^{(1)}) \qquad \text{in } \Omega(t),\ t>0, \tag{2.5b}$$

$$p^{(2)} = c_0^2\rho_2, \tag{2.5c}$$

$$\boldsymbol{v}^{(2)} = -(\nabla \boldsymbol{v}^{(1)})\boldsymbol{u} \qquad \text{on } \Gamma_d(t),\ t>0, \tag{2.5d}$$

$$\boldsymbol{v}^{(2)} = 0, \quad p^{(2)} = 0, \quad \Omega(0) = \Omega_0 \qquad \text{for } t = 0. \tag{2.5e}$$

This system can be interpreted as an instationary Stokes problem describing creeping flow and containing mass sources and body forces.

Define $T := 2\pi\omega$ as the period of the harmonic SAW oscillation. Since we are interested in the acoustic streaming effects observable on the larger time scale, we take a temporal average of the right-hand sides. More precisely, we apply the operation

$$\langle a\rangle := \frac{1}{T}\int_{t_0}^{t_0+T} a(t)\,dt.$$

We thereby obtain the system

$$\begin{aligned}
\rho^{(0)}\frac{\partial \bar{\boldsymbol{v}}^{(2)}}{\partial t} - \nabla\cdot\bar{\Sigma}^{(2)} &= \left\langle -\rho^{(1)}\frac{\partial \boldsymbol{v}^{(1)}}{\partial t} - \rho^{(0)}(\nabla \boldsymbol{v}^{(1)})\boldsymbol{v}^{(1)}\right\rangle && \text{in } \Omega(t),\ t>0,\\
\frac{\partial \bar{\rho}^{(2)}}{\partial t} + \rho^{(0)}\nabla\cdot\bar{\boldsymbol{v}}^{(2)} &= \left\langle -\nabla\cdot(\rho^{(1)}\boldsymbol{v}^{(1)})\right\rangle && \text{in } \Omega(t),\ t>0,\\
\bar{p}^{(2)} &= c_0^2\bar{\rho}_2, &&\\
\bar{\boldsymbol{v}}^{(2)} &= -\left\langle(\nabla \boldsymbol{v}_1)\boldsymbol{u}\right\rangle && \text{on } \Gamma_d(t),\ t>0,\\
\bar{\boldsymbol{v}}^{(2)} = 0, \quad p^{(2)} &= 0, \quad \Omega(0) = \Omega_0 && \text{for } t = 0.
\end{aligned} \tag{2.6}$$

In the following we will deal only with the averaged terms $\bar{\boldsymbol{v}}^{(2)}$, $\bar{p}^{(2)}$, $\bar{\rho}^{(2)}$ and therefore denote these without the bars for simplicity. The corresponding stationary problem is also of interest when performing simulations without free capillary boundaries.

$$-\nabla\cdot\Sigma^{(2)} = \left\langle -\rho^{(1)}\frac{\partial \boldsymbol{v}^{(1)}}{\partial t} - \rho^{(0)}(\nabla \boldsymbol{v}^{(1)})\boldsymbol{v}^{(1)}\right\rangle \qquad \text{in } \Omega, \tag{2.7a}$$

$$\rho^{(0)}\nabla\cdot\boldsymbol{v}^{(2)} = \left\langle -\nabla\cdot(\rho^{(1)}\boldsymbol{v}^{(1)})\right\rangle \qquad \text{in } \Omega, \tag{2.7b}$$

$$p^{(2)} = c_0^2\rho_2, \tag{2.7c}$$

$$\boldsymbol{v}^{(2)} = -\left\langle(\nabla \boldsymbol{v}_1)\boldsymbol{u}\right\rangle \qquad \text{on } \Gamma_d. \tag{2.7d}$$

2.5. Free capillary boundaries. In addition to the problems with fixed channel geometries we will also treat the case of free capillary surfaces along parts of the boundary. To prepare this we first rewrite (2.1) using additional boundary conditions which are valid on a free capillary boundary $\Gamma_f(t)$ between liquid and air. The new boundary conditions are necessary to determine the normal velocity of the moving

interface as well as the stress acting due to capillary forces.

$$\rho\left(\frac{\partial \boldsymbol{v}}{\partial t} + (\nabla \boldsymbol{v})\boldsymbol{v}\right) = \nabla \cdot \Sigma \qquad \text{in } \Omega(t),\ t > 0, \tag{2.8a}$$

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{v}) = 0 \qquad \text{in } \Omega(t),\ t > 0, \tag{2.8b}$$

$$\boldsymbol{v}(t, \boldsymbol{x} + \boldsymbol{u}(t, \boldsymbol{x})) = \frac{\partial \boldsymbol{u}}{\partial t}(t, \boldsymbol{x}) \qquad \text{on } \Gamma_d, \tag{2.8c}$$

$$\Sigma \boldsymbol{n} = \sigma(d-1)\boldsymbol{\kappa} \qquad \text{on } \Gamma_f(t),\ t > 0, \tag{2.8d}$$

$$\boldsymbol{v} \cdot \boldsymbol{n} = V \qquad \text{on } \Gamma_f(t),\ t > 0, \tag{2.8e}$$

$$\boldsymbol{v} = 0, \quad p = p_0, \quad \Omega(0) = \Omega_0 \qquad \text{for } t = 0. \tag{2.8f}$$

Here we have σ as the constant of surface tension between water and vacuum, the difference to the conditions of water/air interfaces being negligible, (cf. [27]), $\Gamma_f \subset \partial\Omega(t)$ as the free boundary, and V as the normal velocity component of the free boundary. Furthermore we have $\boldsymbol{\kappa}$ as the vector of curvature. By definition this vector has magnitude H (mean curvature of $\partial\Omega(t)$) and points in the direction of $\boldsymbol{n}$ (outer unit normal vector of $\partial\Omega(t)$).

The location of $\Gamma_f(t)$ will change significantly in time and now represents one of the unknowns of the problem. To adapt the two-scale model for the inclusion of capillary boundary effects, we include the capillary stress condition (2.8d) in both subproblems.

For the sake of consistency we will continue to neglect the motion of the domain when solving the first order acoustics problem (2.4). However, we will need to take it into account when solving the large time scale acoustic streaming subproblem (2.7).

3. Analytical results. In this section we will analyze a mathematical model of the acoustics and acoustic streaming subproblems introduced in section 2 for a fixed domain without free boundaries. Both subproblems (2.4) and (2.7) have the following general form: Given right-hand sides $\boldsymbol{f}_v$, f_p, boundary values $\boldsymbol{g}$, and initial values $\boldsymbol{v}_0, p_0$, find $\boldsymbol{v}$ and p satisfying

$$\mu_v \frac{\partial \boldsymbol{v}}{\partial t} - \nu_1 \Delta \boldsymbol{v} - \nu_2 \nabla(\nabla \cdot \boldsymbol{v}) + \nabla p = \boldsymbol{f}_v \qquad \text{in } \Omega,\ t > 0, \tag{3.1a}$$

$$\mu_p \frac{\partial p}{\partial t} + \nabla \cdot \boldsymbol{v} = f_p \qquad \text{in } \Omega,\ t > 0, \tag{3.1b}$$

$$\boldsymbol{v} = \boldsymbol{g} \qquad \text{on } \partial\Omega,\ t > 0, \tag{3.1c}$$

$$\boldsymbol{v} = \boldsymbol{v}_0, \quad p = p_0 \qquad \text{in } \Omega,\ t = 0, \tag{3.1d}$$

with positive constant parameters μ_v, μ_p, ν_1, ν_2. We will consider both the time-dependent and the stationary cases $\mu_v = \mu_p = 0$ which requires different methods.

As before, we assume Ω to be a bounded Lipschitz domain. We will use the following variational formulation of (3.1):

$$\mu_v \int_\Omega \frac{\partial \boldsymbol{v}}{\partial t} \cdot \boldsymbol{w} + \nu_1 \int_\Omega \nabla \boldsymbol{v} : \nabla \boldsymbol{w} + \nu_2 \int_\Omega (\nabla \cdot \boldsymbol{v})(\nabla \cdot \boldsymbol{w}) - \int_\Omega p \nabla \cdot \boldsymbol{w} = \int_\Omega \boldsymbol{f}_v \cdot \boldsymbol{w} \qquad \text{for all } \boldsymbol{w} \in \boldsymbol{C}_0^\infty(\Omega),$$

$$\mu_p \int_\Omega \frac{\partial p}{\partial t} q + \int_\Omega q \nabla \cdot \boldsymbol{v} = \int_\Omega f_p q \qquad \text{for all } q \in C_0^\infty(\Omega).$$

In our analysis we will assume $\mu_v = \mu_p$.

To deal with the instationary problem we will introduce three different solution concepts, each requiring a separate theoretical treatment and numerical implementation. The validity of each approach is determined by the form of the problem data (right-hand sides $\boldsymbol{f}_v$, f_p and boundary conditions $\boldsymbol{g}$):

- general time-dependent solutions, requiring no assumptions on the form of the problem data,
- time-periodic solutions, requiring periodic problem data,
- time-harmonic solutions, requiring problem data given as a superposition of sine and cosine terms.

Periodic solutions are of interest, since the application to SAW-driven acoustic streaming often involves time-periodic data $\boldsymbol{f}_v$, f_p, $\boldsymbol{g}$. In the case of time-harmonic problem data it is possible to eliminate the time dependence altogether and reduce the problem to an equivalent stationary system. This so-called *quasi-stationary approach* is treated later in this section.

3.1. Solution spaces and variational formulation. For the variational formulation we define the following spaces:

$$\begin{aligned}
\boldsymbol{V} &= \boldsymbol{H}_0^1(\Omega),\\
Q &= L_0^2(\Omega),\\
U &= \boldsymbol{H}_0^1(\Omega) \times L_0^2(\Omega),\\
H &= \boldsymbol{L}^2(\Omega) \times L_0^2(\Omega),\\
U' &= \boldsymbol{H}^{-1}(\Omega) \times L_0^2(\Omega) \qquad \text{as the dual space of } U.
\end{aligned}$$

This yields a Gelfand triple

$$U \subset H \subset U' \tag{3.2}$$

with each space continuously embedded and dense in the subsequent space; see e.g., [50].

For a given bounded time interval $[0,T] \subset \mathbb{R}$ we define the Hilbert space

$$W(0,T) = \{f \in H^1(0,T;U') \mid f \in L^2(0,T;U)\},$$

with the scalar product

$$(f_1, f_2)_{W(0,T)} = \int_0^T (f_1(t), f_2(t))_U \, dt + \int_0^T (f_1'(t), f_2'(t))_{U'} \, dt$$

for all $f_1, f_2 \in W(0,T)$.

Thanks to well-known continuity properties of this space we will be able to justify the specification of initial values in H. We now define the following bilinear forms:

$$a : \boldsymbol{V} \times \boldsymbol{V} \to \mathbb{R}, \tag{3.3a}$$

$$a(\boldsymbol{v}, \boldsymbol{w}) = \nu_1 \int_\Omega \nabla \boldsymbol{v} : \nabla \boldsymbol{w} + \nu_2 \int_\Omega (\nabla \cdot \boldsymbol{v})(\nabla \cdot \boldsymbol{w}), \tag{3.3b}$$

$$b : Q \times \boldsymbol{V} \to \mathbb{R}, \tag{3.3c}$$

$$b(p, v) = -\int_\Omega p \nabla \cdot \boldsymbol{v}, \tag{3.3d}$$

$$s : U \times U \to \mathbb{R}, \tag{3.3e}$$

$$s((\boldsymbol{v}, p), (\boldsymbol{w}, q)) = a(\boldsymbol{v}, \boldsymbol{w}) + b(q, \boldsymbol{v}) - b(p, \boldsymbol{w}). \tag{3.3f}$$

LEMMA 3.1. *The following properties hold for the bilinear forms defined in* (3.3)*:*

1. *The form a is $\boldsymbol{V}$-*elliptic*, meaning that there exist constants $c_a, C_a > 0$ so that for all $\boldsymbol{v}, \boldsymbol{w} \in \boldsymbol{V}$*

$$|a(\boldsymbol{v}, \boldsymbol{w})| \leq C_a \|\boldsymbol{v}\|_1 \|\boldsymbol{w}\|_1 ,$$
$$a(\boldsymbol{v}, \boldsymbol{v}) \geq c_a \|\boldsymbol{v}\|_1^2 .$$

2. *There exist constants $c_b, C_b > 0$ so that for all $\boldsymbol{v} \in \boldsymbol{V}$ and $p \in Q$*

$$|b(p, \boldsymbol{v})| \leq C_b \|p\|_0 \|v\|_1 , \tag{3.4}$$

$$\inf_{p \in Q \setminus \{0\}} \sup_{\boldsymbol{v} \in \boldsymbol{V} \setminus \{0\}} \frac{b(p, \boldsymbol{v})}{\|p\|_0 \|\boldsymbol{v}\|_1} \geq c_b. \tag{LBB}$$

The inequality (LBB) *is the well-known Ladyzhenskaya-Babuška-Brezzi condition.*

3. *The form s is U-*coercive *(see, e.g.,* [29]*); in other words, there are constants $c_s, C_s > 0$, $k_s \in \mathbb{R}$ so that for all $f, g \in U$*

$$|s(f, g)| \leq C_s \|f\|_U \|g\|_U , \tag{3.5}$$

$$s(f, f) + k_s \|f\|_H^2 \geq c_s \|f\|_U^2 . \tag{3.6}$$

We can choose $c_s = k_s = c_a$.

DEFINITION 3.2. *Using Lemma* 3.1 *we define the following bounded linear operators. Let $\boldsymbol{v}, \boldsymbol{w} \in \boldsymbol{V}$, $p, q \in Q$, $f, g \in U$.*

$$\begin{aligned} A \in L(\boldsymbol{V}, \boldsymbol{V}') : &\qquad \langle A\boldsymbol{v}, \boldsymbol{w} \rangle_{\boldsymbol{V}' \times \boldsymbol{V}} = a(\boldsymbol{v}, \boldsymbol{w}), \\ B \in L(Q, \boldsymbol{V}') : &\qquad \langle Bp, \boldsymbol{v} \rangle_{\boldsymbol{V}' \times \boldsymbol{V}} = b(p, \boldsymbol{v}), \\ B^* \in L(\boldsymbol{V}, Q) : &\qquad \langle B^* \boldsymbol{v}, p \rangle_{Q \times Q} = b(p, \boldsymbol{v}), \\ S \in L(U, U') : &\qquad \langle Sf, g \rangle_{U' \times U} = s(f, g), \\ K \in L(Q, Q) : &\qquad K = B^* A^{-1} B. \end{aligned} \tag{3.7}$$

Also useful will be the bilinear form corresponding to the Schur complement operator K:

$$k : Q \times Q \to \mathbb{R},$$
$$k(p, q) = (Kp, q)_0.$$

3.2. Instationary solutions. We can now formulate the variational formulation of (3.1).

Problem 3.3 (weak formulation of the instationary problem). Let $u_0 = (\boldsymbol{v}_0, p_0) \in H$ and $f = (\boldsymbol{f}_v, f_p) \in L^2(0, T; U')$. Find $u = (\boldsymbol{v}, p) \in W(0, T)$ so that

$$\begin{aligned} \langle w, u'(t) \rangle_{U \times U'} + s(w, u(t)) = \langle w, f(t) \rangle_{U \times U'} &\quad \text{for all } w \in U, \text{ f.a.a.}^1 \ \ t \in (0, T), \\ u(0) = u_0 &\quad \text{in } H. \end{aligned} \tag{P}$$

We remark that Problem 3.3 is well-posed due to the properties of the solution space $W(0, T)$.

[1]The abbreviation "f.a.a." will stand for "for almost all" with respect to the used Lebesgue measure.

LEMMA 3.4. *The problem* (P) *can be equivalently formulated as*
Find $(\boldsymbol{v}, p) \in W(0,T)$ *so that*

$$
\begin{aligned}
\langle \boldsymbol{w}, \boldsymbol{v}'(t)\rangle_{\boldsymbol{V}\times\boldsymbol{V}'} + a(\boldsymbol{w}, \boldsymbol{v}(t)) + b(p(t), \boldsymbol{w}) &= \langle \boldsymbol{w}, \boldsymbol{f}_v(t)\rangle_{\boldsymbol{V}\times\boldsymbol{V}'} && \text{for all } \boldsymbol{w} \in \boldsymbol{V},\\
(q, p'(t))_0 - b(q, \boldsymbol{v}(t)) &= (q, f_p(t))_0 && \text{for all } q \in Q,\\
& && \text{f.a.a. } t \in (0,T),\\
\boldsymbol{v}(0) &= \boldsymbol{v}_0 && \text{in } \boldsymbol{L}^2(\Omega),\\
p(0) &= p_0 && \text{in } Q.
\end{aligned}
\tag{P$'$}
$$

Proof. Assume (P$'$) holds. Set $w = (\boldsymbol{w}, q)$, $u_0 = (\boldsymbol{v}_0, p_0)$, and sum the first two equations to get (P). Now assume (P) holds. Select first $\psi = (\boldsymbol{w}, 0)$ to obtain the first variational equality in (P$'$) and then $\psi = (0, q)$ for the second one. The initial condition is clear. ∎

The stationary problem corresponding to (3.1) can be interpreted as a compressible Stokes system.

Problem 3.5 (weak formulation of the stationary problem). Let $f = (\boldsymbol{f}_v, f_p) \in U'$. Find $u = (\boldsymbol{v}, p) \in U$ so that

$$
\begin{aligned}
a(\boldsymbol{w}, \boldsymbol{v}) + b(p, \boldsymbol{w}) &= \langle \boldsymbol{w}, \boldsymbol{f}_v\rangle_{\boldsymbol{V}\times\boldsymbol{V}'} && \text{for all } \boldsymbol{w} \in \boldsymbol{V},\\
-b(q, \boldsymbol{v}) &= (q, f_p)_0 && \text{for all } q \in Q.
\end{aligned}
\tag{P$_{stat}$}
$$

An equivalent reformulation of (P$_{stat}$) using the form s is also possible using the same methods as in Lemma 3.4.

Remark 3.6. The choice of Q as pressure space is a real restriction as the following example shows.

Given $\Omega = (0,1)^2$, solve the problem

$$
\begin{aligned}
\frac{\partial \boldsymbol{v}}{\partial t}(t) - \Delta \boldsymbol{v} - \nabla(\nabla \cdot \boldsymbol{v}) + \nabla p &= 0 && \text{in } \Omega,\\
\frac{\partial p}{\partial t}(t) + \nabla \cdot \boldsymbol{v} &= 1 && \text{in } \Omega,\\
\boldsymbol{v}(x,t) &= 0 && \text{on } \partial\Omega,\\
\boldsymbol{v}_0(x) = 0, \quad p_0(x) &= 0.
\end{aligned}
\tag{3.8}
$$

We immediately verify that a solution is given by $\boldsymbol{v}(x,t) = 0$, $p(x,t) = t$, which does not lie in $W(0,T)$.

This shows that working with $Q = L_0^2(\Omega)$ is too restrictive in this setting. However, we use this space for simplicity in this presentation. An extension of the existence and uniqueness theorems presented later to the case of generalized solutions $(\boldsymbol{v}, p)$ sought in the larger space $\boldsymbol{H}^1(\Omega) \times L^2(\Omega)$ may be found in [26].

Since the Ladyzhenskaya-Babuška-Brezzi condition (LBB) is fulfilled we can deduce the existence and uniqueness of solutions of Problem 3.5.

LEMMA 3.7. *Problem* (P$_{stat}$) *has a unique solution* $(\boldsymbol{v}, p) \in U$. *The following a priori estimate holds:*

$$\|\boldsymbol{v}\|_1 + \|p\|_0 \le C(\|\boldsymbol{f}_v\|_{\boldsymbol{V}'} + \|f_p\|_0),$$

with a constant $C > 0$.

Proof. Given the properties of the bilinear forms in Lemma 3.1, we can use the saddle point theory specified in [34]. Additional references are [19, 25, 7]. ∎

We observe that (P) defines a parabolic problem due to the properties of $s(\cdot,\cdot)$ explained in Lemma 3.1. We cite the following theorem.

THEOREM 3.8 (existence and uniqueness of (P)). *For $0 < T < \infty$ the problem* (P) *possesses a unique solution $(\boldsymbol{v}, p) \in W(0,T)$. Furthermore, we have the following stability estimate:*

$$\|(\boldsymbol{v},p)\|_{W(0,T)} \le C\left(\|(\boldsymbol{v}_0,p_0)\|_H + \|(\boldsymbol{f}_v,f_p)\|_{L^2(0,T;U')}\right).$$

The constant C will generally depend on T.

Proof. [50, Theorem 26.1]. ∎

We also state the following result for time regularity of the solution.

THEOREM 3.9 (time regularity of the solution of (P)). *Assume $u = (\boldsymbol{v}, p)$ as the solution of the parabolic problem* (P) *stated in the following form:*

$$\begin{aligned} u' + Su &= f && \text{a.e. in } (0,T),\\ u(0) &= u_0 && \text{for } t = 0. \end{aligned}$$

Let $0 < k \in \mathbb{N}$. We assume $f \in H^k(0,T;U')$ and u_0 fulfilling the following compatibility conditions:

$$\begin{aligned} w_0 &:= u_0 \in U,\\ w_j &:= \frac{d^{j-1}f}{dt^{j-1}}(0) - Sw_{j-1} \in U,\ j = 1,\dots,k-1,\\ w_k &:= \frac{d^{k-1}f}{dt^{k-1}}(0) - Sw_{k-1} \in H. \end{aligned}$$

We then have

$$u \in H^k(0,T;U) \qquad \text{and} \qquad \frac{d^{k+1}u}{dt^{k+1}} \in L^2(0,T;U').$$

Proof. [50, Theorem 27.2]. ∎

3.3. Periodic solutions. For the acoustics subproblem (2.4) we generally have a time-periodic driving force $\boldsymbol{u}$. It is therefore of interest to reformulate Problem 3.3 for right-hand sides f which have period T and search for periodic solutions. This changes the character of the problem from an initial value problem to a periodic problem, where $u(0)$ and $u(T)$ are unknown a priori.

Later in this section we will compare the solution u of the initial value problem (P) supplied with a periodic forcing f to the solution u_{per} of the corresponding periodic problem. We expect the solution of the initial value problem to converge in some sense towards a periodic equilibrium state, that is, u_{per}, for large times. This phenomenon is known as *forced oscillation* in physics. In [26] we present a theorem confirming this convergent behavior.

We first specify what is meant by a periodic solution. The definition may be surprising at first since periodicity is not mentioned explicitly. The characterization given in Proposition 3.11 will justify the terminology.

Problem 3.10 (periodic solutions). Let $0 < T < \infty$ and $f = (\boldsymbol{f}_v, f_p) \in L^2(0,T;H)$. Find $u = (\boldsymbol{v}, p) \in W(0,T)$ so that

$$(\mathrm{P}_{per}) \qquad \begin{aligned} \langle w, u'(t)\rangle_{U\times U'} + s(w,u(t)) &= (w,f(t))_H && \text{for all } w \in U, \text{ a.e. in } (0,T),\\ u(0) &= u(T) && \text{in } H. \end{aligned}$$

Note that $u(0)$ and $u(T)$ are unknowns, not data of the problem. We demand more regularity in space, i.e., $f(t) \in H$. This will enable us to use elementary results of Fourier analysis (see [1]) for the derivation of a priori estimates.

PROPOSITION 3.11. *Assume that the right-hand side f is extended to a periodic function $\widetilde{f}$ by*

$$\widetilde{f}(t+kT) = f(t) \qquad \text{for } t \in [0,T),\ k \in \mathbb{Z}.$$

A solution u of (P_{per}) can then be uniquely extended to a periodic solution $\widetilde{u}$ fulfilling

$$\langle w, \widetilde{u}'(t)\rangle_{U\times U'} + s(w, \widetilde{u}(t)) = (w, \widetilde{f}(t))_H \qquad \text{for all } w \in U,\ \text{f.a.a. } t \in \mathbb{R}.$$

Proof. The existence and uniqueness Theorem 3.8 implies that the solution u with initial values $u(0)$ is well defined for an arbitrarily large interval $[0,\tau]$. Define $z(t) := u(T+t) - u(t)$. The function z is a solution of the following initial value problem:

$$\begin{aligned} z'(t) + Sz(t) &= 0, \\ z(0) &= 0. \end{aligned}$$

The a priori estimate of Theorem 3.8 then implies $z = 0$, and therefore $u(t) = u(T+t)$. A similar argument shows that the solution can be uniquely extended backwards in time. ∎

THEOREM 3.12 (existence and uniqueness of (P_{per})). *The problem (P_{per}) has a unique solution $(\boldsymbol{v}, p) \in W(0,T)$. We have the stability estimate*

$$\|(\boldsymbol{v}, p)\|_{W(0,T)} \le C \,\|(\boldsymbol{f}_v, f_p)\|_{L^2(0,T;H)}.$$

The constant C will generally depend on the period T.

The idea of the proof is to use the standard Galerkin method. In a first step we establish stability properties of corresponding finite-dimensional problems. These are extended to the full problem using standard compactness results. For the finite-dimensional problem we used Fourier analysis to eliminate the time dependence of the periodic problem. The key elements of the stability proof are elaborate estimates on the eigenvalues of the compressible Stokes operator S defined in (3.7). These were taken from [4]. For the complete proof see [26, Theorem 3.2.6].

3.4. Quasi-stationary approach. In the physical applications the forcing terms are typically given by a harmonic signal:

$$\begin{aligned} \boldsymbol{f}_v(t,\boldsymbol{x}) &= \widehat{\boldsymbol{f}}_{vc}(\boldsymbol{x})\cos(\omega t) + \widehat{\boldsymbol{f}}_{vs}(\boldsymbol{x})\sin(\omega t), \\ f_p(t,\boldsymbol{x}) &= \widehat{f}_{pc}(\boldsymbol{x})\cos(\omega t) + \widehat{f}_{ps}(\boldsymbol{x})\sin(\omega t), \\ \boldsymbol{g}(t,\boldsymbol{x}) &= \widehat{\boldsymbol{g}}_{c}(\boldsymbol{x})\cos(\omega t) + \widehat{\boldsymbol{g}}_{s}(\boldsymbol{x})\sin(\omega t), \end{aligned} \tag{3.9}$$

with a frequency $\omega = 2\pi/T > 0$. As proven in [26, Theorem 3.2.10] the solution of the acoustic system will converge against a periodic equilibrium state for $t \to \infty$. As will become clear later this periodic equilibrium state must also be time-harmonic. Since the equilibrium state is what we are interested in when solving the full problem, we may attempt to split the spatial and temporal dependence of the acoustic field solution. This eliminates the need for a time discretization. To be precise, we consider the following ansatz:

$$\begin{aligned} \boldsymbol{v}(t,\boldsymbol{x}) &= \widehat{\boldsymbol{v}}_{c}(\boldsymbol{x})\cos(\omega t) + \widehat{\boldsymbol{v}}_{s}(\boldsymbol{x})\sin(\omega t), \\ p(t,\boldsymbol{x}) &= \widehat{p}_{c}(\boldsymbol{x})\cos(\omega t) + \widehat{p}_{s}(\boldsymbol{x})\sin(\omega t), \end{aligned} \tag{3.10}$$

with a frequency ω equal to the driving frequency of the boundary conditions and right-hand sides in (3.9). This idea is standard for solving time-harmonic problems, although a decomposition using the complex terms $\exp(i\omega t)$, $\exp(-i\omega t)$ is usually used instead.

After substituting these formulas for $\boldsymbol{v}$ and p in system (3.1) and using the linear independence of sine and cosine functions, we arrive at four new field equations in the unknowns $\widehat{\boldsymbol{v}}_i$, $\widehat{p}_i$, $i = c, s$:

$$
\begin{aligned}
\omega\mu_v\widehat{\boldsymbol{v}}_s - \nu_1\Delta\widehat{\boldsymbol{v}}_c - \nu_2\nabla(\nabla\cdot\widehat{\boldsymbol{v}}_c) + \nabla\widehat{p}_c &= \widehat{\boldsymbol{f}}_{vc},\\
-\omega\mu_v\widehat{\boldsymbol{v}}_c - \nu_1\Delta\widehat{\boldsymbol{v}}_s - \nu_2\nabla(\nabla\cdot\widehat{\boldsymbol{v}}_s) + \nabla\widehat{p}_s &= \widehat{\boldsymbol{f}}_{vp},\\
\omega\mu_p\widehat{p}_s + \nabla\cdot\widehat{\boldsymbol{v}}_c &= \widehat{f}_{pc},\\
-\omega\mu_p\widehat{p}_c + \nabla\cdot\widehat{\boldsymbol{v}}_s &= \widehat{f}_{ps} \qquad \text{in } \Omega,\\
\widehat{\boldsymbol{v}}_c &= \widehat{\boldsymbol{g}}_c,\\
\widehat{\boldsymbol{v}}_s &= \widehat{\boldsymbol{g}}_s \qquad \text{on } \partial\Omega.
\end{aligned}
\tag{3.11}
$$

We are now faced with solving only one stationary system at the price of twice as many unknowns as before. A variational formulation of this system is derived as before. This variational formulation can be analyzed using well-known results for saddle point problems; see [34, Theorem 2.2], [25], and also [7].

4. Methods for free boundary problems. This section will treat the discretization of free capillary problems associated with (3.1). Most of the available literature for free capillary boundary problems concentrates on the incompressible (nonlinear) Navier–Stokes system; see [3] and references therein. It is interesting to note that the nonlinear convection term, together with the assumption of incompressibility, may actually serve to prove the stability of numerical schemes; see [3, p. 6]. Analytical results of existence and uniqueness, even for the more well known case of incompressible Navier–Stokes, are applicable only to special domains or small times or rely on function spaces with high regularity properties unsuitable for a finite element method (FEM) discretization; see [30, 45, 42].

We will treat the case of Dirichlet and free capillary boundary conditions specified on subsets Γ_d (resp., Γ_f) of $\partial\Omega$, $\Gamma_d \cup \Gamma_f = \partial\Omega$. We will assume that Γ_d is a topologically closed surface, implying that the contact line between the free surface and walls remains fixed. We note that this hypothesis is not always valid in physical situations. However, moving contact lines cause problems in mathematical formulations due to the incompatibility of boundary conditions; see the first section of [45].

This yields the following system:

$$\mu_v\frac{\partial\boldsymbol{v}}{\partial t} - \nabla\cdot\Sigma = \boldsymbol{f}_v \qquad \text{in } \Omega(t), t > 0, \tag{4.1a}$$

$$\mu_p\frac{\partial p}{\partial t} + \nabla\cdot\boldsymbol{v} = f_p \qquad \text{in } \Omega(t),\ t > 0, \tag{4.1b}$$

$$\boldsymbol{v} = \boldsymbol{g} \qquad \text{on } \Gamma_d, \tag{4.1c}$$

$$\Sigma\boldsymbol{n} = \sigma(d-1)\boldsymbol{\kappa} \qquad \text{on } \Gamma_f(t),\ t > 0, \tag{4.1d}$$

$$\boldsymbol{v}\cdot\boldsymbol{n} = V \qquad \text{on } \Gamma_f(t),\ t > 0, \tag{4.1e}$$

$$\Omega(0) = \Omega_0, \tag{4.1f}$$

$$\boldsymbol{v}(t=0,\boldsymbol{x}) = \boldsymbol{v}_0 \qquad \text{in } \Omega_0, \tag{4.1g}$$

$$p(t=0,\boldsymbol{x}) = p_0 \qquad \text{in } \Omega_0, \tag{4.1h}$$

with

$$\Sigma_{ij} = -p\delta_{ij} + \nu_1 \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right) + (\nu_2 - \nu_1)(\nabla \cdot \boldsymbol{v})\delta_{ij} \qquad \text{the stress tensor,}$$

V the normal velocity of $\Gamma_f(t)$,

$\boldsymbol{\kappa}$ the vector of mean curvature,

$\sigma > 0$ the coefficient of surface tension.

We are interested in a sharp interface arbitrary Lagrangian Eulerian (ALE) approach, where the domain essentially follows the motion of the fluid; see [23, 22] and references therein. This seems most promising because of its simplicity, at the price of not being able to deal with domain topology changes, for example, due to the formation of cusps. Examples of more general methods are volume-of-fluid [20] or the more recent level set methods [33, 47, 44, 46, 43]. In the standard physical situations considered we do not expect topology changes.

4.1. Variational formulation including curvature. To incorporate free capillary boundaries in the finite element framework, we will need a variational formulation of the curvature term [12, 13].

We assume $\Gamma \subset \mathbb{R}^d$ to be a subset of an orientable $(d-1)$-dimensional smooth manifold M and $\boldsymbol{\chi} : U \to \Gamma$, U an open subset of $\mathbb{R}^{d-1}$, as a local parameterization of Γ. We define $g_{ij} = \partial_i \boldsymbol{\chi} \cdot \partial_j \boldsymbol{\chi}$ as the metric tensor of Γ and $g^{ij} = (g^{-1})_{ij}$ as its inverse.

For functions $f : \Gamma \to \mathbb{R}$, $\boldsymbol{f} : \Gamma \to \mathbb{R}^d$ we introduce the following differential operators on Γ:

$$(\nabla_\Gamma f) \circ \boldsymbol{\chi} := \sum_{i,j=1}^{d-1} g^{ij} \partial_i (f \circ \boldsymbol{\chi}) \partial_j \boldsymbol{\chi}$$

as the tangential derivative of f,

$$(\nabla_\Gamma \cdot \boldsymbol{f}) \circ \boldsymbol{\chi} := \sum_{i,j=1}^{d-1} g^{ij} \partial_i (\boldsymbol{f} \circ \boldsymbol{\chi}) \cdot \partial_j \boldsymbol{\chi}$$

as the tangential divergence of $\boldsymbol{f}$, and

$$(\Delta_\Gamma f) \circ \boldsymbol{\chi} := (\nabla_\Gamma \cdot \nabla_\Gamma f) \circ \boldsymbol{\chi} = \sum_{i,j=1}^{d-1} \frac{1}{\sqrt{\det g}} \partial_i \left(\sqrt{\det g}\, g^{ij} \partial_j (f \circ \boldsymbol{\chi}) \right)$$

as the Laplace–Beltrami operator of f; see, for instance, [2, 9]. We have the following identity:

$$\Delta_\Gamma \operatorname{id}_\Gamma = \boldsymbol{\kappa},$$

where $\boldsymbol{\kappa}$ is the mean curvature vector on Γ. Let $\boldsymbol{\psi}$ be a smooth vector valued function defined on Γ, with $\boldsymbol{\psi} = 0$ on the relative boundary $\operatorname{relbd}(\Gamma)$ in M. Using partial integration on Γ we get

$$\begin{aligned} \int_\Gamma \boldsymbol{\kappa} \cdot \boldsymbol{\psi}\, d\mathcal{H}^{d-1} &= \int_\Gamma (\Delta_\Gamma \operatorname{id}_\Gamma) \cdot \boldsymbol{\psi}\, d\mathcal{H}^{d-1} \\ &= -\int_\Gamma \nabla_\Gamma \operatorname{id}_\Gamma : \nabla_\Gamma \boldsymbol{\psi}\, d\mathcal{H}^{d-1} = -\int_\Gamma \nabla_\Gamma \cdot \boldsymbol{\psi}\, d\mathcal{H}^{d-1}. \end{aligned} \tag{4.2}$$

There is no boundary term in the partial integration because $\boldsymbol{\psi} = 0$ on relbd(Γ). We now return to the system (4.1). If we test (4.1a) with a smooth function $\boldsymbol{\psi}$ defined on $\bar{\Omega}(t)$ and fulfilling $\boldsymbol{\psi} = 0$ on Γ_d, we get

$$\begin{aligned}\mu_v \int_\Omega \boldsymbol{\psi} \cdot \frac{\partial \boldsymbol{v}}{\partial t} - \int_\Omega \boldsymbol{\psi} \cdot \boldsymbol{f}_v &= \int_\Omega \boldsymbol{\psi} \cdot (\nabla \cdot \Sigma) \\ &= \int_\Omega \nabla \cdot (\Sigma \boldsymbol{\psi}) - \int_\Omega \Sigma : \nabla \boldsymbol{\psi} \\ &= \int_{\partial\Omega} (\Sigma \boldsymbol{\psi}) \cdot \boldsymbol{n} \, d\mathcal{H}^{d-1} - \int_\Omega \Sigma : \nabla \boldsymbol{\psi}.\end{aligned}$$

Thanks to (4.2) and the boundary condition (4.1d) the boundary integral is equal to

$$\int_{\partial\Omega} (\Sigma \boldsymbol{\psi}) \cdot \boldsymbol{n} \, d\mathcal{H}^{d-1} = \int_{\partial\Omega} (\Sigma \boldsymbol{n}) \cdot \boldsymbol{\psi} \, d\mathcal{H}^{d-1} = -\sigma(d-1) \int_{\Gamma_f} \nabla_\Gamma \cdot \boldsymbol{\psi} \, d\mathcal{H}^{d-1}.$$

The second integral evaluates to

$$\begin{aligned}-\int_\Omega \Sigma : \nabla \boldsymbol{\psi} = \int_\Omega p(\nabla \cdot \boldsymbol{\psi}) - \frac{\nu_1}{2} \int_\Omega D(\boldsymbol{v}) : D(\boldsymbol{\psi}) \\ - (\nu_2 - \nu_1) \int_\Omega (\nabla \cdot \boldsymbol{v})(\nabla \cdot \boldsymbol{\psi}),\end{aligned}$$

with the *deformation tensor* $D(\boldsymbol{v}) := \nabla \boldsymbol{v} + \nabla \boldsymbol{v}^T$. Combining the last equations finally yields

$$\begin{aligned}(4.3)\quad \mu_v \int_\Omega \boldsymbol{\psi} \cdot \frac{\partial \boldsymbol{v}}{\partial t} + \frac{\nu_1}{2} \int_\Omega D(\boldsymbol{v}) : D(\boldsymbol{\psi}) + (\nu_2 - \nu_1) \int_\Omega (\nabla \cdot \boldsymbol{v})(\nabla \cdot \boldsymbol{\psi}) \\ - \int_\Omega p(\nabla \cdot \boldsymbol{\psi}) = \int_\Omega \boldsymbol{\psi} \cdot \boldsymbol{f}_v - \sigma(d-1) \int_{\Gamma_f} \nabla_\Gamma \cdot \boldsymbol{\psi} \, d\mathcal{H}^{d-1}.\end{aligned}$$

This will be the starting point of the discrete formulation of the free boundary problem in the following section.

We will first review the ALE approach used to treat the motion of the free capillary surface. The basic idea is to use a special finite element space defined on a reference triangulation which will contain the coordinate function defining the mesh nodes at any time step. We will use a time-marching procedure as before to calculate the evolution of the velocity and pressure fields. The velocity at the free capillary boundary will be used to update the mesh position for the next time step.

4.2. ALE formulation. For the beginning we consider the extension of the evolution problem (P) to a time-dependent but given domain $\Omega(t)$. Let $\hat{\Omega} \subset \mathbb{R}^d$ be a bounded reference domain. We define a time-dependent continuous transformation to map $\hat{\Omega}$ to the current domain $\Omega(t)$:

$$\begin{aligned}\boldsymbol{\Phi} : [0,T] \times \hat{\Omega} &\to \mathbb{R}^d, \\ \boldsymbol{\Phi}(t, \hat{\Omega}) &= \Omega(t), \\ G &= \{(t, \boldsymbol{x}) \in [0,T] \times \mathbb{R}^d \mid \boldsymbol{x} \in \Omega(t)\}.\end{aligned}$$

We define a reverse transformation $\mathbf{\Psi}$ as follows:

$$\begin{aligned}&\mathbf{\Psi} : G \to \hat{\Omega},\\ &\mathbf{\Psi}(t, \mathbf{\Phi}(t, \hat{\boldsymbol{x}})) = \hat{\boldsymbol{x}}.\end{aligned}$$

We will assume that the following natural requirements for a proper space-time domain G are fulfilled.

Assumption 4.1 (admissible transformation). Let $t \in [0, T]$. Then

- $\mathbf{\Phi}(t, \cdot) : \hat{\Omega} \to \Omega(t)$ is injective, and
- $\inf_{\hat{x} \in \hat{\Omega}} \det D_{\hat{x}} \mathbf{\Phi}(t, \hat{x}) > 0$.

Here $D_{\hat{x}} \mathbf{\Phi}(t, \hat{x})$ stands for the Jacobian of $\mathbf{\Phi}$ with respect to the space variable.

For a given function $\varphi : G \to \mathbb{R}$ we can therefore define a related reference function as

$$\begin{aligned}&\hat{\varphi} : [0, T] \times \hat{\Omega} \to \mathbb{R},\\ &\qquad \hat{\varphi}(t, \hat{x}) = \varphi(t, \mathbf{\Phi}(t, \hat{x})).\end{aligned}$$

Let $\mathbf{\Phi}$ and $\varphi : G \to \mathbb{R}$ be sufficiently smooth to apply the chain rule of differentiation. We have

$$\begin{aligned} D_t \hat{\varphi}(t, \hat{x}) &= D_t \varphi(t, \mathbf{\Phi}(t, \hat{x})) + D_x \varphi(t, \mathbf{\Phi}(t, \hat{x})) D_t \mathbf{\Phi}(t, \hat{x}) \\ &= D_t \varphi(t, x) + D_x \varphi(t, x) D_t \mathbf{\Phi}(t, \hat{x}), \end{aligned} \tag{4.4}$$

where $x = \mathbf{\Phi}(t, \hat{x})$. The second term containing spatial derivatives of φ is actually omitted in the numerical discretization, as it is of higher order with respect to the expansion parameter ϵ and since we are mostly interested in a stationary limit. See [26] for details.

For a discretization in space as well as in time we assume that we have a finite-dimensional reference subspace $\hat{U}_h \subset \hat{U}$ defined on $\hat{\Omega} = \Omega_0$. Furthermore, we assume that a sequence of discrete time steps $0 = t_0 < t_1 < \cdots < t_N = T$ are given with corresponding mesh transformations $\mathbf{\Phi}_h^n : \hat{\Omega} \to \Omega_n$, $\mathbf{\Psi}_h^n : \Omega_n \to \hat{\Omega}$.

We now seek the discrete solution

$$u_h^n \in U_{h,n} = \{\hat{\varphi}_h \circ \mathbf{\Psi}_h^n \mid \hat{\varphi}_h \in \hat{U}_h\}$$

for $n = 1, \dots, N$.

4.3. Algorithm. We will now define an algorithm to discretize the free boundary problem (4.1). We assume that a reference domain $\hat{\Omega}$ with a conforming triangulation $\hat{\mathcal{T}}$ is given. We define $\hat{\Gamma}_f \subset \partial\hat{\Omega}$ as the part of the boundary where free capillary boundary conditions are prescribed and $\hat{\Gamma}_d \subset \partial\hat{\Omega}$ as the part with Dirichlet boundary conditions. We choose $\hat{\Gamma}_d$ to be a relatively closed surface in $\partial\hat{\Omega}$ and require $\partial\hat{\Omega} = \hat{\Gamma}_f \cup \hat{\Gamma}_d$, $\hat{\Gamma}_f \cap \hat{\Gamma}_d = \emptyset$.

We assume that $\hat{\mathcal{T}}$ is chosen in such a way that the boundary faces of the simplices $\hat{S} \in \hat{\mathcal{T}}$ induce a conforming triangulation $\hat{\mathcal{R}}$ of dimension $d-1$ of $\hat{\Gamma}_f$. We split the simplices of $\hat{\mathcal{T}}$ into two sets

$$\begin{aligned}&\hat{\mathcal{T}}_i := \{S \in \hat{\mathcal{T}} \mid S \cap \hat{\Gamma}_f = \emptyset\},\\ &\hat{\mathcal{T}}_f := \hat{\mathcal{T}} \setminus \hat{\mathcal{T}}_i.\end{aligned}$$

We thus have the property

$$\hat{\mathcal{R}} = \{\hat{T} \mid \hat{T} \text{ is } d-1 \text{ dim. simplex}, \ \hat{T} = \hat{S} \cap \partial\hat{\Omega}, \ \hat{S} \in \mathcal{T}_f\}.$$

We define the finite element spaces for the discrete velocity, pressure, and mesh position in the interior and at the free boundary as follows:

$$\begin{aligned}
\hat{\boldsymbol{V}}_h &= \{\boldsymbol{w}_h \in C^0(\overline{\hat{\Omega}})^d \mid \boldsymbol{w}_h|_S \in \mathbb{P}_2(S)^d \text{ for all } S \in \hat{\mathcal{T}}, \boldsymbol{w}_h|_{\Gamma_d} = 0\},\\
\hat{Q}_h &= \{q_h \in C^0(\overline{\hat{\Omega}}) \mid q_h|_S \in \mathbb{P}_1(S) \text{ for all } S \in \hat{\mathcal{T}}\},\\
\hat{M}_h &= \{\boldsymbol{\Phi}_h \in C^0(\overline{\hat{\Omega}})^d \mid \boldsymbol{\Phi}_h|_S \in \mathbb{P}_1(S)^d \text{ for } S \in \hat{\mathcal{T}}_i \text{ and } \boldsymbol{\Phi}_h|_S \in \mathbb{P}_2(S)^d \text{ for } S \in \hat{\mathcal{T}}_f\},\\
\hat{R}_h &= \{\boldsymbol{x}_h \in C^0(\overline{\hat{\Gamma}}_f)^d \mid \boldsymbol{x}_h|_T \in \mathbb{P}_2(T)^d \text{ for } T \in \hat{\mathcal{R}}\}.
\end{aligned}$$

In other words, we use the Taylor–Hood element of order 1 for velocity and pressure, piecewise linears for the interior simplices, and piecewise quadratics for the boundary simplices. The choice of piecewise quadratics implies the use of curved simplices at the free boundary which is desirable for precise modeling of discrete curvature; cf. [3].

Given a transformation $\boldsymbol{\Phi}_h^n \in \hat{M}_h$ at time step n, we define the transformed mesh and spaces as follows:

$$\begin{aligned}
\Omega_n &:= \boldsymbol{\Phi}_h^n(\hat{\Omega}), & \Gamma_d &:= \boldsymbol{\Phi}_h^0(\hat{\Gamma}_d) \text{ (stays fixed)},\\
\Gamma_{f,n} &:= \boldsymbol{\Phi}_h^n(\hat{\Gamma}_f), & \mathcal{T}^n &:= \{\boldsymbol{\Phi}_h^n(S) \mid S \in \hat{\mathcal{T}}\},\\
\mathcal{T}_i^n &:= \{\boldsymbol{\Phi}_h^n(S) \mid S \in \hat{\mathcal{T}}_i\}, & \mathcal{T}_f^n &:= \{\boldsymbol{\Phi}_h^n(S) \mid S \in \hat{\mathcal{T}}_f\},\\
\mathcal{R}^n &:= \{\boldsymbol{\Phi}_h^n(T) \mid T \in \hat{\mathcal{R}}\}, & \boldsymbol{\Psi}_h^n &:= (\boldsymbol{\Phi}_h^n)^{-1} : \Omega_n \to \hat{\Omega},\\
\boldsymbol{V}_{h,n} &:= \{\hat{\boldsymbol{w}}_h \circ \boldsymbol{\Psi}_h^n \mid \hat{\boldsymbol{w}}_h \in \hat{\boldsymbol{V}}_h\}, & Q_{h,n} &:= \{\hat{q}_h \circ \boldsymbol{\Psi}_h^n \mid \hat{q}_h \in \hat{Q}_h\},\\
M_{h,n} &:= \{\hat{\boldsymbol{x}}_h \circ \boldsymbol{\Psi}_h^n \mid \hat{\boldsymbol{x}}_h \in \hat{M}_h\}, & R_{h,n} &:= \{\hat{\boldsymbol{x}}_h \circ \boldsymbol{\Psi}_h^n \mid \hat{\boldsymbol{x}}_h \in \hat{R}_h\}.
\end{aligned} \tag{4.5}$$

Since the domain will change with time, we define time-dependent bilinear forms on the domain $\Omega_n = \boldsymbol{\Phi}_h^n(\hat{\Omega})$ for a variational formulation:

$$\begin{aligned}
&a_n : \boldsymbol{V}_{h,n} \times \boldsymbol{V}_{h,n} \to \mathbb{R},\\
&a_n(\boldsymbol{v}_h, \boldsymbol{w}_h) = \frac{\nu_1}{2} \int_{\Omega_n} D(\boldsymbol{v}_h) : D(\boldsymbol{w}_h) + (\nu_2 - \nu_1) \int_{\Omega_n} (\nabla \cdot \boldsymbol{v}_h)(\nabla \cdot \boldsymbol{w}_h),\\
&b_n : Q_{h,n} \times \boldsymbol{V}_{h,n} \to \mathbb{R},\\
&b_n(q_h, \boldsymbol{v}_h) = -\int_{\Omega_n} q_h \nabla \cdot \boldsymbol{v}_h,\\
&d_n : R_{h,n} \times R_{h,n} \to \mathbb{R},\\
&d_n(\boldsymbol{v}_h, \boldsymbol{w}_h) = \tau_n \sigma (d-1) \int_{\Gamma_{f,n}} \nabla_{\Gamma_{f,n}} \boldsymbol{v}_h \cdot \nabla_{\Gamma_{f,n}} \boldsymbol{w}_h \, d\mathcal{H}^{d-1}.
\end{aligned} \tag{4.6}$$

Two more tools are needed to define the discretized free boundary algorithm: a discrete outer unit normal vector $\boldsymbol{n}_h^n \in R_{h,n}$ to the mesh boundary $\Gamma_{f,n}$ and a suitable extension operator $E_h : \hat{R}_h \times \hat{M}_h \to \hat{M}_h$.

The function $\boldsymbol{n}_h^n \in R_{h,n}$ is defined by setting the values in the nodes of the quadratic Lagrange mesh G_2 of $\Gamma_{f,n}$. If $T \in \mathcal{R}_n$ is a $d-1$-dimensional simplex, we set

$$\begin{aligned}
\widetilde{\boldsymbol{n}}_T &:= \boldsymbol{n}_T \mathcal{H}^{d-1}(T),\\
P(x) &:= \{T' \in \mathcal{R}_n \mid x \in T'\}.
\end{aligned}$$

Here $\boldsymbol{n}_T$ is the (well-defined) outer unit normal vector on T. Now define $\boldsymbol{n}_h^n \in R_{h,n}$ as

$$\boldsymbol{n}_h^n(\overline{x}) := \begin{cases} \frac{\sum_{T\in P(\overline{x})} \tilde{\boldsymbol{n}}_T}{|\sum_{T\in P(\overline{x})} \tilde{\boldsymbol{n}}_T|} & \text{if } \overline{x} \in \Gamma_{f,n}, \\ 0 & \text{otherwise} \end{cases}$$

for all nodes $\overline{x}$ of G_2. This means that the (well-defined) normal on faces is averaged to define values on corners or edges.

The extension operator E_h is used to define the interior nodes of Ω_n, based on the position of $\Gamma_{f,n}$ and the old mesh Ω_{n-1}:

$$\boldsymbol{\Phi}_h^n = E_h(\boldsymbol{\Phi}_h^n|_{\hat{\Gamma}_f}, \boldsymbol{\Phi}_h^{n-1}).$$

This is necessary, since the boundary condition (4.1e) prescribes only the motion of the free boundary. The interior nodes should be selected in such a manner as to guarantee a good quality of the mesh triangulation $\mathcal{T}^n$.

ALGORITHM 4.2 (free boundary discretization). *Let $\boldsymbol{\Phi}_h^0 \in \hat{M}_h$ and $\boldsymbol{v}_h^0 \in \boldsymbol{V}_{h,0}$, $p_h^0 \in Q_{h,0}$, as well as $\boldsymbol{f}_v^n$, f_p^n as locally integrable functions on $\mathbb{R}^d$, $n \in \mathbb{N}$, and $\boldsymbol{g}_h$ as the extension by zero of given Dirichlet boundary conditions for the velocity. Loop over $n = 1, 2, \ldots$:*

1. *Solve for $(\tilde{\boldsymbol{v}}_h^n, \tilde{p}_h^n) \in (\boldsymbol{g}_h + \boldsymbol{V}_{h,n-1}) \times Q_{h,n-1}$:*

$$\begin{aligned} (4.7)\quad \frac{1}{\tau_n}\int_{\Omega_{n-1}} \boldsymbol{\psi}_h \cdot \tilde{\boldsymbol{v}}_h^n + a_{n-1}(\boldsymbol{\psi}_h, \tilde{\boldsymbol{v}}_h^n) + d_{n-1}(\boldsymbol{\psi}_h, \tilde{\boldsymbol{v}}_h^n) + b_{n-1}(\tilde{p}_h^n, \boldsymbol{\psi}_h) \\ = \frac{1}{\tau_n}\int_{\Omega_{n-1}} \boldsymbol{\psi}_h \cdot \boldsymbol{v}_h^{n-1} + \int_{\Omega_{n-1}} \boldsymbol{\psi}_h \cdot \boldsymbol{f}_v^{n-1} \\ - \sigma(d-1)\int_{\Gamma_{f,n-1}} \nabla_{\Gamma_{f,n-1}} \cdot \boldsymbol{\psi}_h \, d\mathcal{H}^{d-1} \end{aligned}$$

for all $\boldsymbol{\psi}_h \in \boldsymbol{V}_{h,n-1}$ and

$$\begin{aligned} (4.8)\quad \frac{1}{\tau_n}\int_{\Omega_{n-1}} \varphi_h \cdot \tilde{p}_h^n - b_{n-1}(\varphi_h, \tilde{\boldsymbol{v}}_h) \\ = \frac{1}{\tau_n}\int_{\Omega_{n-1}} \varphi_h \cdot p_h^{n-1} + \int_{\Omega_{n-1}} \varphi_h f_p^{n-1} \end{aligned}$$

for all $\varphi_h \in Q_{h,n-1}$.

2. *Update the position of the free boundary as follows: Define*

$$\begin{aligned} \boldsymbol{\Delta}_h^{n-1} &:= \tau_n(\tilde{\boldsymbol{v}}_h^n|_{\Gamma_{f,n}} \cdot \boldsymbol{n}_h^{n-1})\boldsymbol{n}_h^{n-1} \in R_{h,n-1}, \\ \boldsymbol{\Phi}_h^n|_{\hat{\Gamma}_f} &:= \left(\mathrm{id}_{\Gamma_{f,n-1}} + \boldsymbol{\Delta}_h^n\right) \circ \boldsymbol{\Phi}_h^{n-1}|_{\hat{\Gamma}_f}. \end{aligned}$$

3. *Now update the position of the entire mesh:*

$$\boldsymbol{\Phi}_h^n := E_h(\boldsymbol{\Phi}_h^n|_{\hat{\Gamma}_f}, \boldsymbol{\Phi}_h^{n-1}),$$

with a suitable extension operator E_h. If $\boldsymbol{\Phi}_h^n$ is not an admissible mesh transformation according to Assumption 4.1, then STOP.

4. *Finally, set*

$$\begin{aligned}
\Omega_n &:= \boldsymbol{\Phi}_h^n(\hat{\Omega}),\\
\Gamma_{f,n} &:= \boldsymbol{\Phi}_h^n(\hat{\Gamma}_f),\\
\mathcal{T}^n &:= \{\boldsymbol{\Phi}_h^n(S) \mid S \in \hat{\mathcal{T}}\},\\
\mathcal{R}^n &:= \{\boldsymbol{\Phi}_h^n(T) \mid T \in \hat{\mathcal{R}}\},\\
\boldsymbol{\Psi}_h^n &:= (\boldsymbol{\Phi}_h^n)^{-1},\\
\boldsymbol{v}_h^n &:= \tilde{\boldsymbol{v}}_h^n \circ \boldsymbol{\Phi}_h^{n-1} \circ \boldsymbol{\Psi}_h^n,\\
p_h^n &:= \tilde{p}_h^n \circ \boldsymbol{\Phi}_h^{n-1} \circ \boldsymbol{\Psi}_h^n.
\end{aligned}$$

Remark 4.3. Several remarks about the free boundary algorithm are in order.

- The algorithm employs a decoupling of the bulk equations and the geometry updates, which would otherwise be coupled in a nonlinear way.
- The discretization has the following property:

(4.9)
$$\begin{aligned}
&-d_{n-1}(\boldsymbol{\psi}_h, \tilde{\boldsymbol{v}}_h^n) - \sigma(d-1)\int_{\Gamma_{f,n-1}} \nabla_{\Gamma_{f,n-1}} \cdot \boldsymbol{\psi}_h \, d\mathcal{H}^{d-1}\\
&\quad = -\sigma(d-1)\int_{\Gamma_{f,n-1}} \nabla_{\Gamma_{f,n-1}} \boldsymbol{\psi}_h \cdot \nabla_{\Gamma_{f,n-1}} (\mathrm{id}_{\Gamma_{f,n-1}} + \tau_n \tilde{\boldsymbol{v}}_h^n) \, d\mathcal{H}^{d-1},\\
&\quad \approx \sigma(d-1)\int_{\Gamma_{f,n}} \boldsymbol{\psi}_h \cdot \boldsymbol{\kappa} \, d\mathcal{H}^{d-1};
\end{aligned}$$

  cf. (4.3). The capillary boundary condition is thus treated in a semi-implicit manner, since information about the new boundary position $\Gamma_{f,n}$ enters the definition of the curvature term. Furthermore, note that the term d_n adds additional coercivity on the left-hand side.
  The idea for this discretization of curvature was taken from [3], where it is crucial for proving stability of a discontinuous Galerkin scheme for the free boundary incompressible Navier–Stokes equations.
- We have defined the update of the free boundary as

$$\boldsymbol{\Phi}_h^n|_{\hat{\Gamma}_f} := \big(\mathrm{id}_{\Gamma_{f,n-1}} + \tau_n (\tilde{\boldsymbol{v}}_h^n|_{\Gamma_{f,n}} \cdot \boldsymbol{n}_h^{n-1}) \boldsymbol{n}_h^{n-1}\big) \circ \boldsymbol{\Phi}_h^{n-1}|_{\hat{\Gamma}_f}.$$

  A simpler formula, motivated by (4.9), is

$$\boldsymbol{\Phi}_h^n|_{\hat{\Gamma}_f} := \big(\mathrm{id}_{\Gamma_{f,n-1}} + \tau_n \tilde{\boldsymbol{v}}_h^n|_{\Gamma_{f,n}}\big) \circ \boldsymbol{\Phi}_h^{n-1}|_{\hat{\Gamma}_f}. \tag{4.10}$$

  This method is actually used during the mesh update Algorithm 4.4 below. However, in the case of large variations of the tangential velocity at the boundary (4.10) may lead to a quick distortion of the mesh, making the definition of a suitable mesh extension E_h even more difficult. Moving the free boundary only in the normal direction is justified since the set $\Gamma_{f,n}$ remains essentially the same—only the nodes defining the triangulation $\mathcal{R}_n$ are placed differently.

We conclude the discussion of the algorithm by presenting two implemented algorithms to define the mesh extension $E_h^n : \hat{R}_h \times \hat{M}_h \to \hat{M}_h$.

ALGORITHM 4.4 (mesh update algorithm 1). *Given old mesh positions* $\boldsymbol{\Phi}_h^{n-1} \in \hat{R}_h$ *and the velocity field* $\tilde{\boldsymbol{v}}_h^n \in \boldsymbol{V}_{h,n-1}$ *of step* 1 *in Algorithm* 4.2*, define* $\boldsymbol{\Phi}_h^n =$

$E(\boldsymbol{\Phi}_h^n|_{\hat{\Gamma}_f}, \boldsymbol{\Phi}_h^{n-1})$ as follows:

$$\boldsymbol{\Phi}_h^n|_{\hat{\Omega}\cup\hat{\Gamma}_f} := \left(\mathrm{id}_{\Omega_{n-1}} + \tau_n \tilde{\boldsymbol{v}}_h^n\right) \circ \boldsymbol{\Phi}_h^{n-1}|_{\hat{\Omega}\cup\hat{\Gamma}_f},$$
$$\boldsymbol{\Phi}_h^n|_{\hat{\Gamma}_d} := \boldsymbol{\Phi}_h^0|_{\hat{\Gamma}_d}.$$

In other words, all interior nodes as well as the nodes on the free boundary are moved with the velocity field $\tilde{\boldsymbol{v}}_h^n$.

ALGORITHM 4.5 (mesh update algorithm 2). *Given are old mesh positions $\boldsymbol{\Phi}_h^{n-1} \in \hat{R}_h$ as well as the new positions of the free boundary $\boldsymbol{\Phi}_h^n|_{\hat{\Gamma}_f}$. Determine $\boldsymbol{\Xi}_h \in M_{h,n-1}$ (see (4.5)) as the solution of the elliptic system*

$$\int_{\Omega_{n-1}} \nabla\boldsymbol{\psi}_h : \nabla\boldsymbol{\Xi}_h = 0 \qquad \text{for all } \boldsymbol{\psi}_h \in M_{h,n-1},$$
$$\boldsymbol{\Xi}_h = \boldsymbol{\Phi}_h^n|_{\hat{\Gamma}_f} \circ \boldsymbol{\Psi}_h^{n-1} \quad \text{on } \Gamma_{f,n-1},$$
$$\boldsymbol{\Xi}_h = \mathrm{id}_{\Gamma_d} \qquad \text{on } \Gamma_d.$$

Now set $\boldsymbol{\Phi}_h^n = E(\boldsymbol{\Phi}_h^n|_{\hat{\Gamma}_f}, \boldsymbol{\Phi}_h^{n-1})$ as

$$\boldsymbol{\Phi}_h^n := \boldsymbol{\Xi}_h \circ \boldsymbol{\Phi}_h^{n-1}.$$

This idea of using the Laplace operator to smooth the mesh nodes was taken from [3]. Although simple and efficient to implement, this method has certain drawbacks; e.g., it cannot deal with nonconvex domains without modifications. Nevertheless, this algorithm was sufficient for our purposes. We refer to [39] for more sophisticated ideas on mesh smoothing.

5. Discretization. The simulation of acoustic streaming induced by SAWs was realized by the two-stage model already introduced in section 2. We used the second order Taylor–Hood finite element for the spatial discretization of the Stokes-like acoustic and acoustic streaming subproblems. The resulting discrete saddle point problem is solved using a CG iteration to invert the corresponding Schur complement of the system. For details we refer to [8]. To achieve a temporal discretization we employed either backward Euler or the Crank–Nicolson scheme.

We now present detailed algorithms to calculate solutions. We differentiate between the fixed mesh case and the case of free capillary boundaries.

5.1. Fixed meshes.

ALGORITHM 5.1 (acoustic streaming, fixed domain). *Given is a computational domain $\Omega \subset \mathbb{R}^d$, physical parameters, and a periodic function $\boldsymbol{u}$ as the SAW displacement on $\Gamma_d = \partial\Omega$.*

1. *Solve the acoustics subproblem (2.4) for $(\boldsymbol{v}^{(1)}, p^{(1)})$. We are interested in the periodic equilibrium state corresponding to the periodic forcing terms. Two methods are implemented:*
   (a) *Perform a time discretization with fixed time step size $\tau^{(1)} > 0$ together with Taylor–Hood finite elements with mesh parameter h. We perform either a fixed number of time steps or iterate until a given periodicity condition is fulfilled at a time t_{end}.*
   (b) *Employ the quasi-stationary approach (no time discretization) as described in section 3.4.*
2. *Calculate the right-hand terms and boundary conditions for (2.7).*

3. *Solve the acoustic streaming subproblem for* $(\boldsymbol{v}^{(2)}, p^{(2)})$. *Here we have the following alternatives:*
   (a) *We solve the stationary problem* (2.7).
   (b) *We solve the instationary variant* (2.6) *with fixed time step size* $\tau^{(2)}$. *This is primarily of interest when determining the relaxation time of the acoustic streaming flow.*

5.2. Free capillary boundaries. For the simulation of acoustic streaming with free capillary boundaries, we will need physical parameters as before and additionally a coefficient σ of surface tension. A typical value would be $\sigma = 7.3 \cdot 10^{-2}$ N/m.

The basis of this algorithm was already presented as Algorithm 4.2. It remains to specify the right-hand sides $\boldsymbol{f}_v^n$, f_p^n for each time step. These are now determined by solving the acoustics subproblem as above.

ALGORITHM 5.2 (acoustic streaming, free boundaries). *Given is a starting domain* $\Omega_0 \subset \mathbb{R}^d$ *and a periodic function* $\boldsymbol{u}$ *as the SAW displacement on* $\Gamma_d \subset \partial\Omega$. *Loop over* $n = 0, 1, 2, \ldots$:

1. *Solve the acoustics subproblem* (2.4) *for* $(\boldsymbol{v}^{(1)}, p^{(1)})$ *on the domain* Ω_n *up to the periodic equilibrium state. This is done as above in Algorithm* 5.1, *with either a time discretization or the quasi-stationary approach. In the former case iterate until periodicity is reached.*
2. *Calculate right-hand terms* $\boldsymbol{f}_v^{(2),n}$, $f_p^{(2),n}$ *and boundary conditions for the acoustic streaming solution at time step* t_n *using* $\boldsymbol{v}^{(1)}$ *and* $p^{(1)}$ *as above.*
3. *Perform step* 1 *of Algorithm* 4.2 *to determine the acoustic streaming solution* $(\boldsymbol{v}^{(2),n}, p^{(2),n})$ *at time step* t_n *with* $\tau_n := \tau^{(2)}$.
4. *Update the position of the mesh to determine* Ω_{n+1}, *as described in steps* 2 *and* 3 *of Algorithm* 4.2. *We have the alternatives* 4.4 *and* 4.5 *as mesh smoothing methods.*

Remark 5.3. Recalculating the acoustics solution $(\boldsymbol{v}^{(1)}, p^{(1)})$ in each time step t_n of the main iteration is extremely costly. In practice we have relaxed this requirement by continuing to use $\boldsymbol{f}_v^{(2),n}$, $f_p^{(2),n}$ for several time steps before recomputing. Numerical experiments show that the additional error involved does not have a significant effect on the equilibrium state of the solution $(\Omega_n, \boldsymbol{v}^{(2),n}, p^{(2),n})$ reached for large n.

6. Numerical results. The goal of this section is to present simulation results of SAW-induced streaming with realistic physical parameters. The computational effort for solving fixed domain problems is still acceptable for our 3D setup—the simulation of streaming effects including free capillary boundaries, however, is possible only in a simple two-dimensional (2D) configuration using the current implementation. The reason for this is that the change of the domain in each time step necessitates the recalculation of the acoustic field solution; see below.

All calculations were performed on AMD Opteron$^{\text{TM}}$ 64 bit 2.6 GHz dual-head processors with 8 GB memory. We used the freely available ALBERTA finite element toolbox, [41].

6.1. Fixed domains.

6.1.1. 2D simulation. We refer to (2.1) for the notation. The 2D simulation setting is a fluid-filled square cavity $\Omega = [0, 1 \text{ mm}]^2$ with SAW displacements $\boldsymbol{u}$ prescribed on the bottom edge. The velocity $\boldsymbol{v}$ is set to zero on the other three edges. The SAW propagates from left to right with exponential attenuation. The exact shape assumed in this work is based on wavelengths, wave velocities, amplitudes,

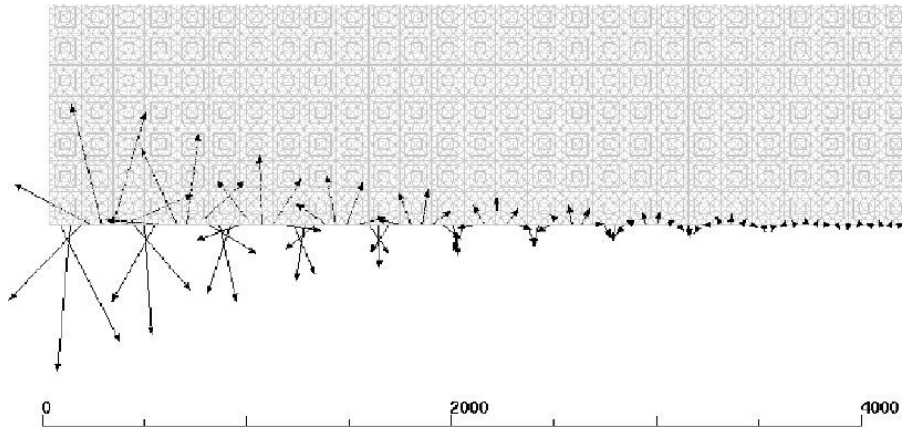


FIG. 6.1. *Detail of computational mesh and SAW displacement field in the bottom left corner of the domain with a computational length scale (*1 *unit* $\equiv 10^{-1}\mu$m*).*

TABLE 6.1
Numerical and physical parameters for the 2*D SAW simulation.*

Parameter	Value and units	Description
V	$1.0 \cdot 10^{-1}$ m/s	Dimensionless velocity scale
L	$1.0 \cdot 10^{-7}$ m	Dimensionless length scale
T	$1.0 \cdot 10^{-8}$ s	Dimensionless time scale
f	$1.0 \cdot 10^{8}$ Hz	Frequency of the SAW device
$c_{0,l}$	$1.484 \cdot 10^{3}$ m/s	Small signal sound speed in water
$c_{0,s}$	$3.8 \cdot 10^{3}$ m/s	Small signal sound speed in solid
ρ_0	$1.0 \cdot 10^{3}$ kg/m^3	Density of liquid
u_0	$1.0 \cdot 10^{-9}$ m	Maximal SAW displacement
C_d	$8.06 \cdot 10^{3}$ 1/m	Damping parameter of the LSAW
ν_1	$1.002 \cdot 10^{-6}$ m^2/s	Kinematic viscosity of water
ν_2	$1.002 \cdot 10^{-6}$ m^2/s	Kinematic bulk viscosity of water

and damping parameters of a typical lithium niobate crystal as described in [18]; see Figure 6.1.

This setup approximates the case of a vertically oriented thin slice of water trapped in a container and touching the SAW-traversed solid surface at the bottom. The fluid is assumed to be initially at rest, i.e., $\boldsymbol{v}_0 = 0$, $p_0 = 0$.

Data of the 2*D SAW device simulation.* The function $\boldsymbol{u}$ to describe the SAW displacement at the lower edge is defined in dimensionless form as

$$
\begin{aligned}
u_1(t, \boldsymbol{x}) &= 0.6\epsilon \exp(-\widehat{C}_d x_1) \sin(2\pi(-\widehat{k} x_1 + \widehat{f} t)), \\
u_2(t, \boldsymbol{x}) &= -\epsilon \exp(-\widehat{C}_d x_1) \cos(2\pi(-\widehat{k} x_1 + \widehat{f} t))
\end{aligned}
\tag{6.1}
$$

for $\boldsymbol{x} \in \partial\Omega$, $x_2 = 0$ with parameters $\epsilon = u_0/L$, $\widehat{C}_d = C_d L$, $\widehat{k} = L/\lambda$, $\widehat{f} = fT$. Note that the discretization parameter ϵ is 0.1; confer (2.3). Additional damping terms, not mentioned here, serve to avoid sharp jumps of the function $\boldsymbol{u}$ at $\boldsymbol{x} = 0$ and $t = 0$. The other parameters used in the simulation are given in Table 6.1.

As described in section 2 the acoustic streaming field $(\boldsymbol{v}^{(2)}, p^{(2)})$ is calculated via a two-stage algorithm using the intermediate acoustics field solution $(\boldsymbol{v}^{(1)}, p^{(1)})$. We use the standard Crank–Nicolson time discretization algorithm for parabolic problems

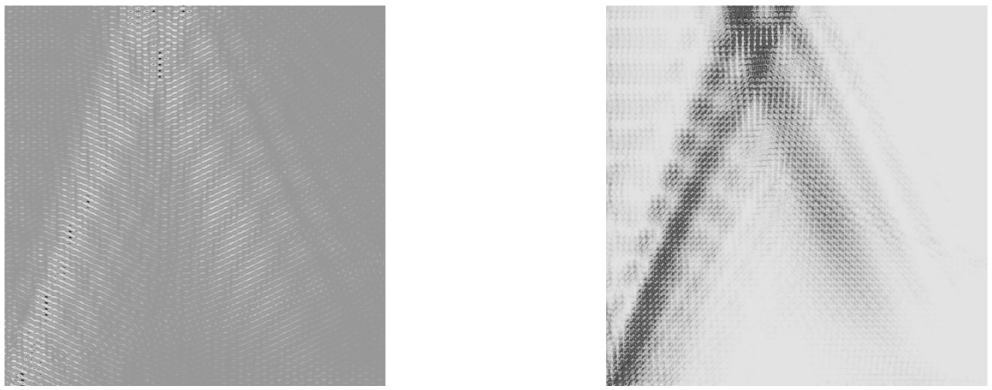


FIG. 6.2. *Left: Acoustics subproblem solution: pressure field* $p^{(1)}$ *at time* $t = 142 \equiv 1.42$ μs. *Right: Plot of effective force* f_v *for the acoustic streaming solution. Clearly visible is the reflection of the pressure wave at the top boundary of the domain as well as the typical Rayleigh angle corresponding to the ratio of sound speeds in liquid and substrate.*

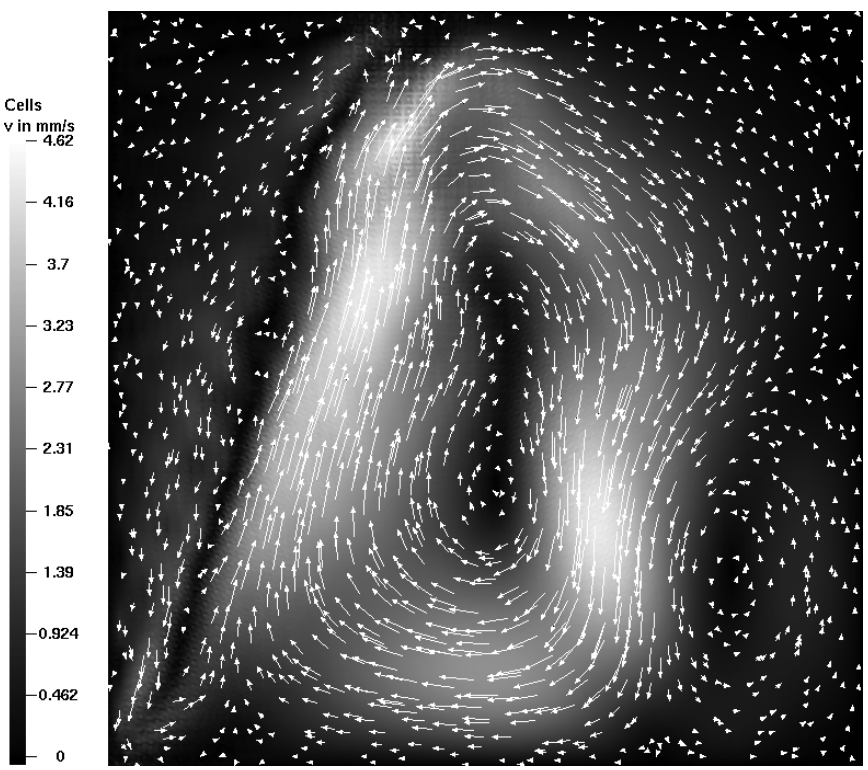


FIG. 6.3. *Acoustic streaming solution: velocity field* $v^{(2)}$ *in* mm/s.

to solve for the acoustics field $(\boldsymbol{v}^{(1)}, p^{(1)})$ at each time step. We calculate a sufficient number of time steps in order to reach an oscillating equilibrium state.

Results of the 2D SAW device simulation. The CPU time necessary for the full calculation was approximately two hours. See Figures 6.2 and 6.3.

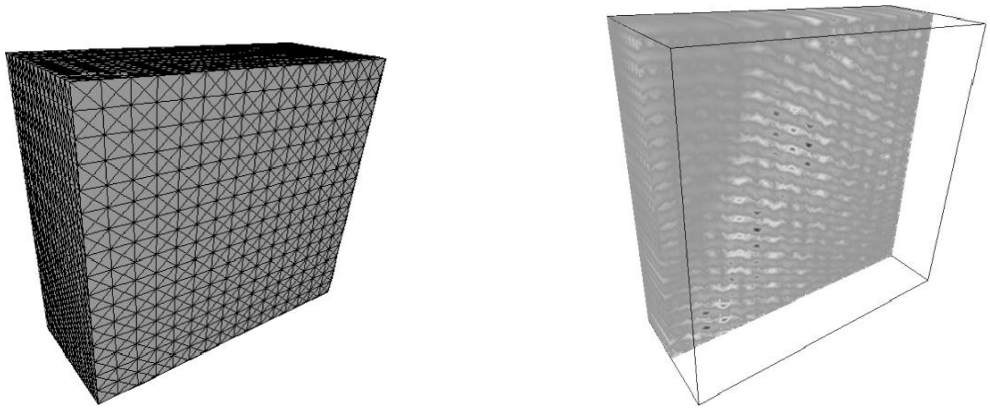


FIG. 6.4. *Left: Domain for the 3D simulation runs. Right: Acoustics subproblem solution: slab* $x_3 \geq 0$ *of the pressure field* $p^{(1)}$ *at time* $t = 200 \equiv 2$ *μs.*

6.1.2. 3D simulation. The parameters used for the 3D simulation are similar to the 2D case. The domain will again be a fluid-filled rectangular cavity. We set $\Omega = [0, 250\ \mu\text{m}]^2 \times [-50\ \mu\text{m}, 50\ \mu\text{m}]$; see Figure 6.4. The SAW traverses the bottom face; on the other walls, we again prescribe zero Dirichlet boundary conditions for fluid velocity.

Data of the 3D SAW device simulation. The parameters are set to the same values as before. The function $\boldsymbol{u}$ is defined in dimensionless form as

$$\begin{aligned} u_1(t, \boldsymbol{x}) &= 0.6\epsilon \exp(-\widehat{C}_d x_1) \sin(2\pi(-\widehat{k}x_1 + \widehat{f}t)) d(x_3), \\ u_2(t, \boldsymbol{x}) &= -\epsilon \exp(-\widehat{C}_d x_1) \cos(2\pi(-\widehat{k}x_1 + \widehat{f}t)) d(x_3), \\ u_3(t, \boldsymbol{x}) &= 0. \end{aligned} \tag{6.2}$$

The terms have the same meaning as in (6.1) above. What is new is the term $d(x_3)$, defined as

$$d(x_3) = 4 \cdot 10^{-6}(x_3 - 500)(x_3 + 500).$$

Observing that the domain depth 100 μm corresponds to the dimensionless depth of 1000 units, we have thus extended the SAW shape of the 2D simulation to a parabolic profile over the bottom face.

Results of the 3D SAW device simulation. The CPU time necessary for the full calculation was approximately six days. The acoustic streaming velocity values $\boldsymbol{v}^{(2)}$ are shown in Figure 6.5.

We additionally took a vertical cross section of the 3D mesh to define a 2D mesh, together with values of the SAW displacement field $\boldsymbol{u}$ along the cross section. This permits a comparison of a 3D solution with a corresponding 2D solution. Figure 6.6 shows that the result is qualitatively close to the 3D field.

6.2. Free boundaries. As described above, we will restrict ourselves to performing a 2D simulation. We will describe the deformation of an originally semicircular drop of water of radius $2500 \equiv 0.25$ mm posed above a SAW wave of the same shape as in the fixed domain 2D simulations. The maximal SAW amplitude was $\boldsymbol{u}_0 = 100$ nm, a much higher value compared to the fixed domain simulations.

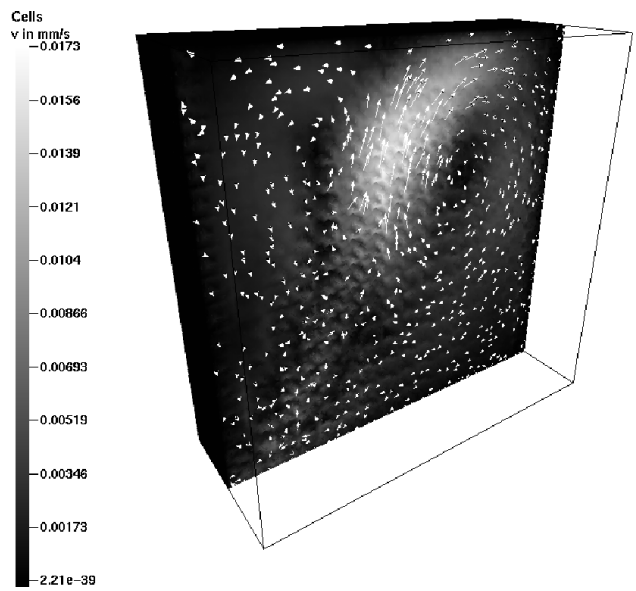


FIG. 6.5. *Acoustic streaming solution: slab* $x_3 \geq 0$ *of the velocity field* $v^{(2)}$.

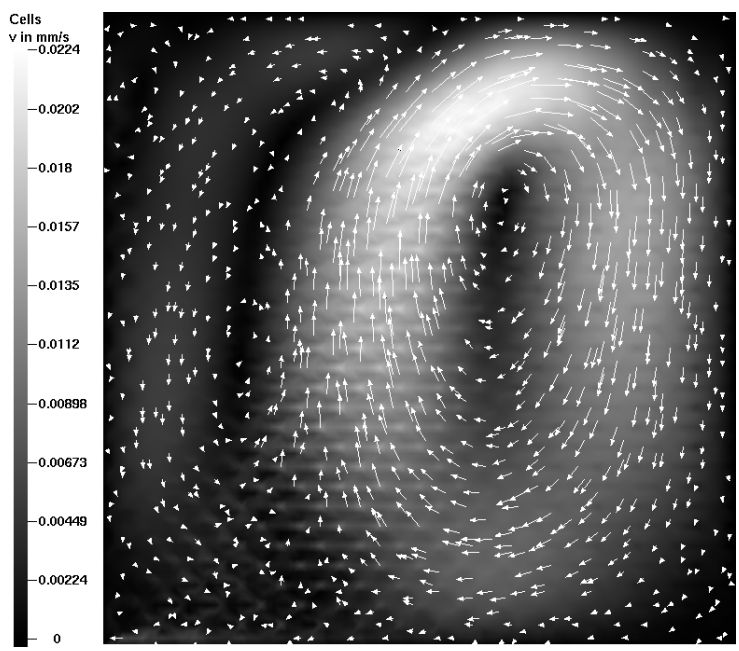


FIG. 6.6. *Comparison with the corresponding* $2D$ *simulation: acoustic streaming velocity field* $v^{(2)}$.

The coefficient of surface tension assumed in the simulations is $\sigma = 7.3 \cdot 10^{-2}$ N/m corresponding to a water/vacuum interface.

We solved 200 microscopic time steps of length $\tau^{(1)} = 0.1 \equiv 1.0 \cdot 1$ ns to determine the equilibrium acoustics solution needed for each macroscopic time step. The acoustic streaming solution was calculated using macroscopic time steps of length $\tau^{(2)} = 1000 \equiv 10$ μs.

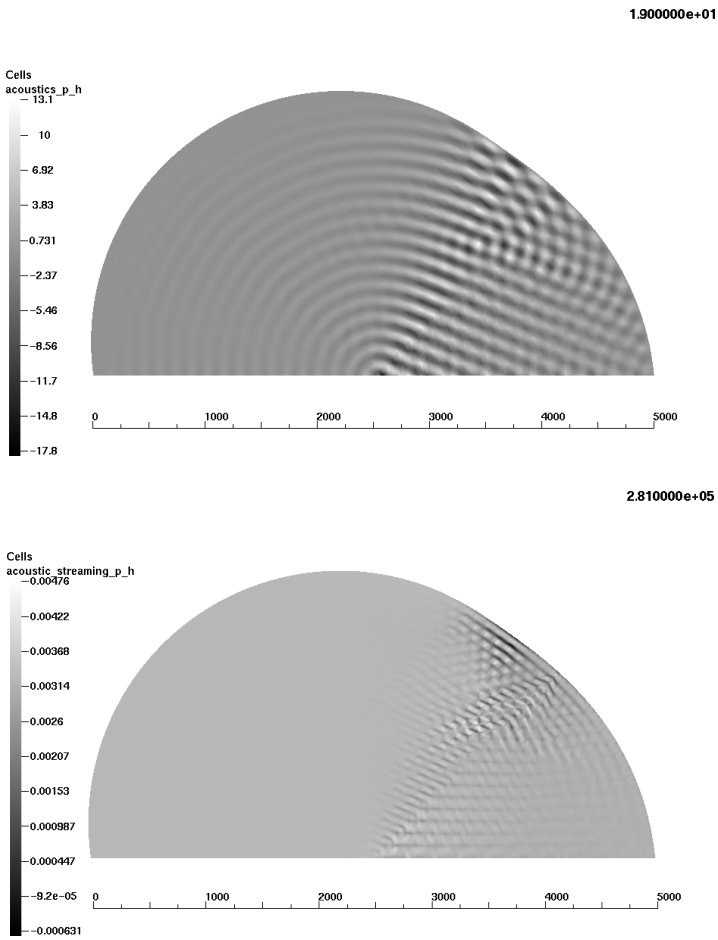


FIG. 6.7. *Free boundary solution. Top: acoustic pressure field* $p^{(1)}$ *at microscopic time* $t^{(1)} = 19$ *and macroscopic time* $t^{(2)} = 2.81 \cdot 10^5$. *Bottom: acoustic streaming pressure field* $p^{(2)}$ *at macroscopic time* $t^{(2)} = 2.81 \cdot 10^5$.

One important change was done in the calculations. The mass generation rate $f_p^{(2),n}$ and outflow $\boldsymbol{g}^{(2),n} \cdot \boldsymbol{n}$ used in step 3 of Algorithm 4.2 will, in general, not be balanced. This implies a change in the pressure mean over time. In the presence of free capillary boundaries a change in the overall pressure leads to a change in volume—this is undesirable and nonphysical behavior. We therefore restored the balance between outflow and mass generation in the numerical calculations by multiplying $\boldsymbol{g}^{(2),n}$ with a correction factor. Numerical experiments performed without this correction indeed displayed a continuously decreasing volume of the droplets.

Results of the free boundary SAW simulation. The maximal CPU time for the calculation of 10 time steps was approximately five hours. The observed behavior of the drop is characterized by initial damped oscillations. A deformed equilibrium position is obtained after a short period of time.

Figure 6.7 shows the acoustic and acoustic streaming pressure fields $p^{(1)}$ (resp., $p^{(2)}$) in the droplet at different times.

REFERENCES

[1] H. W. Alt, *Lineare Funktionalanalysis*, Springer-Verlag, Berlin, 1999.
[2] H. W. Alt, *Analysis* III, Lecture notes at the University of Bonn, 2004.
[3] E. Bänsch, *Numerical Methods for the Instationary Navier–Stokes Equations with a Free Capillary Surface*, Habilitation thesis, University of Freiburg, 1998.
[4] M. Benzi and V. Simoncini, *On the eigenvalues of a class of saddle point matrices*, Numer. Math., 103 (2006), pp. 173–196.
[5] N. D. Botkin and V. L. Turova, *Mathematical models of a biosensor*, Applied Mathematical Modelling, 28 (2004), pp. 573–589.
[6] C. E. Bradley, *Acoustic streaming field structure: The influence of the radiator*, J. Acoust. Soc. Am., 100 (1996), pp. 1399–1408.
[7] F. Brezzi and M. Fortin, *Mixed and Hybrid Finite Element Methods*, Springer-Verlag, Berlin, 1991.
[8] M. O. Bristeau, R. Glowinski, and J. Periaux, *Numerical methods for the Navier–Stokes equations. Applications to the simulation of compressible and incompressible viscous flows*, Comp. Phys. Rep., 6 (1987), pp. 73–187.
[9] M. P. D. Carmo, *Differentialgeometrie von Kurven und Flächen*, Vieweg Verlag, Wiesbaden, 1998.
[10] K. Chakrabarty and J. Zeng, *Design automation for microfluidics-based biochips*, ACM Journal on Emerging Technologies in Computing Systems, 1 (2005), pp. 186–223.
[11] B. Desjardins and C.-K. Lin, *A survey of the compressible Navier–Stokes equations*, Taiwanese J. Math., 3 (1999), pp. 123–137.
[12] G. Dziuk, *Finite elements for the Beltrami operator on arbitrary surfaces*, in Partial Differential Equations and Calculus of Variations, S. Hildebrandt and R. Leis, eds., Springer-Verlag, Berlin, 1988, pp. 142–155.
[13] G. Dziuk, *An algorithm for evolutionary surfaces*, Numer. Math., 58 (1991), pp. 603–611.
[14] W. Engquist and B. Engquist, *The heterogeneous multiscale methods*, Commun. Math. Sci., 1 (2003), pp. 87–132.
[15] W. E Engquist and B. Engquist, *The heterogenous multi-scale method*, http://arxiv.org/abs/physics/0205048v2, 2006.
[16] C. Eckart, *Vortices and streams caused by sound waves*, Phys. Rev., 73 (1947), pp. 68–76.
[17] A. Gantner, R. H. W. Hoppe, D. Köster, K. G. Siebert, and A. Wixforth, *Numerical simulation of piezoelectrically agitated surface acoustic waves on microfluidic biochips*, Comput. Vis. Sci., 10 (2007), pp. 145–161.
[18] Z. Guttenberg, A. Rathgeber, S. Keller, J. O. Rädler, A. Wixforth, M. Kostur, M. Schlinder, and P. Talkner, *Flow profiling of a surface acoustic wave nanopump*, Phys. Rev. E, 70 (2004), paper 056311.
[19] W. Hackbusch, *Elliptic Differential Equations*, Springer Ser. Comput. Mech., 1992.
[20] C. W. Hirt and B. D. Nichols, *Volume of fluid (VOF) method for the dynamics of free boundaries*, J. Comput. Phys., 39 (1981), pp. 201–225.
[21] J. Hoffelner, H. Landes, M. Kaltenbacher, and R. Lerch, *Finite element simulation of nonlinear wave propagation in thermoviscous fluids including dissipation*, IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, 48 (2001), pp. 779–786.
[22] A. Huerta and W. K. Liu, *Viscous flow with large free surface motion*, Comput. Methods Appl. Mech. Engrg., 69 (1988), pp. 277–324.
[23] T. J. Hughes, W. K. Liu, and T. K. Zimmermann, *Lagrangian-Eulerian finite element formulation for incompressible viscous flows*, Comput. Methods Appl. Mech. Engrg., 29 (1981), pp. 329–349.
[24] G. Karniadakis, A. Beskok, and N. Aluru, *Microflows and Nanoflows. Fundamentals and Simulation*, Springer-Verlag, Berlin, 2005.
[25] R. B. Kellogg and B. Liu, *A finite element method for the compressible Stokes equations*, SIAM J. Numer. Anal., 33 (1996), pp. 780–788.
[26] D. Köster, *Numerical Simulation of Acoustic Streaming on SAW-Driven Biochips*, Ph.D. thesis, University of Augsburg, 2006.
[27] L. D. Landau and E. M. Lifschitz, *Lehrbuch der Theoretischen Physik* VI: *Hydrodynamik*, Akademie Verlag, Berlin, 1991.
[28] J. Lighthill, *Acoustic Streaming*, Academic Press, London, 1978.
[29] J.-L. Lions and E. Magenes, *Non-homogeneous Boundary Value Problems and Applications*, Springer-Verlag, Berlin, 1972.
[30] P.-L. Lions and N. Masmoudi, *On a free boundary barotropic model*, Ann. Inst. H. Poincaré, 16 (1999), pp. 373–410.

[31] W. L. Nyborg, *Acoustic streaming*, in Physical Acoustics, W. Mason, ed., Academic Press, New York, 1965, pp. 265–331.
[32] W. L. Nyborg, *Acoustic streaming*, in Nonlinear Acoustics, M. F. Hamilton and D. T. Blackstock, eds., Academic Press, New York, 1998, pp. 207–231.
[33] S. Osher and J. Sethian, *Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi-formulations*, J. Comput. Phys., 79 (1988), pp. 12–49.
[34] P. Ciarlet, Jr., J. Huang, and J. Zou, *Some observations on generalized saddle-point problems*, SIAM J. Matrix Anal. Appl., 25 (2003), pp. 224–236.
[35] L. Rayleigh, *On the circulation of air observed in Kundt's tubes, and on some allied acoustical problems*, Philos. Trans. R. Soc. Lond., 175 (1884), pp. 1–21.
[36] N. Riley, *Oscillating viscous flows*, Mathematika, 12 (1965), pp. 161–175.
[37] N. Riley, *Acoustic streaming*, Theoret. Comput. Fluid Dyn., 10 (1998), pp. 349–356.
[38] L. D. Rozenberg, ed., *Ultrasonic Technology*, Plenum Press, New York, 1971, Chapt. High-Intensity Ultrasonic Fields.
[39] M. Rumpf, *A variational approach to optimal meshes*, Numer. Math., 72 (1996), pp. 523–540.
[40] A. Sathaye and A. Lal, *Numerical simulation of acoustic streaming near embedded microstructures inside microfluidic channels*, in Proceedings of the IEEE Ultrasonics Symposium, 2002.
[41] A. Schmidt and K. G. Siebert, *Design of Adaptive Finite Element Software: The Finite Element Toolbox ALBERTA*, Springer-Verlag, Berlin, 2005.
[42] B. Schweizer, *Free boundary fluid systems in a semigroup approach and oscillatory behavior*, SIAM J. Math. Anal., 28 (1997), pp. 1135–1157.
[43] J. A. Sethian, *Level Set Methods and Fast Marching Methods*, Cambridge Monogr. Appl. Comput. Math., 1999.
[44] P. Smereka, *Level Set Methods for Two-Fluid Flows*, Lecture notes from a short course given at INRIA, 1996.
[45] V. A. Solonnikov, *On some free boundary problems for the Navier–Stokes equations with moving contact points and lines*, Math. Ann., 302 (1995), pp. 743–772.
[46] M. Sussman, E. Fatemi, P. Smereka, and S. Osher, *An improved level set method for incompressible two-phase flows*, Comput. & Fluids, 27 (1997).
[47] M. Sussman, P. Smereka, and S. J. Osher, *A level set approach for computing solutions to incompressible two-phase flow*, J. Comput. Phys., 114 (1994), pp. 146–159.
[48] T. Uchida, T. Suzuki, and S. Shiokawa, *Investigation of acoustic streaming excited by surface acoustic waves*, in Proceedings of the 1995 IEEE Ultrasonics Symposium, IEEE, 1995, pp. 1081–1084.
[49] A. Wixforth, C. Strobl, C. Gauer, et al., *Acoustic manipulation of small droplets*, Anal. Bioanal. Chem., 379 (2004), pp. 982–991.
[50] J. Wloka, *Partielle Differentialgleichungen*, Teubner, Stuttgart, 1982.

SIAM J. SCI. COMPUT.
Vol. 29, No. 6, pp. 2381–2401
© 2007 Society for Industrial and Applied Mathematics

ADAPTIVE SEMIDISCRETE CENTRAL-UPWIND SCHEMES FOR NONCONVEX HYPERBOLIC CONSERVATION LAWS*

ALEXANDER KURGANOV†, GUERGANA PETROVA‡, AND BOJAN POPOV‡

Abstract. We discover that the choice of a piecewise polynomial reconstruction is crucial in computing solutions of nonconvex hyperbolic (systems of) conservation laws. Using semidiscrete central-upwind schemes, we illustrate that the obtained numerical approximations may fail to converge to the unique entropy solution or the convergence may be so slow that achieving a proper resolution would require the use of (almost) impractically fine meshes. For example, in the scalar case, all computed solutions seem to converge to solutions that are entropy solutions for some entropy pairs. However, in most applications, one is interested in capturing the unique (Kruzhkov) solution that satisfies the entropy condition for all convex entropies. We present a number of numerical examples that demonstrate the convergence of the solutions, computed with the dissipative second-order minmod reconstruction, to the unique entropy solution. At the same time, more compressive and/or higher-order reconstructions may fail to resolve composite waves, typically present in solutions of nonconvex conservation laws, and thus may fail to recover the Kruzhkov solution. In this paper, we propose a simple and computationally inexpensive adaptive strategy that allows us to simultaneously capture the unique entropy solution and to achieve a high resolution of the computed solution. We use the dissipative minmod reconstruction near the points where convexity changes and utilize a fifth-order weighted essentially nonoscillatory (WENO5) reconstruction in the rest of the computational domain. Our numerical examples (for one- and two-dimensional scalar and systems of conservation laws) demonstrate the robustness, reliability, and nonoscillatory nature of the proposed adaptive method.

Key words. hyperbolic systems of conservation laws, semidiscrete central-upwind schemes, piecewise polynomial reconstructions, nonconvex flux

AMS subject classification. Primary, 65M12; Secondary, 35L65

DOI. 10.1137/040614189

1. Introduction.

We consider initial-value problems (IVPs) for hyperbolic (systems of) conservation laws in one and two space dimensions:

$$\mathbf{u}_t + \nabla_{\mathbf{x}} \cdot \mathbf{f}(\mathbf{u}) = \mathbf{0}, \tag{1.1}$$

$$\mathbf{u}(\mathbf{x}, 0) = \mathbf{u}_0(\mathbf{x}), \tag{1.2}$$

whose entropy solutions may admit composite waves. Such examples are scalar conservation laws with nonconvex fluxes and systems of conservation laws, for which (at least) one of the structure coefficients, defined in (2.7) in section 2, changes sign. Further, we refer to all these cases as *nonconvex* hyperbolic (systems of) conservation laws.

An important example of such a system is the Euler equations of gas dynamics with a nonconvex equation of state (EOS). In general, in this case, anomalous wave structure should be expected near a phase transition, because the wave speed need

*Received by the editors August 30, 2004; accepted for publication (in revised form) February 20, 2007; published electronically October 17, 2007.
http://www.siam.org/journals/sisc/29-6/61418.html

†Mathematics Department, Tulane University, New Orleans, LA 70118 (kurganov@math.tulane.edu). This author's work was supported in part by NSF grants DMS-0310585 and DMS-0610430.

‡Department of Mathematics, Texas A&M University, College Station, TX 77843 (gpetrova@math.tamu.edu, popov@math.tamu.edu). The work of the second author was supported in part by NSF grants DMS-0296020 and DMS-0505501. The work of the third author was supported in part by NSF grant DMS-0510650.

not be monotonic along Hugoniot loci and isentropes, and the usual construction of the wave curves for the Riemann problem must be modified (see [27] and the references therein). Another important example of a nonconvex system of conservation laws is the polymer system used to model polymer flooding processes in enhanced oil recovery (see, e.g., [29]). The solution of the Riemann problem for these two systems is known (see, for example, [2], [23], [24], [27], [36], [38], [39], and [11], [13], respectively). However, designing reliable and robust high-order numerical methods for general initial data remains a challenging problem.

Many classical numerical schemes have been extended to solve nonconvex (systems of) conservation laws. We refer the reader to [1], [4], [5], [6], [10], [31], [37], [40] for several upwind Godunov-type schemes based on both the exact and approximate Riemann problem solvers. A low dissipative front tracking method for the polymer system was proposed in [9]. At the same time, note that the "simplest" extensions of the Roe scheme typically cannot be applied in practice to nonconvex systems of conservation laws. We would also like to point out that many of the aforementioned schemes may require computationally expensive evaluation of integrals or a very careful estimation of the CFL number, since otherwise the computed solution may converge to a weak solution that is not the unique entropy solution (see, for example, [19]).

In this paper, we apply the Godunov-type semidiscrete central-upwind schemes from [15] to numerically solve nonconvex hyperbolic (systems of) conservation laws. These schemes are an attractive alternative to other existing methods because they are simple (no Riemann problem solvers are employed), universal, and efficient, and can be used as a "black-box solver." They consist of three steps: projection (averaging), a piecewise polynomial reconstruction, and evolution of the reconstructed interpolant to the next time level according to the integral form of the conservation law (obtained by integrating (1.1) over the control volume). Assuming the exact averaging and evolution operators are used, the nonoscillatory nature of the computed solution is guaranteed provided the piecewise polynomial reconstruction is nonoscillatory, which is typically achieved with the help of nonlinear limiters. When solving the nonconvex problem (1.1)–(1.2) using the semidiscrete central-upwind schemes from [15], necessary adjustments, related to the estimation of the local speeds of propagation, have to be made. Moreover, it turns out that the choice of reconstruction is crucial for capturing of the unique entropy solution. One may expect that, as in the convex case, the use of more compressive and/or higher-order reconstruction will result in sharper resolution of the computed solution. However, this is not the case here, where sharper reconstructions may lead to the convergence of the computed solutions toward a weak solution, which satisfies the entropy condition for only some, but not all, convex entropies. At the same time, the use of more dissipative reconstructions typically results in computing the unique entropy solution, but with relatively low resolution. For example, the implementation of the dissipative minmod reconstruction (see, e.g., [35], [22], [28], [34]) seems to ensure the convergence toward the unique entropy solution, while more compressible reconstructions based on the generalized minmod [22], [28], [34] or superbee [22], [30], [34] limiters as well as the fifth-order essentially nonoscillatory (WENO5) [3], [12], [26], [32], [33] reconstruction may lead to capturing another weak solution (at least in the case when the mesh is of the reasonably coarse size typically used in practical computations).

In this paper, we introduce a simple scheme adaption strategy for solving nonconvex problems, which utilizes the advantages of both the compressive/high-order and the dissipative reconstructions, namely, computing the entropy solution with high

resolution. This is achieved by the proposed *adaptive semidiscrete central-upwind scheme*. In the new scheme, the choice of reconstruction is based on the so far computed solution and switches adaptively from one type to another at or near the points where the flux convexity changes.

The paper is organized as follows. In section 2, we briefly describe the semidiscrete central-upwind schemes. Several piecewise polynomial reconstructions and their impact on the computed solutions are discussed in section 3. Our scheme adaption strategy is presented in section 4. The numerical examples are shown in section 5, where we test our method on one-dimensional (1-D) and two-dimensional (2-D) nonconvex conservation laws, including scalar equations, the Euler equations of gas dynamics, and the polymer system.

2. Semidiscrete central-upwind schemes—an overview. In this section, we briefly describe the semidiscrete central-upwind schemes proposed in [15]. Here we emphasize the details needed for their implementation for nonconvex conservation laws.

For simplicity, we will focus on the 1-D case, where we consider a uniform spatial grid $x_\alpha = \alpha\Delta x$, and assume that the computed cell averages of the solution at some time level t,

$$\bar{\mathbf{u}}_j(t) \approx \frac{1}{\Delta x}\int_{x_{j-\frac12}}^{x_{j+\frac12}} \mathbf{u}(x,t)\,dx,$$

are already available. Using these values, we reconstruct a (nonoscillatory) piecewise polynomial interpolant (from now on, we will omit the time-dependence of all computed and reconstructed quantities), $\sum_j \mathbf{p}_j(x)\chi_j(x)$, where χ_j is the characteristic function of $[x_{j-\frac12}, x_{j+\frac12}]$ and $\mathbf{p}_j(x)$ are the corresponding polynomial pieces. For example, conservative linear pieces have the following form:

$$\mathbf{p}_j(x) = \bar{\mathbf{u}}_j + \mathbf{s}_j(x - x_j). \tag{2.1}$$

Piecewise polynomial reconstructions are, in general, discontinuous at the cell interfaces, $x = x_{j+\frac12}$, and therefore their evolution is locally described by the solutions of generalized Riemann problems. The sizes of the corresponding Riemann fans are determined by the right- and left-sided local speeds of propagation, $\{a^{\pm}_{j+\frac12}\}$, which can be estimated as follows:

$$\begin{aligned} a^+_{j+\frac12} &:= \max_{\boldsymbol{\omega}\in C(\mathbf{u}^-_{j+\frac12},\mathbf{u}^+_{j+\frac12})}\left\{\lambda_N\left(\frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\boldsymbol{\omega})\right),0\right\},\\ a^-_{j+\frac12} &:= \min_{\boldsymbol{\omega}\in C(\mathbf{u}^-_{j+\frac12},\mathbf{u}^+_{j+\frac12})}\left\{\lambda_1\left(\frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\boldsymbol{\omega})\right),0\right\}. \end{aligned} \tag{2.2}$$

Here, $\lambda_1 < \cdots < \lambda_N$ are the eigenvalues of the Jacobian $\frac{\partial \mathbf{f}}{\partial \mathbf{u}}$, and $C(\mathbf{u}^-_{j+\frac12}, u^+_{j+\frac12})$ is the curve in the phase space that connects the left and right values of the reconstruction at $x = x_{j+\frac12}$:

$$\mathbf{u}^-_{j+\frac12} := \mathbf{p}_j(x_{j+\frac12}), \quad \mathbf{u}^+_{j+\frac12} := \mathbf{p}_{j+1}(x_{j+\frac12}). \tag{2.3}$$

The cell averages $\{\bar{\mathbf{u}}_j\}$ are then evolved in time according to the semidiscrete central-upwind scheme:

$$\frac{d}{dt}\bar{\mathbf{u}}_j(t) = -\frac{\mathbf{H}_{j+\frac{1}{2}}(t) - \mathbf{H}_{j-\frac{1}{2}}(t)}{\Delta x}, \tag{2.4}$$

where the numerical fluxes $\mathbf{H}_{j+\frac{1}{2}}$ are

$$\mathbf{H}_{j+\frac{1}{2}}(t) = \frac{a^+_{j+\frac{1}{2}}\mathbf{f}(\mathbf{u}^-_{j+\frac{1}{2}}) - a^-_{j+\frac{1}{2}}\mathbf{f}(\mathbf{u}^+_{j+\frac{1}{2}})}{a^+_{j+\frac{1}{2}} - a^-_{j+\frac{1}{2}}} + \frac{a^+_{j+\frac{1}{2}}a^-_{j+\frac{1}{2}}}{a^+_{j+\frac{1}{2}} - a^-_{j+\frac{1}{2}}}\left[\mathbf{u}^+_{j+\frac{1}{2}} - \mathbf{u}^-_{j+\frac{1}{2}}\right]. \tag{2.5}$$

Notice that a practical implementation of (2.4)–(2.5) requires the following:

(i) A stable, preferably a strong stability preserving (SSP) [7], solver for the ODE system (2.4). In all numerical experiments presented below, we have used the third-order SSP Runge–Kutta method. To guarantee that no wave interaction occurs during each time step, the CFL number has been set to 0.5.

(ii) A choice of a piecewise polynomial reconstruction (see the discussion in section 3). Notice that the spatial order of the resulting scheme is determined by the order of the reconstruction.

(iii) A practical version of formula (2.2) for computing the one-sided local speeds of propagation. It is well known that in the convex case, they can be computed by

$$\begin{aligned} a^+_{j+\frac{1}{2}} &= \max\left\{\lambda_N\left(\frac{\partial\mathbf{f}}{\partial\mathbf{u}}(\mathbf{u}^-_{j+\frac{1}{2}})\right), \lambda_N\left(\frac{\partial\mathbf{f}}{\partial\mathbf{u}}(\mathbf{u}^+_{j+\frac{1}{2}})\right), 0\right\},\\ a^-_{j+\frac{1}{2}} &= \min\left\{\lambda_1\left(\frac{\partial\mathbf{f}}{\partial\mathbf{u}}(\mathbf{u}^-_{j+\frac{1}{2}})\right), \lambda_1\left(\frac{\partial\mathbf{f}}{\partial\mathbf{u}}(\mathbf{u}^+_{j+\frac{1}{2}})\right), 0\right\}. \end{aligned} \tag{2.6}$$

In the nonconvex case, however, this simple formula is incorrect if the flux convexity changes near $\mathbf{u}^\pm_{j+\frac{1}{2}}$, and one has to use a more accurate estimate of $a^\pm_{j+\frac{1}{2}}$ there.

In the scalar case, the speeds in (2.2) can be easily evaluated exactly:

$$a^+_{j+\frac{1}{2}} = \max_{u\in[u^{\min}_{j+\frac{1}{2}}, u^{\max}_{j+\frac{1}{2}}]}\{f'(u), 0\}, \quad a^-_{j+\frac{1}{2}} = \min_{u\in[u^{\min}_{j+\frac{1}{2}}, u^{\max}_{j+\frac{1}{2}}]}\{f'(u), 0\},$$

where $u^{\min}_{j+\frac{1}{2}} = \min\{u^-_{j+\frac{1}{2}}, u^+_{j+\frac{1}{2}}\}$ and $u^{\max}_{j+\frac{1}{2}} = \max\{u^-_{j+\frac{1}{2}}, u^+_{j+\frac{1}{2}}\}$.

In the case of a system, we follow the approach in [37] if the structure coefficients,

$$\kappa_i = \nabla_{\mathbf{u}}\lambda_i \cdot \mathbf{r}_i, \quad i = 1, \ldots, N, \tag{2.7}$$

change sign near $\mathbf{u}^\pm_{j+\frac{1}{2}}$ (here $\mathbf{r}_i$ is the ith right eigenvector, corresponding to the ith eigenvalue λ_i). The left and right local speeds are then estimated by

$$\begin{aligned} a^+_{j+\frac{1}{2}} &= \max\left\{\lambda_N\left(\frac{\partial\mathbf{f}}{\partial\mathbf{u}}(\mathbf{u}^-_{j+\frac{1}{2}})\right), \lambda_N\left(\frac{\partial\mathbf{f}}{\partial\mathbf{u}}(\mathbf{u}^+_{j+\frac{1}{2}})\right), \widetilde{\mathcal{H}}_N, 0\right\},\\ a^-_{j+\frac{1}{2}} &= \min\left\{\lambda_1\left(\frac{\partial\mathbf{f}}{\partial\mathbf{u}}(\mathbf{u}^-_{j+\frac{1}{2}})\right), \lambda_1\left(\frac{\partial\mathbf{f}}{\partial\mathbf{u}}(\mathbf{u}^+_{j+\frac{1}{2}})\right), \widetilde{\mathcal{H}}_1, 0\right\}. \end{aligned} \tag{2.8}$$

Here, the notation is

$$\widetilde{\mathcal{H}}_N := \max_{\sigma\in[0,1]} \mathcal{H}(\lambda_N(\mathbf{u}^-_{j+\frac12}),\lambda_N(\mathbf{u}^+_{j+\frac12}),\kappa^-_N,\kappa^+_N;\sigma),$$

$$\widetilde{\mathcal{H}}_1 := \min_{\sigma\in[0,1]} \mathcal{H}(\lambda_1(\mathbf{u}^-_{j+\frac12}),\lambda_1(\mathbf{u}^+_{j+\frac12}),\kappa^-_1,\kappa^+_1;\sigma),$$

where

$$\kappa^\pm_i := \nabla_{\mathbf{u}}\lambda_i(\mathbf{u}^\pm_{j+\frac12})\cdot\mathbf{r}_i(\mathbf{u}^{\rm mid}_{j+\frac12}),\quad \mathbf{u}^{\rm mid}_{j+\frac12}=\frac{\mathbf{u}^+_{j+\frac12}+\mathbf{u}^-_{j+\frac12}}{2},\qquad i=1,N, \tag{2.9}$$

are the left and right structure coefficients at $x_{j+\frac12}$, and $\mathcal{H}(a,b,a',b';\sigma)$ is the unique Hermite cubic polynomial that satisfies the following interpolation conditions:

$$\mathcal{H}(a,b,a',b';0)=a,\ \ \mathcal{H}(a,b,a',b';1)=b,\ \ \mathcal{H}'(a,b,a',b';0)=a',\ \ \mathcal{H}'(a,b,a',b';1)=b'.$$

Away from the regions where the convexity changes, the local speeds are computed by (2.6).

Remark. In the 2-D case, we use a higher-order version of the central-upwind scheme from [15], where a fourth-order central-upwind numerical flux was obtained. Proceeding along the lines of the numerical flux derivation in [15] and replacing Simpson's rule with the 3-point Gaussian quadrature in the evaluation of the spatial flux integrals, we arrive at the following semidiscrete central-upwind scheme:

$$\frac{d}{dt}\bar{\mathbf{u}}_{j,k}(t)=-\frac{\mathbf{H}^x_{j+\frac12,k}(t)-\mathbf{H}^x_{j-\frac12,k}(t)}{\Delta x}-\frac{\mathbf{H}^y_{j,k+\frac12}(t)-\mathbf{H}^y_{j,k-\frac12}(t)}{\Delta y}, \tag{2.10}$$

where the numerical fluxes are

(2.11)

$$\begin{aligned}\mathbf{H}^x_{j+\frac12,k} :=\ & \frac{a^+_{j+\frac12,k}}{18(a^+_{j+\frac12,k}-a^-_{j+\frac12,k})}\Big[5\mathbf{f}(\mathbf{u}^{\rm E1}_{j,k})+8\mathbf{f}(\mathbf{u}^{\rm E2}_{j,k})+5\mathbf{f}(\mathbf{u}^{\rm E3}_{j,k})\Big]\\ &-\frac{a^-_{j+\frac12,k}}{18(a^+_{j+\frac12,k}-a^-_{j+\frac12,k})}\Big[5\mathbf{f}(\mathbf{u}^{\rm W1}_{j+1,k})+8\mathbf{f}(\mathbf{u}^{\rm W2}_{j+1,k})+5\mathbf{f}(\mathbf{u}^{\rm W3}_{j+1,k})\Big]\\ &+\frac{a^+_{j+\frac12,k}a^-_{j+\frac12,k}}{18(a^+_{j+\frac12,k}-a^-_{j+\frac12,k})}\\ &\times\Big[5(\mathbf{u}^{\rm W1}_{j+1,k}-\mathbf{u}^{\rm E1}_{j,k})+8(\mathbf{u}^{\rm W2}_{j+1,k}-\mathbf{u}^{\rm E2}_{j,k})+5(\mathbf{u}^{\rm W3}_{j+1,k}-\mathbf{u}^{\rm E3}_{j,k})\Big]\end{aligned}$$

and

(2.12)

$$\begin{aligned}\mathbf{H}^y_{j,k+\frac12} :=\ & \frac{b^+_{j,k+\frac12}}{18(b^+_{j,k+\frac12}-b^-_{j,k+\frac12})}\Big[5\mathbf{g}(\mathbf{u}^{\rm N1}_{j,k})+8\mathbf{g}(\mathbf{u}^{\rm N2}_{j,k})+5\mathbf{g}(\mathbf{u}^{\rm N3}_{j,k})\Big]\\ &-\frac{b^-_{j,k+\frac12}}{18(b^+_{j,k+\frac12}-b^-_{j,k+\frac12})}\Big[5\mathbf{g}(\mathbf{u}^{\rm S1}_{j,k+1})+8\mathbf{g}(\mathbf{u}^{\rm S2}_{j,k+1})+5\mathbf{g}(\mathbf{u}^{\rm S3}_{j,k+1})\Big]\end{aligned}$$

$$+\frac{b^+_{j,k+\frac12}b^-_{j,k+\frac12}}{18(b^+_{j,k+\frac12}-b^-_{j,k+\frac12})}$$

$$\times\Big[5(\mathbf{u}^{\mathrm{S}1}_{j,k+1}-\mathbf{u}^{\mathrm{N}1}_{j,k})+8(\mathbf{u}^{\mathrm{S}2}_{j,k+1}-\mathbf{u}^{\mathrm{N}2}_{j,k})+5(\mathbf{u}^{\mathrm{S}3}_{j,k+1}-\mathbf{u}^{\mathrm{N}3}_{j,k})\Big].$$

Here, the Cartesian grid is assumed to be uniform with $x_\alpha=\alpha\Delta x$, $y_\beta=\beta\Delta y$; the point values $\mathbf{u}^{\mathrm{E}i}_{j,k}$, $i=1,2,3$, are the values of the polynomial piece over the cell $[x_{j-\frac12},x_{j+\frac12}]\times[y_{k-\frac12},y_{k+\frac12}]$, evaluated at the corresponding Gaussian nodes along the eastern edge ($x=x_{j+\frac12}$, $y_{k-\frac12}<y<y_{k+\frac12}$); the point values $\mathbf{u}^{\mathrm{N}i}_{j,k}$, $i=1,2,3$, are the values of the polynomial piece over the same cell evaluated at the corresponding Gaussian nodes along the northern edge ($x_{j-\frac12}<x<x_{j+\frac12}$, $y=y_{k+\frac12}$); and the rest of the point values, $\mathbf{u}^{\mathrm{W}i}_{j,k}$, $\mathbf{u}^{\mathrm{S}i}_{j,k}$, $i=1,2,3$, are calculated at the Gaussian nodes along the remaining two edges. Recall that the Gaussian nodes G_i, $i=1,2,3$, on an arbitrary segment AB with $A=(a_1,a_2)$ and $B=(b_1,b_2)$ have the following coordinates: $G_1=(\alpha_+a_1+\alpha_-a_2,\alpha_+b_1+\alpha_-b_2)$, $G_2=((a_1+a_2)/2,(b_1+b_2)/2)$, and $G_3=(\alpha_-a_1+\alpha_+a_2,\alpha_-b_1+\alpha_+b_2)$, where $\alpha_\pm=\frac12(1\pm\sqrt{3/5})$.

The local speeds of propagation in the x- and y-direction, $a^\pm_{j+\frac12,k}$ and $b^\pm_{j,k+\frac12}$, respectively, are computed at the midpoint of each cell interface in a dimension-by-dimension manner according to the above 1-D recipe.

If the point values in (2.11)–(2.12) are computed using an rth-order piecewise polynomial reconstruction, the resulting spatial order of the semidiscrete central-upwind scheme (2.10)–(2.12) will be $\min(r,6)$ due to the accuracy limitation of the Gaussian formula. When a second-order piecewise linear reconstruction is utilized, the sixth-order fluxes (2.11)–(2.12) can be replaced by their simpler second-order versions (see, e.g., [15], [17]).

3. Piecewise polynomial reconstructions. In this section, we discuss the effect of different piecewise polynomial reconstructions on the numerical solution, computed by the central-upwind schemes. We will consider several second-order reconstructions and the fifth-order WENO5 reconstruction, each of which will be applied to the conservative variables componentwise (therefore, all the variables used in this section are scalar). In the nonconvex case, the choice of reconstruction is particularly important for computing the unique entropy solution, especially when composite waves are present.

In general, in order to achieve high resolution, the piecewise polynomial reconstruction should be of an appropriate order of accuracy and nonoscillatory. For second-order piecewise linear reconstructions (see (2.1)), the first requirement can be easily achieved by ensuring $s_j=u_x(x_j,t)+\mathcal{O}(\Delta x)$, while to satisfy the nonoscillatory property, the reconstruction has to utilize a nonlinear limiter. A library of such reconstructions is available (see, e.g., [35], [19], [8], [22], [28], [30], [34]). In the numerical experiments, reported in section 5, we have tested several well-known limiters, such as the generalized minmod limiter ([8], [22], [28], [34]),

$$s_j=\mathrm{minmod}\left(\theta\frac{\bar u_j-\bar u_{j-1}}{\Delta x},\ \frac{\bar u_{j+1}-\bar u_{j-1}}{2\Delta x},\ \theta\frac{\bar u_{j+1}-\bar u_j}{\Delta x}\right),\quad \theta\in[1,2], \tag{3.1}$$

where the minmod function is defined by

$$\text{minmod}(z_1, z_2, \ldots) := \begin{cases} \min_j\{z_j\} & \text{if } \ z_j > 0 \ \ \forall j, \\ \max_j\{z_j\} & \text{if } \ z_j < 0 \ \ \forall j, \\ 0 & \text{otherwise,} \end{cases}$$

and the superbee limiter ([19], [22], [30], [34]),

$$s_j = \frac{\bar{u}_j - \bar{u}_{j-1}}{\Delta x} \max\left\{0, \min(2r, 1), \min(r, 2)\right\}, \quad r = \frac{\bar{u}_{j+1} - \bar{u}_j}{\bar{u}_j - \bar{u}_{j-1}}. \tag{3.2}$$

Note that the generalized minmod limiter (3.1) with $\theta = 1$ reduces to the standard (dissipative) minmod limiter, which will be referred to as the MM1 limiter. The other limiting case ($\theta = 2$) corresponds to the most compressive minmod limiter, which from now on, will be denoted by MM2. The superbee limiter will be denoted by SB.

In many cases, especially when smooth parts of the solution have a complicated structure, it may be advantageous to use higher (than second)-order reconstructions. We refer the reader to [3], [16], [19], [20], [21], [25] for numerous examples of higher-order reconstructions. A very popular family of such reconstructions are the so-called WENO reconstructions. They were first introduced in [26] and since have been extended to higher orders, multiple numbers of dimensions, and finite-difference setting (see, e.g., [3], [12], [32], [33] and the references therein). In our numerical experiments, we have used the fifth-order finite-volume WENO5 reconstruction, applied componentwise and extended to the 2-D case in a dimension-by-dimension manner as described in [32].

In practice, the reconstruction selection depends on the problem at hand. When the entropy solution does not admit composite waves, the implementation of a compressive second-order reconstruction or a higher-order one typically results in a high quality numerical solution, since both smooth and discontinuous (shocks, contacts) parts of the solution are approximated more accurately than by a more dissipative reconstruction, such as the MM1 one. However, the situation is more complicated in the nonconvex case, in which a formation of composite waves is possible. This problem is best understood in the case of a 1-D scalar equation, which is the simplest conservation law whose solutions admit composite waves.

3.1. One-dimensional scalar conservation laws. Composite waves (see [19]) consist of a sequence of joined rarefactions and shocks. A generic case is a shock adjacent to a rarefaction wave, in the computation of which the reconstruction selection seems to be crucial. As we illustrate below, a numerical scheme that uses the generalized minmod limiter (3.1) with $\theta > 1$ or even the more compressive SB may result in capturing a numerical solution that converges to a weak solution, consisting of a shock, adjacent to an artificial constant state, followed by a rarefaction wave. Moreover, the rarefaction may be completely missing (see Figure 3.4), and the shock location and strength may be totally different from those in the unique entropy solution.

We now consider a particular example: a Riemann problem for the 1-D scalar equation

$$u_t + f(u)_x = 0 \tag{3.3}$$

with the C^1 monotone increasing flux, whose convexity changes at $u = 1/2$:

$$f(u) = \begin{cases} \dfrac{u(1-u)}{4}, & u < \dfrac{1}{2}, \\ \dfrac{1}{2}u^2 - \dfrac{1}{2}u + \dfrac{3}{16}, & u \geq \dfrac{1}{2}. \end{cases} \tag{3.4}$$

While the Kruzhkov entropy solution can be easily obtained for any (nonconvex) smooth flux function (see, e.g., [18]), numerically solving (3.3) by a high-order scheme may be quite a challenging problem.

We apply the central-upwind semidiscretization (2.2)–(2.5) to (3.3)–(3.4). Due to the monotonicity of the flux (3.4), $a^-_{j+\frac{1}{2}} = 0$, and thus the central-upwind scheme (2.2)–(2.5) reduces to a simple semidiscrete upwinding:

$$\frac{d}{dt}\bar{u}_j(t) = -\frac{f(u^-_{j+\frac{1}{2}}) - f(u^-_{j-\frac{1}{2}})}{\Delta x}. \tag{3.5}$$

We first solve (3.3)–(3.4) subject to the initial condition:

$$u(x,0) = \begin{cases} 0, & x < 0.25, \\ 1, & 0.25 < x. \end{cases} \tag{3.6}$$

In Figure 3.1, we show the solutions computed at time $t = 1$ on a uniform grid with $\Delta x = 1/1600$ using the second-order upwind scheme (3.5), (2.1), (2.3) with compressive MM2 and SB reconstructions. One can clearly see that in both cases, the captured solution is not the Kruzhkov entropy solution, which consists of a shock, located at $x = (\sqrt{6} - 1)/4$, followed by a rarefaction wave. Moreover, we have performed a numerical convergence study which clearly indicates that both numerical approximations fail to converge to the entropy solution.

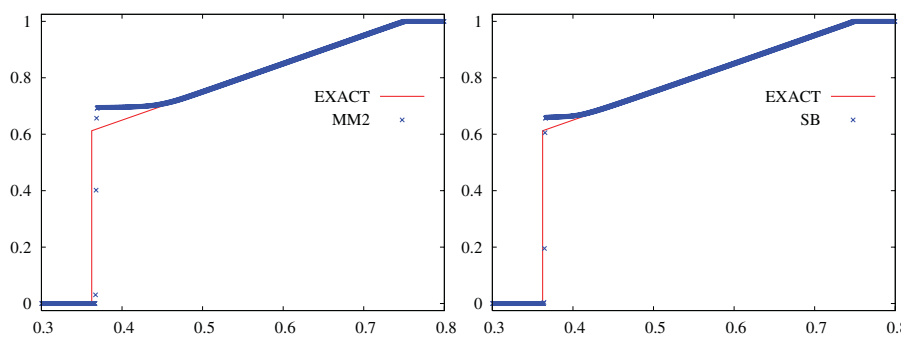


FIG. 3.1. *Solutions of* (3.3)–(3.4), (3.6) *computed with MM2 (left) and SB (right).*

At the same time, we observe that the same method, but with the dissipative MM1 reconstruction, leads to the convergence of the computed solutions toward the unique entropy solution; see Figure 3.2 (left) and Figure 3.2 (right), where a mesh refinement study is performed.

In this example, the evolution step is performed using the semidiscrete upwind scheme (3.5), (2.1), (2.3) and the third-order SSP Runge–Kutta ODE solver. A natural question to ask is whether the observed effect is due to the particular spatial

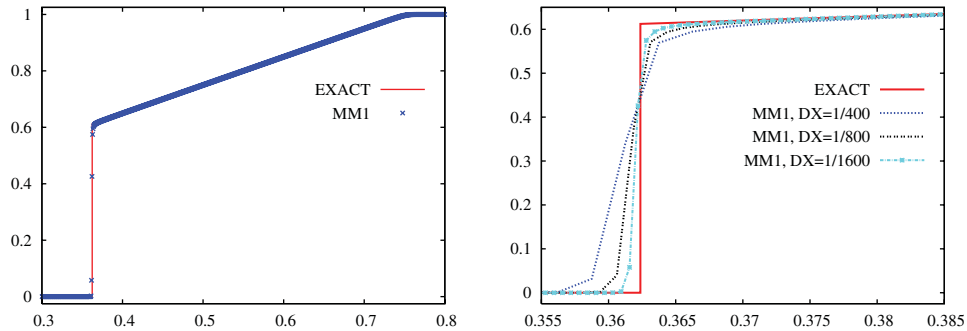


FIG. 3.2. *Solutions of* (3.3)–(3.4), (3.6) *computed with MM*1 *(left) and its convergence test (right, zoom at the composite wave area).*

discretization and time integration used or is a consequence of the implemented reconstruction. It turns out that if we start with a compressive piecewise linear reconstruction at time $t = t^n$ and evolve it exactly to the next time level $t = t^{n+1}$, according to the integral formulation of (3.3),

(3.7)
$$\bar{u}_j(t^{n+1}) = \bar{u}_j(t^n) - \frac{1}{\Delta x}\left[\int_{t^n}^{t^{n+1}} f(u(x_{j+\frac{1}{2}},\tau))\,d\tau - \int_{t^n}^{t^{n+1}} f(u(x_{j-\frac{1}{2}},\tau))\,d\tau\right],$$

the same behavior persists. Here, we only need to exactly compute the temporal integrals in (3.7), which can be done analytically when $\Delta t \le 0.5\Delta x$, since the formulae for both the flux function and the exact solution are available (we have taken $\Delta t = 0.5\Delta x$, but smaller time steps lead to the same results).

Snapshots at times $t = 0.1, 0.2, \ldots, 1.0$ of the exactly evolved solutions of the IVP (3.3)–(3.4), (3.6) are presented in Figure 3.3, where the MM1 and MM2 reconstructions are used. In the case of the MM2 reconstruction, the wrong shock location as well as the formation of an artificial constant state between the shock and the rarefaction wave can be clearly seen, while the MM1 solution accurately approximates the composite wave.

Here, the implementation of the MM2 reconstruction (or another compressive reconstruction, for example, the generalized minmod with any $\theta > 1$ or the SB) leads to an initial overshoot in a single cell—the last cell with a cell-average value less than 1. As time progresses, the overshoot persists and, because of the error propagation, spreads over neighboring cells. At the same time, the moving shock cannot balance the effect of the overshoot, because (i) the overshoot speed is higher than the shock speed, and (ii) the large jumps from the shock feed the overshoot and thus oppose its smearing by the rarefication process and the numerical diffusion. As a result, an artificial constant state between the shock and rarefaction wave is formed. In the case of a generalized minmod reconstruction, its value depends on the value of θ and the size of the initial shock.

The above problem seems to get resolved if we use the dissipative MM1 reconstruction. The successful resolution of composite waves in this case is due to the fact that the smaller of the jumps, $\bar{u}_{j+1} - \bar{u}_j$ or $\bar{u}_j - \bar{u}_{j-1}$, is used to determine the corresponding slope s_j in (2.1). The calculation of the new local cell averages over/near the border between the shock and rarefaction is mostly based on the information from

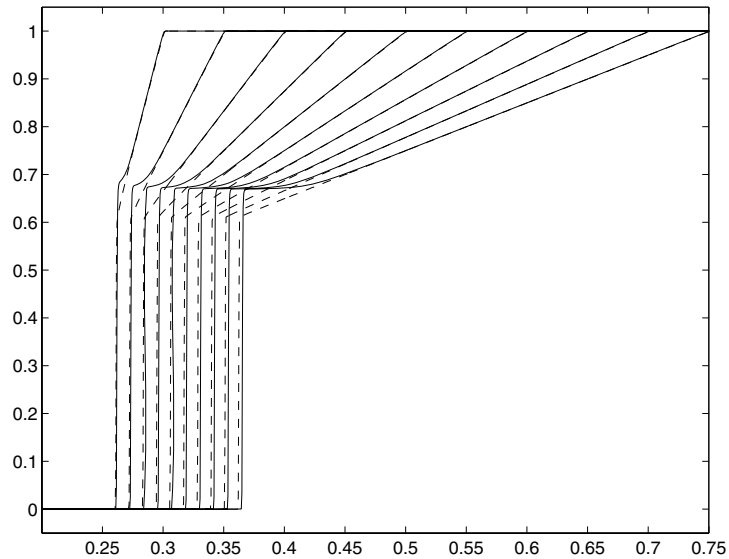


FIG. 3.3. *Exact evolution of the MM*1 *and MM*2 *reconstructions;* $\Delta x = 1/2000$.

the rarefaction part of the computed solution (where the jumps are smaller), and thus the shock impact on the evaluation of these averages is substantially reduced.

So far, the discussion has been restricted to second-order piecewise linear reconstructions since in this case one can relatively easily understand the effects of different reconstructions on the computed solutions. The situation gets much more complicated when a higher-order (for example, the fifth-order WENO5) reconstruction is employed. In the scalar 1-D case, we were unable to find an example in which the WENO5 reconstruction leads to capturing a non-Kruzhkov weak solution. However, our example of a 1-D system (section 5.2) and the 2-D scalar example (section 5.3) clearly illustrate that the WENO5-based scheme may fail to resolve composite waves; see Figures 5.4, 5.7, and 5.10, respectively.

We conclude this section with yet another numerical example, which illustrates that if a compressive reconstruction is employed, a rarefaction part of the composite wave can be completely missed, while the use of the dissipative MM1 reconstruction leads to capturing the unique entropy solution. We compute the solution of the same equation (3.3)–(3.4) subject to different initial data:

$$u(x,0) = \begin{cases} 1, & x < 0.25, \\ 0, & 0.25 < x. \end{cases} \tag{3.8}$$

In Figure 3.4, we show the solutions computed at time $t = 2$ on a uniform grid with $\Delta x = 1/1600$ using the second-order upwind scheme (3.5), (2.1), (2.3) with compressive MM2 and SB reconstructions. One can clearly see that in both cases, the captured solution is a simple shock wave—not the Kruzhkov entropy solution, which consists of a shock, located at $x = (2\sqrt{3}-1)/4$, followed by a rarefaction wave. In Figure 3.5, we present the solution, computed by the same method but with the dissipative MM1 reconstruction. As in the previous example, one can observe the convergence of the numerical solutions toward the unique entropy solution.

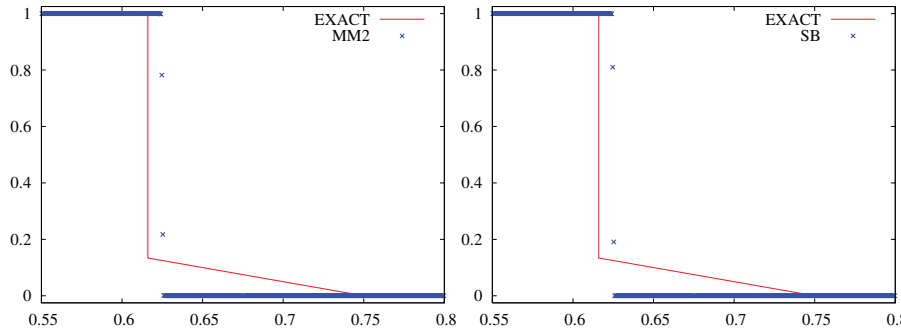


FIG. 3.4. *Solutions of* (3.3)–(3.4), (3.8) *computed with MM*2 *(left) and SB (right).*

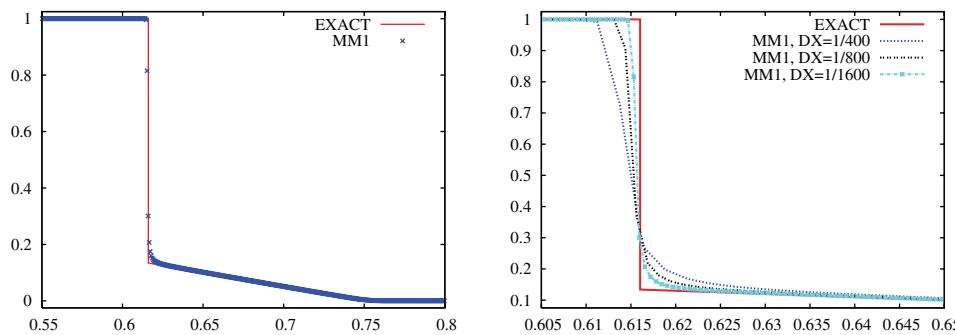


FIG. 3.5. *Solutions of* (3.3)–(3.4), (3.8) *computed with MM*1 *(left) and its convergence test (right, zoom at the composite wave area).*

4. Scheme adaption algorithm. It is a well-known fact that more compressive second-order and/or higher-order piecewise polynomial reconstructions typically lead to a higher resolution than the one that can be achieved with the help of the dissipative MM1 reconstruction. However, as observed in the previous section, the use of higher resolution reconstructions may be dangerous at the presence of composite waves, while the use of the MM1 reconstruction seems to ensure the convergence toward these waves.

We now describe a simple adaption strategy that will utilize the advantages of higher resolution reconstructions away from the composite waves, which, in turn, will be locally approximated using the MM1 reconstruction.

• Use the MM1 reconstruction only in the transition zones—in a small number of cells around points where the flux convexity changes (that is, where f'' or one of the κ_i's in (2.9) changes sign).

• Use a higher resolution reconstruction otherwise.

Remarks.

1. In all numerical experiments presented below, we have used the finite-volume WENO5 reconstruction—applied componentwise to the 1-D systems and in a dimension-by-dimension manner in the 2-D case—as a higher resolution component of our adaptive algorithm.
2. Our extensive numerical experiments suggest that in order to ensure convergence to the unique entropy solution, the number of cells where the MM1

reconstruction is used should be relatively small and could be taken proportional to $|\ln(\Delta x)|$ as the grid is refined.

5. Numerical examples. In this section, we numerically solve nonconvex 1-D hyperbolic systems of conservation laws and a 2-D scalar conservation law with nonconvex fluxes. We apply the semidiscrete central-upwind schemes described in section 2 with different piecewise polynomial reconstructions including the adaptive reconstruction proposed in section 4.

5.1. One-dimensional Euler equations of gas dynamics. We consider the 1-D Euler equations of gas dynamics,

$$\frac{\partial}{\partial t}\begin{bmatrix}\rho\\ m\\ E\end{bmatrix}+\frac{\partial}{\partial x}\begin{bmatrix}m\\ \rho u^2+p\\ u(E+p)\end{bmatrix}=0, \tag{5.1}$$

with the following nonconvex EOS (see [37]):

$$p=p(\rho,e)=((\gamma-1)\rho+f(\rho))e,\quad e=\frac{E}{\rho}-\frac{1}{2}u^2,\quad \gamma=1.4, \tag{5.2}$$

where the function f is given by

$$f(\rho)=\begin{cases}10\exp\left(\dfrac{\rho^2}{1-2\rho+0.75\rho^2}\right), & \dfrac{2}{3}<\rho<2,\\ 0 & \text{otherwise,}\end{cases} \tag{5.3}$$

and ρ, u, $m=\rho u$, p, and E are the density, velocity, momentum, pressure, and total energy, respectively. It can be shown that system (5.1) with the EOS given by (5.2)–(5.3) is hyperbolic.

The computation of the numerical solution of (5.1)–(5.3), using the scheme (2.4)–(2.5), (2.3), (2.8), combined with different piecewise polynomial reconstructions requires the calculation of the one-sided local speeds of propagation, and therefore the computation of the corresponding structure coefficients (2.9). Here, we have

$$\kappa_1^{\pm}=\left(-\frac{m^{\pm}}{(\rho^{\pm})^2}-c_\rho^{\pm}\right)+\left(\frac{1}{\rho^{\pm}}-c_m^{\pm}\right)(u_{mid}-c_{mid})-c_E^{\pm}\left(\frac{E_{mid}+p_{mid}}{\rho_{mid}}-u_{mid}c_{mid}\right),$$

$$\kappa_3^{\pm}=\left(-\frac{m^{\pm}}{(\rho^{\pm})^2}+c_\rho^{\pm}\right)+\left(\frac{1}{\rho^{\pm}}+c_m^{\pm}\right)(u_{mid}+c_{mid})+c_E^{\pm}\left(\frac{E_{mid}+p_{mid}}{\rho_{mid}}+u_{mid}c_{mid}\right),$$

where c is the sound speed,

$$c^2=p_\rho+\frac{pp_e}{\rho^2},$$

and

$$\mathcal{R}_{mid}:=\frac{\mathcal{R}^{+}+\mathcal{R}^{-}}{2}$$

for $\mathcal{R}$ being ρ, u, c, p, or E. The corresponding partial derivatives are

$$c_m=-\frac{m}{2\rho^2}T,\qquad c_E=\frac{1}{2\rho}T,$$

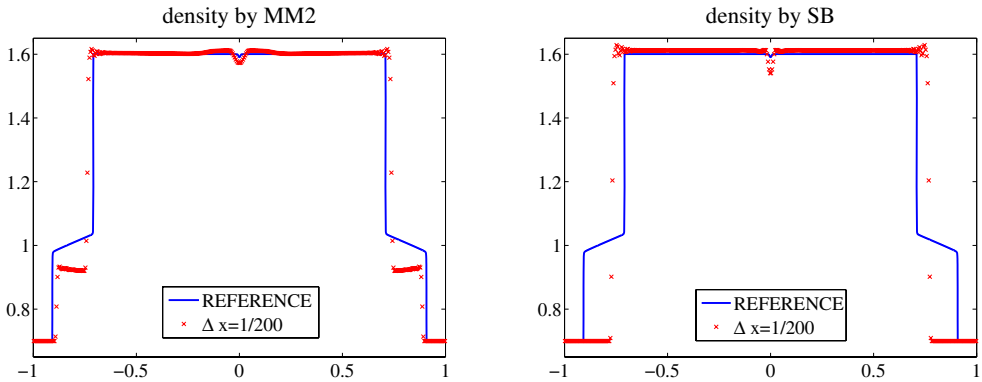


FIG. 5.1. *Solutions of* (5.1)–(5.4), *computed with MM*2 *(left) and SB (right).*

$$c_\rho = \frac{m^2 - E\rho}{2\rho^3}T + \frac{1}{2T}\left(f''(\rho) + \frac{(0.8\rho + 2f(\rho))(f'(\rho)\rho - f(\rho))}{\rho^3}\right),$$

where

$$T = \left(\frac{0.4 + f'(\rho) + (0.4 + f(\rho)\rho^{-1})^2}{E\rho^{-1} - 0.5m^2\rho^{-2}}\right)^{\frac{1}{2}}.$$

We solve the system (5.1)–(5.3) subject to the initial data (see [37]):

$$(p, \rho, u)_L = (0.5, 0.6998, 1), \quad (p, \rho, u)_R = (0.5, 0.6998, -1), \tag{5.4}$$

using the MM1, MM2, SB, and the adaptive reconstruction proposed in section 4. To apply the adaptive strategy from section 4, we need to know where the structure coefficients change sign, which in this case is the same as checking where $p_{\tau\tau}$, $\tau = \rho^{-1}$, changes sign. In this example, $p_{\tau\tau}$ has two isolated zeros $\tau^* = 1/\rho^* = 0.862$ and $\tau^{**} = 1/\rho^{**} = 1.21$. We perform the adaption in the following way. First, we mark all cell pairs with indexes $j, j+1$, for which either $(\rho_{j+1} - \rho^*)(\rho_j - \rho^*) < 0$ or $(\rho_{j+1} - \rho^{**})(\rho_j - \rho^{**}) < 0$. Then we use the MM1 reconstruction in the marked cells and in the cells located within $K\Delta x$ from them, while the WENO5 reconstruction is used in the rest of the computational domain.

We compare the density profiles of the numerical solutions, calculated at time $t = 1$ on a uniform grid with $\Delta x = 1/200$, with the profile of a reference solution, obtained with the "reliable" MM1 limiter and $\Delta x = 1/5000$. The results are presented in Figures 5.1–5.3, where one can clearly see that the composite waves are incorrectly resolved when a compressive piecewise linear reconstruction (either MM2 or SB) is employed, while the central-upwind scheme based on the dissipative MM1 reconstruction accurately captures their structure. Note that in this example, the use of the higher-order WENO5 reconstruction also leads to capturing the entropy solution (see Figure 5.2 (right)). However, the obtained numerical approximation is quite oscillatory, which, together with the failure of the WENO5-based scheme to resolve composite waves in the examples from sections 5.2 and 5.3, makes one doubt the reliability of a pure WENO5 reconstruction.

The solutions, computed by our adaptive central-upwind scheme, are shown in Figure 5.3. When the adaption constant $K = 1$, the obtained solution is basically

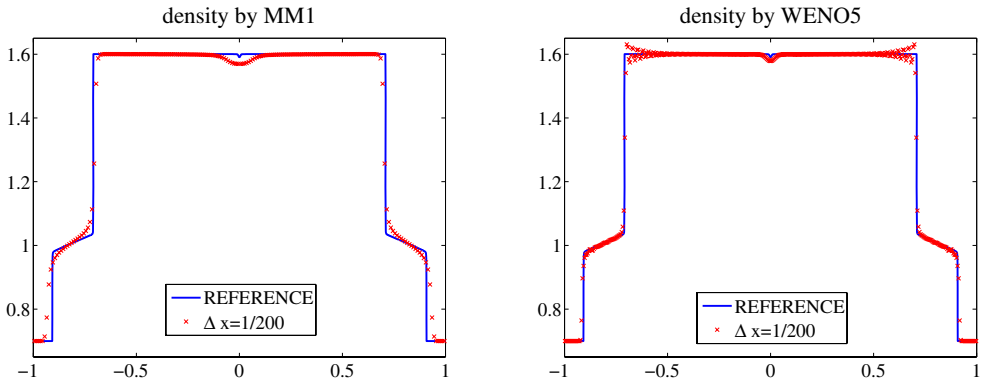


FIG. 5.2. *Solutions of* (5.1)–(5.4), *computed with MM*1 *(left) and WENO*5 *(right).*

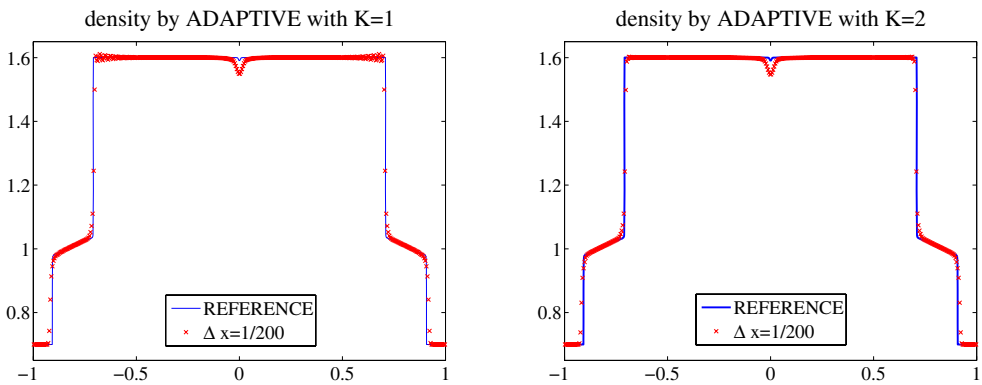


FIG. 5.3. *Solutions of* (5.1)–(5.4), *computed adaptively with* $K = 1$ *(left) and* $K = 2$ *(right).*

as well resolved as that obtained by the WENO5-based scheme, but it still has some oscillations, which are completely removed when a larger ($K = 2$) adaption constant is used. As expected, the adaptive reconstruction offers a better overall resolution of the numerical solution, and thus the adaptive strategy is preferable.

5.2. The polymer system. We consider the 1-D polymer system

$$\frac{\partial}{\partial t}\begin{bmatrix} s \\ b \end{bmatrix} + \frac{\partial}{\partial x}\begin{bmatrix} f(s,c) \\ cf(s,c) \end{bmatrix} = \mathbf{0}, \tag{5.5}$$

where s is the water saturation, c is the polymer concentration in the water, $f(s,c)$ is the fractional flow function of water, $b(s,c) = sc + a(c)$, and $a(c)$ is the adsorption function. Here, we take (see, e.g., [9])

$$f(s,c) = \frac{s^2}{s^2 + (0.5+c)(1-s)^2}, \qquad a(c) = \frac{c}{5(1+c)}. \tag{5.6}$$

Notice that the system (5.5)–(5.6) can be rewritten in the nonconservative form

$$\mathbf{u}_t + A(\mathbf{u})\mathbf{u}_x = \mathbf{0},$$

where $\mathbf{u} = (s, c)^T$ and $A(\mathbf{u})$ has eigenvalues $\lambda_s = f_s(s, c)$ and $\lambda_c = f(s, c)(s + a'(c))^{-1}$ with right eigenvectors $\mathbf{r}_s = (1, 0)^T$, $\mathbf{r}_c = (f_c(s, c), \lambda_c - \lambda_s)^T$ for $s \in (0, 1)$ and $\mathbf{r}_c = (0, 1)^T$ for $s = 0, 1$, respectively. The computation of the one-sided local speeds of propagation (2.8) requires the calculation of the corresponding structure coefficients

$$\kappa_s^{\pm} = f_{ss}(s^{\pm}, c^{\pm}),$$

$$\kappa_c^{\pm} = \frac{f_c^{mid}(f_s^{\pm}(s^{\pm} + a'(c^{\pm})) - f^{\pm})}{(s^{\pm} + a'(c^{\pm}))^2} + \frac{(f^{mid}(s_{mid} + a'(c_{mid}))^{-1} - f_s^{mid})(f_c^{\pm}(s^{\pm} + a'(c^{\pm})) - a''(c^{\pm})f^{\pm})}{(s^{\pm} + a'(c^{\pm}))^2}$$

if $s_{mid} \neq 0, 1$, and

$$\kappa_c^{\pm} = \frac{f_c^{\pm}(s^{\pm} + a'(c^{\pm})) - a''(c^{\pm})f^{\pm}}{(s^{\pm} + a'(c^{\pm}))^2}$$

for $s_{mid} \in \{0, 1\}$. Here, we have used the notation $\mathcal{R}^{mid} := \mathcal{R}(s_{mid}, c_{mid})$, where $s_{mid} := (s^+ + s^-)/2$ and $c_{mid} := (c^+ + c^-)/2$, and $\mathcal{R}^{\pm} := \mathcal{R}(s^{\pm}, c^{\pm})$ for $\mathcal{R}$ being f_s, f_c, or f.

We consider the system (5.5)–(5.6) subject to the Riemann initial data taken from [14]:

$$(s(x, 0), c(x, 0)) = \begin{cases} (0.45, 0), & x \leq 0.252, \\ (0.20, 1), & x > 0.25. \end{cases} \tag{5.7}$$

The exact solution of this IVP is rather complicated and consists of both simple and composite waves (see [13]).

We numerically solve the IVP (5.5)–(5.7) using the central-upwind scheme (2.4)–(2.6), (2.8) with different piecewise polynomial reconstructions of the conserved variables s and b.

For adaption, we check at every grid cell whether a point, where one of the structure coefficients, either

$$\kappa_s = f_{ss}$$

or

$$\kappa_c = \begin{cases} a''f(s + a')^{-3}(f_s(s + a') - f), & 0 < s < 1, \\ (s + a')^{-2}(f_s(s + a') - a''f), & s \in \{0, 1\}, \end{cases}$$

changes sign, is nearby. If this is the case, namely, if the distance between the current grid point and one of the aforementioned points is smaller than $K\Delta x$, then the MM1 reconstruction is used; otherwise the WENO5 reconstruction is used.

In Figure 5.4, we plot the numerical solutions at time $t = 1$, calculated using the MM1 reconstruction with $\Delta x = 1/250$ and $\Delta x = 1/8000$ (the latter is used as a reference solution in all other figures shown in this section).

In Figure 5.5, we compare the c-component of the numerical solutions, computed using the MM1, MM2, SB, and WENO5 reconstructions on three different uniform meshes with $\Delta x = 1/250$, $1/500$, and $1/1000$. As one can clearly see in this figure, only the MM1 numerical solution recovers the correct structure of the composite

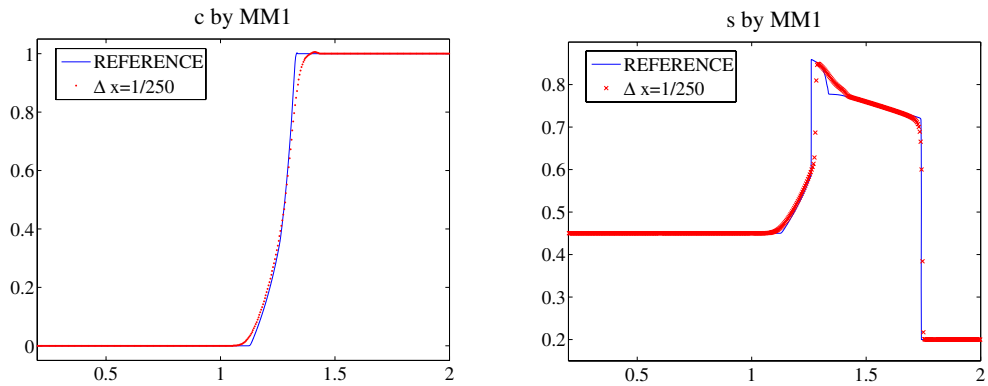


FIG. 5.4. *Solutions of* (5.5)–(5.7), *computed with MM*1*; (left) c- and (right) s-components.*

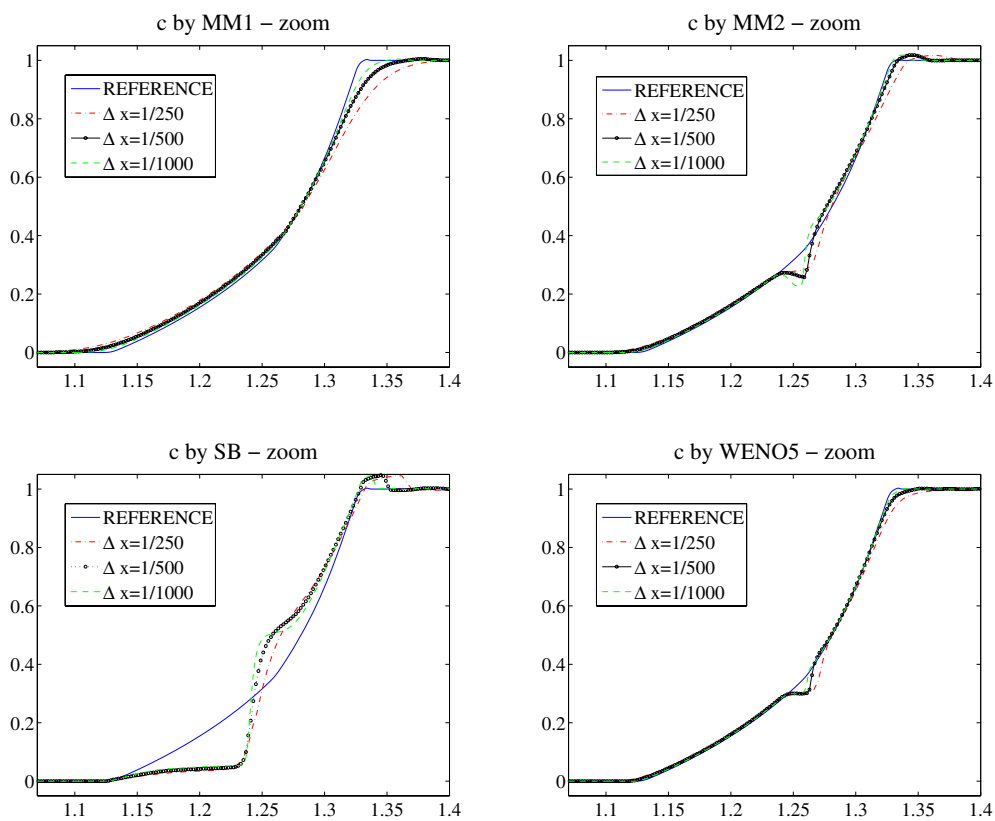


FIG. 5.5. *Solutions (c-component) of* (5.5)–(5.7), *computed with MM*1*, MM*2*, SB, and WENO*5*.*

wave, while the MM2, SB, and WENO5 reconstructions consistently fail to achieve this goal. Note that $\Delta x = 1/1000$ is a reasonably small mesh size that one usually uses in practical computations for such a problem (in the 2-D case, one would have to use an even coarser grid). To perform a numerical convergence study, we have computed the solution with even smaller Δx. It turns out that the numerical approximation

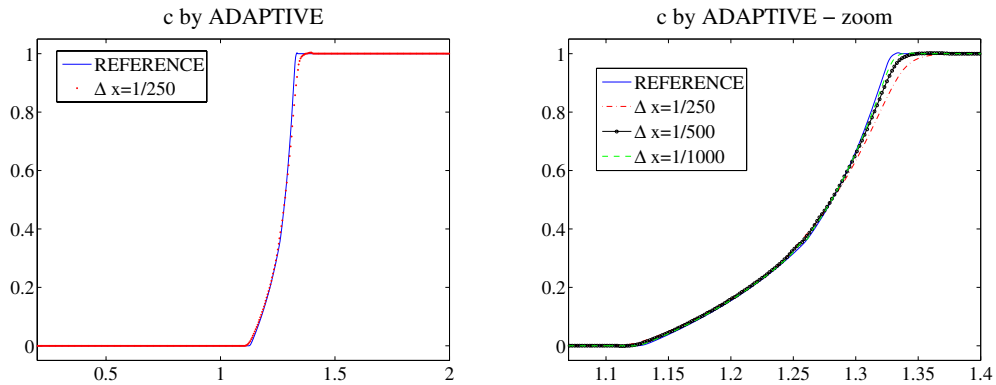


FIG. 5.6. *Solutions (c-component) of* (5.5)–(5.7), *computed adaptively with* $K = 2$ *(left) and its mesh refinement study (right).*

obtained with the WENO5 reconstruction seems to slowly converge to the reference solution, but the values of Δx, required in order to be able to observe this convergence, are so small that this positive result is not of practical interest. On the other hand, the results shown in Figure 5.6 demonstrate that our adaptive strategy both achieves a very high resolution and accurately captures the structure of the composite wave, and thus it is superior to both the lower resolution MM1 solution and the less reliable WENO5 solution.

Next, in Figures 5.7–5.8, we present the s-component of the computed solutions. While the wave structure of the s-component is quite different from that of the c-component, we observe the same effect as before: The central-upwind scheme based on either MM2, SB, or WENO5 reconstruction fails to recover the reference solution even when a very fine mesh with $\Delta x = 1/1000$ is used. At the same time, the schemes based on the MM1 and the adaptive reconstructions perform well. However, as expected, the overall resolution achieved by the adaptive scheme is higher than that achieved by the more dissipative MM1 scheme.

5.3. Two-dimensional scalar conservation law. In this section, we numerically solve the 2-D scalar conservation law

$$u_t + (\sin u)_x + (\cos u)_y = 0, \tag{5.8}$$

subject to the initial data

$$u(x, y, 0) = \begin{cases} 3.5\pi, & x^2 + y^2 < 1, \\ 0.25\pi, & \text{otherwise}, \end{cases} \tag{5.9}$$

using the scheme (2.10)–(2.12) and the piecewise polynomial reconstructions described in sections 3 and 4.

We have computed the solution at time $t = 1$ on a uniform grid with $\Delta x = \Delta y = 1/100$ using the MM1, MM2, SB, WENO5, and the adaptive reconstructions. For adaption, we mark the cells where the convexity of the flux changes, which occurs at $u = k\pi/2$, $k \in \mathbb{Z}$. In these cells and in all cells located within $K \max(\Delta x, \Delta y)$ from them, we use the MM1 reconstruction; otherwise the WENO5 reconstruction is employed.

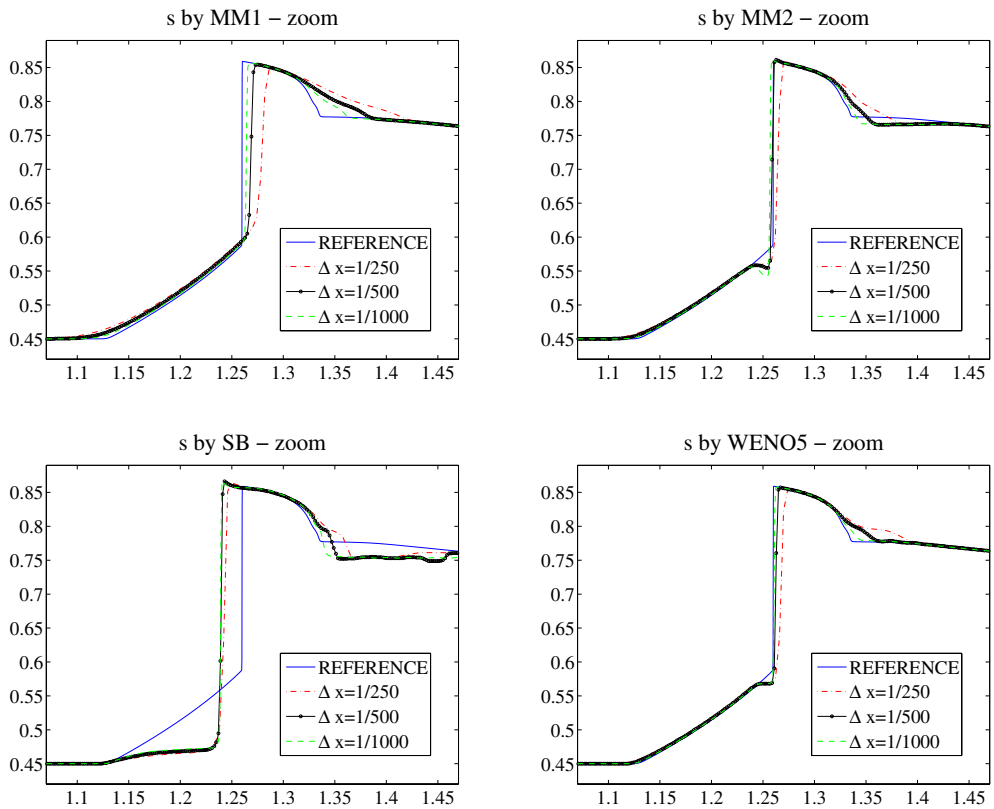


FIG. 5.7. *Solutions (s-component) of* (5.5)–(5.7)*, computed with MM*1*, MM*2*, SB, and WENO*5*.*

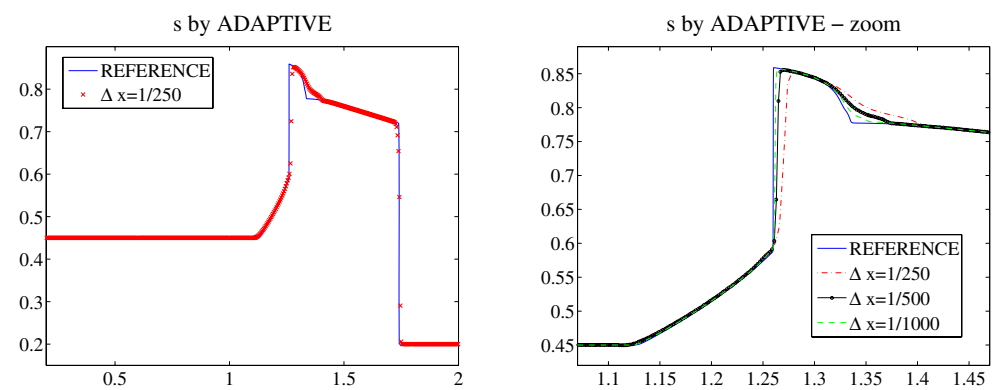


FIG. 5.8. *Solutions (s-component) of* (5.5)–(5.7)*, computed adaptively with* $K = 2$ *(left) and its mesh refinement study (right).*

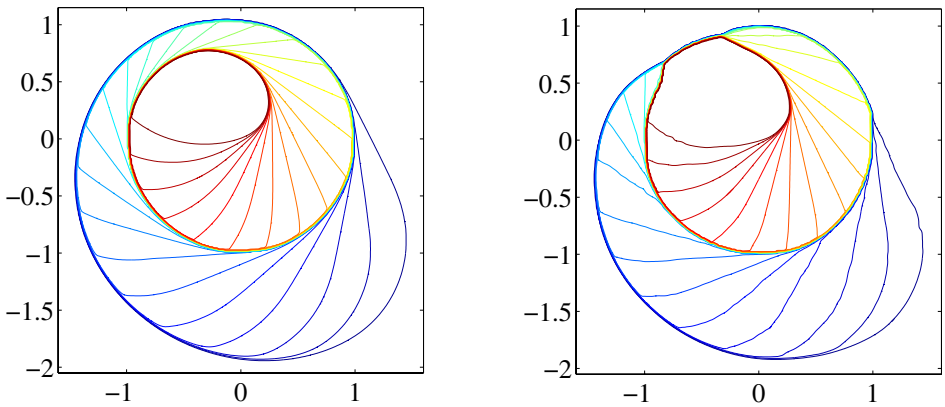


FIG. 5.9. *Solution of* (5.8)–(5.9), *computed with MM*1 *(left) and MM*2 *(right).*

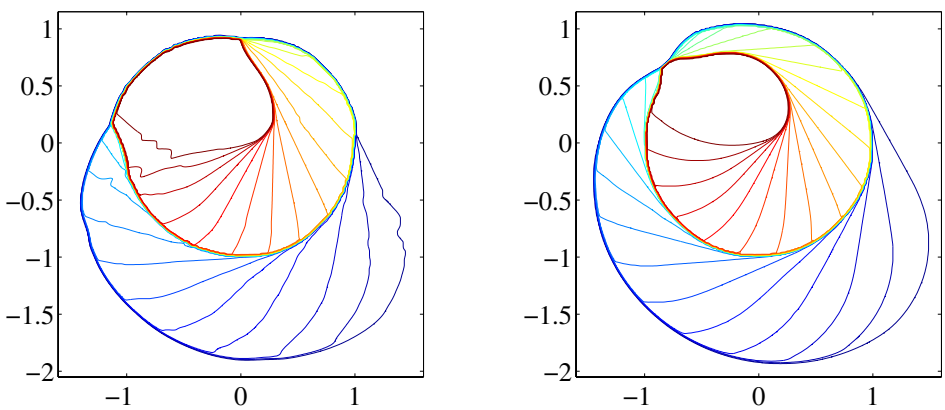


FIG. 5.10. *Solution of* (5.8)–(5.9), *computed with SB (left) and WENO*5 *(right).*

We present our results in Figures 5.9–5.11, where we observe the same effects as in the 1-D case. The central-upwind schemes based on the MM2, SB, and WENO5 reconstructions fail to resolve the composite wave structure, while the use of the MM1 or the adaptive reconstruction leads to capturing the accurate solution. We would like to point out that the adaptive strategy does not seem to be sensitive to the choice of the adaption constant K. We also show (see Figure 5.11 (right)), the areas where the MM1 (white) and the WENO5 (black) reconstructions were used. It can be clearly seen that the MM1 limiter is turned on only at or around transition areas.

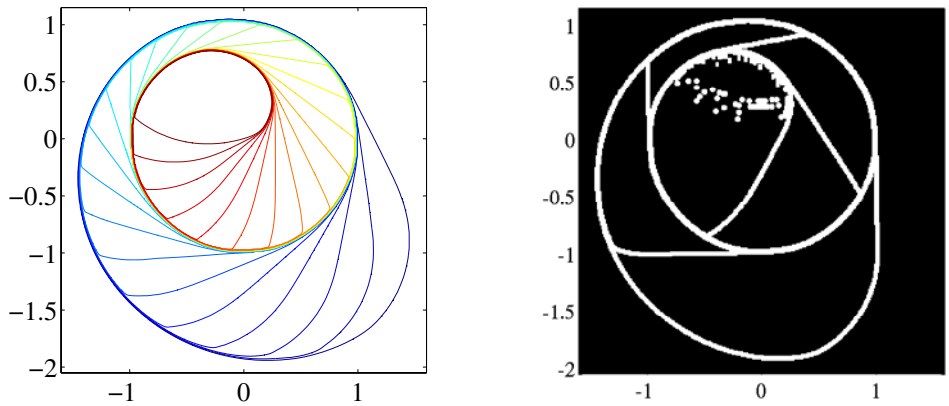


FIG. 5.11. *Solution of* (5.8)–(5.9), *computed adaptively with* $K = 2$ *(left) and the corresponding MM1/WENO5 regions (right).*

Acknowledgments. Part of this work was done in the Fall of 2005, when A. Kurganov visited the Department of Mathematics at the University of Michigan during the semester following Hurricane Katrina. A. Kurganov thanks the chairman, Professor A. Bloch, and all members of the department for their extremely warm hospitality.

REFERENCES

[1] P. BELL, P. COLELLA, AND J. TRANGENSTEIN, *Higher order Godunov methods for general systems of hyperbolic conservation laws*, J. Comput. Phys., 59 (1985), pp. 264–289.

[2] A. BETHE, *Report on the Theory of Shock Waves for an Arbitrary Equation of State*, Technical report PB-32189, U.S. Department of Commerce, 1942.

[3] B. COCKBURN, C. JOHNSON, C.-W. SHU, AND E. TADMOR, *Advanced numerical approximation of nonlinear hyperbolic equations*, CIME Lecture Notes, Lecture Notes in Math. 1697, A. Quarteroni, ed., Springer-Verlag, Berlin, 1997.

[4] P. COLELLA, *Multidimensional upwind methods for hyperbolic conservation laws*, J. Comput. Phys., 87 (1990), pp. 171–200.

[5] P. COLELLA AND H. GLAZ, *Efficient solution algorithms for the Riemann problem for real gases*, J. Comput. Phys., 59 (1985), pp. 264–289.

[6] F. COQUEL AND B. PERTHAME, *Relaxation of energy and approximate Riemann solvers for general pressure laws in fluid dynamics*, SIAM J. Numer. Anal., 35 (1998), pp. 2223–2249.

[7] S. GOTTLIEB, C.-W. SHU, AND E. TADMOR, *Strong stability-preserving high-order time discretization methods*, SIAM Rev., 43 (2001), pp. 89–112.

[8] A. HARTEN AND S. OSHER, *Uniformly high-order accurate nonoscillatory schemes*. I, SIAM J. Numer. Anal., 24 (1987), pp. 279–309.

[9] V. HAUGSE, K. H. KARLSEN, K.-A. LIE, AND J.R. NATVIG, *Numerical solution of the polymer system by front tracking*, Transp. Porous Media, 44 (2001), pp. 63–83.

[10] K. HOLING, J. ALVESTAD, AND J. A. TRANGENSTEIN, *The use of second-order Godunov methods for simulating EOR processes in realistic reservoir models*, in Second European Conference on the Mathematics of Oil Recovery, D. Guérillot and O. Guillon, eds., 1990, pp. 101–111.

[11] E. ISAACSON, *Global Solution of a Riemann Problem for a Non-Strictly Hyperbolic System of Conservation Laws Arising in Enhanced Oil Recovery*, preprint, Rockefeller University, New York, 1981.

[12] G.-S. JIANG AND C.-W. SHU, *Efficient implementation of weighted ENO schemes*, J. Comput. Phys., 126 (1996), pp. 202–228.

[13] T. Johansen and R. Winther, *The solution of the Riemann problem for a hyperbolic system of conservation laws modeling polymer flooding*, SIAM J. Math. Anal., 19 (1988), pp. 541–566.

[14] K. H. Karlsen, K.-A. Lie, J. R. Natvig, H. F. Nordhaug, and H. K. Dahle, *Operator splitting methods for systems of convection-diffusion equations: Nonlinear error mechanisms and correction strategies*, J. Comput. Phys., 173 (2001), pp. 636–663.

[15] A. Kurganov, S. Noelle, and G. Petrova, *Semidiscrete central-upwind schemes for hyperbolic conservation laws and Hamilton–Jacobi equations*, SIAM J. Sci. Comput., 23 (2001), pp. 707–740.

[16] A. Kurganov and G. Petrova, *A third-order semi-discrete genuinely multidimensional central scheme for hyperbolic conservation laws and related problems*, Numer. Math., 88 (2001), pp. 683–729.

[17] A. Kurganov and E. Tadmor, *Solution of two-dimensional Riemann problems for gas dynamics without Riemann problem solvers*, Numer. Methods Partial Differential Equations, 18 (2002), pp. 584–608.

[18] P. G. LeFloch, *Hyperbolic Systems of Conservation Laws. The Theory of Classical and Nonclassical Shock Waves*, Lectures Math. ETH Zürich, Birkhäuser Verlag, Basel, 2002.

[19] R. LeVeque, *Finite Volume Methods for Hyperbolic Problems*, Cambridge Texts Appl. Math., Cambridge University Press, Cambridge, UK, 2002.

[20] D. Levy, G. Puppo, and G. Russo, *Central WENO schemes for hyperbolic systems of conservation laws*, M2AN Math. Model. Numer. Anal., 33 (1999), pp. 547–571.

[21] D. Levy, G. Puppo, and G. Russo, *Compact central WENO schemes for multidimensional conservation laws*, SIAM J. Sci. Comput., 22 (2000), pp. 656–672.

[22] K.-A. Lie and S. Noelle, *On the artificial compression method for second-order nonoscillatory central difference schemes for systems of conservation laws*, SIAM J. Sci. Comput., 24 (2003), pp. 1157–1174.

[23] T.-P. Liu, *The Riemann problem for general* 2×2 *conservation laws*, Trans. Amer. Math. Soc., 199 (1974), pp. 89–112.

[24] T.-P. Liu, *The Riemann problem for general systems of conservation laws*, J. Differential Equations, 18 (1975), pp. 218–234.

[25] X.-D. Liu and S. Osher, *Nonoscillatory high order accurate self-similar maximum principle satisfying shock capturing schemes* I, SIAM J. Numer. Anal., 33 (1996), pp. 760–779.

[26] X.-D. Liu, S. Osher, and T. Chan, *Weighted essentially non-oscillatory schemes*, J. Comput. Phys., 115 (1994), pp. 200–212.

[27] R. Menikoff and B. Plohr, *The Riemann problem for fluid flow of real materials*, Rev. Modern Phys., 61 (1989), pp. 75–130.

[28] H. Nessyahu and E. Tadmor, *Non-oscillatory central differencing for hyperbolic conservation laws*, J. Comput. Phys., 87 (1990), pp. 408–463.

[29] G. A. Pope, *The application of fractional flow theory to enhanced oil recovery*, Soc. Pet. Eng. J., 20 (1980), pp. 191–205.

[30] P. L. Roe, *Characteristic-based schemes for the Euler equations*, in Annual Review of Fluid Mechanics, Vol. 18, Annual Reviews, Palo Alto, CA, 1986, pp. 337–365.

[31] R. Saurel, M. Larini, and J. C. Loraud, *Exact and approximate Riemann solvers for real gases*, J. Comput. Phys., 112 (1994), pp. 126–137.

[32] J. Shi, C. Hu, and C.-W. Shu, *A technique of treating negative weights in WENO schemes*, J. Comput. Phys., 175 (2002), pp. 108–127.

[33] C.-W. Shu, *High-order finite difference and finite volume WENO schemes and discontinuous Galerkin methods for CFD*, Int. J. Comput. Fluid Dyn., 17 (2003), pp. 107–118.

[34] P. K. Sweby, *High resolution schemes using flux limiters for hyperbolic conservation laws*, SIAM J. Numer. Anal., 21 (1984), pp. 995–1011.

[35] B. van Leer, *Towards the ultimate conservative difference scheme*, V. *A second-order sequel to Godunov's method*, J. Comput. Phys., 32 (1979), pp. 101–136.

[36] A. Voss, *Exact Riemann Solution for the Euler Equations with Nonconvex and Nonsmooth Equation of State*, Ph.D. thesis, Technical University of Aachen, Aachen, Germany, 2004.

[37] B. Wang and H. Glaz, *Second Order Godunov-Like Schemes for Gas Dynamics with a Nonconvex Equation of State*, AIAA Report AIAA-99-3256, AIAA, Reston, VA, 1999.

[38] B. Wendroff, *The Riemann problem for materials with nonconvex equations of state*. I. *Isentropic flow*, J. Math. Anal. Appl., 38 (1972), pp. 454–466.

[39] B. Wendroff, *The Riemann problem for materials with nonconvex equations of state*. II. *General flow*, J. Math. Anal. Appl., 38 (1972), pp. 640–658.

[40] A. L. Zachary, A. Malagoli, and P. Colella, *A higher-order Godunov method for multidimensional ideal magnetohydrodynamics*, SIAM J. Sci. Comput., 15 (1994), pp. 263–284.

SIAM J. SCI. COMPUT.
Vol. 29, No. 6, pp. 2402–2425
© 2007 Society for Industrial and Applied Mathematics

OPTIMIZED MULTIPLICATIVE, ADDITIVE, AND RESTRICTED ADDITIVE SCHWARZ PRECONDITIONING*

A. ST-CYR†, M. J. GANDER‡, AND S. J. THOMAS§

Abstract. We demonstrate that a small modification of the multiplicative, additive, and restricted additive Schwarz preconditioner at the algebraic level, motivated by optimized Schwarz methods defined at the continuous level, leads to a significant reduction in the iteration count of the iterative solver. Numerical experiments using finite difference and spectral element discretizations of the positive definite Helmholtz problem and an idealized climate simulation illustrate the effectiveness of the new approach.

Key words. domain decomposition, restricted additive Schwarz method, optimized Schwarz methods, multiplicative Schwarz, spectral elements, high-order methods

AMS subject classifications. 65F19, 65N22, 65N35

DOI. 10.1137/060652610

1. Introduction.

The classical Schwarz method employs Dirichlet transmission conditions between subdomains. By introducing a more general Robin transmission condition, it is possible to optimize the convergence of the original algorithm; see [9] and the references therein. In this paper, general results are derived for using optimized transmission conditions at the algebraic level of restricted additive Schwarz (RAS), multiplicative Schwarz (MS), and additive Schwarz (AS) on an augmented system. These methods are then applied to the positive definite Helmholtz problem $(\eta - \Delta)u = f$, $\eta > 0$, discretized with finite differences and spectral finite elements, by a simple modification of already existing classical implementations of RAS, MS, and AS, with and without overlap.

Optimized Schwarz methods were originally derived from Fourier analysis of the continuous elliptic partial differential equation (PDE); see [9] and references therein. Until now, it was not clear how to introduce optimized transmission conditions in the classical forms of the AS, MS, and RAS preconditioners at the algebraic level. The present work closes this gap by showing that small modifications of the subdomain matrices in these preconditioners lead to optimized Schwarz methods. The modified preconditioners must satisfy specific compatibility conditions. For optimized RAS

*Received by the editors February 21, 2006; accepted for publication (in revised form) February 27, 2007; published electronically October 17, 2007. This work was partially supported by National Science Foundation Collaborations in Mathematics and Geosciences grant 0222282 and the Department of Energy Climate Change Prediction Program. Computer time was provided by NSF MRI grants CNS-0421498, CNS-0420873, and CNS-0420985; NSF sponsorship of the National Center for Atmospheric Research (NCAR); the University of Colorado; and a grant from the IBM Shared University Research program. This work was performed by an employee of the U.S. Government or under U.S. Government contract. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights.
http://www.siam.org/journals/sisc/29-6/65261.html
†Corresponding author. NCAR, 1850 Table Mesa Drive, Boulder, CO 80305 (amik@ucar.edu). Traveling funds for this author were made available by the Early Career Scientist Association and the Scientific Computing Division (both at NCAR) and by Geneva University.
‡Section de Mathématiques, Université de Genève, CP 64, CH-1211 Genève, (mgander@math.unige.ch). Traveling funds for this author were made available by the Early Career Scientist Association and the Scientific Computing Division (both at NCAR) and by Geneva University.
§NCAR, 1850 Table Mesa Drive, Boulder, CO 80305 (thomas@ucar.edu).

(ORAS), an overlap condition must be satisfied. In the case of optimized MS (OMS), there is no such condition on the overlap. Optimized AS (OAS) requires an augmented system, permitting the use of nonoverlapping regions, where the common unknowns at subdomain boundaries are duplicated. Because these results are algebraic, they readily apply to any space discretization of continuous PDEs for which optimized Schwarz methods have been analyzed in the literature; see [9] for positive definite Helmholtz problems, [12, 10] for indefinite Helmholtz problems, and [14, 6] for advection diffusion problems. Greatly enhanced convergence factors have been shown for one-level optimized Schwarz methods in these references at the continuous level, with much weaker dependence on the overlap than for classical one-level Schwarz methods. There are also first attempts to directly construct algebraically good transmission conditions, which are related to approximations of Schur complement matrices; see [22, 17, 28, 11]. There are, however, no two-level variants yet for optimized Schwarz methods.

Schwarz algorithms are reviewed in the following section, and the link between the discrete and continuous problems is established. The equivalence of the iterative forms of optimized Schwarz methods with ORAS and OMS is demonstrated for multiple domains, as well as with OAS applied to an augmented system. These results are then applied to the positive definite Helmholtz problem using finite difference and spectral element discretizations. The size of the overlap region is varied for finite differences, whereas nonoverlapping subdomains are considered for spectral elements. We finally apply the new techniques to a spectral element climate model [27].

2. Classical Schwarz methods. Schwarz methods for elliptic PDEs subdivide the physical domain into subdomains and iterate the subdomain solutions in order to obtain the global solution. When the subdomain problems are discretized, Schwarz methods can be formulated at the discrete or algebraic level.

2.1. Schwarz methods at the algebraic level. Discretization of the elliptic PDE

$$\mathcal{L}u = f \quad \text{in } \Omega, \quad \mathcal{B}u = g \quad \text{on } \partial\Omega, \tag{2.1}$$

where $\mathcal{L}$ is an elliptic differential operator, $\mathcal{B}$ is a boundary operator, and Ω is a bounded domain, leads to a linear system of equations

$$A\boldsymbol{u} = \boldsymbol{f}. \tag{2.2}$$

A stationary iterative method for (2.2) is given by

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + M^{-1}(\boldsymbol{f} - A\boldsymbol{u}^n). \tag{2.3}$$

An initial guess $\boldsymbol{u}^0$ is required to start the iteration. Algebraic domain decomposition methods group the unknowns into subsets, $\boldsymbol{u}_j = R_j\boldsymbol{u}$, $j = 1, \ldots, J$, where R_j are rectangular restriction matrices such that each entry u_i of the vector $\boldsymbol{u}$ is contained in at least one $\boldsymbol{u}_j$. Coefficient matrices for subdomain problems are defined by $A_j = R_j A R_j^T$. The classical MS method is a stationary iteration (2.3), where the preconditioner M is

$$M_{MS}^{-1} = \left[I - \prod_{j=1}^{J} \left(I - R_j^T A_j^{-1} R_j A \right) \right] A^{-1}. \tag{2.4}$$

MS is often written using fractional steps as

$$\boldsymbol{u}^{n+\frac{j}{J}} = \boldsymbol{u}^{n+\frac{j-1}{J}} + R_j^T A_j^{-1} R_j(\boldsymbol{f} - A\boldsymbol{u}^{n+\frac{j-1}{J}}), \quad j = 1, \ldots, J, \tag{2.5}$$

which shows that it is equivalent to a Gauss–Seidel iteration on the subdomains [25]. It thus corresponds to the original alternating method introduced by Schwarz at the continuous level [23]. A more parallel variant is the AS preconditioner introduced in [5],

$$M_{AS}^{-1} = \sum_{j=1}^{J} R_j^T A_j^{-1} R_j. \tag{2.6}$$

In general, the AS iteration (2.3), (2.6) does not converge, nor does it correspond to a classical iteration per subdomain [7]. The RAS method [3] corrects this problem by introducing prolongation operators $\tilde{R}_j^T$, corresponding to a nonoverlapping decomposition. Each entry u_i of the vector $\boldsymbol{u}$ occurs in $\tilde{R}_j\boldsymbol{u}$ for exactly one j. The RAS preconditioner is

$$M_{RAS}^{-1} = \sum_{j=1}^{J} \tilde{R}_j^T A_j^{-1} R_j. \tag{2.7}$$

Another way to alleviate the convergence problems of AS is to apply M_{AS} as a preconditioner in a Krylov iteration. In general, MS and RAS also will be accelerated by Krylov iteration.

The algebraic formulation of Schwarz methods has an important feature: a subdomain matrix A_j is not necessarily the restriction of A to a subdomain j. For example, if A represents a spectral element discretization of a differential operator, then A_j can be obtained from a finite element discretization on the Gaussian quadrature points. Furthermore, subdomain matrices A_j can be chosen to accelerate convergence, and this is the focus of the next section.

2.2. Schwarz methods at the continuous level. Historically, domain decomposition methods were formulated at the continuous level. Consider a decomposition of the original domain Ω into two overlapping subdomains Ω_1 and Ω_2, and denote the interfaces by $\Gamma_{ij} = \partial\Omega_i \cap \Omega_j$, $i \neq j$, and the outer boundaries by $\partial\Omega_j = \partial\Omega \cap \bar{\Omega}_j$.

The original alternating Schwarz method introduced in [23] for (2.1) is

$$\begin{array}{rclll} \mathcal{L}u_1^{n+1} & = & f & \text{in } \Omega_1, & \mathcal{L}u_2^{n+1} = f \quad \text{in } \Omega_2, \\ \mathcal{B}(u_1^{n+1}) & = & g & \text{on } \partial\Omega_1, & \mathcal{B}(u_2^{n+1}) = g \quad \text{on } \partial\Omega_2, \\ u_1^{n+1} & = & u_2^n & \text{on } \Gamma_{12}, & u_2^{n+1} = u_1^{n+1} \quad \text{on } \Gamma_{21}, \end{array} \tag{2.8}$$

corresponding to Gauss–Seidel traversal of the subdomains. The Jacobi variant of the algorithm introduced in [16] is

$$\begin{array}{rclll} \mathcal{L}u_1^{n+1} & = & f & \text{in } \Omega_1, & \mathcal{L}u_2^{n+1} = f \quad \text{in } \Omega_2, \\ \mathcal{B}(u_1^{n+1}) & = & g & \text{on } \partial\Omega_1, & \mathcal{B}(u_2^{n+1}) = g \quad \text{on } \partial\Omega_2, \\ u_1^{n+1} & = & u_2^n & \text{on } \Gamma_{12}, & u_2^{n+1} = u_1^n \quad \text{on } \Gamma_{21}. \end{array} \tag{2.9}$$

It was shown in [7] that the discrete form of (2.8), represented by

$$A_1\boldsymbol{u}_1^{n+1} = \boldsymbol{f}_1 + B_{12}\boldsymbol{u}_2^n, \quad A_2\boldsymbol{u}_2^{n+1} = \boldsymbol{f}_2 + B_{21}\boldsymbol{u}_1^{n+1}, \tag{2.10}$$

is equivalent to MS in (2.4), and that the discrete form of (2.9),

$$A_1 \boldsymbol{u}_1^{n+1} = \boldsymbol{f}_1 + B_{12}\boldsymbol{u}_2^n, \quad A_2 \boldsymbol{u}_2^{n+1} = \boldsymbol{f}_2 + B_{21}\boldsymbol{u}_1^n, \tag{2.11}$$

is equivalent to RAS in (2.7).

3. Optimized Schwarz methods. Optimized Schwarz methods were developed at the continuous level using Fourier analysis and led to significantly faster convergence rates than classical Schwarz methods [9]. However, an algebraic formulation of optimized Schwarz methods is lacking. Here it is shown how optimized Schwarz methods can be embedded within the framework of RAS, MS, and AS.

3.1. Optimized Schwarz methods at the continuous level. In optimized Schwarz methods, the Dirichlet transmission conditions on the interfaces Γ_{ij} in (2.8) and (2.9) are replaced by more effective conditions including derivatives. The latter permits the use of nonoverlapping domains. The optimized algorithm for the classical case is

$$\begin{array}{rclll rcll} \mathcal{L}u_1^{n+1} & = & f & \text{in } \Omega_1, & \mathcal{L}u_2^{n+1} & = & f & \text{in } \Omega_2, \\ \mathcal{B}(u_1^{n+1}) & = & g & \text{on } \partial\Omega_1, & \mathcal{B}(u_2^{n+1}) & = & g & \text{on } \partial\Omega_2, \\ \tilde{\mathcal{B}}_{12}u_1^{n+1} & = & \tilde{\mathcal{B}}_{12}u_2^n & \text{on } \Gamma_{12}, & \tilde{\mathcal{B}}_{21}u_2^{n+1} & = & \tilde{\mathcal{B}}_{21}u_1^{n+1} & \text{on } \Gamma_{21}, \end{array} \tag{3.1}$$

where the transmission operators $\mathcal{B}_{12}$ and $\mathcal{B}_{21}$ are chosen to obtain faster convergence. Similarly, the more parallel Jacobi version of the optimized Schwarz method is

$$\begin{array}{rclll rcll} \mathcal{L}u_1^{n+1} & = & f & \text{in } \Omega_1, & \mathcal{L}u_2^{n+1} & = & f & \text{in } \Omega_2, \\ \mathcal{B}(u_1^{n+1}) & = & g & \text{on } \partial\Omega_1, & \mathcal{B}(u_2^{n+1}) & = & g & \text{on } \partial\Omega_2, \\ \tilde{\mathcal{B}}_{12}u_1^{n+1} & = & \tilde{\mathcal{B}}_{12}u_2^n & \text{on } \Gamma_{12}, & \tilde{\mathcal{B}}_{21}u_2^{n+1} & = & \tilde{\mathcal{B}}_{21}u_1^n & \text{on } \Gamma_{21}. \end{array} \tag{3.2}$$

The interface formulation was extensively tested in [9] for $\mathcal{L} = \eta - \Delta$, and optimized transmission conditions of Robin and second-order type were developed, leading to significantly faster convergence rates.

A discretization with optimized transmission conditions leads to discrete versions of the form (2.10) and (2.11). The subdomain matrices A_j are replaced by $\tilde{A}_j$, and the transmission operators B_{jk} are replaced by $\tilde{B}_{jk}$, corresponding to the optimized transmission conditions derived at the continuous level for various types of PDEs; see [9, 12, 10, 14, 6] and references therein.

3.2. Optimized restricted additive Schwarz methods. The ORAS method is based on the RAS method (2.3) and (2.7) but uses different subdomain matrices, which come from the optimized Jacobi–Schwarz method

$$\tilde{A}_j \boldsymbol{u}_j^{n+1} = \boldsymbol{f}_j + \sum_{k=1}^{J} \tilde{B}_{jk}\boldsymbol{u}_k^n, \quad j = 1, \ldots, J, \tag{3.3}$$

where $\tilde{B}_{jj} = 0$ to simplify the notation. We show that, for sufficient overlap, the subdomain matrices A_j in the RAS algorithm (2.3) and (2.7) can be replaced by the optimized subdomain matrices $\tilde{A}_j$ from (3.3) to obtain an ORAS method,

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \left(\sum_{j=1}^{J} \tilde{R}_j^T \tilde{A}_j^{-1} R_j \right) (\boldsymbol{f} - A\boldsymbol{u}^n), \tag{3.4}$$

which produces the same iterates as (3.3). An advantage of the ORAS implementation (3.4) over the optimized Schwarz method (3.3) is that the additional transmission operators $\tilde{B}_{jk}$ are not needed in ORAS, which greatly simplifies the transition from an existing RAS implementation to ORAS. For a simplified proof in the case of two subdomains and a link to the Schur complement system, consult [28].

DEFINITION 3.1 (consistency). *Let R_j, $j = 1, \ldots, J$ be restriction operators covering the entire discrete domain, and let $\boldsymbol{f}_j = R_j \boldsymbol{f}$. The matrix splitting R_j, $\tilde{A}_j$, $\tilde{B}_{jk}$, $j = 1, \ldots, J$ and $k = 1, \ldots, J$, in* (3.3) *is consistent if for all $\boldsymbol{f}$ and associated solution $\boldsymbol{u}$ of* (2.2), *$\boldsymbol{u}_j = R_j \boldsymbol{u}$ satisfy*

$$\tilde{A}_j \boldsymbol{u}_j = \boldsymbol{f}_j + \sum_{k=1}^{J} \tilde{B}_{jk} \boldsymbol{u}_k. \tag{3.5}$$

LEMMA 3.2. *Let A in* (2.2) *have full rank. For a consistent matrix splitting R_j, $\tilde{A}_j$, $\tilde{B}_{jk}$, $j = 1, \ldots, J$ and $k = 1, \ldots, J$, we have the matrix identities*

$$\tilde{A}_j R_j - \sum_{k=1}^{J} \tilde{B}_{jk} R_k = R_j A, \quad j = 1, \ldots, J. \tag{3.6}$$

Proof. For an arbitrary $\boldsymbol{f}$, we apply R_j to (2.2) and obtain, using consistency (3.5),

$$R_j A \boldsymbol{u} = R_j \boldsymbol{f} = \boldsymbol{f}_j = \tilde{A}_j \boldsymbol{u}_j - \sum_{k=1}^{J} \tilde{B}_{jk} \boldsymbol{u}_k.$$

Now using $\boldsymbol{u}_j = R_j \boldsymbol{u}$ and $\boldsymbol{u}_k = R_k \boldsymbol{u}$ on the right-hand side yields

$$\left(\tilde{A}_j R_j - \sum_{k=1}^{J} \tilde{B}_{jk} R_k - R_j A \right) \boldsymbol{u} = 0.$$

Because $\boldsymbol{f}$ was arbitrary and A is invertible, the identity is true for all $\boldsymbol{u}$, and therefore the identity (3.6) is established. ∎

While the definition of consistency is simple, it has important consequences: if the classical submatrices are used, i.e., $\tilde{A}_j = A_j = R_j A R_j^T$, then the restriction operators R_j can be overlapping or nonoverlapping, and with the associated B_{jk}, we obtain a consistent splitting R_j, A_j, B_{jk}, $j = 1, \ldots, J$, $k = 1, \ldots, J$. However, if alternative subdomain matrices $\tilde{A}_j$ are employed, then the restriction operators R_j must be such that the unknowns in $\boldsymbol{u}_j$ affected by the change in $\tilde{A}_j$ are also available in neighboring $\boldsymbol{u}_k$ to compensate via $\tilde{B}_{jk}$ in (3.5). Hence consistency implies for all nonclassical splittings a condition on the overlap in the R_j. The following further restriction on the overlap is necessary for the optimized RAS algorithm.

ASSUMPTION 1. *We assume that the restriction operators R_j and the prolongation operators $\tilde{R}_j^T$ from the ORAS method* (3.4) *and the transmission operators $\tilde{B}_{jk}$ from the optimized parallel Schwarz method* (3.3) *satisfy*

$$\tilde{B}_{jk} R_k \tilde{R}_m^T = 0, \quad m \neq k. \tag{3.7}$$

Remark 1. The conditions (3.7) in Assumption 1 relate the overlap to the operators $\tilde{B}_{jk}$. Classical Schwarz methods only employ interface values from neighboring

subdomains, and thus these conditions are automatically satisfied. The optimized Schwarz operators $\tilde{B}_{jk}$ approximate derivatives, requiring values at the interface and neighboring points. The operators $\tilde{R}_j$ must not overwrite these values, hence the requirement of a small overlap in the ORAS formulation.

LEMMA 3.3. *Let R_j, $j = 1, \dots, J$, be restriction operators covering the entire discrete domain, and let $\tilde{R}_j$ be the corresponding RAS versions of these operators. If Assumption* 1 *holds, then $\tilde{B}_{jk} R_k \tilde{R}_k^T = \tilde{B}_{jk}$ for all $j = 1, \dots, J$, $k = 1, \dots, J$.*

Proof. We first note that by the nonoverlapping definition of $\tilde{R}_j$, $j = 1, \dots, J$, the identity matrix I can be written as

$$I = \sum_{j=1}^{J} \tilde{R}_j^T \tilde{R}_j = \sum_{j=1}^{J} \tilde{R}_j^T R_j. \tag{3.8}$$

Now, multiplying for $m \neq k$ the identity $\tilde{B}_{jk} R_k \tilde{R}_m^T = 0$ on the right by R_m and substituting the term $\tilde{R}_m^T R_m$ using (3.8) leads to

$$\tilde{B}_{jk} R_k \left(I - \sum_{l \neq m} \tilde{R}_l^T R_l \right) = 0 \quad \Longrightarrow \quad \tilde{B}_{jk} R_k = \sum_{l \neq m} \tilde{B}_{jk} R_k \tilde{R}_l^T R_l,$$

which, using Assumption 1, gives $\tilde{B}_{jk} R_k = \tilde{B}_{jk} R_k \tilde{R}_k^T R_k$. Hence we obtain $(\tilde{B}_{jk} - \tilde{B}_{jk} R_k \tilde{R}_k^T) R_k = 0$, which completes the proof, because the restriction matrix R_k has full rank. ∎

We are now ready to prove the main result of this subsection.

THEOREM 3.4. *Let R_j, $\tilde{A}_j$, $\tilde{B}_{jk}$, $j = 1, \dots, J$ and $k = 1, \dots, J$, be a consistent matrix splitting, and let $\tilde{R}_j$ be the corresponding RAS versions of R_j. If the initial iterates $\boldsymbol{u}_j^0$, $j = 1, \dots, J$, of the optimized Jacobi–Schwarz method* (3.3) *and the initial iterate $\boldsymbol{u}^0$ of the ORAS method* (3.4) *satisfy*

$$\boldsymbol{u}^0 = \sum_{j=1}^{J} \tilde{R}_j^T \boldsymbol{u}_j^0, \tag{3.9}$$

and if Assumption 1 *holds, then the two methods* (3.3) *and* (3.4) *generate an identical sequence of iterates,*

$$\boldsymbol{u}^n = \sum_{j=1}^{J} \tilde{R}_j^T \boldsymbol{u}_j^n. \tag{3.10}$$

Remark 2. While condition (3.9) is important in proving that the sequence of iterates of the two methods, and thus the two methods, are equivalent, in practice one would not need to compute an initial condition $\boldsymbol{u}^0$ from given initial conditions $\boldsymbol{u}_j^0$ or vice versa, even though this can be achieved directly using the formula (3.9). Note that for given $\boldsymbol{u}^0$, the choice of the $\boldsymbol{u}_j^0$ is not unique: entries in $\boldsymbol{u}_j^0$ erased by $\tilde{R}_j^T$ can be chosen arbitrarily.

Proof. The proof is by induction. For $n = 0$, (3.10) follows by assumption (3.9) on the initial iterates. We now assume that $\boldsymbol{u}^n = \sum_{j=1}^{J} \tilde{R}_j^T \boldsymbol{u}_j^n$ and show that the identity (3.10) holds for $n + 1$. Applying Lemma 3.2 to the jth term of the sum in

(3.4), we obtain

$$
\begin{aligned}
\tilde{R}_j^T \tilde{A}_j^{-1} R_j(\boldsymbol{f} - A\boldsymbol{u}^n) &= \tilde{R}_j^T \tilde{A}_j^{-1}(\boldsymbol{f}_j - R_j A\boldsymbol{u}^n) \\
&= \tilde{R}_j^T \tilde{A}_j^{-1}\left(\boldsymbol{f}_j - \left(\tilde{A}_j R_j - \sum_{k=1}^{J} \tilde{B}_{jk} R_k\right)\boldsymbol{u}^n\right) \\
&= \tilde{R}_j^T \left(\tilde{A}_j^{-1}\boldsymbol{f}_j - R_j\boldsymbol{u}^n + \tilde{A}_j^{-1}\sum_{k=1}^{J} \tilde{B}_{jk} R_k \boldsymbol{u}^n\right).
\end{aligned} \tag{3.11}
$$

Substituting this expression into (3.4) and using the identity (3.8) leads to

$$
\boldsymbol{u}^{n+1} = \sum_{j=1}^{J} \tilde{R}_j^T \tilde{A}_j^{-1}\left(\boldsymbol{f}_j + \sum_{k=1}^{J} \tilde{B}_{jk} R_k \boldsymbol{u}^n\right).
$$

Now using the induction hypothesis to replace $\boldsymbol{u}^n$ by $\sum_{m=1}^{J} \tilde{R}_m^T \boldsymbol{u}_m^n$ on the right-hand side, we obtain

$$
\boldsymbol{u}^{n+1} = \sum_{j=1}^{J} \tilde{R}_j^T \tilde{A}_j^{-1}\left(\boldsymbol{f}_j + \sum_{k=1}^{J}\sum_{m=1}^{J} \tilde{B}_{jk} R_k \tilde{R}_m^T \boldsymbol{u}_m^n\right).
$$

By Assumption 1, the inner sum over m vanishes except for $m = k$, and together with Lemma 3.3, the above relation becomes

$$
\boldsymbol{u}^{n+1} = \sum_{j=1}^{J} \tilde{R}_j^T \tilde{A}_j^{-1}\left(\boldsymbol{f}_j + \sum_{k=1}^{J} \tilde{B}_{jk} R_k \tilde{R}_k^T \boldsymbol{u}_k^n\right) = \sum_{j=1}^{J} \tilde{R}_j^T \tilde{A}_j^{-1}\left(\boldsymbol{f}_j + \sum_{k=1}^{J} \tilde{B}_{jk} \boldsymbol{u}_k^n\right).
$$

Finally, using (3.3), we find

$$
\boldsymbol{u}^{n+1} = \sum_{j=1}^{J} \tilde{R}_j^T \boldsymbol{u}_j^{n+1},
$$

which concludes the proof by induction. ∎

3.3. Optimized multiplicative Schwarz methods. The OMS method is an alternative to ORAS. In contrast to ORAS with an additional condition on the overlap, the consistency of the splitting suffices here. However, OMS needs a coloring of subdomains to remain a parallel method. To implement a parallel optimized Schwarz method without conditions on the overlap, an augmented formulation is required, as shown in subsection 3.4.

The OMS method is based on the MS method (2.5), but uses different subdomain matrices, which come from the discretized optimized alternating Schwarz method

$$
\tilde{A}_j \boldsymbol{u}_j^{n+1} = \boldsymbol{f}_j + \sum_{k=1}^{j-1} \tilde{B}_{jk} \boldsymbol{u}_k^{n+1} + \sum_{k=j+1}^{J} \tilde{B}_{jk} \boldsymbol{u}_k^n, \quad j = 1, \ldots, J. \tag{3.12}
$$

This method differs from (3.3), because subdomain solves are performed in a Gauss–Seidel fashion. The MS method (2.5) always uses the most recently updated values in the iteration by definition. To guarantee the same property for the alternating Schwarz

method (3.12), we need the following assumption which states this in mathematical terms.

ASSUMPTION 2. *We assume that the restriction operators R_j and the transmission operators $\tilde{B}_{jk}$ in the alternating Schwarz method* (3.12) *satisfy*

$$\tilde{B}_{jk} R_k R_m^T = 0$$

for all $j = 1, \ldots, J$ *and*

$$k \neq \begin{cases} j-1 & \text{if } j > 1, \\ J & \text{otherwise,} \end{cases} \qquad m \in \begin{cases} \{1, \ldots, j-1, k+1, \ldots, J\} & \text{if } k > j, \\ \{k+1, \ldots, j-1\} & \text{otherwise.} \end{cases}$$

Remark 3. Assumption 2 appears deceptively similar to the overlap Assumption 1 in the ORAS method, but it is not a constraint on the overlap. It simply states that the most recently updated values must be used by the transmission operators $\tilde{B}_{jk}$ in (3.12), i.e., we need $\tilde{B}_{jk} R_k R_m^T = 0$, if values from subdomain m have been updated more recently than from subdomain k, which with the ordering implied by (3.12) leads to the conditions in Assumption 2.

We now show that by replacing the subdomain matrices A_j, $j = 1, \ldots, J$, in the MS method (2.5) by their optimized variants $\tilde{A}_j$ from (3.12), the new OMS method

$$\boldsymbol{u}^{n+\frac{j}{J}} = \boldsymbol{u}^{n+\frac{j-1}{J}} + R_j^T \tilde{A}_j^{-1} R_j (\boldsymbol{f} - A \boldsymbol{u}^{n+\frac{j-1}{J}}), \quad j = 1, \ldots, J, \tag{3.13}$$

produces the same iterates as (3.12), again without the explicit use of the transmission operators $\tilde{B}_{jk}$. Three lemmas are needed to prove this result.

LEMMA 3.5. *For $j = 1, \ldots, J$, the matrix identity $R_j(I - R_j^T R_j) = 0$ holds.*

Proof. The result follows by multiplying through by R_j:

$$R_j(I - R_j^T R_j) = (R_j - R_j R_j^T R_j) = (I - R_j R_j^T) R_j = 0,$$

because $R_j R_j^T = I$. ∎

LEMMA 3.6. *For overlapping or nonoverlapping restriction operators R_j, $j = 1, \ldots, J$, covering the entire domain, we have*

$$\prod_{j=1}^{J} (I - R_j^T R_j) = 0.$$

Proof. For $A \in \mathbb{R}^{m \times m}$, the rectangular Boolean matrices R_j are of dimensions $n_j \times m$ and have rank n_j. Let S_j be the index set formed by the integers representing columns of R_j containing a 1. Since the restriction operators R_j are covering the entire domain, we have $\cup_{j=1}^{J} S_j = \{1, 2, 3, \ldots, m\} \equiv M$. If for the index set S we denote by I_S the matrix with 1's on the diagonal indicated by the indices in S and zeros everywhere else, we have $I_{S_j^c} = I - R_j^T R_j$, where S_j^c denotes the complement of S_j in M. We now obtain

$$\prod_{j=1}^{J} (I - R_j^T R_j) = \prod_{j=1}^{J} I_{S_j^c} = I_{\cap_{j=1}^{J} S_j^c} = 0,$$

since $\cap_{j=1}^{J} S_j^c = (\cup_{j=1}^{J} S_j)^c = M^c = \emptyset$ by De Morgan's law. ∎

LEMMA 3.7. *For any given set of subdomain vectors* $\boldsymbol{u}_l$, $l = 1, \ldots, J$, *and under Assumption* 2 *we have*

$$\tilde{B}_{jk} R_k \sum_{l=1}^{j-1} \prod_{i=l+1}^{j-1} (I - R_i^T R_i) R_l^T \boldsymbol{u}_l = \begin{cases} \tilde{B}_{jk} \boldsymbol{u}_k & \text{if } k < j, \\ 0 & \text{otherwise,} \end{cases} \tag{3.14}$$

and

$$\tilde{B}_{jk} R_k \prod_{i=1}^{j-1} (I - R_i^T R_i) \left(\sum_{p=1}^{J} \prod_{q=p+1}^{J} (I - R_q^T R_q) R_p^T \boldsymbol{u}_p \right) = \begin{cases} \tilde{B}_{jk} \boldsymbol{u}_k & \text{if } k > j, \\ 0 & \text{otherwise.} \end{cases} \tag{3.15}$$

Proof. We first start by showing (3.14): if $k < j$, we split the left-hand side in (3.14) into three parts:

$$\begin{aligned} \tilde{B}_{jk} R_k \sum_{l=1}^{j-1} \prod_{i=l+1}^{j-1} (I - R_i^T R_i) R_l^T \boldsymbol{u}_l &= \sum_{l=1}^{k-1} \tilde{B}_{jk} R_k \prod_{i=l+1}^{j-1} (I - R_i^T R_i) R_l^T \boldsymbol{u}_l \\ &+ \tilde{B}_{jk} R_k \prod_{i=k+1}^{j-1} (I - R_i^T R_i) R_k^T \boldsymbol{u}_k + \sum_{l=k+1}^{j-1} \tilde{B}_{jk} R_k \prod_{i=l+1}^{j-1} (I - R_i^T R_i) R_l^T \boldsymbol{u}_l. \end{aligned} \tag{3.16}$$

The first sum on the right vanishes, because $k < j$ and the product includes the term $i = k$, and because the product terms commute, each term in the sum contains $R_k(I - R_k^T R_k)$ which is identically zero by Lemma 3.5. In the second term on the right in (3.16), two cases need to be distinguished: if $k = j - 1$, then the product is empty, and because $R_k R_k^T = I$, the term equals $\tilde{B}_{jk} \boldsymbol{u}_k$. On the other hand, if $k < j - 1$, we obtain

$$\tilde{B}_{jk} R_k \prod_{i=k+1}^{j-1} (I - R_i^T R_i) R_k^T \boldsymbol{u}_k = \tilde{B}_{jk} R_k R_k^T \boldsymbol{u}_k = \tilde{B}_{jk} \boldsymbol{u}_k,$$

where the assumption that $\tilde{B}_{jk} R_k R_i^T = 0$ for $k < j - 1$ and $i \in \{k + 1, \ldots, j - 1\}$ was used, leading to $R_k R_k^T = I$. Hence, the second term always equals $\tilde{B}_{jk} \boldsymbol{u}_k$. Now for the last term in (3.16), two cases are again possible: if $k = j - 1$, then the sum is empty and thus the term is zero. If $k < j - 1$, then

$$\sum_{l=k+1}^{j-1} \tilde{B}_{jk} R_k \prod_{i=l+1}^{j-1} (I - R_i^T R_i) R_l^T \boldsymbol{u}_l = \sum_{l=k+1}^{j-1} \tilde{B}_{jk} R_k R_l^T \boldsymbol{u}_l = 0,$$

because again by assumption $\tilde{B}_{jk} R_k R_i^T = 0$ for $k < j - 1$ and $i \in \{k + 1, \ldots, j - 1\}$, and also $\tilde{B}_{jk} R_k R_l^T = 0$ for $k < j - 1$ and $l \in \{k + 1, \ldots, j - 1\}$, which concludes the proof for the first result (3.14) if $k < j$. Now if $k = j$, then the left-hand side of (3.14) is zero because by definition $\tilde{B}_{jj} = 0$. Finally, if $k > j$, zero is obtained for $j = 1$, because the sum is empty, and for $j > 1$

$$\sum_{l=1}^{j-1} \tilde{B}_{jk} R_k \prod_{i=l+1}^{j-1} (I - R_i^T R_i) R_l^T \boldsymbol{u}_l = \sum_{l=1}^{j-1} \tilde{B}_{jk} R_k R_l^T \boldsymbol{u}_l = 0,$$

where the assumption $\tilde{B}_{jk} R_k R_i^T = 0$ for $k > j$ and $i \in \{2, \ldots, j-1\}$ and $\tilde{B}_{jk} R_k R_l^T = 0$ for $k > j$ and $l \in \{1, \ldots, j-1\}$ was again employed. This concludes the proof of (3.14).

To show the second identity (3.15), first note that for $k \le j$,

$$\tilde{B}_{jk} R_k \prod_{i=1}^{j-1} (I - R_i^T R_i) = 0,$$

because for $k = j$, $\tilde{B}_{jk} = 0$, and for $k < j$ the product includes the term $R_k(I - R_k^T R_k)$ which is identically zero by Lemma 3.5. Now if $k > j$, then

$$\tilde{B}_{jk} R_k \prod_{i=1}^{j-1} (I - R_i^T R_i) = \tilde{B}_{jk} R_k,$$

because for $j = 1$ the product is empty, and for $j > 1$ we have by assumption $\tilde{B}_{jk} R_k R_i^T = 0$ for $k > j$ and $i \in \{1, \dots, j-1\}$. The remaining sum is separated into three terms:

(3.17)
$$\begin{aligned} \tilde{B}_{jk} R_k \sum_{p=1}^{J} \prod_{q=p+1}^{J} (I - R_q^T R_q) R_p^T \boldsymbol{u}_p &= \sum_{p=1}^{k-1} \tilde{B}_{jk} R_k \prod_{q=p+1}^{J} (I - R_q^T R_q) R_p^T \boldsymbol{u}_p \\ &+ \tilde{B}_{jk} R_k \prod_{q=k+1}^{J} (I - R_q^T R_q) R_k^T \boldsymbol{u}_k + \sum_{p=k+1}^{J} \tilde{B}_{jk} R_k \prod_{q=p+1}^{J} (I - R_q^T R_q) R_p^T \boldsymbol{u}_p. \end{aligned}$$

The first term on the right vanishes, because the product always contains the term $i = k$, and because the product terms commute, each term in the sum contains $R_k(I - R_k^T R_k)$ which is identically zero by Lemma 3.5. The second term on the right in (3.17) simplifies to

$$\tilde{B}_{jk} R_k \prod_{q=k+1}^{J} (I - R_q^T R_q) R_k^T \boldsymbol{u}_k = \tilde{B}_{jk} R_k R_k^T \boldsymbol{u}_k = \tilde{B}_{jk} \boldsymbol{u}_k,$$

because by assumption for $k > j$ and $k \neq J$ we have $\tilde{B}_{jk} R_k R_q^T = 0$ for $q \in \{k+1, \dots, J\}$, and for $k = J$ the product is empty. Finally the last term in (3.17) becomes

$$\sum_{p=k+1}^{J} \tilde{B}_{jk} R_k \prod_{q=p+1}^{J} (I - R_q^T R_q) R_p^T \boldsymbol{u}_p = \sum_{p=k+1}^{J} \tilde{B}_{jk} R_k R_p^T \boldsymbol{u}_p = 0,$$

because for $k = J$ the sum is empty, and for $j < k < J$, by assumption, $\tilde{B}_{jk} R_k R_q^T = 0$ for $q \in \{k+2, \dots, J\}$, and also $\tilde{B}_{jk} R_k R_p^T = 0$ for $p \in \{k+1, \dots, J\}$, which concludes the proof. ∎

THEOREM 3.8. *Let R_j, $\tilde{A}_j$, $\tilde{B}_{jk}$, $j = 1, \dots, J$, $k = 1, \dots, J$, be a consistent matrix splitting. If the initial iterates $\boldsymbol{u}_j^0$, $j = 1, \dots, J$, of the optimized Schwarz method* (3.12) *and the initial iterate $\boldsymbol{u}^0$ of the OMS method* (3.13) *satisfy*

$$\boldsymbol{u}^0 = \sum_{j=1}^{J} \prod_{i=j+1}^{J} (I - R_i^T R_i) R_j^T \boldsymbol{u}_j^0 \tag{3.18}$$

and Assumption 2 holds, then (3.12) *and* (3.13) *generate an equivalent sequence of iterates,*

$$\boldsymbol{u}^n = \sum_{j=1}^{J} \prod_{i=j+1}^{J} (I - R_i^T R_i) R_j^T \boldsymbol{u}_j^n. \tag{3.19}$$

Proof. The proof has two steps, both shown by induction. We first assume that for a fixed n, relation (3.19) holds and show by induction that this implies the relation

$$\boldsymbol{u}^{n+\frac{j}{J}} = \prod_{i=1}^{j}(I - R_i^T R_i)\boldsymbol{u}^n + \sum_{l=1}^{j}\prod_{i=l+1}^{j}(I - R_i^T R_i)R_l^T \boldsymbol{u}_l^{n+1} \tag{3.20}$$

for $j = 0, 1, \ldots, J$. This relation trivially holds for $j = 0$. Assuming that it holds for $j-1$, we find from (3.13) using the matrix identity (3.6)

$$\begin{aligned}
\boldsymbol{u}^{n+\frac{j}{J}} &= \boldsymbol{u}^{n+\frac{j-1}{J}} + R_j^T \tilde{A}_j^{-1} R_j(\boldsymbol{f} - A\boldsymbol{u}^{n+\frac{j-1}{J}}) \\
&= \boldsymbol{u}^{n+\frac{j-1}{J}} + R_j^T \tilde{A}_j^{-1}\left(\boldsymbol{f}_j - \tilde{A}_j R_j \boldsymbol{u}^{n+\frac{j-1}{J}} + \sum_{k=1}^{J}\tilde{B}_{jk}R_k\boldsymbol{u}^{n+\frac{j-1}{J}}\right) \\
&= (I - R_j^T R_j)\boldsymbol{u}^{n+\frac{j-1}{J}} + R_j^T \tilde{A}_j^{-1}\left(\boldsymbol{f}_j + \sum_{k=1}^{J}\tilde{B}_{jk}R_k\boldsymbol{u}^{n+\frac{j-1}{J}}\right).
\end{aligned}$$

Now, replacing relation (3.20) at $j-1$ in the last sum, together with Lemma 3.7, leads to

$$\begin{aligned}
\sum_{k=1}^{J}\tilde{B}_{jk}R_k\boldsymbol{u}^{n+\frac{j-1}{J}} &= \sum_{k=1}^{J}\tilde{B}_{jk}R_k\left(\prod_{i=1}^{j-1}(I - R_i^T R_i)\boldsymbol{u}^n + \sum_{l=1}^{j-1}\prod_{i=l+1}^{j-1}(I - R_i^T R_i)R_l^T\boldsymbol{u}_l^{n+1}\right) \\
&= \sum_{k=1}^{j-1}\tilde{B}_{jk}\boldsymbol{u}_k^{n+1} + \sum_{k=j+1}^{J}\tilde{B}_{jk}\boldsymbol{u}_k^n.
\end{aligned}$$

Substituting this result back into the expression for $\boldsymbol{u}^{n+\frac{j}{J}}$, we find, using (3.12) and relation (3.20) at $j-1$ again,

$$\begin{aligned}
\boldsymbol{u}^{n+\frac{j}{J}} &= (I - R_j^T R_j)\boldsymbol{u}^{n+\frac{j-1}{J}} + R_j^T \tilde{A}_j^{-1}\left(\boldsymbol{f}_j + \sum_{m=1}^{j-1}\tilde{B}_{jk}\boldsymbol{u}_k^{n+1} + \sum_{m=j+1}^{J}\tilde{B}_{jk}\boldsymbol{u}_k^n\right) \\
&= \prod_{i=1}^{j}(I - R_i^T R_i)\boldsymbol{u}^n + \sum_{l=1}^{j-1}\prod_{i=l+1}^{j}(I - R_i^T R_i)R_l^T\boldsymbol{u}_l^{n+1} + R_j^T\boldsymbol{u}_j^{n+1} \\
&= \prod_{i=1}^{j}(I - R_i^T R_i)\boldsymbol{u}^n + \sum_{l=1}^{j}\prod_{i=l+1}^{j}(I - R_i^T R_i)R_l^T\boldsymbol{u}_l^{n+1},
\end{aligned}$$

which concludes the first proof by induction. The main result (3.19) can now be proved by induction on n. By the assumption on the initial iterate, (3.19) holds for $n = 0$. Thus assuming it holds for n, we obtain from the first part for this n that relation (3.20) holds for $j = 0, 1, \ldots, J$. In particular, for $j = J$, we have

$$\boldsymbol{u}^{n+1} = \prod_{i=1}^{J}(I - R_i^T R_i)\boldsymbol{u}^n + \sum_{l=1}^{J}\prod_{i=l+1}^{J}(I - R_i^T R_i)R_l^T\boldsymbol{u}_l^{n+1},$$

and by Lemma 3.6 the first product vanishes, which completes the proof. ∎

3.4. Optimized additive Schwarz methods for the augmented system. To obtain parallel discrete optimized Schwarz methods, which in contrast to ORAS can be used in a nonoverlapping fashion, we need to introduce an augmented system, where unknowns in the overlap and at the interfaces are stored for each subdomain. Tang [29] called this an *enhanced* system, whereas the term *augmented* system was employed in [33].

At convergence, observe that the discretized optimized Schwarz method (3.3) leads naturally to the augmented system

$$\begin{bmatrix} \tilde{A}_1 & -\tilde{B}_{12} & \cdots & -\tilde{B}_{1,J-1} & -\tilde{B}_{1,J} \\ -\tilde{B}_{21} & \tilde{A}_2 & -\tilde{B}_{23} & \cdots & -\tilde{B}_{2,J} \\ \vdots & & \ddots & & \vdots \\ -\tilde{B}_{J-1,1} & \cdots & -\tilde{B}_{J-1,J-2} & \tilde{A}_{J-1} & -\tilde{B}_{J-1,J} \\ -\tilde{B}_{J,1} & -\tilde{B}_{J,2} & \cdots & -\tilde{B}_{J,J-1} & \tilde{A}_J \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \vdots \\ \boldsymbol{u}_{J-1} \\ \boldsymbol{u}_J \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \\ \vdots \\ \boldsymbol{f}_{J-1} \\ \boldsymbol{f}_J \end{bmatrix},$$

or in matrix notation, specifying augmented quantities by a hat,

$$\hat{A}\hat{\boldsymbol{u}} = \hat{\boldsymbol{f}}. \tag{3.21}$$

Applying a standard AS method with nonoverlapping restriction operators $\hat{R}_j$ to the augmented system (3.21), $\boldsymbol{u}_j = \hat{R}_j\hat{\boldsymbol{u}}$, we obtain the iterative method

$$\hat{\boldsymbol{u}}^{n+1} = \hat{\boldsymbol{u}}^n + \sum_{j=1}^{J} \hat{R}_j^T \tilde{A}_j^{-1} \hat{R}_j(\hat{\boldsymbol{f}} - \hat{A}\hat{\boldsymbol{u}}^n). \tag{3.22}$$

Note that the nonoverlapping restriction operators $\hat{R}_j$ are related to the restriction operators for the consistent splitting R_j, $\tilde{A}_j$, $\tilde{B}_j$, $j = 1, \ldots, J$: their range is the same, only their domain is enlarged to account for the augmented matrix $\hat{A}$.

THEOREM 3.9. *Let R_j, $\tilde{A}_j$, $\tilde{B}_{jk}$, $j = 1, \ldots, J$, $k = 1, \ldots, J$, be a consistent matrix splitting. If the initial iterates $\boldsymbol{u}_j^0$, $j = 1, \ldots, J$, of the optimized Jacobi–Schwarz method* (3.3) *and the initial iterate $\hat{\boldsymbol{u}}^0$ of the AS method applied to the augmented system* (3.22) *satisfy*

$$\hat{\boldsymbol{u}}^0 = \left[(\boldsymbol{u}_1^0)^T,\ (\boldsymbol{u}_2^0)^T, \ldots, (\boldsymbol{u}_J^0)^T\right]^T, \tag{3.23}$$

then (3.3) *and* (3.22) *generate the same sequence of iterates, i.e.,*

$$\hat{\boldsymbol{u}}^n = \left[(\boldsymbol{u}_1^n)^T,\ (\boldsymbol{u}_2^n)^T, \ldots, (\boldsymbol{u}_J^n)^T\right]^T. \tag{3.24}$$

Proof. The proof is again by induction. By assumption, (3.24) holds for $n = 0$. Assuming it holds for n, we obtain from (3.22), using the fact that the restriction operators $\hat{R}_j$, $j = 1, \ldots, J$, are nonoverlapping, Lemma 3.2, and (3.3),

$$\begin{aligned} \hat{\boldsymbol{u}}^{n+1} &= \hat{\boldsymbol{u}}^n + \sum_{j=1}^{J} \hat{R}_j^T \tilde{A}_j^{-1} \hat{R}_j(\hat{\boldsymbol{f}} - \hat{A}\hat{\boldsymbol{u}}^n) = \hat{\boldsymbol{u}}^n + \sum_{j=1}^{J} \hat{R}_j^T \tilde{A}_j^{-1}(\boldsymbol{f}_j - \hat{R}_j\hat{A}\hat{\boldsymbol{u}}^n) \\ &= \hat{\boldsymbol{u}}^n + \sum_{j=1}^{J} \hat{R}_j^T \tilde{A}_j^{-1}\left(\boldsymbol{f}_j - \tilde{A}_j\boldsymbol{u}_j^n + \sum_{l=1}^{J} \tilde{B}_{jl}\boldsymbol{u}_l^n\right) = \sum_{j=1}^{J} \hat{R}_j^T \tilde{A}_j^{-1}\left(\boldsymbol{f}_j + \sum_{l=1}^{J} \tilde{B}_{jl}\boldsymbol{u}_l^n\right) \\ &= \sum_{j=1}^{J} \hat{R}_j^T \boldsymbol{u}_j^{n+1} = \left[(\boldsymbol{u}_1^{n+1})^T,\ (\boldsymbol{u}_2^{n+1})^T, \ldots, (\boldsymbol{u}_J^{n+1})^T\right]^T = \hat{\boldsymbol{u}}^{n+1}, \end{aligned} \tag{3.25}$$

which concludes the proof. ∎

An advantage of this formulation is that nonoverlapping parallel preconditioners can be used, such as the optimized Jacobi–Schwarz method (3.3) and optimized alternating Schwarz method (3.12). A disadvantage is that the operators $\tilde{B}_{jk}$ must be constructed explicitly, because they are needed in the application of $\tilde{A}_j^{-1}$ in (3.22).

4. Application to the positive definite Helmholtz problem. We now apply the results of the previous sections to finite difference and spectral element discretizations of the positive definite Helmholtz problem on the domain Ω,

$$\mathcal{L}u = (\eta - \Delta)u = f \quad \text{in } \Omega, \tag{4.1}$$

where Ω is an open set in two dimensions, and we impose homogeneous Dirichlet conditions to simplify the exposition.

4.1. Finite differences. Discretizing (4.1) using a standard five-point discretization on an equidistant grid on the domain $\Omega = (0,1) \times (0,1)$, as shown in Figure 4.1, and using a lexicographic ordering enumerating the unknowns from bottom to top and left to right leads to the matrix problem

$$A\boldsymbol{u} = \boldsymbol{f}, \quad A = \frac{1}{h^2}\begin{bmatrix} T_\eta & -I & \\ -I & T_\eta & \ddots \\ & \ddots & \ddots \end{bmatrix}, \quad T_\eta = \begin{bmatrix} \eta h^2 + 4 & -1 & \\ -1 & \eta h^2 + 4 & \ddots \\ & \ddots & \ddots \end{bmatrix},$$

where $A \in \mathbb{R}^{m^2 \times m^2}$, $T_\eta \in \mathbb{R}^{m \times m}$, $I \in \mathbb{R}^{m \times m}$ denotes the identity matrix and $h = \frac{1}{m+1}$. Decomposing the domain Ω into the two subdomains $\Omega_1 = (0, \beta) \times (0,1)$ and $\Omega_2 = (\alpha, 1) \times (0,1)$, as indicated in Figure 4.1, we obtain the rectangular restriction matrices $R_1 \in \mathbb{R}^{mb \times m^2}$ and $R_2 \in \mathbb{R}^{m(m-a+1) \times m^2}$ of the form $R_1 = [I\ 0]$ and $R_2 = [0\ I]$, where the 1's in the identity matrices I of appropriate size correspond to the grid points indicated in Figure 4.1 by D_1 and D_2, respectively. The new restriction matrices $\tilde{R}_j$ are of the same size and form as R_j, $j = 1, 2$, but have 1's only on the diagonals for the grid points indicated by the nonoverlapping $\tilde{D}_j$ in Figure 4.1. The classical subdomain matrices associated with this decomposition are by definition $A_j = R_j A R_j^T$, $j = 1, 2$, which implies that $A_1 \in \mathbb{R}^{mb \times mb}$ and $A_2 \in \mathbb{R}^{m(m-a+1) \times m(m-a+1)}$.

To obtain optimized RAS, MS, and AS methods, it suffices according to the results in section 3 to simply replace the classical subdomain matrices A_j by optimized subdomain matrices $\tilde{A}_j$ from optimized Schwarz methods. Since one changes only the transmission conditions to obtain optimized Schwarz methods, it suffices to replace the last diagonal block T_η in A_1, which is an interface block to the second subdomain, and the first diagonal block T_η in A_2, which is an interface block to the first subdomain, by the blocks corresponding to optimized transmission conditions to obtain the optimized subdomain matrices $\tilde{A}_j$. We choose here interface blocks based on the continuous analysis of the underlying PDE and transmission conditions [9], which suggests replacing the two interface blocks T_η by the matrix

$$\tilde{T} = \frac{1}{2}T_\eta + phI + \frac{q}{h}(T_0 - 2I), \quad T_0 = T_\eta|_{\eta=0}, \tag{4.2}$$

which corresponds to a general optimized transmission condition of order 2 with two parameters p and q; see [9]. The optimal choice for the parameters p and q in the

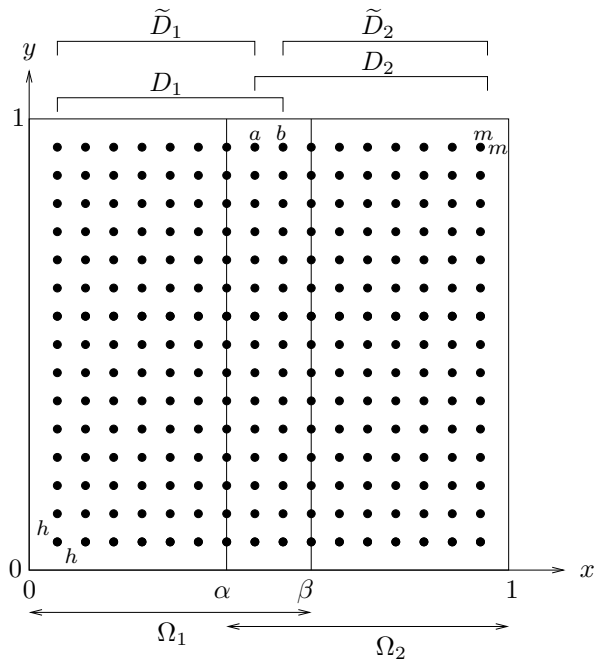


FIG. 4.1. *Finite difference discretization points and a two-subdomain decomposition.*

TABLE 4.1

Hierarchy of choices for the parameters p and q in the new interface blocks $\tilde{T}$ in (4.2). Here, TOj stands for "Taylor of order j," and OOj stands for "optimized of order j."

	p	q
TO0	$\sqrt{\eta}$	0
TO2	$\sqrt{\eta}$	$\frac{1}{2\sqrt{\eta}}$
OO0, no overlap	$\sqrt{\pi}(k_{\min}^2+\eta)^{1/4}h^{-1/2}$	0
OO0, overlap Ch	$2^{-1/3}(k_{\min}^2+\eta)^{1/3}(Ch)^{-1/3}$	0
OO2, no overlap	$2^{-1/2}\pi^{1/4}(k_{\min}^2+\eta)^{3/8}h^{-1/4}$	$2^{-1/2}\pi^{-3/4}(k_{\min}^2+\eta)^{-1/8}h^{3/4}$
OO2, overlap Ch	$2^{-3/5}(k_{\min}^2+\eta)^{2/5}(Ch)^{-1/5}$	$2^{-1/5}(k_{\min}^2+\eta)^{-1/5}(Ch)^{3/5}$

new block $\tilde{T}$ depends on the problem parameter η, the overlap in the method, the mesh parameter h, and the lowest frequency along the interface, $k_{\min}$, which in our case would be $k_{\min}=\pi$, since this is the lowest possible mode at the interfaces fixed at $y=0$ and $y=1$ by a Dirichlet condition. In [9], the hierarchy of choices in Table 4.1 was derived for h small for the PDE (4.1). Figure 4.2 displays the effect of replacing the classical subdomain matrices A_j by the optimized subdomain matrices $\tilde{A}_j$, $j=1,2$, on the convergence of RAS for the model problem (4.1) on the unit square with $\eta=1$ and $h=1/30$. We applied the asymptotic formulas from Table 4.1 for the various choices of the parameters p and q in the new interface blocks $\tilde{T}$ (4.2) of the optimized subdomain matrices $\tilde{A}_j$. Clearly, the convergence of RAS is greatly enhanced with the new subdomain matrices $\tilde{A}_j$, and exchanging the interface blocks T_η by $\tilde{T}$ does not change the number of operations per iteration in any way.

There is an interesting observation concerning the overlap: in the experiment above, we used an overlap at the matrix level of two blocks in the definition of the

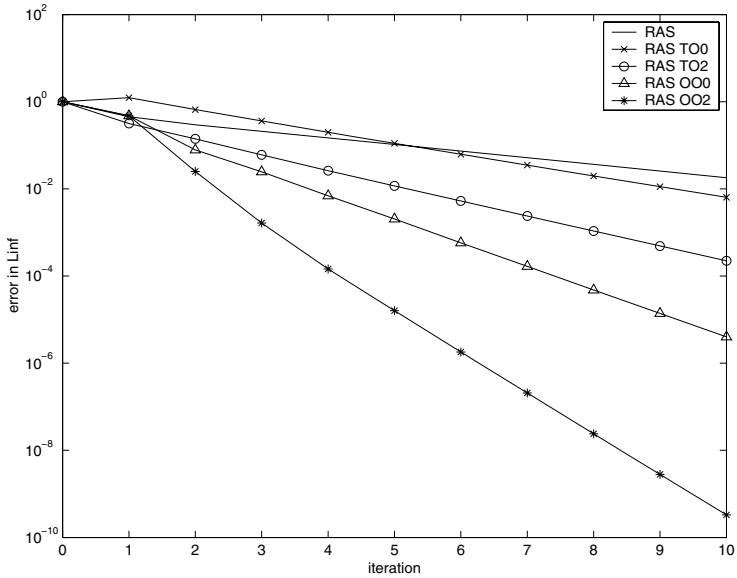


FIG. 4.2. *Convergence curves of classical RAS (top), with overlap* $3h$*, compared to the hierarchy of ORAS methods with overlap* h*.*

restriction operators R_j, $j = 1, 2$, which corresponds to the drawing in Figure 4.1 and is the smallest possible satisfying the overlap condition in Theorem 3.4. This corresponds for the classical RAS method to an overlap of $3h$, because the matrices $B_{12} = B_{21} = h^{-2}I$, which pick up the Dirichlet values from the other subdomains in (2.11), are a distance $3h$ apart in physical space; see the decomposition in Figure 4.1. This distance remains the same in ORAS, but the values picked up by the new $\tilde{B}_{12}$ and $\tilde{B}_{21}$ now include, in addition to those picked up by B_{12} and B_{21}, values in the neighboring row corresponding to the changes in $\tilde{A}_j$, and the values from B_{12} and B_{21} are used for the normal derivatives inside the neighboring domain. This implies that the physical interface has shifted by h on each subdomain, and hence the physical overlap is reduced by $2h$ and is thus only of size h for ORAS. In other words, the new physical interface is now on the grid lines denoted by a and b in Figure 4.1. We therefore used $C = 1$ for overlap h in the formulas in Table 4.1 to determine the parameters p and q. This shows that ORAS cannot be used without physical overlap, because we chose the smallest overlap according to Theorem 3.4, even though optimized Schwarz methods can be used without overlap.

Another interesting equivalence is now apparent: from formula (4.2), we observe that the new interface blocks $\tilde{T}$ and the classical interface blocks T coincide if one chooses

$$p = \frac{2 + \eta h^2}{2h}, \quad q = \frac{h}{2}, \tag{4.3}$$

and thus the ORAS method with parameters (4.3) and overlap h is identical to the classical RAS method with overlap $3h$. The numerical results in Figure 4.2 demonstrate, however, that the reduction in overlap is more than compensated for by a good choice of the parameters p and q.

In Figure 4.3, we illustrate the impact of the new subdomain matrices on the

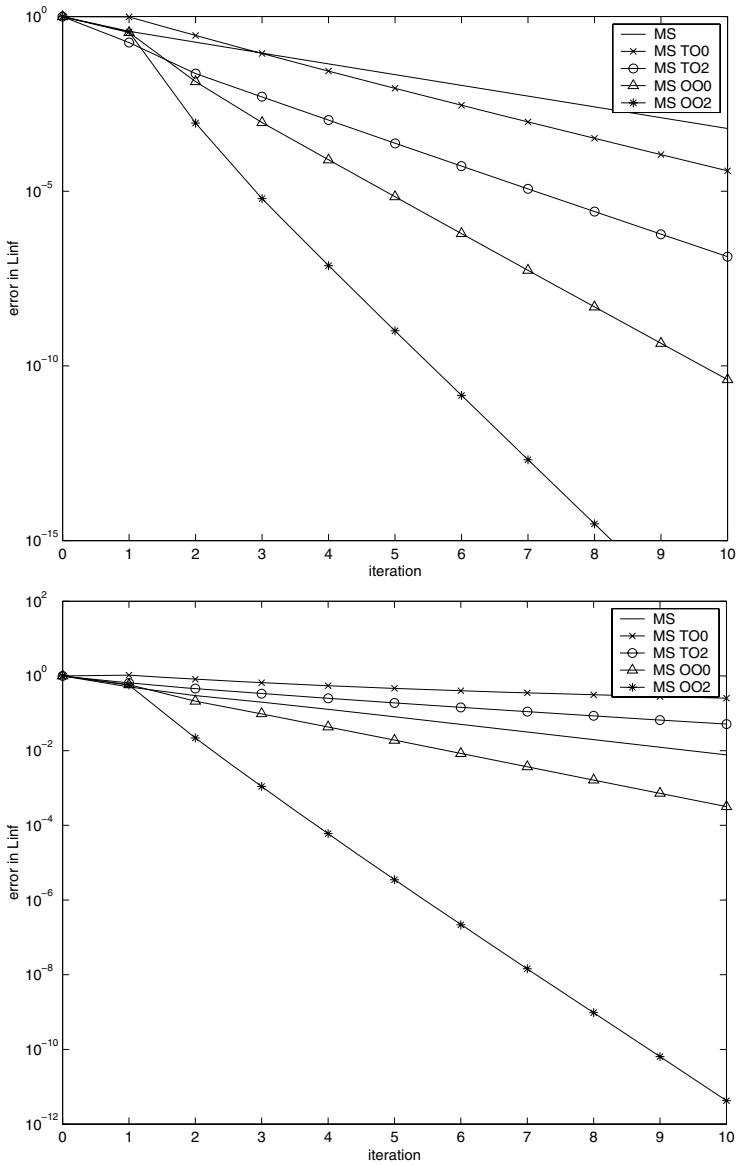


FIG. 4.3. *Convergence curves of classical MS compared to the hierarchy of OMS methods, at the top for overlap h (3h for classical MS), and at the bottom for zero overlap (2h for classical MS).*

convergence of MS. The error is plotted on top for the same configuration as in the previous experiments with RAS. As expected, the convergence is about two times faster than for RAS. At the bottom we show the results obtained for OMS with zero overlap, which corresponds in the classical case to an overlap of $2h$. Here, one can see that the Taylor transmission conditions are not effective enough to compensate for the loss of overlap, even though asymptotically they are comparable to classical MS with overlap $2h$, as shown in [9]. The optimized transmission conditions without overlap again improve upon the classical results with overlap by a large margin.

4.2. Spectral elements. In a nodal spectral element discretization, the computational domain Ω is partitioned into J quadrilateral elements Ω_j in which the dependent and independent variables are approximated by Nth order tensor-product polynomial expansions. A function $u : \mathbb{R}^2 \to \mathbb{R}$ is expanded, on element Ω_j, in terms of the Nth degree Lagrangian interpolants h_i as defined in [21],

$$u^h(\mathbf{x}(r_1^j, r_2^j))|_{\Omega_j} \equiv u_j(\mathbf{x}(r_1^j, r_2^j)) \equiv \sum_{k=0}^{N}\sum_{l=0}^{N} (\mathbf{u}_j)_{kl}\, h_k(r_1^j(\mathbf{x}))\, h_l(r_2^j(\mathbf{x})), \tag{4.4}$$

where $\mathbf{x} \to (r_1^j(\mathbf{x}), r_2^j(\mathbf{x}))$ is an affine transformation from the quadrilateral element Ω_j, on the original domain, to the reference element $[-1,1]\times[-1,1]$, and $(\mathbf{u}_j)_{kl}$ are the nodal basis coefficients defined at the Gauss–Lobatto–Legendre (GLL) quadrature points $\{\xi_i\}_{i=0}^N$. The weak formulation is obtained by integrating the differential equation (4.1) with respect to test functions in $H_0^1(\Omega)$ if Dirichlet conditions are assumed on the physical boundary $\partial\Omega$. By directly evaluating integrals using Gaussian quadrature, the generalized Galerkin problem consists of finding u in $X^N \equiv \mathbb{P}_{N,J}^2 \cap H_0^1(\Omega)$ such that

$$a^h(\phi, u) \equiv \eta(\phi, u)_{GLL} + (\nabla\phi, \nabla u)_{GLL} = (\phi, f)_{GLL} \equiv f^h(\phi) \tag{4.5}$$

for all ϕ in X^N, where the polynomial space $\mathbb{P}_{N,J}^2$ of degree N is defined as in [8]. The integrals are obtained as follows:

$$(f,g)_{GLL} \equiv \sum_{j=1}^{J}\sum_{k,l=0}^{N} f(\mathbf{x}^j(\xi_k,\xi_l))|_{\Omega_k} \cdot g(\mathbf{x}^j(\xi_k,\xi_l))|_{\Omega_j}\ |\mathcal{J}^j(\xi_k,\xi_l)|\rho_k\rho_l, \tag{4.6}$$

where $|\mathcal{J}^j(\xi_k,\xi_l)|$ is the Jacobian of the transformation $\mathbf{x}^j(\mathbf{r})$ and $\{\rho_i\}_{i=0}^N$ are the GLL weights associated with the quadrature points $\{\xi_i\}_{i=0}^N$. In our numerical experiments, the Jacobian is simply $\frac{h^2}{4}$ for all quadrilaterals Ω_j.

In order to construct a nonoverlapping algorithm, the augmented system must be explicitly derived to find an expression for the matrices $\tilde{B}_{ij}$. This can be accomplished in two different ways. The first approach employs the weak formulation directly on each domain. The second is simply to use the consistency relation in Lemma 3.2 and to construct the matrices $\tilde{B}_{jk}$. The latter can be extracted by specifying the form of the operator $\tilde{A}_j$ on each element. The complete derivation can be found in [26] for general curvilinear domains. Assuming that the spectral elements Ω_j coincide with the subdomains and that a general transmission condition between each element is used,

$$\left[\frac{\partial u_j}{\partial \mathbf{n}} + T(u_j,p,q,\tau)\right]_{\Gamma_{ij}}^{n+1} = \left[\frac{\partial u_i}{\partial \mathbf{n}} + T(u_j,p,q,\tau)\right]_{\Gamma_{ij}}^{n},$$

where $T(u_j,p,q,\tau) \equiv pu_j - q\frac{\partial^2 u_j}{\partial\tau^2}$, we now construct the OAS algorithm. At the continuous level, the optimized Schwarz algorithm solves the following problem for u_j^{n+1} on each Ω_j:

$$\begin{array}{rcll} \eta u_j^{n+1} - \Delta u_j^{n+1} & = & f_j(x,y) & \text{in } \Omega_j, \\ T(u_j^{n+1},p,q,\tau) + \frac{\partial u_j^{n+1}}{\partial \mathbf{n}_{jl}} & = & T(u_l^n,p,q,\tau) + \frac{\partial u_l^n}{\partial \mathbf{n}_{jl}} & \text{on } \Gamma_{jl} \equiv \partial\Omega_j \cap \partial\Omega_l, \\ u_j^{n+1} & = & g(x,y) & \text{on } \partial\Omega_j \cap \partial\Omega, \end{array} \tag{4.7}$$

where $l \in \mathcal{N}(\Omega_j)$, which represents the set of integers containing the indices of the elements which are neighbors of Ω_j sharing a complete edge or a vertex, $\mathbf{n}_{jl}$ is a normal vector pointing from element Ω_j to element Ω_l along the boundary Γ_{jl}, and superscript n denotes the iterate of the Schwarz algorithm. After some manipulation, one step of the discrete optimized Schwarz algorithm for the spectral element discretization is to find u_j^{n+1} in $X_j^N \equiv \mathbb{P}_{N,J}^2 \cap H^1(\Omega_j) \cap H_0^1(\Omega)$ such that

$$\begin{aligned} a_j^h(u_j^{n+1}, \phi_j) + \sum_{l \in \mathcal{N}(\Omega_j)} \langle \phi_j, T(u_j^{n+1}, p, q, \tau) \rangle|_{\Gamma_{jl}} &= f_j^h(\phi_j) + \sum_{l \in \mathcal{N}(\Omega_j)} f_l^h(\phi_l|_{\Gamma_{jl}}) \\ - \sum_{l \in \mathcal{N}(\Omega_j)} a_l^h(u_l^n, \phi_l|_{\Gamma_{jl}}) + \sum_{l \in \mathcal{N}(\Omega_j)} \langle \phi_l, T(u_l^n, p, q, \tau) \rangle|_{\Gamma_{jl}} & \end{aligned} \tag{4.8}$$

for all ϕ_j in X_j^N. The bracket $\langle \cdot, \cdot \rangle|_{\Gamma_{jl}}$ is the discrete one-dimensional integral on an edge or vertex Γ_{jl} performed using the GLL quadrature, and $a_j^h(\cdot, \cdot)$ is identical to $a^h(\cdot, \cdot)$ with the exception that the sum is taken for element Ω_j only with the same convention for $f_j^h(\cdot)$. The notation $\phi_l|_{\Gamma_{jl}}$ denotes the trace of the test function on the edge Γ_{jl}, including end points, and $\phi_l|_{\Gamma_{jl}}(x) = \phi_j|_{\Gamma_{jl}}(x)$ since, at convergence, the functions must be in $H_0^1(\Omega)$; see [19], for example.

Remark 4. The vertex integrals are performed in the same way as their edge or surface counterparts except that the test function is nonzero only at the vertex point and zero at all other collocation points. Using the basis described in [8], such a test function for the vertex situated at $-(1,1)$, on the reference element, would be $h_0(r_1^j)h_0(r_2^j)$.

The proposed algorithm (4.8) is a relaxation of the assembly procedure at the heart of any finite element method. The following links to Lemma 3.2 can be established:

$$\underline{\phi}_j^T \tilde{A}_j \hat{R}_j \hat{\mathbf{u}} = a_j^h(\mathbf{u}_j, \underline{\phi}_j) + \sum_{l \in \mathcal{N}(\Omega_j)} \langle \underline{\phi}_j, T(\mathbf{u}_j, p, q, \tau) \rangle|_{\Gamma_{jl}}, \tag{4.9}$$

$$\underline{\phi}_j^T \sum_{l=1}^{J} \tilde{B}_{jl} \hat{R}_l \hat{\mathbf{u}} = - \sum_{l \in \mathcal{N}(\Omega_j)} a_l^h(\mathbf{u}_l, \underline{\phi}_l|_{\Gamma_{jl}}) + \sum_{l \in \mathcal{N}(\Omega_j)} \langle \underline{\phi}_l, T(\mathbf{u}_l, p, q, \tau) \rangle|_{\Gamma_{jl}}, \tag{4.10}$$

$$\underline{\phi}_j^T \hat{R}_j A \hat{\mathbf{u}} = a_j^h(\mathbf{u}_j, \underline{\phi}_j) + \sum_{l \in \mathcal{N}(\Omega_j)} a_l^h(\mathbf{u}_l, \underline{\phi}_l|_{\Gamma_{jl}}), \tag{4.11}$$

$$\underline{\phi}_j^T \hat{R}_j \hat{\mathbf{f}} = f_j^h(\underline{\phi}_j) + \sum_{l \in \mathcal{N}(\Omega_j)} f_l^h(\underline{\phi}_l|_{\Gamma_{jl}}) \tag{4.12}$$

for all $1 \le j \le J$ and vector $\underline{\phi}_j$ such that $\mathbf{u}_j = \hat{R}_j \hat{\mathbf{u}}$. Finally, the preconditioned system can be written as

$$\left(I - \sum_{j=1}^{J} \hat{R}_j^T \tilde{A}_j^{-1} \sum_{l=1}^{J} \tilde{B}_{jl} \hat{R}_l \right) \hat{\mathbf{u}} = \sum_{j=1}^{J} \hat{R}_j^T \tilde{A}_j^{-1} \hat{R}_j \hat{\mathbf{f}}. \tag{4.13}$$

Notice that no Lagrange multipliers were introduced to find the value of the normal derivative at corners as in [15]. Coerciveness must be ensured for each local problem. The coefficient $p = O(\sqrt{\eta})$, in the case of a zeroth-order Taylor approximation, can be too small for spectral elements. For a zeroth-order optimized p, this seems not to be a

problem. This was also observed in the case of locally discontinuous Galerkin approximations where good bounds on the penalty coefficient p are available for triangles of degree up to 8 (see [24], for instance).

In order to test the spectral element optimized Schwarz algorithm, the positive definite Helmholtz problem is solved with the C^∞ right-hand side

$$f(x,y) = (1 + 5\pi^2/4)\sin(\pi x + \pi/4)\sin(\pi y/2 + \pi/4). \tag{4.14}$$

The domain Ω is the rectangle $[0,2] \times [0,4]$. The solution is depicted in the top panel of Figure 4.4. In the case of GMRES, in the most favorable situation, one should expect the number of iterations to be proportional to the square root of the condition number of the preconditioned spectral element matrix $\kappa(M^{-1}A)$. The bottom panel of Figure 4.4 depicts the number of GMRES iterations with respect to the polynomial degree N. In all tests, a starting vector with random entries between $[0,1]$ was employed, which guarantees that all frequencies are contained in the error during the iteration, and is important when verifying asymptotics; see [9]. For a classical Schwarz preconditioner applied to the augmented system, the asymptotic behavior in the number of iterations reveals the expected growth in the condition number, namely, $\kappa(M_{AS}^{-1}A) = O(N^2)$; see, for instance, [31]. For the zeroth-order optimized Robin transmission condition, the convergence rate is $\kappa(M_{OO0}^{-1}A) = O(N)$, which is an order N better than that of the classical Schwarz or a substructuring method. The most dramatic gains are obtained when the second-order tangential derivatives are added to the transmission operator. A very slow growth in the condition number is observed: $\kappa(M_{OO2}^{-1}A) = O(\sqrt{N})$. This result is in agreement with the theory developed in [18]. In addition to the change in the asymptotic behavior, the number of iterations also decreases for a fixed polynomial degree and an increasingly sophisticated transmission operator.

The behavior of the algorithm with respect to the number of spectral elements, which equals the number of subdomains, is considered next. In the top panel of Figure 4.5, if the parameter η is kept fixed, then the number of iterations behaves like $O(H^{3/4})$, which is well within the limits established in [18]. However, if the parameter η is changed following the rule $O(JN^4)$, where J is the number of spectral elements in two dimensions and N is the polynomial degree, then the number of iterations is independent of both H and N (see bottom panel of Figure 4.5). This was observed by Cai in [2] for the classical Schwarz method and by Qin and Xu in [18] for optimized Schwarz methods in the general case.

The method was also tested in an atmospheric general circulation model (GCM). At each semi-implicit Eulerian time step, a positive definite Helmholtz problem must be solved (see [20]). The computational domain is the projection of a cube onto the surface of the sphere where the primitive equations are solved in a thin layer above the surface. Vertical acceleration is ignored due to the hydrostatic assumption. Details of the semi-implicit time discretization with spectral elements can be found in Thomas and Loft [30]. The vertical finite difference discretization results in a set of uncoupled positive definite Helmholtz problems where the parameter η in (4.1) depends on the vertical structure of the atmosphere. It should be noted that the problem stiffness increases with height above the surface due to the increasing linear gravity wave speeds. Here, the test proposed by Held and Suarez [13] is considered where an analytical solution is not available. In the experimental setup, a diabatic heating is prescribed together with a frictional forcing. This gives rise to climate simulations similar to those obtained with more complex physics; see Figure 4.6. The domain is partitioned into 6144 spectral elements, and each one of these elements is considered as

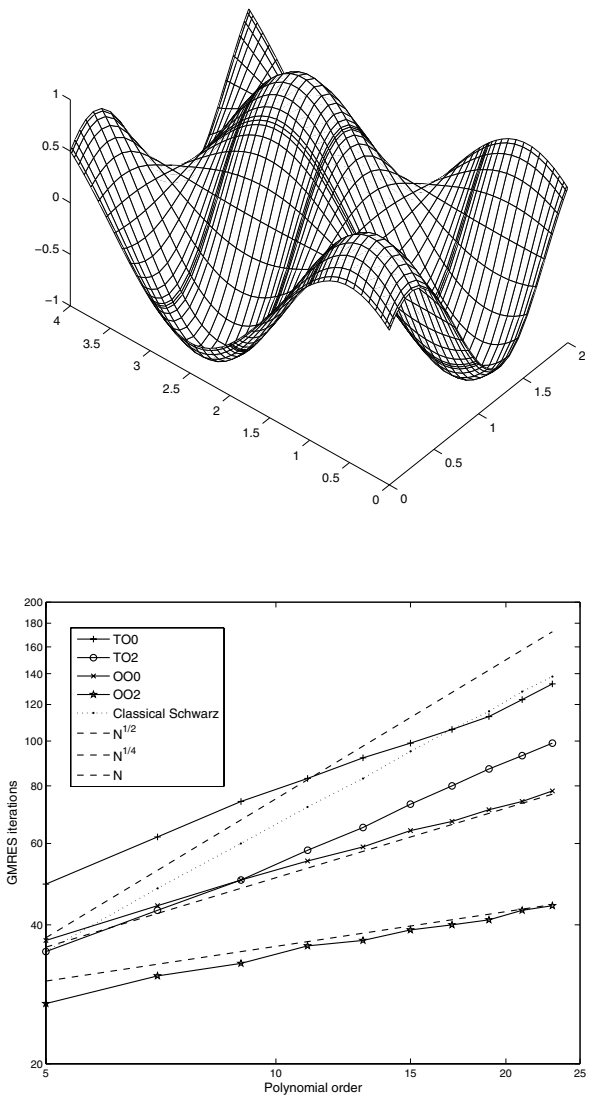


FIG. 4.4. *Top: spectral elements method (SEM) solution to the positive definite Helmholtz problem on a biperiodic domain for a C^∞ right-hand side, polynomial order* 15 *and* 8 *spectral elements. Bottom: number of GMRES iterations for different choices of preconditioning and polynomial degree for the augmented SEM matrix. The expected asymptotic behavior of each preconditioner is also displayed.*

a subdomain in the optimized Schwarz algorithm. A fifth-order polynomial expansion is employed on each element. Parallel performance results obtained on the IBM Blue Gene machine are presented in Figure 4.7 and are shown to be competitive with the explicit time-stepping approach in terms of scaling. For integration rates of the model for various computer architectures, see [4] and [26]. The number of operations, or computational cost, in two dimensions, required to perform a preconditioned matrix vector multiplication behaves asymptotically like $O(JN^3)$ as compared to $O(JN^4)$

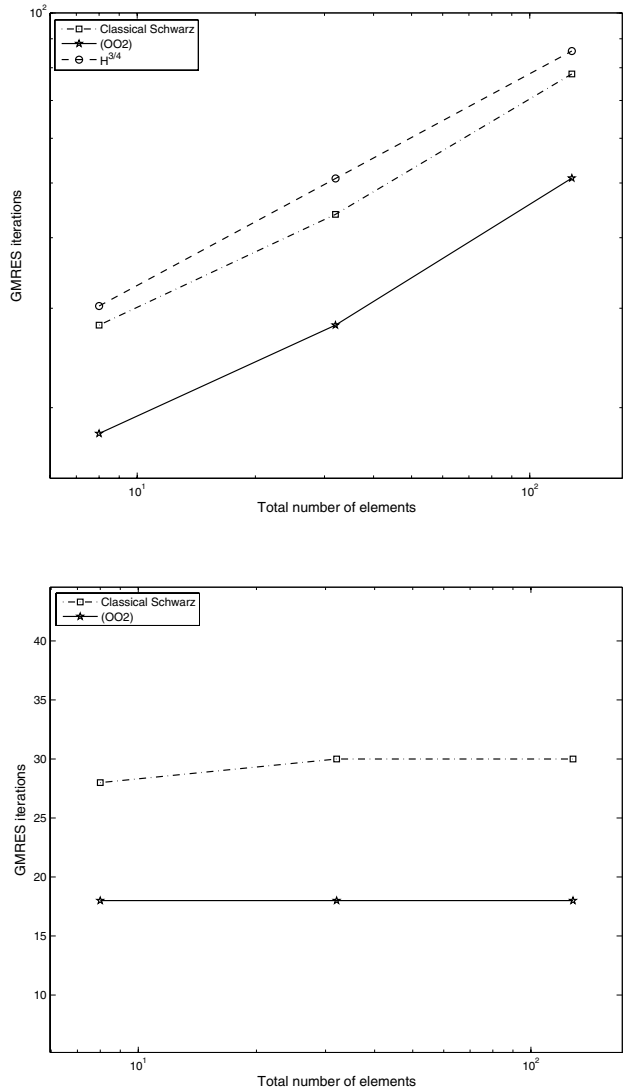


FIG. 4.5. *Top and bottom: number of GMRES iterations for classical Schwarz and optimized Schwarz (OO2) using polynomial degree $N = 5$ and a varying number of elements and no coarse solver. Bottom only: the formula used for η is $\frac{J}{4}(\frac{N}{4})^4$, where J is the number of elements and N the polynomial degree.*

when a fast diagonalization method cannot be employed. The speedup of the explicit time stepping was scaled with respect to the semi-implicit one. For an overview of the machine, its two modes (coprocessor and virtual node), and techniques for optimizing the communication patterns, consult [1]. Note that the augmented system formulation enables one to store only $O(N^3)$ matrix entries per spectral element for application of the inverse instead of the $O(N^4)$ entries required in the case of an LU factorization. Since the L2 cache available on the Blue Gene machine is rather small (2KB), the savings are advantageous. No coarse solver is needed in the GCM.

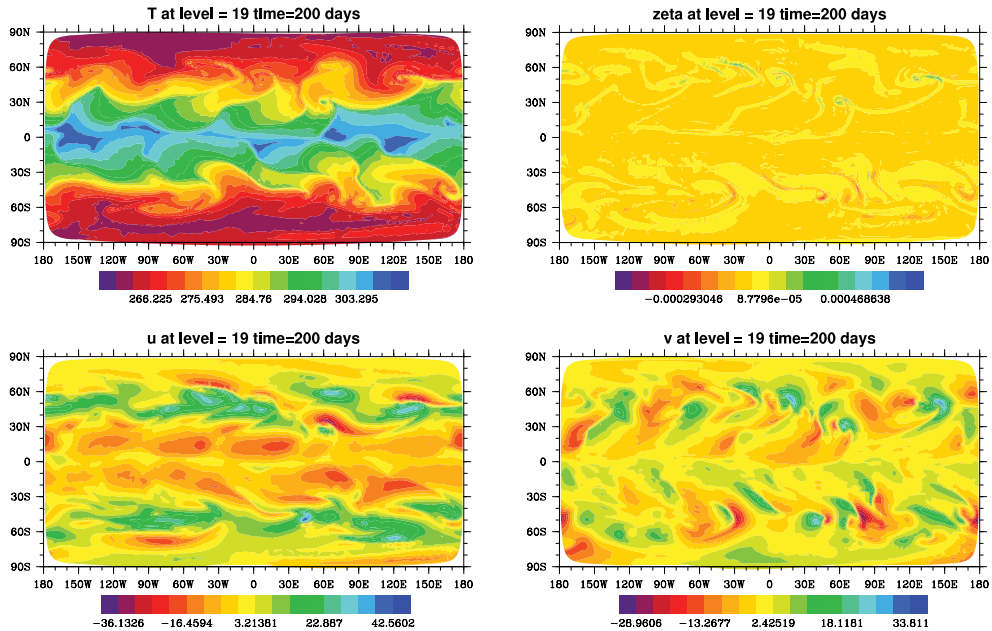


FIG. 4.6. *Held–Suarez test, temperature field, vorticity and velocities (latitudinal and longitudinal) at the surface of the planet after* 200 *days of simulation obtained with the spectral element general circulation model (GCM).*

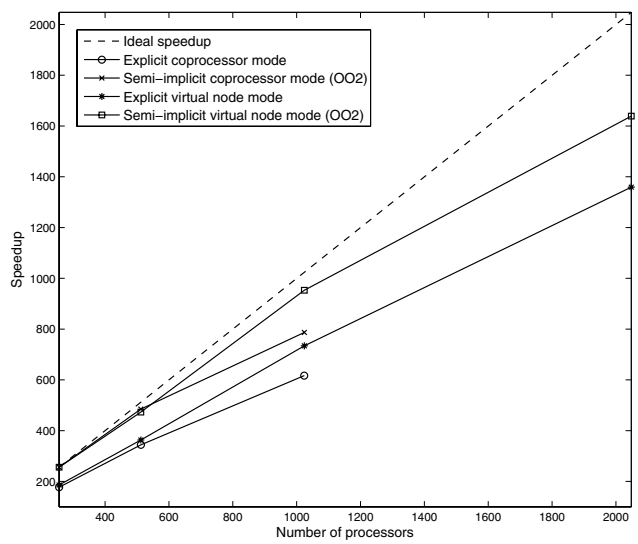


FIG. 4.7. *Speedup for the GCM on a single rack of the IBM Blue Gene machine for* 6144 *spectral elements,* 20 *vertical levels, and polynomial order* 5.

This is a consequence of the semi-implicit time discretization. The model time step is restricted by the advective Courant–Friedrichs–Levy condition which scales the parameter $\eta = \eta(\Delta t)$ as $O(JN^4)$. On massively parallel computers such as the Blue Gene machine, avoiding the coarse solve is of utmost importance. A quantification of how a coarse solver affects parallel performance on such machines can be found in [32].

5. Conclusion. Our presentation showed how the one-level RAS and MS algorithms can be modified to increase their respective convergence rates in the context of multiple domains. For the optimized restricted additive Schwarz algorithm, a condition on the overlap is required to produce optimized iterates. It was also shown how an augmented system can be constructed to generate optimized iterates for domains with nonoverlapping regions. Theoretical results were obtained at the algebraic level and are independent of the underlying spatial discretization. Finite difference and high-order spectral element examples document the simplicity and significant convergence acceleration. The augmented system is appropriate for high-order methods, where the construction of an overlapping grid is cumbersome and technically challenging. For zeroth-order optimized transmission conditions, the condition number of the preconditioned spectral element matrix is asymptotically N times better than the condition number resulting from a classical Schwarz method or a substructuring method with similar computational cost. For a second-order optimized transmission operator, the behavior with respect to the polynomial order confirms the asymptotic results found in [9, 18]. When the stiffness induced by η in the model equation is modified following the Eulerian semi-implicit GCM advective time-step restriction, all block preconditioners are optimal with respect to H refinement, thus alleviating the use of a coarse solver. Regardless, the second-order optimized Schwarz approach yields the smallest number of iterations. Application of the method to a spectral element climate model results in a significant reduction of the computing time, demonstrating the practicality of the theoretical results. It is conjectured that more realistic subgrid scale physical forcing will result in a much broader range of unresolved frequencies, in which case the optimized algorithm will be the method of choice.

REFERENCES

[1] G. Bhanot, A. Gara, P. Heildelberger, E. Lawless, J. C. Sexton, and R. Walkup, *Optimizing task layout on the Blue Gene/L supercomputer*, IBM J. Res. Dev., 49 (2005), pp. 489–500.

[2] X.-C. Cai, *Additive Schwarz algorithms for parabolic convection-diffusion equations*, Numer. Math., 60 (1991), pp. 41–61.

[3] X.-C. Cai and M. Sarkis, *A restricted additive Schwarz preconditioner for general sparse linear systems*, SIAM J. Sci. Comput., 21 (1999), pp. 792–797.

[4] J. Dennis, A. Fournier, W. F. Spotz, A. St-Cyr, M. A. Taylor, S. J. Thomas, and H. Tufo, *High-resolution mesh convergence properties and parallel efficiency of a spectral element atmospheric dynamical core*, Int. J. High Perform. Comput. Appl., 19 (2005), pp. 225–235.

[5] M. Dryja and O. B. Widlund, *An Additive Variant of the Schwarz Alternating Method for the Case of Many Subregions*, Tech. report 339, Department of Computer Science, Courant Institute, New York University, NY, 1987.

[6] O. Dubois, *Optimized Schwarz methods with Robin conditions for the advection-diffusion equation*, in Domain Decomposition Methods in Science and Engineering XVI, Lect. Notes Comput. Sci. Eng. 55, D. E. Keyes and O. B. Widlund, eds., Springer-Verlag, Berlin, Heidelberg, 2007, pp. 181–188.

[7] E. Efstathiou and M. J. Gander, *Why restricted additive Schwarz converges faster than additive Schwarz*, BIT, 43 (2003), pp. 945–959.

[8] P. F. Fischer, *An overlapping Schwarz method for spectral element solution of the incompressible Navier-Stokes equations*, J. Comput. Phys., 133 (1997), pp. 84–101.
[9] M. J. Gander, *Optimized Schwarz methods*, SIAM J. Numer. Anal., 44 (2006), pp. 699–731.
[10] M. J. Gander, L. Halpern, and F. Magoulès, *An optimized Schwarz method with two-sided Robin transmission conditions for the Helmholtz equation*, Internat. J. Numer. Methods Fluids, 55 (2007), pp. 163–175.
[11] M. J. Gander, L. Halpern, F. Magoulès, and F.-X. Roux, *Patch substructuring methods*, C. R. Math. Acad. Sci. Paris, submitted.
[12] M. J. Gander, F. Magoulès, and F. Nataf, *Optimized Schwarz methods without overlap for the Helmholtz equation*, SIAM J. Sci. Comput., 24 (2002), pp. 38–60.
[13] I. H. Held and M. J. Suarez, *A proposal for the intercomparison of the dynamical cores of atmospheric general circulation models*, Bull. Amer. Math. Soc., 75 (1994), pp. 1825–1830.
[14] C. Japhet, F. Nataf, and F. Rogier, *The optimized order 2 method. Application to convection-diffusion problems*, Future Generation Computer Systems, 18 (2001), pp. 17–30.
[15] J.-H. Kimn, *A convergence theory for an overlapping Schwarz algorithm using discontinuous iterates*, Numer. Math., 100 (2005), pp. 117–139.
[16] P.-L. Lions, *On the Schwarz alternating method.* I, in First International Symposium on Domain Decomposition Methods for Partial Differential Equations, R. Glowinski, G. H. Golub, G. A. Meurant, and J. Périaux, eds., SIAM, Philadelphia, 1988, pp. 1–42.
[17] F. Magoulès, F.-X. Roux, and S. Salmon, *Optimal discrete transmission conditions for a nonoverlapping domain decomposition method for the Helmholtz equation*, SIAM J. Sci. Comput., 25 (2004), pp. 1497–1515.
[18] L. Qin and X. Xu, *On a parallel Robin-type nonoverlapping domain decomposition method*, SIAM J. Numer. Anal., 44 (2006), pp. 2539–2558.
[19] A. Quarteroni and A. Valli, *Numerical Approximations of Partial Differential Equations*, 2nd ed., Springer Ser. Comput. Math. 23, Springer-Verlag, Berlin, 1997.
[20] A. J. Robert, *A semi-Lagrangian and semi-implicit numerical integration scheme for the primitive meteorological equations*, J. Math. Soc. Japan, 60 (1982), pp. 319–325.
[21] E. Ronquist, *Optimal Spectral Element Methods for the Unsteady Three-Dimensional Incompressible Navier-Stokes Equations*, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA, 1988.
[22] F.-X. Roux, F. Magoules, S. Salmon, and L. Series, *Optimization of interface operator based on algebraic approach*, in Proceedings of the 14th International Conference on Domain Decomposition Methods, Cocoyoc, Mexico, I. Herrera, D. E. Keyes, O. B. Widlund, and R. Yates, eds., 2002.
[23] H. A. Schwarz, *Über einen Grenzübergang durch alternierendes Verfahren*, Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich, 15 (1870), pp. 272–286.
[24] K. Shahbazi, *An explicit expression for the penalty parameter of the interior penalty method*, J. Comput. Phys., 205 (2005), pp. 401–407.
[25] B. F. Smith, P. E. Bjørstad, and W. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, Cambridge, MA, 1996.
[26] A. St-Cyr, *Optimized Schwarz preconditioning for a semi-implicit spectral elements GCM*, in preparation.
[27] A. St-Cyr, J. M. Dennis, R. Loft, S. Thomas, H. Tufo, and T. Voran, *Early experiences with the 360TF IBM BlueGene/L platform*, Int. J. Comput. Methods, accepted.
[28] A. St-Cyr, M. J. Gander, and S. J. Thomas, *Optimized restricted additive Schwarz methods*, in Domain Decomposition Methods in Science and Engineering XVI (NewYork, 2007), Lect. Notes Comput. Sci. Eng. 55, O. B. Widlund and D. E. Keyes, eds., Springer-Verlag, Berlin, Heidelberg, 2007, pp. 213–220.
[29] W. P. Tang, *Generalized Schwarz splittings*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 573–595.
[30] S. J. Thomas and R. D. Loft, *Semi-implicit spectral element atmospheric model*, J. Sci. Comput., 17 (2002), pp. 339–350.
[31] A. Toselli and O. Widlund, *Domain Decomposition Methods—Algorithms and Theory*, Springer Ser. Comput. Math. 34, Springer-Verlag, Berlin, 2005.
[32] H. M. Tufo and P. F. Fischer, *Fast parallel direct solvers for coarse grid problems*, J. Parallel Distrib. Comput., 61 (2001), pp. 151–177.
[33] P. Wilders and E. Brakkee, *Schwarz and Schur: An algebraical note on equivalence properties*, SIAM J. Sci. Comput., 20 (1999), pp. 2297–2303.

SIAM J. SCI. COMPUT.
Vol. 29, No. 6, pp. 2426–2441
© 2007 Society for Industrial and Applied Mathematics

ON RESTART AND ERROR ESTIMATION FOR KRYLOV APPROXIMATION OF $w = f(A)v$*

HILLEL TAL-EZER†

Abstract. Krylov algorithms are widely used for the solution of large scale linear systems. A Krylov approach is implemented also for the approximation of the vector w which results from operating with a function of a matrix on a vector v. For example, when solving a set of linear ODEs we have $w = f(A)v$, where $f(z) = \exp(tz)$. The main drawback of the Krylov algorithm lies in the need to store all the vectors spanning the approximation space. When solving linear systems, it is possible to overcome this drawback by restarting. In this paper we show how to apply restarts in the general case of approximating $w = f(A)v$. This new algorithm allows a much more efficient approximation of the exponential operator than the standard Krylov algorithm, and it is especially useful in the case of functions which cannot be factored into a product of functions. Yet another interesting subject is the error estimate. The scheme given here provides error estimates "for free," a trait which enables us to stop the algorithm whenever the desired accuracy is achieved.

Key words. function of a matrix, matrix exponential, Krylov methods, restarted Krylov, initial value problem

AMS subject classifications. 65F10, 65F99, 65M20

DOI. 10.1137/040617868

1. Introduction.

Let us assume that A is a square matrix of a large dimension N. The need to approximate a vector w which satisfies

$$w = f(A)v \tag{1.1}$$

may be encountered in diverse ways. The solution of ODEs, for example, entails $U = f(A)U^0$, where $f(z) = \exp(zt)$ in the homogeneous case, and where $f(z) = (\exp(zt) - 1)/z$ in the nonhomogeneous case. In the case of a one-way wave equation, for example, we have $U = f(A)U^0$, while $f(z) = \exp(\alpha\sqrt{z})$. Even the basic problem of solving the linear system $Ax = b$ stems from this family, since the solution vector is $x = f(A)b$, whereas $f(z) = z^{-1}$. Extensive research has been carried out in this field. There are two main approaches:

1. Algorithms based on a polynomial approximation of $f(z)$ in domain D of the complex plane, which includes all the eigenvalues of A [1], [8], [9], [21], [22];

2. Algorithms based on the projection of the problem onto the Krylov space: $Kr = \mathrm{Span}\{v, Av, \ldots, A^{m-1}v\}$ [3], [7], [14], [25].

Each of these two approaches has both advantages and disadvantages. In fact, one may even say that the advantage of one is the disadvantage of the other. The main benefits of the Krylov algorithm are as follows: It does not require knowledge on the domain of eigenvalues, and the algorithm is capable of adjusting itself to the information contained in the vector v. These two qualities combine to make the algorithm highly efficient and robust. On the other hand, its main drawback lies in the need to store all the vectors spanning the Krylov space. When dealing with a large domain of eigenvalues and/or when $f(z)$ is characterized by large gradients, the Krylov space will be of considerable size as well, so that we might not have sufficient memory

*Received by the editors October 28, 2004; accepted for publication (in revised form) March 9, 2007; published electronically October 17, 2007.
http://www.siam.org/journals/sisc/29-6/61786.html
†Computer Sciences, Academic College of Tel-Aviv Yaffo, Tel Aviv, Israel (hillel@mta.ac.il).

to store the basis vectors. Yet another drawback typical of a large Krylov space lies in the fact that the number of scalar products is proportional to m^2, where m is the dimension of this particular space. In the case of a large m, the CPU time needed to compute the scalar products might be significant. When applying the Krylov algorithm to linear systems $Ax = b$, it is still possible to implement Krylov space of small dimension through the use of restarts. The application of a Krylov algorithm of dimension m leads to the approximated solution vector x_m. If the solution does not meet the required accuracy, we may employ yet another Krylov algorithm, solving $A(x - x_m) = b - b_m$. Using restart for the approximation of $w = f(A)$ is not as straightforward a process as in the case of linear systems. This general problem is dealt with in this paper through the development of a restarted Krylov algorithm.

Another important issue relates to the need to find a good and yet relatively cheap error estimate, which will enable us to stop the algorithm when the desired accuracy is reached. This issue will be addressed too.

The paper is organized as follows. Section 2 provides a short description of the Krylov algorithm. In [14] it is shown that the Krylov approach can be regarded as an interpolating polynomial approach, whereby the interpolating points are eigenvalues of a small matrix H_m, resulting from the restriction of A on the Krylov space. Based on this observation, section 3 includes an algorithm for the approximation of $w = f(A)v$ through the use of restarts. Section 4 provides a simple expression of the error estimate. A byproduct of the research described in this paper is a modified algorithm applicable to the standard, nonrestarted algorithm. In [14] there is a corrected scheme which enlarges the approximation subspace by using the vector v_{m+1} (produced by the Krylov algorithm). Unlike the standard case, it allows the solution in the Krylov space of dimension $m + 1$, rather than of m, without necessitating any additional matrix-vector multiplication. It is shown that the resulting w_{m+1} satisfies $w_{m+1} = P_m(A)v$, where $P_m(z)$ interpolates $\exp(z)$ at $\{\lambda_1, \ldots, \lambda_m, 0\}$ while $\lambda_1, \ldots, \lambda_m$ are the eigenvalues of H_m. This method is in turn generalized to allow the development of a modified algorithm which results in $w_{m+1} = P_m(A)v$, where $P_m(z)$ interpolates $f(z)$ at $\{\lambda_1, \ldots, \lambda_m, \lambda\}$ and λ is subsequently chosen optimally. The modified algorithm is described in section 5. We conclude the paper with the numerical results given in section 6.

The restarted algorithm presented in this paper is based on the Arnoldi approach. As in the case of linear systems, one can also develop an algorithm based on the Lanczos approach. The main advantage of the Lanczos approach lies in the fact that there is absolutely no need to restart. On the other hand, in this case one has to be aware of the hazard of losing orthogonality. Yet another disadvantage stems from the need to operate with the transpose of the operator unless some modifications are done as in the transpose-free algorithms. These issues will be addressed in a future paper.

2. Krylov algorithm. We want to approximate w, where

$$w = f(A)v. \tag{2.1}$$

First we build an orthonormal basis of the Krylov space

$$K_m = \mathrm{Span}\{v, Av, \ldots, A^{m-1}v\} \tag{2.2}$$

by applying modified Gram–Schmidt orthogonalization (Arnoldi).

ARNOLDI ALGORITHM.

$\beta = ||v||_2,\ v_1 = \frac{1}{\beta}v.$
for $j = 1 : m$
  $v_{j+1} = Av_j$
  for $i = 1 : j$
    $h_{i,j} = v_i^H v_{j+1}$
    $v_{j+1} = v_{j+1} - h_{i,j}v_i$
  end
  $h_{j+1,j} = ||v_{j+1}||_2$
  if $h_{j+1,j} == 0$
    $m = j$ break
  end
  $v_{j+1} = \frac{1}{h_{j+1,j}}v_{j+1}$
end

If V is an $N \times m$ matrix whose columns are $v_1, \ldots, v_m$ (the orthonormal basis vectors of K_m), we have

$$AV = VH_m + h_{m+1,m}v_{m+1}e_m^H. \tag{2.3}$$

Hence

$$H_m = V^H AV. \tag{2.4}$$

H_m is actually the matrix restriction of A on K_m. Using a Krylov method means that we project the original problem onto the subspace K_m and solve it exactly there. Since $v = \beta V e_1$, we have

$$V^H w = V^H f(A)v = \beta V^H f(A)Ve_1 \approx \beta f(H_m)e_1. \tag{2.5}$$

Defining w_m as

$$w_m = \beta V f(H_m)e_1, \tag{2.6}$$

we get that

$$w_m \approx VV^H w, \tag{2.7}$$

where $VV^H w$ is the projection of w on K_m.

Since m is much smaller than N (the dimension of A), the CPU time needed to compute $f(H_m)e_1$ is negligible. A possible stable algorithm for computing $y = f(H_m)e_1$ can be based on the identity $f(H_m) = P_{m-1}(H_m)$, where the polynomial

$$P_{m-1}(z) = a_1 + a_2(z - \lambda_1) + \cdots + a_m(z - \lambda_1)\cdots(z - \lambda_{m-1}) \tag{2.8}$$

interpolates $f(z)$ at $\lambda_1, \ldots, \lambda_m$, the eigenvalues of H_m. Using Horner's scheme, we get

$y = a_m e_1$
for $i = m - 1 : -1 : 1$
  $y = a_i + (H_m - \lambda_i I)\, y$
end

(If m is large, it is recommended that one stabilize the algorithm by using Leja ordering of the interpolating points [13].)

Here is the Krylov algorithm for approximating $w = f(A)v$ in a space of dimension m.

KRYLOV ALGORITHM.

$\beta = ||v||_2,\ v_1 = \frac{1}{\beta}v.$
for $j = 1 : m$
  $v_{j+1} = Av_j$
  for $i = 1 : j$
    $h_{i,j} = v_i^H v_{j+1}$
    $v_{j+1} = v_{j+1} - h_{i,j}v_i$
  end
  $h_{j+1,j} = ||v_{j+1}||_2$
  if $h_{j+1,j} == 0$
    $m = j$ break
  end
  $v_{j+1} = \frac{1}{h_{j+1,j}}v_{j+1}$
end
$H_m = h(1 : m, 1 : m),\ y = \beta f(H_m)e_1$
$w_m = V(:, 1 : m)y$

Saad has proved in [14] (for $f(z) = \exp(z)$) that w_m satisfies

$$w_m = P_{m-1}(A)v, \tag{2.9}$$

where $P_{m-1}(z)$ is the unique polynomial of degree $m-1$ which interpolates $f(z)$ at the eigenvalues of H_m (if an eigenvalue λ has multiplicity k, then $P_{m-1}^{(i)}(\lambda) = f^{(i)}(\lambda),\ \ i = 0, \ldots, k-1$). The theorem and proof hold for any function $f(z)$ analytic in an open set which contains the spectrum of A and H_m. Based on the equivalence of Krylov and polynomial algorithms, we have developed a restarted algorithm as well as an easily computed error estimate, both of which are described in sections 3 and 4, respectively.

3. Restart. As mentioned above, when the domain of eigenvalues is large and/or whenever $f(z)$ has large gradients, the size of the Krylov space required for a good approximation will also be considerable, so that at times we might not have enough memory to store the basis vectors. In addition, the number of scalar products which is proportional to the square of the dimension of the Krylov domain can result in significant CPU times. We shall now describe a restarted algorithm, which allows a cyclical application of the basic Krylov algorithm. Each cycle (restart) uses a Krylov space of dimension m.

Let us assume that $p_{m-1}(z)$ is the polynomial which interpolates $f(z)$ at $z_1, \ldots, z_m$. The use of Newton representation leads to the following expression of the error:

$$E(z) = f(z) - p_{m-1}(z) = f[z_1, \ldots, z_m, z]r_m(z), \tag{3.1}$$

where $f[z_1, \ldots, z_m, z]$ is the divided differences whose recursive definition is

$f[z_1, z] = \frac{f(z)-f(z_1)}{z-z_1}$
for $k = 1 : m - 1$
  $f[z_1, \ldots, z_{k+1}, z] = \frac{f[z_1,\ldots,z_k,z]-f[z_1,\ldots,z_{k+1}]}{z-z_{k+1}}$
end

and $r_m(z) = \prod_{i=1}^{m}(z - z_i)$.

Hence, approximating w by w_m results in

$$e_m = w - w_m = f(A)v - p_{m-1}(A)v = g(A)r, \tag{3.2}$$

where

$$g(z) = f[z_1, \ldots, z_m, z] \tag{3.3}$$

and

$$r = r_m(A)v = \prod_{i=1}^{m}(A - z_i I)\, v. \tag{3.4}$$

In the appendix we prove that in the case of an analytic function $f(z)$, $g(z)$ is analytic too. When the interpolation points are the eigenvalues of H_m, it is known (see, for example, Exercise 6.8 on p. 216 of [16]) that the vector r given above is parallel to v_{m+1} and satisfies

$$r = \alpha v_{m+1}, \tag{3.5}$$

where

$$\alpha = ||v||_2 h_{m+1,m} q_m \tag{3.6}$$

and q_m is the last element of the first column of $r_m(H_m)$. Hence, (3.2) means that

$$w = w_m + g(A)r, \tag{3.7}$$

where

$$g(z) = f[\lambda_1, \ldots, \lambda_m, z] \tag{3.8}$$

and

$$r = \alpha v_{m+1}. \tag{3.9}$$

Restarted Krylov is based on the concept that if the result is not accurate enough (approximating the accuracy is described in the next section), we continue by applying the Krylov algorithm to allow the approximation of $g(A)r$. Restarting the Krylov algorithm with a Krylov space of dimension m means that each restart is marked by a different function $g(z)$. The functions can be defined recursively as

$$g_1(z) = f(z), \tag{3.10}$$

$$g_{k+1}(z) = g_k[\lambda_1^k, \ldots, \lambda_m^k, z], \qquad k \geq 1, \tag{3.11}$$

where $f(z)$ is the original function and $\lambda_1^k, \ldots, \lambda_m^k$ are the eigenvalues of H_m^k (the matrix restriction of A on the Krylov space at the kth restart). Based on Lemma 3 given in the appendix, $g_k(z)$ is an analytic function.

Below is a recursive MATLAB routine which computes $y = g_k(z)$. Here zp is a two dimensional matrix whose kth column holds the m eigenvalues of H_m at the

$(k-1)$th restart. Similarly, wp is a two dimensional matrix whose kth column holds the values of g_k at the kth column of zp.

```
function y = g(z, k)
global zp, wp
if k == 0
    y = f(z)
    return
end
zp_0 = zp(:, k)
wp_0 = wp(:, k)
m = length(zp_0)
zp_0(m + 1) = z
wp_0(m + 1) = g(z, k − 1)
for j = m : −1 : 1
    wp_0(j, 1) = (wp_0(j + 1) − wp_0(j))/(zp_0(m + 1) − zp_0(j))
end
y = wp_0(1)
```

Observe that, computing $g_k(z)$, we do one computation of $g_{k-1}(z)$ and m divisions. Hence, by induction on k, we get that computing the value of $g_k(z)$ at one interpolating point involves one computation of $f(z)$ and $k \times m$ divisions. Thus, for practical problems where $m \ll N$, the CPU time devoted to this part of the computation is insignificant.

For a given tolerance (tol), maximal Krylov dimension (m), and maximal number of restarts (maxrstrt), the formal restarted algorithm for approximating $w = f(A)v$ is the following.

RESTARTED KRYLOV ALGORITHM.

```
r_1 = v, w = 0, g_0 = f
for k = 1 : maxrstrt
    [u_{k+1}, r_{k+1}, flag] = Krylov (r_k, m, g_{k−1}, tol)
    w = w + u_{k+1}
    if flag == 1
        break
    end
    tol = tol||w||_2
    g_k(z) =def g_{k−1}[λ_1^{k−1}, ..., λ_m^{k−1}, z]
end
```

where r_{k+1} computed by the Krylov algorithm satisfies

$$r_{k+1} = ||r_k||_2 h^k_{m+1,m} q^k_m v^k_{m+1} \tag{3.12}$$

and $flag$ is a parameter which accepts the value 1 when the Krylov algorithm has achieved the desired accuracy. (A Krylov algorithm with error estimate is described in the next section.)

The formal solution of linear system $Ax = b$ is $x = f(A)b$, where $f(z) = z^{-1}$. It can be shown that the application of the restarted Krylov algorithm described above to this function is equivalent to the FOM (full orthogonal method) algorithm [15]. It is well known that restarted FOM, similar to restarted GMRES [17], can exhibit slow

convergence or complete stagnation. This problem (mainly for the GMRES case) was addressed by many researchers in an attempt to provide theoretical explanations and to come up with valid modifications which would allow this hurdle to be overcome. Morgan [11] suggested modifying the restarted algorithm through the deflation of small eigenvalues, using the thick restarting technique developed by Wu and Simon [27]. In another paper [10] Morgan suggests retaining information during the move from one restart to another, by using the implicit Arnoldi methodology developed by Sorensen [20]. Embree [5] provides detailed analysis of the stagnation phenomenon and disproves the common belief that a larger Krylov space leads to faster convergence. This, in turn, gives rise to a very important issue, namely, what is the optimal size of a Krylov space. Zhang and Nodera address this subject in [28]. The source of the stagnation phenomenon can be traced to the singularity of the function $f(z) = z^{-1}$ at $z = 0$. If, for small $\varepsilon > 0$, the ε-pseudospectrum [2] of A contains the singular point, then stagnation is quite probable. Consequently, similar difficulties might arise following the application of the new algorithm to functions having singularities. The concepts discussed in the above-mentioned papers might be useful in these cases. On the other hand, when $f(z)$ is an analytic function, we would not face stagnation.

A widely known example of using the polynomial approximation of a function of a matrix is the numerical solution of a system of ODEs,

$$U_t = GU, \qquad U(t=0) = U^0. \tag{3.13}$$

The formal solution at $t = T$ is

$$U(T) = f(G)U^0, \tag{3.14}$$

where $f(z) = e^{zT}$. The implementation of a Krylov approach for the purpose of approximating the matrix exponential is described and analyzed in many papers, e.g., [6], [7], [14], [19], [23], [24]. In this case $f(z)$ is an analytic function. The approximation of $U(T)$ for large T through a Krylov algorithm which is marked by a restricted size m means that we have to split the time domain into time steps of size Δt. We get

$$U(t + \Delta t) = f(G)U(t), \tag{3.15}$$

where $f(z) = e^{z\Delta t}$. By choosing a Δt which is small enough, the domain of eigenvalues of the matrix $\Delta t G$ remains small, and the Krylov algorithm which utilizes space of small size m can be accurate. The parameters m and Δt control this accuracy. Usually, m is dictated by memory limitations, and Δt is chosen in accordance with the accuracy requirements.

When using the algorithm described in this paper for the approximation of the matrix exponential operation, we do not split the time domain into time steps. The following lemma shows that the implementation of the new algorithm is equivalent to the approximation of the evolution operator by one large degree polynomial which interpolates the exponential function at the union of points produced by the algorithm.

LEMMA 1. *Interpolating $f(z)$ at $z_1^{(1)}, \ldots, z_m^{(1)}$, we have*

$$f(z) = p_{m-1}^{(1)}(z) + g_1(z)(z - z_1^{(1)}) \ldots (z - z_m^{(1)}), \tag{3.16}$$

where

$$g_1(z) = f[z_1^{(1)}, \ldots z_m^{(1)}, z]. \tag{3.17}$$

Interpolating $g_1(z)$ *at* $z_1^{(2)}, \ldots, z_k^{(2)}$, *we have*

$$g_1(z) = p_{k-1}^{(2)}(z) + g_2(z)(z - z_1^{(2)}) \ldots (z - z_k^{(2),}), \tag{3.18}$$

where

$$g_2(z) = g_1[z_1^{(2)}, \ldots z_k^{(2)}, z]. \tag{3.19}$$

Let $w = f(A)v$. *Approximating* w *by* $w_m^{(1)} + w_k^{(2)}$, *where*

$$w_m^{(1)} = p_{m-1}^{(1)}(A)v, \qquad w_k^{(2)} = p_{k-1}^{(2)}(A)r_1 \tag{3.20}$$

while

$$r_1 = (A - z_1^{(1)} I) \ldots (A - z_m^{(1)} I)v, \tag{3.21}$$

is identical to approximating w *by*

$$w_{m+k} = p_{m+k-1}(A)v, \tag{3.22}$$

where the polynomial $p_{m+k-1}(z)$ *interpolates* $f(z)$ *at* $z_1^{(1)}, \ldots z_m^{(1)}, z_1^{(2)}, \ldots, z_k^{(2)}$.

Proof. We have

$$w_m^{(1)} + w_k^{(2)} = p_{m-1}^{(1)}(A)v + p_{k-1}^{(2)}(A)r_1. \tag{3.23}$$

Since $r_1 = (A - z_1^{(1)} I) \ldots (A - z_m^{(1)} I)v$ we get

$$w_m^{(1)} + w_k^{(2)} = p_{m+k-1}(A)v, \tag{3.24}$$

where $p_{m+k-1}(z) = p_{m-1}^{(1)}(z) + p_{k-1}^{(2)}(z)(z - z_1^{(1)}) \ldots (z - z_m^{(1)})$. Hence

$$p_{m+k-1}(z_i^{(1)}) = p_{m-1}^{(1)}(z_i^{(1)}) = f(z_i^{(1)}), \qquad i = 1, \ldots, m. \tag{3.25}$$

It is left to show that

$$p_{m+k-1}(z_i^{(2)}) = f(z_i^{(2)}), \qquad i = 1, \ldots, k. \tag{3.26}$$

Using (3.16), we have

$$g_1(z) = \frac{f(z) - p_{m-1}^{(1)}(z)}{(z - z_1^{(1)}) \ldots (z - z_m^{(1)})}. \tag{3.27}$$

Since $p_{k-1}^{(2)}(z)$ interpolates $g_1(z)$ at $z_1^{(2)}, \ldots, z_k^{(2)}$ we get

$$p_{k-1}^{(2)}(z_i^{(2)}) = \frac{f(z_i^{(2)}) - p_{m-1}^{(1)}(z_i^{(2)})}{(z_i^{(2)} - z_1^{(1)}) \ldots (z_i^{(2)} - z_m^{(1)})}, \qquad i = 1, \ldots, k, \tag{3.28}$$

which results in

$$p_{m-1}^{(1)}(z_i^{(2)}) + p_{k-1}^{(2)}(z_i^{(2)})(z_i^{(2)} - z_1^{(1)}) \ldots (z_i^{(2)} - z_m^{(1)}) = f(z_i^{(2)}), \tag{3.29}$$

$$i = 1, \ldots, k.$$

Hence

$$p_{m+k-1}(z_i^{(2)}) = f(z_i^{(2)}), \qquad i = 1, \ldots, k, \tag{3.30}$$

and the proof is concluded.

The above lemma shows the main difference between the restarted Krylov and the standard marching algorithm (an algorithm where the time domain is cut into time intervals and at each interval we use the Krylov algorithm to compute the solution at the next time step). At each step of the restarted Krylov the interpolating points are considered optimal (in the Arnoldi sense) with regard to the error vector. Hence, the union of points will be distributed across the domain at each step, as obligated by the Arnoldi optimization. On the other hand, when applying the Krylov algorithm at each step of the standard marching algorithm, the interpolating points are considered optimal with respect to the converged solution vector. In this case, the union of points will be more localized. Let us assume now that for each algorithm we have computed an approximation to $\exp(TA)v$ by making M matrix-vector multiplications. The application of a restarted Krylov routine means that we have approximated the evolution operator by $P_M(A)v$, where $P_M(z)$ interpolates $\exp(Tz)$ at a set of points distributed all over the domain. The use of k steps in the marching algorithm, while each time applying a polynomial of degree m, means that$P_M(z) = \Pi_{i=1}^{k} P_m^{(i)}(z)$. In this case, $P_M(z)$ interpolates $\exp(Tz)$ at a localized set of points. In general, the set of points inherent in the restarted Krylov will result in a much more accurate interpolating polynomial. Alternatively, for a given tolerance we will need a lesser number of matrix-vector multiplications. The numerical results presented in section 5 demonstrate the high efficiency of the new algorithm within the overall approximation of the exponential operator.

The approximation of the evolution operator by cutting the time domain into time steps is based on the property of the exponential function

$$\exp(Tz) = (\exp(\Delta t z))^n \qquad (T = n\Delta t). \tag{3.31}$$

However, not every function features this useful property. When the domain of eigenvalues is large and/or when $f(z)$ has large gradients, using a standard Krylov algorithm requires a Krylov space of large dimensions, which may result in an exceedingly large demand on memory. Moreover, since the number of inner products is proportional to the square of the Krylov dimension, it can lead to huge computational demands. The algorithm described in this paper is particularly useful for these kind of functions.

4. Accuracy. Approximating $w = f(A)v$ by $w_m = p_{m-1}(A)v$, we have (3.2),

$$e_m = w - w_m = g(A)r, \tag{4.1}$$

where $g(z) = f[z_1, \ldots, z_m, z]$. Since $r = ||v||_2 h_{m+1,m} q_m v_{m+1}$ we get (for normal matrix A) that the relative error satisfies

$$\frac{||w - w_m||_2}{||w_m||_2} \leq \frac{||v||_2}{||w_m||_2} h_{m+1,m} |q_m| \max_{z \in D} |g(z)|, \tag{4.2}$$

where D is the domain of eigenvalues. Since, in general, the domain D is not known, we will approximate $\max_{z \in D} |g(z)|$ by $\max_{z \in Z} |g(z)|$, where Z is a large set of points distributed across D. Let us describe now a possible way to create the set Z. When

using the Krylov algorithm we create the matrices $\{H_j\}_{j=1}^m$ which are the restrictions of the matrix A on the subspaces $V_j = \text{Span}\{v, Av, \ldots, A^{j-1}v\}$. The eigenvalues of H_j are known as Ritz values [12]. We can take Z as the union of these sets of eigenvalues.

Below is the Krylov algorithm with error estimate applied.

KRYLOV ALGORITHM + ERROR ESTIMATE.

$\beta = ||v||_2,\ v_1 = \frac{1}{\beta}v,\ Z = \{\},$ given tol, maxit
for $j = 1 : \text{maxit}$
  $v_{j+1} = Av_j$
  for $i = 1 : j$
    $h_{i,j} = v_i^H v_{j+1}$
    $v_{j+1} = v_{j+1} - h_{i,j}v_i$
  end
  $h_{j+1,j} = ||v_{j+1}||_2$
  $v_{j+1} = v_{j+1}/h_{j+1,j}$
  $H = h(1 : j, 1 : j)$
  $[U, \lambda] = \text{eig}(H)$
  $Z = Z \cup \text{diag}(\lambda)$
  $y = g_j(H)e_1$ (g_j are defined in (3.11))
  $q = (r_j(H))_{j1}$, $\bar{g} = \max_{z \in Z} |g_j(z)|$
  $e = \beta h_{j+1,j}|q|\bar{g}/||y||_2$
  if $e \leq$ tol
    $w = \beta V(:, 1 : j)y$
    break
  end
end

In the algorithm above we have to compute the vector $y = g_j(H)e_1$ and the scalar $q = r_j(H)_{j1}$. y can be computed by a Horner scheme applied to the interpolating polynomial (2.8), and for the scalar q we can do the following:

$u = [1, 0, \ldots, 0]^H$
for $i = 1 : j - 1$
  $u = Hu - \lambda_{ii}u$
end
$q = u(j)$

In the case of a large degree polynomial r_j, it might be a good idea to stabilize the algorithm described above by implementing Leja ordering [13] on the set of λ_{ii}. The error estimate given above is based on the assumption that the matrix is normal. When the matrix is close to normal, one can account for the nonnormality by changing the expression for e to

$$e = \beta h_{j+1,j}|q|\bar{g}||U||_2||U^{-1}||_2/||y||_2. \tag{4.3}$$

This error estimate is not valid for highly nonnormal matrices.

5. Modified algorithm. As a byproduct of the research on the restarted algorithm, we have developed a modified algorithm which is applicable in the nonrestarted case.

Interpolating $f(z)$ at $\lambda_1, \ldots, \lambda_m$ results in

$$f(z) = P_{m-1}(z) + f[\lambda_1, \ldots, \lambda_m, z] \prod_{j=1}^{m} (z - \lambda_j)\,. \tag{5.1}$$

Adding another interpolating point λ, we have

$$f(z) = P_{m-1}(z) + f[\lambda_1, \ldots, \lambda_m, \lambda] \prod_{j=1}^{m} (z - \lambda_j) \tag{5.2}$$

$$+ f[\lambda_1, \ldots, \lambda_m, \lambda, z]\,(z - \lambda) \prod_{j=1}^{m} (z - \lambda_j)\,.$$

Since

$$w = f(A)v, \qquad w_m = P_{m-1}(A)v, \qquad r = \prod_{j=1}^{m} (A - \lambda_j I)\, v, \tag{5.3}$$

we get

$$w = w_m + f[\lambda_1, \ldots, \lambda_m, \lambda]r + f[\lambda_1, \ldots, \lambda_m, \lambda, A]\,(A - \lambda I)\, r. \tag{5.4}$$

Hence, using (3.9), we can get an improved approximation without any additional matrix-vector operation:

$$w_{m+1} = w_m + \gamma v_{m+1}, \tag{5.5}$$

where

$$\gamma = ||v||_2 h_{m+1,m} q_m f[\lambda_1, \ldots, \lambda_m, \lambda]. \tag{5.6}$$

The idea is to choose λ in some optimal way. In the numerical examples presented in the next section we took

$$\lambda = \min_{z \in Z} \mid f[\lambda_1, \ldots, \lambda_m, z] \mid, \tag{5.7}$$

where Z is the union all the eigenvalues of the matrices $\{H_j\}_{j=1}^{m}$ created in the Krylov algorithm.

The modification described here is applicable only to a restart-free Krylov algorithm, since in the case of the restarted algorithm we employ the vector r. The gain is particularly significant when the Krylov space is of low dimension. As for the error, when estimating it through the last term in (5.4) we have to carry out yet another matrix-vector operation, and there will be no subsequent gain in the modification. Thus we have to settle for the nonmodified estimate (4.2). In general it will be a pessimistic one.

TABLE 1

Restart	Err
1	3.1038e-001
2	1.1262e-002
3	1.2727e-006
4	3.4623e-010

TABLE 2

m	Nonrestarted Krylov	Restarted Krylov
5	2015	40
6	948	42
7	546	42
8	384	40
9	288	42
10	230	40

6. Numerical results.

6.1. Restart. In the first example we used the new restarted algorithm to approximate $w = f(A)v$, where $f(z) = \sin(\cos(2z))$, $v = [1, \ldots, 1]^H$, and A is a 500×500 Toeplitz matrix whose elements are

$$A_{ij} = \begin{cases} -1, & j = i - 1, \\ 1, & 0 \le j - i \le 5. \end{cases}$$

The dimension of the Krylov space was 10. The results obtained following each restart are given in Table 1, where

$$\text{Err} = \frac{||w - w_m||_2}{||w||_2} \tag{6.1}$$

and w_m is the approximated solution.

Since we have used Krylov space of dimension 10, a total number of 40 matrix-vector multiplications was needed in order to achieve relative accuracy of $3.4623e-010$. The following example demonstrates the efficiency of the restarted algorithm when applied to the evolution operator. Approximating $w = \exp(tA)v$ by a standard Krylov algorithm, as described in [14] for example, means that we have to split the time domain into time steps. The size of the time step is dictated by the number of Krylov vectors which may be stored in memory. We selected the ORANI678 example given in Expokit [18]. In this example we solve for $U = \exp(tA)U^0$, where A is a nonsymmetric matrix of size 2529×2529 taken from an economic model of Australia. We took $U^0 = [1, 1, \ldots, 1]^H$ and $t = 5$. The results are given in Table 2, where m is the size of the Krylov space and the two columns give the total number of matrix-vector multiplications. The desired accuracy was $5.E-8$.

Observe that the total number of matrix-vector multiplications, in the restarted case, remains almost the same when we change the size of the Krylov space. This means that in each case, the union of Ritz values has a similar quality as a set of interpolating points.

TABLE 3

m	Exact	Estim
6	1.8200E-3	3.5032E-3
9	2.7980E-5	2.8993E-5
12	7.7956E-8	9.0906E-8

TABLE 4

m	Exact	Estim
6	1.3238E-3	1.5938E-3
9	3.3782E-5	4.2253E-5
12	4.9040E-7	6.0743E-7

TABLE 5

m	Error	Error (modified)
6	9.6128E-4	1.8021E-4
9	1.9555E-6	3.3465E-7
12	1.9840E-9	2.3744E-10

6.2. Error estimate.
Example 2. Let $w = f(A)v$, where $f(z) = \exp(z)\sin(z)$, $v = [1, \ldots, 1]^H$, and

$$A_{i,j} = \begin{cases} -2, & i = j, \\ 1, & |i-j| = 1, \\ 0, & \text{else.} \end{cases}$$

Let $N = 300$ be the dimension of A and m the dimension of the Krylov space. Results for different m are given in Table 3, where

$$\text{Exact} = \frac{||w - w_m||_2}{||w_m||_2}, \qquad \text{Estim} = \frac{||v||_2}{||w_m||_2} \max_{z \in Z} |g(z)| h_{m+1,m} |q_m|, \tag{6.2}$$

Z is the set of Ritz values, and $g(z) = f[\lambda_1, \ldots, \lambda_m, z]$. The next experiment is the same as the previous one except for the matrix. We have created a random normal matrix whose eigenvalues are randomly spread in the square $0 \le \mathrm{Re}(z) \le 1$, $0 \le \mathrm{Im}(z) \le 1$. The results are given in Table 4.

These two examples clearly demonstrate that, for normal matrices, the error estimate is close to the true error.

6.3. Modified algorithm. Once again we took a random normal matrix with $N = 300$. The function is $f(z) = \sin(2z)$, and the initial data $v = [1, \ldots, 1]^H$. In Table 5 we present the relative error for the original and modified schemes for different values of m (the dimension of the Krylov space).

Conclusions. In this paper we have presented a new algorithm for the polynomial approximation of a vector w which satisfies $w = f(A)v$, where $f(z)$ is an analytic function in a domain D, A is a linear operator, and v is a given vector. This problem is typical of various fields of applied mathematics. The new algorithm utilizes a restart technique similar to the one which constitutes a basic ingredient in many iterative algorithms used for solving linear systems. The numerical results presented

in the paper demonstrate the high efficiency of the algorithm in terms of CPU time and memory.

Appendix.

LEMMA 3. *Let $f(z)$ be an analytic function in D. Let $g(z)$ be the divided differences function*

$$g(z) = f[z_1, \ldots, z_n, z], \qquad z_k \in D; \tag{A.1}$$

then $g(z)$ is analytic in D.

Proof. Define

$$g^i(z) = f[z^i, z], \tag{A.2}$$

where z^i is any subset of i points from $z_1, \ldots, z_n$. We will prove the lemma by induction on i. Let us show first that $g^1(z)$ is analytic. Without loss of generality we assume that $z^1 = z_1$.

Hence

$$g^1(z) = \begin{cases} \frac{f(z)-f(z_1)}{z-z_1}, & z \neq z_1, \\ f'(z_1), & z = z_1. \end{cases}$$

Since $f(z)$ is analytic, we have

$$f(z) = \sum_{j=0}^{\infty} \frac{f^{(j)}(z_1)}{j!}(z - z_1)^j. \tag{A.3}$$

Thus

$$g^1(z) = \sum_{j=0}^{\infty} \frac{f^{(j+1)}(z_1)}{(j+1)!}(z - z_1)^j \tag{A.4}$$

and therefore it is analytic.

Now, let us show by induction that $g^{i+1}(z) = f[z_1, \ldots, z_{i+1}, z]$ is analytic. If $z_k = z_j$ for all k and j, then

$$g^{i+1}(z) = \sum_{j=0}^{\infty} \frac{f^{(i+j+1)}(z_1)}{(i+j+1)!}(z - z_1)^j, \tag{A.5}$$

which is analytic. If not, there are two points, let us say z_k and z_j, such that $z_k \neq z_j$. Upon defining

$$\begin{aligned} g_k^i(z) &= f[z_1, \ldots, z_{k-1}, z_{k+1}, \ldots, z_{i+1}, z], \\ g_j^i(z) &= f[z_1, \ldots, z_{j-1}, z_{j+1}, \ldots, z_{i+1}, z], \end{aligned} \tag{A.6}$$

we get

$$g^{i+1}(z) = \frac{g_k^i(z) - g_j^i(z)}{z_j - z_k}. \tag{A.7}$$

Hence, by induction, we get that $g^{i+1}(z)$ is an analytic function.

Acknowledgment. The author would like to thank the referees for their many helpful comments.

Note added in proof. While this manuscript was in the refereeing process, the paper [4] was published. The two papers, ours and [4], are very close and deal with the same problem. The main difference between the two is in the approaches used to construct the approximating algorithm. While the present paper makes use of Newton interpolation, the algorithm described in [4] is based on enlarging the dimension of the Arnoldi subspace while storing a fixed number of basis vectors.

REFERENCES

[1] L. Bergamaschi, M. Caliari, and M. Vianello, *Efficient computation of the exponential operator for discrete* $2d$ *advection-diffusion equations*, Numer. Linear Algebra Appl., 10 (2003), pp. 271–289.

[2] T. A. Driscoll, K.-C. Toh, and L. N. Trefethen, *From potential theory to matrix iterations in six steps*, SIAM Rev., 40 (1998), pp. 547–578.

[3] V. I. Druskin and L. A. Knizhnerman, *Krylov subspace approximation of eigenpairs and matrix functions in exact and computer arithmetics*, Numer. Linear Algebra Appl., 2 (1995), pp. 205–217.

[4] M. Eiermann and O. G. Ernst, *Restarted Krylov subspace method for the evaluation of matrix functions*, SIAM J. Numer. Anal., 44 (2006), pp. 2481–2504.

[5] M. Embree, *Restarted GMRES Dynamics*, slide set available online at http://www.caam.rice.edu/∼embree/householder.pdf.

[6] E. Gallopoulos and Y. Saad, *Efficient solution of parabolic equations by Krylov approximation methods*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 1236–1264.

[7] M. Hochbruck and Ch. Lubich, *On Krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal., 34 (1997), pp. 1911–1925.

[8] W. Huisinga, L. Pesco, and R. Kosloff, *Faber and Newton polynomial integrators for open-system density matrix propagation*, J. Chem. Phys., 110 (1999), pp. 5538–5547.

[9] I. Moret and P. Novati, *An interpolatory approximation of the matrix exponential based on Faber polynomials*, J. Comput. Appl. Math., 131 (2001), pp. 361–380.

[10] R. B. Morgan, *A restarted GMRES method augmented with eigenvectors*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 1154–1171.

[11] R. B. Morgan, *GMRES with deflated restarting*, SIAM J. Sci. Comput., 24 (2002), pp. 20–37.

[12] C. C. Paige, B. N. Parlett, and H. A. van der Vorst, *Approximate solutions and eigenvalue bounds from Krylov subspaces*, Numer. Linear Algebra Appl., 2 (1995), pp. 115–133.

[13] L. Reichel, *Newton interpolation at Leja points*, BIT, 30 (1990), pp. 332–346.

[14] Y. Saad, *Analysis of some Krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal., 29 (1992), pp. 209–228.

[15] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing, London, 1996; available online from http://www-users.cs.umn.edu/∼saad/books.html.

[16] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, Halstead Press, New York, 1992; available online from http://www-users.cs.umn.edu/∼saad/books.html.

[17] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.

[18] R. B. Sidje, *Expokit: A software package for computing matrix exponentials*, ACM Trans. Math. Software, 24 (1998), pp. 130–156.

[19] R. B. Sidje and W. J. Stewart, *A survey of methods for computing large sparse matrix exponentials arising in Markov chains*, Comput. Statist. Data Anal., 29 (1999), pp. 345–368.

[20] D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 357–385.

[21] H. Tal-Ezer, *Polynomial approximation of functions of matrices and applications*, J. Sci. Comput., 4 (1989), pp. 25–60.

[22] H. Tal-Ezer, *High degree polynomial interpolation in Newton form*, SIAM J. Sci. Stat. Comput., 12 (1991), pp. 648–667.

[23] H. Tal-Ezer and R. Kosloff, *An accurate and efficient scheme for propagating the time dependant Schrodinger equation*, J. Chem. Phys., 81 (1984), pp. 3967–3971.

[24] H. Tal Ezer, R. Kosloff, and C. Cerjan, *Low order polynomial approximation of propagators for the time dependent Schrodinger equation*, J. Comput. Phys., 100 (1992), pp. 179–187.

[25] H. A. van der Vost, *Solution of $f(A)x = b$ with projection methods for the matrix A*, in Numerical Challenges in Lattice Quantum Chromodynamics, A. Frommer, T. Lippert, B. Medeke, and K. Schilling, eds., Lect. Notes Comput. Sci. Eng. 15, Springer-Verlag, Berlin, 2000, pp. 18–28.

[26] J. L. Walsh, *Interpolation and Approximation by Rational Functions in the Complex Plane*, Amer. Math. Soc. Colloq. Publ. 20, AMS, Providence, RI, 1956. 45

[27] K. Wu and H. Simon, *Thick-restart Lanczos method for symmetric eigenvalue problems*, SIAM J. Matrix Anal. Appl., 22 (2000), pp. 602–616.

[28] L. Zhang and T. Nodera, *New adaptive restart for GMRES(m) method*, Anziam J., 46E (2005), pp. C409–C425.

SIAM J. SCI. COMPUT.
Vol. 29, No. 6, pp. 2442–2463
© 2007 Society for Industrial and Applied Mathematics

A NUMERICAL BIFURCATION ANALYSIS OF THE ORNSTEIN–ZERNIKE EQUATION WITH HYPERNETTED CHAIN CLOSURE*

R. E. BEARDMORE†, A. T. PEPLOW†, AND F. BRESME‡

Abstract. We study the codimension-one and -two bifurcations of the Ornstein–Zernike equation with hypernetted chain (HNC) closure with Lennard–Jones intermolecular interaction potential. The main purpose of the paper is to present the results of a numerical study undertaken using a suite of algorithms implemented in MATLAB and based on pseudo arc-length continuation for the codimension-one case and a Newton-GMRES method for the codimension-two case. Through careful consideration of the results of our computations, an argument is formulated which shows that *spinodal* isothermal solution branches arising in this model cannot be reproduced numerically. Furthermore, we show that the existence of an upper bound on the density that can be realized on a vapor isothermal solution branch, which must be present at a spinodal, causes the existence of at least one fold bifurcation along that *vapor* branch when density is used as the bifurcation parameter. This provides an explanation for previous inconclusive attempts to compute solutions using Newton–Picard methods that are popular in the physical chemistry literature.

Key words. Ornstein–Zernike equation, spinodal, liquid phase, vapor phase

AMS subject classifications. 47J10, 45G10, 82B26, 82B80

DOI. 10.1137/060650659

Invitation. We invite the interested reader to email author R. E. Beardmore in order to obtain a copy of the MATLAB codes developed for this study.

1. Introduction.

The subject of this paper is the Ornstein–Zernike (OZ) equation that was presented almost a century ago to model the molecular structure of a fluid at varying densities [18]:

$$h(\|\mathbf{x}\|) = c(\|\mathbf{x}\|) + \rho \int_{\mathbb{R}^3} h(\|\mathbf{x}-\mathbf{y}\|)c(\|\mathbf{y}\|)d\mathbf{y}. \tag{1.1}$$

Here, $\mathbf{x}, \mathbf{y} \in \mathbb{R}^3$ are spatial coordinates and due to the assumed isotropy of the fluid we need only the radial coordinate $r = \|\mathbf{x}\|$, where $\|\cdot\|$ is the Euclidean distance function. The parameter ρ is the mean particle density of the fluid at temperature T, where $\beta = 1/(k_B T)$ will be used to denote the Boltzmann factor and $u(r)$ the intermolecular potential, a typical example of which is given below. The total (h) and direct (c) correlation functions yield the indirect correlation function of the fluid, $\gamma = h - c$, and the radial distribution function $g(r) = 1 + h(r)$ is the pair correlation function of molecules in the fluid.

The last two years or so have seen an increased use of continuation algorithms in liquid state theory. This can be seen both in [10, 9] in the context of density functional theory and in a study of the OZ equation [15]. The authors of [10, 9] make the express comment that continuation algorithms are needed to track phase transitions, and the

*Received by the editors January 23, 2006; accepted for publication (in revised form) April 5, 2007; published electronically October 17, 2007.
http://www.siam.org/journals/sisc/29-6/65065.html
†Department of Mathematics, Imperial College, 180 Queen's Gate, South Kensington, London, SW7 2AZ (r.beardmore@ic.ac.uk, atpeplow@kth.se).
‡Department of Chemistry, Imperial College, Exhibition Road, South Kensington, London, SW7 2AZ (f.bresme@ic.ac.uk).

purpose of this paper is to analyze the correspondence between the solutions of the OZ equation and its discretization using continuation algorithms. In particular, we show that for the OZ equation with *hypernetted chain closure*, a term that is defined below, there is no finite-dimensional projection of the equations and hence no algorithm that can approximate physically meaningful solutions when seeking a *spinodal*, a precursor to a phase transition.

1.1. Closures. In order to form a well-posed mathematical system of equations from (1.1) that can be solved, at least in principle, we impose a closure relationship. This is an algebraic equation or pointwise constraint that augments (1.1) and which is deemed to hold throughout the fluid.

Some closures are structured so that the total correlation function is a perturbation of the *Mayer f-function* that is defined by $f(r) = -1 + e^{-\beta u(r)}$. These closures are obtained beginning with the observation that there is a function $B(r)$ such that

$$h = -1 + e^{-\beta u + h - c + B} \tag{1.2}$$

holds throughout the fluid; B is called the Bridge function [6].

Due to the fact that the potential u will have singularities that may prevent $-\beta u + h - c$ from being well defined, let us rewrite (1.2) in the form $h = -1 + e^{-\beta u} + e^{-\beta u}\left(-1 + e^{h-c+B}\right)$. The hypernetted chain (HNC) closure that interests us dates from the late 1950's (see [23]) and is obtained by setting $B(r)$ to zero, something that is not possible to justify on physical grounds in any generality. Thus, the OZ equation with HNC closure (OZ-HNC) is (1.1) coupled to

$$h = f + e^{-\beta u} \cdot (-1 + e^{h-c}). \tag{1.3}$$

Such an approximation suffers from *thermodynamic inconsistency*, the property that different routes to computing thermodynamic quantities such as compressibility and pressure yield different results. This is often seen as a fundamental flaw of theories based on closures like HNC.

Another often-studied problem in the physical chemistry literature (again see [6] for background and further references) is the OZ equation with Percus–Yevick (PY) closure [19]. This also dates from the 1950's and is obtained by setting $B = \ln(\gamma + 1) - \ln \gamma$, leading to the PY closure

$$h = f + e^{-\beta u} \cdot (h - c). \tag{1.4}$$

We do not study this particular thermodynamically inconsistent form here. These and other closures have been in use for around thirty years, as explained in the review [6], and combinations of different closures have also been proposed, some of which are reviewed in the latter reference, but we have chosen to concentrate on OZ-HNC.

The HNC and PY closures can be written in a general form involving the potential, temperature, and the indirect correlation function through a nonlinearity G, say, as follows:

$$h = f + e^{-\beta u} G(h - c), \qquad G(0) = 0. \tag{1.5}$$

The suite of algorithms we have developed are able to solve (1.1) and (1.5) for any function G with β and ρ as bifurcation parameters. Moreover, solution branches obtained by varying ρ but holding β fixed are called *isotherms*; those obtained by fixing ρ but varying β are *isochores*.

1.2. Local existence. Note that the OZ equation with closure of the form (1.5) can be written as a single integral equation

$$\gamma = \rho(f + e^{-\beta u}G(\gamma)) * (f + e^{-\beta u}G(\gamma) - \gamma), \tag{1.6}$$

and in the remainder we shall use $N_\beta(\gamma)$ to denote the operator formed by the right-hand side of (1.6), so that this integral equation can be written $\gamma = \rho N_\beta(\gamma)$ for brevity.

The following basic existence theorem for (1.6) can be easily proven using the implicit function theorem, but we state it here without proof.

THEOREM 1. *If* $G(\gamma) = -1 + e^\gamma, \beta > 0$, *and* $f \in L^1(\mathbb{R}^3), \cap L^\infty(\mathbb{R}^3)$, *then there is a* $\rho' > 0$ *such that* (1.6) *has a solution* $\gamma \in L^1(\mathbb{R}^3) \cap L^\infty(\mathbb{R}^3)$ *for each* $0 \le \rho < \rho'$.

As a result, one generally requires the short-range condition that f is integrable and bounded over $\mathbb{R}^3$ for $\beta > 0$, a property that follows from the assumption that $\int_1^\infty r^2 u(r)dr < \infty$ such that u is measurable on $(0, \infty)$ and continuous on a half-neighborhood of zero and $\lim_{r\to 0_+} u(r) = +\infty$.

Our algorithms are implemented with the assumption that u is smooth on $(0, \infty)$; we do not perform any computations where the potential is nonsmooth, although it is common in the literature to do so [6, 13, 16]. Indeed, all computations presented here use the Lennard–Jones (LJ) 6-12 potential,

$$u_{LJ}(r) = 4\epsilon\left(\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right),$$

where σ is particle diameter and ϵ controls the well-depth. In accordance with [12], we shall express thermodynamic quantities such as density and temperature in reduced units, $\rho^* = \rho\sigma^3$, $T^* = k_B T/\epsilon = 1/(\epsilon\beta)$, although $\sigma = \epsilon = 1$ hold throughout the paper so that $\rho = \rho^*$ and $T^* = 1/\beta$.

1.3. Spinodals. Let us define the *inverse isothermal compressibility* of a fluid

$$\chi^{-1}(c, \rho) := 1 - \rho \int_{\mathbb{R}^3} c(\|\mathbf{x}\|)d\mathbf{x}. \tag{1.7}$$

This quantity plays a fundamental role in the solution of (1.1) and (1.5) for the following reason.

DEFINITION 1. *Let* $\beta > 0$ *be fixed. A spinodal is said to occur in the OZ-HNC equation* (1.1) *and* (1.5) *at* $\rho = \rho_{sp}$ *if there is a sequence* (h_n, c_n, ρ_n) *of solutions such that* $\rho_n \to \rho_{sp}$ *as* $n \to \infty$ *and* $\chi^{-1}(c_n, \rho_n) \to 0$.

As the temperature is reduced the separation of two phases in the fluid, one liquid and one vapor, begins with the creation of two spinodals at a critical value of temperature and density, T_c and ρ_c. Thus the search for two thermodynamic phases begins with the search for a codimension-two point at which a single critical solution can be found, with $T = T_c$ and $\rho = \rho_c$, say, at which inverse isothermal compressibility is zero.

One consequence of the existence of two spinodals that lie on vapor and liquid isotherms is a region of densities between the two where no solution of (1.6) exists; this region is sometimes called a *no-solution region*. Several analyses based on numerical solutions of the discretized OZ equations appear to have established that a *no-solution region* exists [24, 3, 2, 17, 22], and the purpose of this paper is to investigate these claims further, noting that *none* of the cited references use bifurcation algorithms in their studies. Note in particular [20, Fig. 4] whereby a conjectured series of solution

branches have been drawn; we shall demonstrate that this figure is essentially correct but that there may well be more fold bifurcations present than are shown in [20].

Although we shall show that we are not strictly able to locate phase transitions by applying continuation algorithms to (1.6), we shall use the following suggestive terminology. Isotherms that contain the low-density base solution of (1.6) given by $h = f, c = f$, and $\rho = 0$ will be called *vapor* branches throughout. If there are two isothermal solution branches, the one associated with higher densities will be called the *liquid* branch. We shall also use the prefix *discrete* to denote that these terms apply to solution branches of a discretization of (1.6).

1.3.1. Fold bifurcations and pseudospinodals. Let us consider for a moment the following definition of fold bifurcation, noting that it also incorporates solution branches that intersect in pitchfork or transcritical bifurcations.

DEFINITION 2 (fold bifurcation). *Let X be a Banach space. The nonlinear equation $F(\gamma, \rho) = 0$, where $F : X \times \mathbb{R} \to X$, is said to possess a fold bifurcation at ρ_{FB} if there is a $\gamma_{FB} \in X$ and at least two distinct sequences $(\gamma_n) \in X$ and $(\overline{\gamma}_n) \in X$ such that for some sequence $(\rho_n) \subset \mathbb{R}$*

(i) *$F(\gamma_n, \rho_n) = 0$ and $F(\overline{\gamma}_n, \rho_n) = 0$ for all n,*
(ii) *$\rho_n < \rho_{FB}$ for all n, and*
(iii) *$\lim_{n\to\infty}(\gamma_n, \rho_n) = \lim_{n\to\infty}(\overline{\gamma}_n, \rho_n) = (\gamma_{FB}, \rho_{FB}) \in X \times \mathbb{R}$.*

The reason for introducing this definition is that while phase transitions begin with a search for spinodals, it was noted as long ago as 1968 for the PY closure [1] (see [1, Fig. 1] in particular) that fold bifurcations could occur where one might anticipate the existence of spinodals. As a result, fold bifurcations in the context of (1.6) are often called *pseudospinodals*, although *square-root branch points* [2] are also used. Let us again remark that Definition 2 encompasses a greater class of solution curves than simply those with the typical folded structure at a bifurcation point, but the theorems later in the paper are only able to demonstrate the existence of a fold bifurcation in the sense of Definition 2.

Using a standard codimension-one formulation of (1.6), we shall compute curves that illustrate how fold bifurcations of (1.6) vary as a function of temperature and density, and these rather resemble *phase diagrams* and may be thought of as *an approximation* of the boundary of the region of liquid-vapor coexistence.

The basic Newton method has been seen as the most appropriate algorithm for studying (1.6), which is why continuation algorithms were not used to study the OZ equation until very recently [15]. Moreover, the various codimension-two problems associated with OZ have yet to be tackled directly in the physical chemistry literature; hence this paper addresses these issues. This is perhaps a little surprising given the debate that took place in [11, 5, 4, 7] regarding the errors committed when employing inappropriate numerical techniques. Moreover, the study of the correspondence between analytical and numerical solutions of the OZ equation with PY closure performed in [7] shows surprising discrepancies that, to the best of the authors' knowledge, have not received adequate attention in the literature.

Typical behavior: Unphysical isotherms part I. In anticipation of the results to follow, in Figure 1.1 we have plotted a locus of fold bifurcations for (1.6) that has been computed for an LJ fluid using a codimension-one formulation of the OZ-HNC equation; note its resemblance to a phase diagram. (The meaning of the parameter $R = 10\sigma$ that appears in the caption of this figure is explained later in the paper.)

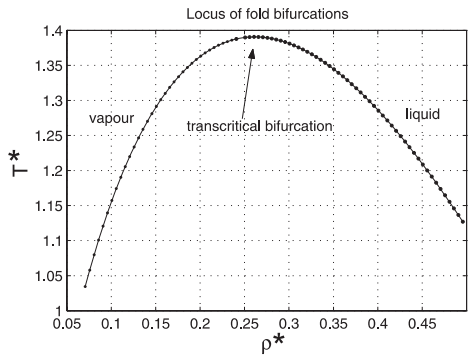


FIG. 1.1. *(LJ potential, HNC closure, $R = 10\sigma$ with $2^{12} + 1$ mesh points.) A locus of fold bifurcations showing liquid and vapor branches created at a transcritical bifurcation. Below the parabolic curve there are regions where no solution exists bounded by two fold bifurcations that occur as ρ is varied and β is fixed; one occurs on the liquid branch and one occurs on the vapor branch (see Figure* 1.2*).*

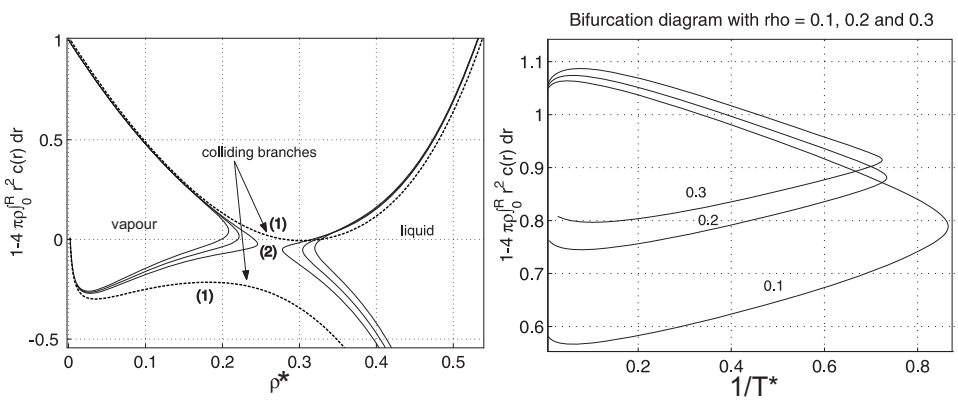


FIG. 1.2. *(LJ potential, HNC closure, $R = 10\sigma$, $2^{12} + 1$ mesh points.) (Left) Two supercritical branches (at* (1)*) touch (near* (2)*) and exchange singularities: Inverse compressibility plotted against density with label* (1) $\beta = 0.696$*, label* (2) $\beta = 0.72$*; two more liquid-vapor isothermal pairs are shown with lower temperatures at* $\beta = 0.725$ *and* 0.73*. (Right) Three isochores showing fold bifurcations at low temperatures; density is fixed at* $\rho = 0.1, 0.2, 0.3$*.*

In Figure 1.2 (left) we have computed a set of isothermal solution branches that show density plotted on the horizontal axis against inverse isothermal compressibility on the vertical axis; to produce each curve, temperature is lowered and a pair of computations is performed, and the resulting figure is an *unfolding* of the apex from Figure 1.1. Label (2) in Figure 1.2 (left) sees the creation of two fold bifurcations or *pseudospinodals* that arise due to a collision of two solution branches, one of which is a vapor branch with positive compressibility, but the other has negative compressibility and is therefore not a branch of physically meaningful solutions.

1.4. Notation. Given a nonlinear mapping $N(\gamma)$ on a Banach space X that contains elements γ and h, $dN(\gamma)[h]$ denotes the Fréchet derivative and $BL(X)$ is the space of bounded, linear mappings from X to itself. We shall write $N_\beta(\gamma)$ to emphasize the dependence of N on a parameter β, but we shall suppress the subscript otherwise. The dual space of all bounded linear functionals on X is written X^*.

Numerical objects that correspond to infinite-dimensional objects will be written using a bold font, so that $\boldsymbol{\gamma}$ will be a vector that corresponds to a correlation function γ projected onto a finite-dimensional subspace of X. The set $\{\mathbf{e}_j\}_{j=0}^n$ is used to denote the standard basis vectors of $\mathbb{R}^{n+1}$. We shall write $B_R(\mathbf{0}) = \{\mathbf{x} \in \mathbb{R}^3 : \|\mathbf{x}\| < R\}$ for a ball in $\mathbb{R}^3$, and $\langle \mathbf{u}, \mathbf{v} \rangle$ is the standard Euclidean inner product of vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^{n+1}$.

We depart a little from convention by using $L^p(\mathbb{R}^3)$ to denote the space $\{u : \mathbb{R}^3 \to \mathbb{R} : \int_{\mathbb{R}^3} |u(\mathbf{x})|^p d\mathbf{x} < \infty\}$, where $p \geq 1$, and $C^0(\mathbb{R}^3)$, $C_0^\infty(\mathbb{R}^3)$, and $L^\infty(\mathbb{R}^3)$ are defined analogously (the point is that the range of functions in these spaces is always $\mathbb{R}$). By $C^{\mathrm{Lip}}(\Omega)$ for a domain $\Omega \subset \mathbb{R}^3$ we understand the space $\{u \in C^0(\overline{\Omega}) : \exists K > 0 \text{ s.t. } |u(\mathbf{x}) - u(\mathbf{y})| \leq K\|\mathbf{x} - \mathbf{y}\|\}$ endowed with a standard norm.

We will use a majuscule to represent the Fourier transform of a minuscule, as in

$$\mathcal{F}(c)(\mathbf{s}) := C(\mathbf{s}) = \int_{\mathbb{R}^3} c(\mathbf{x}) e^{i\mathbf{x}\cdot\mathbf{s}} d\mathbf{x}.$$

The convolution of radially symmetric functions can be written using the convolution theorem and the Hankel transform $\mathcal{H}$, the Fourier transform of radially symmetric functions that is given by

$$\mathcal{H}(a)(s) = 4\pi \int_0^\infty \operatorname{sinc}(rs) r^2 a(r) dr$$

so that $\mathcal{H}^{-1}(a)(s) = \frac{1}{2\pi^2} \int_0^\infty \operatorname{sinc}(rs) r^2 a(r) dr$. Thus we may define a convolution operator by $a * b = \mathcal{H}^{-1}((\mathcal{H}a) \cdot (\mathcal{H}b))$, where $a, b : [0, \infty) \to \mathbb{R}$ are given functions.

2. Observations on the OZ-HNC equation. Throughout the remainder of the paper let us consider (1.6) with the HNC closure $G(\gamma) = -1 + e^\gamma$:

$$\gamma = \rho(f + e^{-\beta u(r)}(-1 + e^\gamma)) * (f + e^{-\beta u(r)}(-1 + e^\gamma) - \gamma). \tag{2.1}$$

Formulation (2.1) is advantageous from a theoretical point of view because G can be regarded as a function that maps $L^1(\mathbb{R}^3) \cap L^\infty(\mathbb{R}^3)$ to itself, a property that is not shared by either of the functions $\gamma \mapsto e^\gamma$ or $\gamma \mapsto \gamma - 1$. As this paper is written from a computational perspective, (2.1) will be rewritten in the more common form

$$\gamma = \rho(-1 + e^{-\beta u(r)} e^\gamma) * (-1 + e^{-\beta u(r)} e^\gamma - \gamma).$$

2.1. Spinodals: Bifurcations at infinity in $L^1(\mathbb{R}^3)$. Applying the Fourier transform to (1.1), we obtain the quadratic algebraic equation that holds almost everywhere,

$$H(s)(1 - \rho C(s)) = C(s), \tag{2.2}$$

where $s \in \mathbb{R}$ is the wave number and H and C are the Fourier transforms of h and c, respectively.

LEMMA 2.1. *There is no pair* $(h, c) \in L^1(\mathbb{R}^3) \times L^1(\mathbb{R}^3)$ *that satisfies* (1.1) *and* $\chi^{-1}(c, \rho) = 0$.

Proof. Suppose that a solution (h, c) of (1.1) exists at a given ρ and that $\chi^{-1}(c, \rho) = 0$. Then $1 = \rho C(0) = \rho \int_{\mathbb{R}^3} c d\mathbf{x}$ but (2.2) yields (on allowing $s \to 0$)

$$H(0) \cdot (1 - \rho C(0)) = C(0) \Longrightarrow \int_{\mathbb{R}^3} h d\mathbf{x} \cdot \chi^{-1}(c, \rho) = \int_{\mathbb{R}^3} c d\mathbf{x} = 0,$$

which is a contradiction. ∎

Suppose that an isothermal solution branch of (1.1) and (1.5) exists (call it $\mathcal{C}$) and suppose that it lies in $L^1(\mathbb{R}^3) \times L^1(\mathbb{R}^3) \times \mathbb{R}$. Suppose also that there is a sequence of low-density solutions in $\mathcal{C}$, so that $\mathcal{C}$ connects to the vapor base solution where $\rho = 0$ and therefore $\chi^{-1}(c, \rho)$ is positive along the low-density part of $\mathcal{C}$.

Now suppose that $\mathcal{C}$ contains a *spinodal* in the sense that there is a sequence $(h_n, c_n, \rho_n) \in \mathcal{C}$ such that $\rho_n \to \rho_{sp} > 0$ and $\lim_{n\to\infty} \chi^{-1}(c_n, \rho_n) = 0$. From (2.2) upon setting $s = 0$ we find

$$\int_{\mathbb{R}^3} h_n d\mathbf{x} \cdot \chi^{-1}(c_n, \rho_n) = \int_{\mathbb{R}^3} c_n d\mathbf{x} \to 1/\rho_{sp},$$

and we must conclude that $\int_{\mathbb{R}^3} h_n(\|\mathbf{x}\|)d\mathbf{x} \to \infty$ as $n \to \infty$. Thus, a spinodal is associated with a bifurcation from infinity in $L^1(\mathbb{R}^3)$. Let us record this in a lemma.

LEMMA 2.2. *Suppose that* $(h_n, c_n, \rho) \subset L^1(\mathbb{R}^3) \times L^1(\mathbb{R}^3) \times \mathbb{R}$ *is a spinodal sequence that accumulates at a nonzero density* $\rho = \rho_{sp}$*; then* $\|h_n\|_{L^1(\mathbb{R}^3)} \to \infty$ *as* $n \to \infty$.

Thus, the fact that a spinodal sequence is a bifurcation at infinity in an $L^1(\mathbb{R}^3)$-norm is a property of the OZ equation alone; whether or not a spinodal is actually present depends essentially on the nature of the closure relationship used in (1.5). For example, the following simple lemma shows that the HNC closure leads to the boundedness of h in the $L^2(\mathbb{R}^3)$-norm at a spinodal, provided that the direct correlation function c in (1.1) is well behaved.

LEMMA 2.3. *Suppose that* $u \in L^1(\mathbb{R}^3 \backslash B_\epsilon(\mathbf{0}))$ *for some* $\epsilon > 0$ *and* (h, c) *solves* (1.1) *and* (1.3) *such that* $\|h\|_{L^\infty} \le M$ *and* $\|c\|_{L^1} + \|c\|_\infty \le M$*. Then there is a* $K_M > 0$ *such that*

$$\int_{\mathbb{R}^3} h(\|\mathbf{x}\|)^2 d\mathbf{x} \le \frac{4\pi\epsilon^2 M^2}{3} + \frac{1}{K_M} \int_{\|\mathbf{x}\|\ge\epsilon} (\beta u + c(\|\mathbf{x}\|))d\mathbf{x}. \tag{2.3}$$

Hence $h \in L^p(\mathbb{R}^3)$ *for all* $2 \le p \le \infty$.

Proof. If we define the function $F(x) = \ln(x + 1) - x$ for all $x > -1$, then $F(0) = 0, F'(0) = 0$, and $F''(x) = -1/(x + 1)^2 < 0$ so that F is concave. Clearly, for each $M > 0$ fixed, there is a $K_M > 0$ such that $F(x) \le -K_M x^2$ for all x such that $-1 < x \le M$. Now, if (h, c) solves (1.1) and (1.3), then from (1.3) we deduce

$$K_M \int_{\|\mathbf{x}\|\ge\epsilon} h^2 d\mathbf{x} \le -\int_{\|\mathbf{x}\|\ge\epsilon} F(h)d\mathbf{x} = \int_{\|\mathbf{x}\|\ge\epsilon} (\beta u + c)d\mathbf{x}$$

because $h(\|\mathbf{x}\|) > -1$ if $\|\mathbf{x}\| > \epsilon$ from the HNC closure. Hence,

$$\begin{aligned}\int_{\mathbb{R}^3} h^2 d\mathbf{x} &\le \int_{\|\mathbf{x}\|\le\epsilon} h^2 d\mathbf{x} + \frac{1}{K_M} \int_{\|\mathbf{x}\|\ge\epsilon} (\beta u + c)d\mathbf{x} \\ &\le M^2 \text{meas}\{\mathbf{x} : \|\mathbf{x}\| \le \epsilon\} + \frac{1}{K_M} \int_{\|\mathbf{x}\|\ge\epsilon} (\beta u + c)d\mathbf{x},\end{aligned}$$

and the result follows. ∎

We can now use the bound from Lemma 2.3 to obtain the following result which states that if a spinodal sequence exists and the correlation functions stay uniformly bounded, and the direct correlation function that is known to be integrable is in fact absolutely integrable, then there are functions $h_0 \in L^2(\mathbb{R}^3) \backslash L^1(\mathbb{R}^3)$ and c_0 that

together satisfy the OZ equation. Any function $h_0 \in L^\infty(\mathbb{R}^3)$ that satisfies an almost-everywhere bound of the form

$$\frac{C}{r^3} < |h_0(\mathbf{x})| < \frac{C'}{r^{(3+\delta)/2}} \qquad \text{(a.a. } \mathrm{r} \geq \mathrm{R})$$

for constants C, C', and R, where $\delta \in (0,3)$ is arbitrary and $r = \|\mathbf{x}\|_2$, the Euclidean length of $\mathbf{x}$, satisfies $h_0 \in L^2(\mathbb{R}^3) \backslash L^1(\mathbb{R}^3)$, and this may give us some clue as to how the total correlation function behaves at a spinodal.

LEMMA 2.4. *Suppose that ρ_{sp} is a density associated with a spinodal sequence $(h_n, c_n, \rho_n) \in L^1(\mathbb{R}^3) \times L^1(\mathbb{R}^3) \times \mathbb{R}$ for $n \geq 1$ and there is an $M > 0$ such that $\|h_n\|_{L^\infty} \leq M$ and $\|c_n\|_{L^1} + \|c_n\|_{L^\infty} \leq M$. Then there are functions $h_0 \in L^2(\mathbb{R}^3) \backslash L^1(\mathbb{R}^3)$ and $c_0 \in L^2(\mathbb{R}^3) \cap L^\infty(\mathbb{R}^3)$ such that*

$$h_0 = c_0 + \rho_{sp}(h_0 * c_0) \quad \text{and} \quad \chi^{-1}(c_0, \rho_{sp}) = 0.$$

Proof. (In this proof, although all functions are defined on $\mathbb{R}^3$, we assume them to be radially symmetric.) The result follows from the continuity of the convolution operator $*$ with respect to weak convergence. For if (h_n, c_n, ρ_n) has the stated properties, then, although $\|h_n\|_{L^1(\mathbb{R}^3)} \to \infty$ as $n \to \infty$, there are bounds of the form

$$\|h_n\|_{L^\infty} \leq M \quad \text{and} \quad \|c_n\|_{L^1} + \|c_n\|_{L^\infty} \leq M$$

that come from the definition of an admissible spinodal sequence. From Lemma 2.3 we may assume that $\|h_n\|_{L^2} \leq M$. We can now apply the Banach–Alaoglu theorem to extract weak and weak* convergent subsequences to find an $h_0 \in L^2$ and a $c_0 \in L^\infty \cap L^2$ such that

$$h_n \rightharpoonup h_0 \text{ in } L^2, c_n \rightharpoonup c_0 \text{ in } L^2, \text{ and } c_n \overset{*}{\rightharpoonup} c_0 \text{ in } L^\infty$$

as $n \to \infty$, using $(L^1)^* \cong L^\infty$ and the fact that $(c_n) \subset (L^1)^*$, which is a bounded sequence.

Now suppose that $\phi \in C_0^\infty(\mathbb{R}^3)$ has support on some closed ball Ω and let $\epsilon > 0$. From the symmetry properties of convolution there is an N_0 such that

$$\begin{aligned} |\langle \phi, h * c\rangle - \langle \phi, h_n * c_n\rangle| &= |\langle h * \phi, c - c_n\rangle - \langle (h_n - h) * \phi, c_n\rangle| \\ &= |\langle h * \phi, c - c_n\rangle - \langle (h_n - h) * \phi, c_n - c\rangle - \langle (h_n - h) * \phi, c\rangle| \\ &\leq \epsilon + |\langle (h_n - h) * \phi, c_n - c\rangle| + \epsilon \end{aligned}$$

for all $n \geq N_0$.

Suppose now that $u_n := h_n - h_0 \rightharpoonup 0$ in L^2 and $v_n := c_n - c_0 \rightharpoonup 0$ in L^2 as $n \to \infty$; then $u_n * \phi$ has support in $2 \cdot \Omega$, where $2 \cdot \Omega = \{2\mathbf{x} : \mathbf{x} \in \Omega\}$. Moreover, from the Cauchy–Schwarz inequality

$$\operatorname{ess\,sup}_{\mathbf{x} \in 2\cdot\Omega} |(u_n * \phi)(\mathbf{x})| \leq \|u_n\|_{L^2} \|\phi\|_{L^2} < \infty$$

and

$$|(u_n * \phi)(\mathbf{x}_1) - (u_n * \phi)(\mathbf{x}_2)| \leq \operatorname{meas}(2 \cdot \Omega)^{1/2} \|\phi\|_{C^1} \|\mathbf{x}_1 - \mathbf{x}_2\| \|u_n\|_{L^2},$$

and we may assume that $u_n * \phi$ is continuous and uniformly bounded. Clearly the Lipschitz norm $\|u_n * \phi\|_{C^{\mathrm{Lip}}(2\cdot\Omega)}$ is bounded and because the embedding $C^{\mathrm{Lip}}(2 \cdot \Omega) \subset$

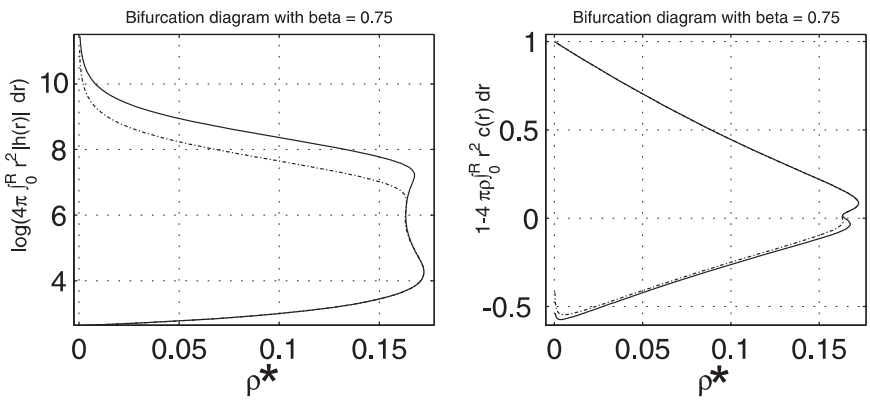


FIG. 2.1. *(LJ potential, HNC closure,* $2^{12}+1$ *mesh points, and* $\beta = 0.75$*.) Two computations with different cut-off parameters* $R = 50\sigma$ *(dashed line) and* 80σ *(full line). Spinodal-like behavior is clearly visible, and note the presence of multiple fold bifurcations.*

$C^0(2\cdot\Omega)$ is compact and $u_n \rightharpoonup 0$ in L^2, it follows that $u_n * \phi \rightharpoonup 0$ in L^2, and so we may assume that $u_n * \phi \to 0$ uniformly on $2\cdot\Omega$.

Hence $|\langle u_n * \phi, v_n\rangle| \le \|u_n * \phi\|_{C^0(2\cdot\Omega)}\|v_n\|_{L^2}$ and on setting $u_n := h_n - h_0$, $v_n := c_n - c_0$ we have shown that $h_n * v_n \rightharpoonup h_0 * v_0$ in L^2 because $C_0^\infty(\mathbb{R}^3)$ is dense in $L^2(\mathbb{R}^3)$. Since $-h_n + c_n + \rho_n h_n * v_n = 0$ for all n we find that the pair (h_0, c_0) satisfies the OZ equation.

To obtain the zero inverse compressibility of this pair, because $\chi^{-1}(c_n, \rho_n) = 1 - \rho_n \int_{\mathbb{R}^3} c_n(\|\mathbf{x}\|)d\mathbf{x} \to 0$ it suffices to show that $\int_{\mathbb{R}^3} c_n d\mathbf{x} \to \int_{\mathbb{R}^3} c_0 d\mathbf{x}$, which itself follows if $\int_{\mathbb{R}^3} c_n\phi d\mathbf{x} \to \int_{\mathbb{R}^3} c_0\phi d\mathbf{x}$ for all $\phi \in L^\infty$. Now, $c_n \in L^1(\mathbb{R}^3)$ can be viewed as a continuous linear functional in $(L^\infty)^*$ acting via integration, as in $c_n(\psi) = \int_{\mathbb{R}^3} c_n\psi d\mathbf{x}$ and $\|c_n\|_{(L^\infty)^*} \le \|c_n\|_{L^1}$. The result now follows by applying the Banach–Alaoglu theorem and choosing $\phi = 1$. □

One might call the functions h_0 and c_0 spinodal *semisolutions* because, although they satisfy the OZ equation (1.1), they may well not satisfy the constraint (1.3). Establishing the existence of the a priori bounds needed for Lemma 2.4 depends on the closure (1.5) and is not a straightforward task. However, given that (1.1) is a description of the fluid that holds under any thermodynamic conditions, it is essential that conditions exist such that (1.1) continues to hold whether or not the fluid is at a spinodal.

Computing spinodals: Unphysical isotherms part II. The property of Lemma 2.3 of having a sequence of total correlation functions (h_n) that are $L^2(\mathbb{R}^3)$-bounded but divergent in $L^1(\mathbb{R}^3)$ will cause difficulties for any numerical discretization of (1.1) and (1.5). Indeed, it will force such a discretization to possess uniform a priori bounds because of the equivalence of different norms on finite-dimensional spaces.

This simple observation ensures that admissible spinodal sequences cannot be reproduced easily with a computational procedure, and this may go some way to explaining the claims made in [2, 20] that apparent spinodals in numerically computed solutions are not true spinodals. Moreover, if Lemma 2.3 does provide a description of the fluid through a spinodal, the resulting computations must behave in a manner that produces nonphysical results.

We have illustrated this problem in Figure 2.1 containing two plots. The left-hand figure shows the $L^1(\mathbb{R}^3)$-norm of h plotted on a log scale against density where a very rapid change in this norm takes place over a very small density interval. This is clear

numerical evidence of the existence of a spinodal, and the right-hand plot shows the numerically obtained inverse compressibility χ^{-1} against density, with this variable passing through zero, which seems to confirm this observation. However, yet again we find that there are physically meaningless solutions with negative compressibility.

Clearly, the values of χ^{-1} are very close to zero along the *vertical* part of the branch in the left-hand plot of Figure 2.1. However, a number of fold bifurcations can be seen (three in this plot) and there is a final fold after which the density monotonically decreases before finally heading toward zero as the norm grows further; this is also physically meaningless behavior. We shall give a tentative explanation of this behavior later in the paper and prove that the existence of a spinodal along a vapor branch in a discretization of (1.1) with the HNC closure must yield the existence of a fold bifurcation.

Remark 1. The existence of an apparent solution whereby $\chi^{-1}(c,\rho)=0$ is troublesome in light of Lemma 2.1 and is an artifact of the discretization used to solve (2.1). However, the discretization procedure used to produce Figure 2.1 is standard in the physical chemistry literature: as described in the following section, the fluid is truncated to a ball of radius R, called the cut-off parameter, and (2.1) is solved as an integral equation on the resulting finite domain. In Figure 2.1 we have used two different values for R—one of $R=50\sigma$ and one of $R=80\sigma$, where σ is the particle diameter, here taken to be 1.

3. Computational procedures.

3.1. The cut-off parameter *R*. In all of the liquid-state physics literature that we have cited (see [24, 3, 17, 2, 15], for example), one solves (1.1) numerically by first truncating the region of integration to a finite region, and so we also adopt this procedure. So, rather than solving (1.1), one seeks solutions of a *windowed* integral equation,

$$h(r)=c(r)+\int_{\mathbb{R}^3}[X_{_R}h](\|\mathbf{x}-\mathbf{y}\|)[X_{_R}c](\|\mathbf{y}\|)d\mathbf{y}\qquad(0\le r\le R),\tag{3.1}$$

where $X_{_R}$ is the windowing operator:

$$(X_{_R}v)(\mathbf{x})=\begin{cases} v(\mathbf{x}) & : \quad \|\mathbf{x}\|\le R,\\ 0 & : \quad \text{otherwise.}\end{cases}$$

The region of integration in (3.1) is therefore finite, which permits a straightforward method of discretization to be applied to (3.1) in tandem with the HNC constraint.

Clearly, $X_{_R}$ is a projection operator in the sense that $X_{_R}(X_{_R}(u))=X_{_R}(u)$ for all $u\in L^1(\mathbb{R}^3)$, and so if we set $h_R=X_{_R}h$ and $c_R=X_{_R}c$, (3.1) is equivalent to

$$h_R=c_R+\rho X_{_R}(h_R*c_R),\tag{3.2}$$

so that when we take the Fourier transform of (3.2) we obtain a relationship involving convolution with the transform of the windowing operator applied to the constant function that is identically one, here denoted 1,

$$H_R=C_R+\rho\widehat{X_{_R}(1)}*(H_R\cdot C_R),\tag{3.3}$$

where $\widehat{X_{_R}(1)}(s)=(2R)^{-1/2}s^{-3/2}J_{3/2}(Rs)$ is a certain Bessel function.

The point of this is the Fourier transform of the windowed OZ equation does not lead to an algebraic equation but rather to a convolution equation with a fixed

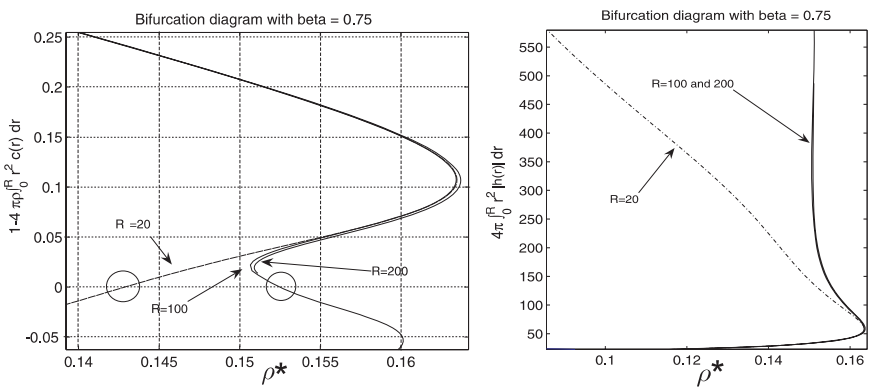


FIG. 3.1. *(LJ potential, HNC closure, $R = 20\sigma, 100\sigma, 200\sigma, \beta = 0.75$, and $2^{13} + 1$ mesh points.) An illustration of the high sensitivity of the bifurcation diagram to changes in the windowing parameter R near an apparent spinodal (marked by circles). (Left) Inverse compressibility against ρ^*. (Right) $L^1(\mathbb{R}^3)$-norm of h against ρ^*.*

kernel, and this makes it difficult to define a *compressibility* functional of the windowed problem (3.1) with the correct qualitative properties. For example, the argument of Lemma 2.1 breaks down when the cut-off procedure is applied, and so the relationship between spinodals and bifurcations from infinity is no longer apparent.

Moreover, if one still defines inverse compressibility in the sense of (1.7), then it may well be possible to realize $L^1(\mathbb{R}^3)$ solutions of the windowed OZ equation (3.1) with zero inverse compressibility. This is illustrated in Figure 3.1 (left), where two regions have been circled that indeed appear to contain numerically computed solutions with zero inverse compressibility.

Based on Figure 3.1 a further comment can be made in terms of the order of convergence of approximate solutions of (3.1) and (1.5) in a suitable norm as $R \to \infty$. One needs estimates of $\|h - X_R h\|_Z$ in some space Z, and, bearing in mind the $L^1(\mathbb{R}^3)$-divergence at a spinodal, the bounds

$$\|h - X_R h\|_Z \leq \begin{cases} 1/(4\pi R^2)\|h - X_R h\|_{L^1(\mathbb{R}^3)} = o(R^{-2}), \\ 1/(4\pi R)^{1/2}\|h - X_R h\|_{L^2(\mathbb{R}^3)} = o(R^{-1/2}) \end{cases}$$

apply with $Z = L^1(0, \infty)$ if $h \in L^1(\mathbb{R}^3) \cap L^2(\mathbb{R}^3)$.

One *may* subsequently anticipate a degradation of the convergence of approximate solutions with respect to R as one nears a spinodal. Indeed, the relatively poor convergence of the bifurcation picture with respect to R is illustrated in Figure 3.1, where a computation is made along an apparent near-spinodal solution branch. Three choices are used, $R = 20\sigma$, 100σ, and 200σ, recalling that the particle diameter σ is 1, where only the larger two of the values resolve a number of bifurcations that seem to be missed when the smaller value is used for R.

3.2. Numerical implementation of OZ-HNC. Suppose that $v : [0, \infty) \to \mathbb{R}$ is a given function, and define the windowed Hankel transforms

$$\mathcal{H}_R v = X_R \mathcal{H}(X_R v) \text{ and } \mathcal{H}_R^{-1} v = X_R \mathcal{H}^{-1}(X_R v);$$

we then finally arrive at the equation that is to be solved as part of our computational study:

$$\gamma = \rho \mathcal{H}_R^{-1}[\mathcal{H}_R(f + e^{-\beta u} G(\gamma)) \cdot \mathcal{H}_R(f + e^{-\beta u} G(\gamma) - \gamma)], \tag{3.4}$$

which is the windowed analogue of (1.6). Now let us briefly explain our discretization of (3.4). First recall that our computations are performed using only the HNC form for G given by $G(\gamma) = -1 + e^{\gamma}$, and note that in the following description, we shall use a subscript applied to a vector in $\mathbb{R}^{n+1}$ to denote a coordinate from that vector.

So, begin by choosing a set of basis functions $\{\phi_i(r)\}_{i=0}^n$ centered on nodes (r_i) of a uniform mesh, taking mesh points $r_i = iR/n$ for $0 \le i \le n$. We seek an approximate solution of (3.4) of the form $\gamma = \sum_{i=0}^n \gamma_i \phi_i(r)$, and the resulting system is projected onto ϕ_j for all $j = 0, \dots, n$. For any such set of basis functions, this procedure yields a finite-dimensional system of equations of dimension $n+1$ and, writing

$$e^{-\beta \mathbf{U}} = (e^{-\beta u(r_i)})_{i=0}^n \text{ such that } (e^{-\beta \mathbf{U}})_0 = 0,$$

$\mathbf{h} = (h_i)_{i=0}^n$, and $\mathbf{c} = (c_i)_{i=0}^n$, the HNC closure yields

$$h_0 = -1,\ h_i = -1 + e^{-\beta u(r_i)} \exp(\gamma_i), \text{ and } c_i = h_i - \gamma_i \qquad (i = 1, \dots, n), \tag{3.5}$$

and we obtain the following system of equations for $\mathbf{h}$ and $\mathbf{c}$:

$$\mathbf{h} = \mathbf{c} + \rho B(\mathbf{h}, \mathbf{c}), \tag{3.6}$$

$$\mathbf{h} = \mathbf{f} + e^{-\beta \mathbf{U}}(-\mathbf{1} + \exp(\mathbf{h} - \mathbf{c})). \tag{3.7}$$

Here $\mathbf{1} = (1, 1, \dots, 1)$ and $B : \mathbb{R}^{n+1} \times \mathbb{R}^{n+1} \to \mathbb{R}^{n+1}$ is a bilinear form that represents a discrete convolution operator and we shall also write, here and throughout,

$$\mathbf{f} = -\mathbf{1} + e^{-\beta \mathbf{U}}$$

for the so-called discrete Mayer f-function.

The convolution operator from (1.1) is the only part of the problem that is discretized in a nontrivial way to give (3.6); the algebraic constraint given by the HNC closure (1.3) simply yields the elementwise constraint (3.7). This means that the bilinear form B in (3.6) is central to any analysis, and we shall assume that B is nonzero and satisfies properties B1–B2 defined as follows:

B1. $B(\mathbf{u}_1 + \mathbf{u}_2, \mathbf{v}) = B(\mathbf{u}_1, \mathbf{v}) + B(\mathbf{u}_2, \mathbf{v})$ and $B(\lambda \mathbf{u}, \mathbf{v}) = \lambda B(\mathbf{u}, \mathbf{v})$,

B2. $B(\mathbf{u}, \mathbf{v}) = B(\mathbf{v}, \mathbf{u})$,

B3. $B(\mathbf{u}, \mathbf{u}) = 0 \Longrightarrow \mathbf{u} = 0$, and

B4. $\langle \mathbf{u}, B(\mathbf{u_1}, \mathbf{u_2}) \rangle = \langle \mathbf{u}_1, B(\mathbf{u}, \mathbf{u_2}) \rangle$

for all $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}, \mathbf{v} \in \mathbb{R}^{n+1}$, and $\lambda \in \mathbb{R}$.

Properties B1 and B2 are the bilinearity and symmetry properties of convolution, whereas B3 reflects the property of convolution over $\mathbb{R}^d$ whereby if the autocorrelation function is identically zero, that is, if $v * v = 0$, then v is zero almost everywhere. It turns out that B3 is not satisfied by the Hankel transform implemented here and by Kelley and Montgomery–Pettitt [15] for (3.6)–(3.7) because it is based on the sine transform, and a weaker property akin to B3 does hold, but only on a codimension-two subspace of $\mathbb{R}^{n+1}$. Property B4 holds whenever $B(\mathbf{u}, \mathbf{v}) = \varphi F^T(F\mathbf{u} \otimes F\mathbf{v})$ for some constant φ and some linear mapping F, and then B3 holds for such a B if F is invertible, where here and below we use $\otimes$ to denote the elementwise product of two vectors.

The finite-dimensional analogy of (3.4) under the HNC closure is a single equation for $\boldsymbol{\gamma} = (\gamma_i)_{i=0}^n$:

$$\boldsymbol{\gamma} = \rho B(\mathbf{f} + e^{-\beta \mathbf{U}} G(\gamma)), \mathbf{f} + e^{-\beta \mathbf{U}} G(\boldsymbol{\gamma} - \boldsymbol{\gamma}), \tag{3.8}$$

where $G(\boldsymbol{\gamma}) = -1 + \exp(\boldsymbol{\gamma})$. We shall use $\mathbf{N}_\beta(\boldsymbol{\gamma})$ to denote the operator that appears in the right-hand side of (3.8).

For the particular case of piecewise-constant basis functions used to locate approximate solutions of (3.4), we obtain the convolution operator B_n given by

$$B_n(\mathbf{u}, \mathbf{v}) = H^*((H\mathbf{u}) \otimes (H\mathbf{v})) \tag{3.9}$$

and H is a discrete Hankel transform that is defined elementwise by

$$(H\mathbf{u})_j = \left(\frac{R}{n}\right)^3 \frac{4n}{j^2} \sum_{i=1}^{n-1} \left[ij \sin\left(\frac{\pi ij}{n}\right) u_i\right] \qquad (\text{for } 1 \le j \le n-1)$$

and where $(H\mathbf{u})_0 = 2(H\mathbf{u})_1 - (H\mathbf{u})_2$ and $(H\mathbf{u})_n = 2(H\mathbf{u})_{n-1} - (H\mathbf{u})_{n-2}$. Similarly, H^* is the transform defined by

$$(H^*\mathbf{u})_j = \frac{n}{2R^2 j^2} \sum_{i=1}^{n-1} \left[ij \sin\left(\frac{\pi ij}{n}\right) u_i\right] \qquad (\text{for } 1 \le j \le n-1),$$

and this is also extended to the zeroth and nth coordinates by extrapolation. Of course, by the nature of the Hankel transform one can scale H to satisfy $H^2 = I$, but we have not done this; it is, however, clear that there is a constant φ depending on R and n such that $H^* = \varphi H = \varphi H^T$.

3.2.1. Localized vectors. The practical evaluation of the convolution of two vectors via the form B_n can be evaluated using three FFT-sine transforms, and note that the resulting convolution satisfies Properties B1–B2 above: B1 comes from the linearity of H, and B2 is true by definition. However, B3 is *not* satisfied for the following reason:

$$B_n(\mathbf{u}, \mathbf{u}) = \mathbf{0} \Longrightarrow H^*((H\mathbf{u})^2) = \mathbf{0} \Longrightarrow (H\mathbf{u})^2 \in \ker(H^*) = \operatorname{span}\{\mathbf{e}_0, \mathbf{e}_n\},$$

because both H and H^*, when acting on either of the vectors $\mathbf{e}_0 = (1, 0, \ldots, 0)$ or $\mathbf{e}_n = (0, \ldots, 0, 1)$, yield the zero vector.

As a result, $B_n(\mathbf{u}, \mathbf{u}) = \mathbf{0}$ for all $\mathbf{u} \in \operatorname{span}\{\mathbf{e}_0, \mathbf{e}_n\}$, and so we shall call any element of span $\{\mathbf{e}_0, \mathbf{e}_n\}$ a *localized* vector. Note that H is only a pseudoinverse of H^* in the sense that they are mutual inverses only on the space $\{0\} \times \mathbb{R}^{n-1} \times \{0\}$, the orthogonal complement of the space of localized vectors.

3.3. Algorithm implementation. Any smooth system of equations of the form (1.6), (3.4), or (3.8) is amenable to a Picard iteration algorithm for small values of ρ, and so we use this approach to locate a base solution on a solution branch. In the vicinity of a bifurcation, both Picard- and Newton-based iteration methods will fail to converge and one must modify the system to be solved in order to account for this instability. We have therefore adopted the well-known pseudo arc-length (PAL) strategy due to Keller [14] and others: view a solution branch $\mathcal{B}$ as a one-dimensional manifold that can be parameterized by an arc-length parameter s and augment (1.6) with a constraint in order to obtain a well-posed system of equations.

The system we solve in practice, again with a Newton method, is $F(\boldsymbol{\gamma}, \rho, s) = (0, 0)$, where

$$\mathbf{F}(\boldsymbol{\gamma}, \rho, s) \equiv (\boldsymbol{\gamma} - \rho \mathbf{N}_\beta(\boldsymbol{\gamma}), \mathbf{g}(\boldsymbol{\gamma}, \rho, s)), \tag{3.10}$$

and $\mathbf{g}$ is an additional arc-length constraint. In principle, there are many possible choices for the function $\mathbf{g}$, but the constraint used in [14] is

$$\mathbf{g}(\boldsymbol{\gamma}, \rho, s) = \langle \mathbf{s}_0, \boldsymbol{\gamma} - \boldsymbol{\gamma}_0 \rangle + \sigma_0(\rho - \rho_0) - (s - s_0),$$

where $(\mathbf{s}_0, \sigma_0)$ is the unit tangent vector at a given point $(\gamma_0, \rho_0, s_0) \in \mathcal{B}$, so that $\mathbf{F}(\boldsymbol{\gamma}_0, \rho_0, s_0) = 0$.

The difference $ds := s - s_0$ is a fixed and typically small quantity but it is sometimes necessary to reduce ds if $\mathcal{B}$ nears an apparent spinodal as there can be many fold bifurcations over a small interval of densities. We do this by monitoring the number of Newton iterations required to reduce the residual to a prescribed tolerance and reduce ds by a constant factor if this number is four or greater.

The Newton step of the PAL algorithm requires a linear solver, and we use an unpreconditioned, restarted GMRES(20) algorithm. As discussed in [15], this is appropriate for integral equations posed on a finite domain as the nonlinear convolution operator in the right-hand side of (1.6) can be formulated as a compact mapping in a suitable function space, and this leads to robust convergence properties for GMRES without preconditioning.

3.3.1. Computing loci of fold bifurcations. Since the problem (1.1) and (1.5) contains the parameters ρ and β, the location of a fold bifurcation may be considered to vary as a function of either ρ or β. We have therefore implemented an algorithm to compute the locus of bifurcation points as they change with respect to changes in temperature (or, rather, changes in β) by applying a Newton method to a discretization of the following augmented or codimension-two formulation:

$$\boldsymbol{\gamma} = \rho\, \boldsymbol{N}_\beta(\boldsymbol{\gamma}), \tag{3.11}$$

$$\boldsymbol{k} = \rho\, d_{\boldsymbol{\gamma}} \boldsymbol{N}_\beta(\boldsymbol{\gamma})[\boldsymbol{k}], \tag{3.12}$$

$$1 = \frac{1}{n} \sum_{j=0}^{n} \langle \boldsymbol{k}, \mathbf{e}_j \rangle\,. \tag{3.13}$$

Applying a Newton iteration to (3.11)–(3.12) allows us to locate a locus of fold points parameterized by ρ, noting that (3.11)–(3.12) ask that $\boldsymbol{\gamma}$ solve (3.8), but also that the linearization of the system not be invertible due to the existence of a nonzero null vector, $\boldsymbol{k}$, that appears in (3.12). The integral constraint in (3.13) is present only to ensure that $\boldsymbol{k}$ is nonzero.

Again, we used GMRES(20) as the linear solver for the Newton iteration method applied to (3.11)–(3.13). Moreover, because we were able to locate $(\boldsymbol{\gamma}, \boldsymbol{k}, \beta)$ for each candidate ρ in practice, a property that can be seen in Figure 1.1, we had no need to implement the PAL algorithm on (3.11)–(3.13).

3.4. Other details.

3.4.1. A fourth-order convolution operator. In order to increase the order of approximation of the finite-dimensional convolution operator $B_n(\mathbf{u}, \mathbf{v})$ with respect to the mesh parameter n, we utilized a simple Richardson extrapolation scheme that provides a fourth-order convolution form denoted $B_n^4(\mathbf{u}, \mathbf{v})$ that utilizes six FFT-sine operations and a single cubic interpolation.

Analytic derivatives were used for the discretization of (3.10), but these were calculated numerically using B_n rather than B_n^4, although the map $\mathbf{N}_\beta(\boldsymbol{\gamma})$ itself was

calculated using B_n^4. To form the numerical derivative operators of (3.11)–(3.13) we again used a difference formula based on the convolution operator B_n, rather than its fourth-order extension B_n^4. However, to evaluate (3.11)–(3.13) as a nonlinear operator acting on $(\boldsymbol{\gamma}, \boldsymbol{k}, \rho)$, we did use the fourth-order discrete convolution operator B_n^4. Irrespective of the parameter values, we used a high tolerance of 10^{-3} for GMRES, controlling the arc-length *ds* in such a way that three Newton iterations (or four for some computations) were needed for a nonlinear residual of 10^{-11}.

4. The fundamental property of discrete vapor spinodals. Let us begin this section with a definition.

DEFINITION 3. *A solution branch* $\boldsymbol{\mathcal{C}}$ *of the discrete OZ equation* (3.8) *(with any closure function* G *and any bilinear form* B*) is said to possess a* discrete vapor spinodal *if* $(\rho, \boldsymbol{\gamma}) = (0, \mathbf{0}) \in \boldsymbol{\mathcal{C}}$ *and* $\sup\{\rho : (\rho, \boldsymbol{\gamma}) \in \boldsymbol{\mathcal{C}}\} < \infty$.

Let us now assume a structure on the function G in (3.8) that is chosen to mimic the behavior of the floating-point realization of the exponential function $-1 + e^{\gamma}$:

G1. $G(\gamma) \geq -1$ and there exists an $m > 0$ such that $G(\gamma) = -1$ for all $\gamma \leq -m$,

G2. G is continuous and monotonic increasing, and it is C^1 smooth on $(-m, +\infty)$, and

G3. G is superlinear in the sense that $\lim_{\gamma \to +\infty} G(\gamma)/\gamma = +\infty$.

Assumption G1 effectively states that e^{-m} is indistinguishable from zero if m is sufficiently large and so is not actually satisfied by the choice $G(\gamma) = -1 + e^{\gamma}$ that yields the HNC closure. However, by taking m to be as large as we like, one can find functions G that satisfy G1–G3 and which are as close as we like to the HNC choice. Given the approximate nature of the integral equation theories considered in this paper, perhaps this is not the unreasonable assumption it first appears to be.

Throughout the remainder of the paper, we shall write $\mathtt{g}$ for the inverse of $1 + G$ on the domain $[-m, \infty)$ so that the domain of $\mathtt{g}$ is $[0, \infty)$ with range $[-m, \infty)$ and $\lim_{x \to \infty} \mathtt{g}(x)/x = 0$ by G3.

The following proposition shows that the finite-dimensionality of (3.8) ensures that one cannot observe a divergent-in-norm total correlation function independently of the direct correlation function, quite unlike the continuous case.

PROPOSITION 1. *For fixed* $\beta > 0$ *and any discrete convolution form* B *that satisfies* B1 *and* B2*, there is a connected set* $\mathcal{C} \subset \mathbb{R}^{n+1}$ *of solutions of* (3.8) *such that* $(\boldsymbol{\gamma}, \rho) = (0, 0) \in \mathcal{C}$ *and* $\mathcal{C}$ *is unbounded in the following sense: for each* $M > 0$*, a solution pair* $(\boldsymbol{\gamma}, \rho) \in \mathcal{C}$ *can be found such that* $\|\boldsymbol{\gamma}\| + \rho = M$ *with* $\rho > 0$*. As a result, if* $\mathcal{C}$ *possesses a discrete vapor spinodal, then there is a sequence* $(\boldsymbol{\gamma}_k, \rho_k)$ *and a* $\rho_{\rm sp} \geq 0$ *such that* $\|\boldsymbol{\gamma}_k\| \to \infty$ *and* $\rho_k \to \rho_{\rm sp}$ *as* $k \to \infty$*.*

The discrete total and direct correlation functions $\mathbf{h}_k$ *and* $\mathbf{c}_k$ *corresponding to* $\boldsymbol{\gamma}_k$ *satisfy*

$$\mathbf{h} = \mathbf{c} + \rho_k B(\mathbf{h}, \mathbf{c}), \tag{4.1}$$

$$\mathbf{h} = \mathbf{f} + e^{-\beta \mathbf{U}} G(\mathbf{h} - \mathbf{c}). \tag{4.2}$$

If $\|\boldsymbol{h}_k\| \to \infty$*, then* $\|\boldsymbol{c}_k\| \to \infty$ *and* $\|\boldsymbol{c}_k\|_\infty \sim \|\boldsymbol{h}_k\|_\infty + o(1)$ *as* $k \to \infty$*.*

Finally, suppose B *also satisfies* B4 *and has the property that* $B(\mathbf{e}_0, \mathbf{e}_0) = \mathbf{0}$*. Under these additional assumptions if* $\|\boldsymbol{c}_k\| \to \infty$*, then* $\|\boldsymbol{h}_k\| \to \infty$*.*

Proof. The existence of $\mathcal{C}$ with the stated properties follows immediately from the implicit function theorem and the Leray–Schauder continuation principle (see [25] for example).

Now suppose that $\mathcal{C}$ is a discrete vapor spinodal; then the only solution of (3.8) with $\rho = 0$ is the base solution $\boldsymbol{\gamma} = \mathbf{0}$, and it follows immediately that any solution pair $(\boldsymbol{\gamma}, \rho) \in \mathcal{C}$ with $\boldsymbol{\gamma} \neq \mathbf{0}$ must satisfy $0 < \rho \leq \rho_{\max}$, where $\rho_{\max} := \sup\{\rho \geq 0 : (\boldsymbol{\gamma}, \rho) \in \mathcal{C}\}$. Hence it follows from the unboundedness of $\mathcal{C}$ in the sense stated that the norm of $\boldsymbol{\gamma}$ must diverge along $\mathcal{C}$.

From (4.2) and G1 it follows that $\mathbf{h} \geq -\mathbf{1}$ and that the zeroth component of $\mathbf{h}$ is $\mathbf{h}_0 = -1$ so that (4.2) can be written

$$\mathsf{g}(e^{\beta \mathbf{U}}(\mathbf{h} + \mathbf{1})) = \mathbf{h} - \mathbf{c}, \tag{4.3}$$

ignoring the undefined zeroth component of $e^{\beta \mathbf{U}}(\mathbf{h}+\mathbf{1})$ *for the moment.* Hence, considering the vectors $\mathbf{h}$ and $\mathbf{c}$ and supposing that the ith entry in $\mathbf{h}$ is nonzero, assuming $i \geq 1$, we obtain

$$c_i = h_i - \mathsf{g}(e^{\beta u(r_i)}(h_i + 1)) = h_i \left(1 - \frac{\mathsf{g}(e^{\beta u(r_i)}(h_i + 1))}{h_i}\right),$$

and therefore, if $h_j = \langle \mathbf{e}_j, \mathbf{h} \rangle = \|\mathbf{h}\|_\infty > 1$ and so $j \geq 1$ by the lower uniform bound $\mathbf{h} \geq -\mathbf{1}$, then

$$c_j = \|\mathbf{h}\|_\infty \left(1 - \frac{\mathsf{g}(e^{\beta u(r_j)}(\|\mathbf{h}\|_\infty + 1))}{\|\mathbf{h}\|_\infty}\right),$$

where $\|\cdot\|_\infty$ is the usual maximum norm of a vector. Hence, we may assume by choosing a subsequence if necessary that if $\mathcal{C}$ has a discrete vapor spinodal, then for some fixed, nonzero $j \in \{1, 2, \ldots, n\}$ there results

$$\frac{\|\mathbf{c}\|_\infty}{\|\mathbf{h}\|_\infty} \geq \frac{|c_j|}{\|\mathbf{h}\|_\infty} = \left|1 - \frac{\mathsf{g}(e^{\beta u(r_j)}(\|\mathbf{h}\|_\infty + 1))}{\|\mathbf{h}\|_\infty}\right|,$$

again provided $\|\mathbf{h}\|_\infty > 1$.

Thus, if $(\mathbf{h}_k, \mathbf{c}_k)$ is a sequence of solutions of (4.1)–(4.2) and $\|\mathbf{h}_k\| \to \infty$ as $k \to \infty$ in any norm, then $\|\mathbf{h}_k\|_\infty \to \infty$ as all finite-dimensional norms are equivalent. Since $\lim_{x\to\infty} \mathsf{g}(x)/x = 0$ it is clear that $\|\mathbf{c}_k\|_\infty / \|\mathbf{h}_k\|_\infty \to 1$ as $k \to \infty$ as claimed.

For the last part, suppose, without loss of any generality, that there is an $M > 0$ such that $\|\mathbf{h}_k\|_\infty \leq M$; then (4.3) shows that the jth component of $\mathbf{c}_k$, $|\langle \mathbf{e}_j, \mathbf{c}_k \rangle|$ for $j \geq 1$, is bounded above by

$$|\langle \mathbf{e}_j, \mathbf{c}_k \rangle| \leq M + \max_{-1 \leq x \leq M} g((x+1)e^{\beta \max_{j\geq 1}(u(r_j))}) < \infty, \tag{4.4}$$

which is potentially very large in practice but finite nevertheless.

Thus, the only way in which $\|\mathbf{c}_k\|$ can diverge is if $c_k := \langle \mathbf{e}_0, \mathbf{c}_k \rangle$ diverges as $k \to \infty$. From (4.1) there results

$$\langle \mathbf{e}_0, \mathbf{h}_k \rangle = c_k + \rho_k \langle \mathbf{e}_0, B(\mathbf{h}_k, \mathbf{c}_k) \rangle$$

so that

$$|c_k| \leq |\langle \mathbf{e}_0, \mathbf{h}_k \rangle| + \rho_{\max} |\langle \mathbf{e}_0, B(\mathbf{h}_k, \mathbf{c}_k) \rangle| \leq |-1| + \rho_{\max} |\langle \mathbf{h}_k, B(\mathbf{c}_k, \mathbf{e}_0) \rangle|.$$

Writing $\mathbf{c}_k = c_k \mathbf{e}_0 + \mathbf{c}_k^\perp$, where $\langle \mathbf{e}_0, \mathbf{c}_k^\perp \rangle \equiv 0$, we obtain

$$|c_k| \leq 1 + \rho_{\max} |\langle \mathbf{h}_k, B(c_k \mathbf{e}_0 + \mathbf{c}_k^\perp, \mathbf{e}_0) \rangle| \leq 1 + \rho_{\max} |\langle \mathbf{h}_k, B(\mathbf{c}_k^\perp, \mathbf{e}_0) \rangle|.$$

As a result, we finally obtain a bound on $|c_k|$ because of the existence of the pointwise upper bound (4.4) on $\mathbf{c}_k^\perp$, and so the final claim holds. ∎

We can now prove the following.

THEOREM 2. *Suppose that* G1–G3 *are satisfied by* G, *that* $B = B_n$, *and that the resulting equation* (3.8) *has a solution branch* $\mathcal{C}$ *with a discrete vapor spinodal. Take any sequence of solutions* $(\mathbf{h}_k, \boldsymbol{c}_k)$ *of* (4.1)–(4.2) *with density values* ρ_k *corresponding to solutions from* $\mathcal{C}$ *such that* $\|\mathbf{h}_k\| \to \infty$ *and* $(\mathbf{h}_k)$ *does not* localize at the cut-off *in the sense that*

$$\liminf_{k\to\infty} \frac{|\langle \mathbf{e}_j, \mathbf{h}_k\rangle|}{\|\mathbf{h}_k\|} > 0 \qquad \text{for some } j \in \{1,\dots,n-1\}.$$

Assuming also that $\langle \mathbf{e}_0, \mathbf{c}_k\rangle$ *is a bounded sequence, if* $\rho_k \to \rho_{\rm sp}$ *as* $k \to \infty$, *then* $\rho_{\rm sp} = 0$.

Proof. Let us begin by reformulating (4.1)–(4.2) by writing $\mathbf{c} = c_0\mathbf{e}_0 + \mathbf{c}^\perp$ and $\mathbf{h} = h_0\mathbf{e}_0 + \mathbf{h}^\perp$, where $\mathbf{c}^\perp, \mathbf{h}^\perp \in \text{span}\{\mathbf{e}_0\}^\perp$. Then (4.1) becomes

$$c_0\mathbf{e}_0 + \mathbf{c}^\perp = h_0\mathbf{e}_0 + \mathbf{h}^\perp + \rho B(c_0\mathbf{e}_0 + \mathbf{c}^\perp, h_0\mathbf{e}_0 + \mathbf{h}^\perp),$$

and projecting this onto $\text{span}\{\mathbf{e}_0\}$ gives

$$\begin{aligned} c_0 &= h_0 - \rho\left\langle \mathbf{e}_0, B(c_0\mathbf{e}_0 + \mathbf{c}^\perp, h_0\mathbf{e}_0 + \mathbf{h}^\perp)\right\rangle \\ &= h_0 - \rho\left\langle B(\mathbf{e}_0, c_0\mathbf{e}_0 + \mathbf{c}^\perp), h_0\mathbf{e}_0 + \mathbf{h}^\perp\right\rangle \quad \text{(using B1 and B4)} \\ &= h_0 - \rho\left\langle B(\mathbf{e}_0, \mathbf{c}^\perp), h_0\mathbf{e}_0 + \mathbf{h}^\perp\right\rangle \\ &= h_0 - \rho\left\langle \mathbf{c}^\perp, B(\mathbf{e}_0, h_0\mathbf{e}_0 + \mathbf{h}^\perp)\right\rangle \quad (\text{since } B_n(\mathbf{e}_0, \mathbf{e}_0) = \mathbf{0}) \\ &= h_0 - \rho\left\langle \mathbf{c}^\perp, B(\mathbf{e}_0, \mathbf{h}^\perp)\right\rangle = h_0 - \rho\left\langle \mathbf{e}_0, B(\mathbf{c}^\perp, \mathbf{h}^\perp)\right\rangle. \end{aligned}$$

However, from (4.2), $h_0 = -1$ as $(e^{-\beta\mathbf{U}})_0$ is interpreted as zero, whence

$$c_0 = -1 - \rho\left\langle \mathbf{e}_0, B(\mathbf{c}^\perp, \mathbf{h}^\perp)\right\rangle,$$

which defines a bilinear functional of $\mathbf{c}^\perp$ and $\mathbf{h}^\perp$ that depends affinely on ρ.

Now (4.1) can be projected onto $\text{span}\{\mathbf{e}_0\}^\perp$ with projector Π to yield

$$\mathbf{h}^\perp = \mathbf{c}^\perp + \rho\Pi B(c_0(\rho, \mathbf{c}^\perp, \mathbf{h}^\perp)\mathbf{e}_0 + \mathbf{c}^\perp, -\mathbf{e}_0 + \mathbf{h}^\perp), \tag{4.5}$$

$$\mathbf{h}^\perp = -\mathbf{1}^\perp + e^{-\beta\mathbf{U}^\perp} G(\mathbf{h}^\perp - \mathbf{c}^\perp), \tag{4.6}$$

where the advantage of this formulation is that $e^{-\beta\mathbf{U}^\perp}$ is strictly positive for any intermolecular potential, thus removing the problematic singularity at $r = 0$. Here, we are using a superscript $\perp$, as in $\mathbf{v}^\perp$, to denote the projection $\Pi(\mathbf{v})$. As a result, the singularity in (4.6) has been removed and (4.6) is now equivalent to

$$\mathbf{c}^\perp = \mathbf{h}^\perp - \mathbf{g}(e^{\beta\mathbf{U}^\perp}(\mathbf{h}^\perp + \mathbf{1}^\perp)). \tag{4.7}$$

Returning to the sequence of solutions from $\mathcal{C}$, define new sequences

$$P_k := \mathbf{h}_k^\perp/\|\mathbf{h}_k\| \quad \text{and} \quad Q_k := \mathbf{c}_k^\perp/\|\mathbf{h}_k\|,$$

as defined in the statement of the theorem; then

$$\frac{P_k - Q_k}{\|\mathbf{h}_k\|} = \rho_k \Pi B \left(\frac{\langle \mathbf{e}_0, \mathbf{c}_k \rangle \mathbf{e}_0}{\|\mathbf{h_k}\|} + Q_k, -\frac{\mathbf{e}_0}{\|\mathbf{h}_k\|} + P_k \right),$$

and

$$Q_k = P_k - \frac{\mathbf{g}(e^{\beta \mathbf{U}^\perp}(\mathbf{h}_k^\perp + \mathbf{1}^\perp))}{\|\mathbf{h}_k\|}. \tag{4.8}$$

Using standard compactness properties, one can find P' of unit norm such that $P_k \to P'$, and therefore $Q_k \to P'$ by (4.8) using the sublinearity of $\mathbf{g}$. As a result, we find that

$$\rho_k \Pi B \left(\frac{\langle \mathbf{e}_0, \mathbf{c}_k \rangle}{\|\mathbf{h_k}\|} + P', P' \right) \to 0$$

as $k \to \infty$.

By hypothesis we can find a $\rho_{\text{sp}} \geq 0$ such that by extracting a further subsequence if necessary $\rho_{\text{sp}} \Pi B\,(P', P') = \mathbf{0}$. If $\rho_{\text{sp}} > 0$ then $\Pi B\,(P', P') = \mathbf{0}$ follows and so $B\,(P', P') = \alpha \mathbf{e}_0$. Recalling that $B = B_n$ in the statement of the theorem, $H^*(HP')^2 = \alpha \mathbf{e}_0$ whence $\alpha = 0$ by the fact that $H\mathbf{e}_0 = \mathbf{0}$. Hence $(HP')^2 = \alpha \mathbf{e}_0$ for some $\alpha \in \mathbb{R}$, so that $HP' = \sqrt{\alpha}\mathbf{e}_0$ giving $\alpha = 0$ as H is self-adjoint, so that finally $P' \in \text{span}\{\mathbf{e}_0, \mathbf{e}_n\}$. By construction of the sequence (P_k), it follows that $P' \in \text{span}\{\mathbf{e}_n\}$ which states that $\mathbf{h}_k$ localizes at the cut-off, contradicting an assumption of the theorem. □

From Theorem 2 we have the following trichotomy in terms of discrete vapor spinodals when using the convolution form $B = B_n$ in (4.1)–(4.2). If $\boldsymbol{\mathcal{C}}$ is that discrete vapor spinodal, at least one of the following applies along a sequence $(\mathbf{h}_k, \mathbf{c}_k, \rho_k) \in \boldsymbol{\mathcal{C}}$:

T1. Density ρ_k heads to zero,

T2. $\langle \mathbf{e}_0, \mathbf{c}_k \rangle$ is unbounded, or

T3. $\mathbf{h}_k / \|\mathbf{h}_k\|$ (up to a subsequence) tends toward the localized vector $\mathbf{e}_n$.

Note that each of T1, T2, and T3 represents a different, nonphysical possibility, and both T1 and T3 have been observed in practice when G takes the HNC form, even though G1 does not apply in that context. On the other hand, we have not observed T2 in practice. It is an immediate corollary of T1 that (4.1)–(4.2) has a fold bifurcation in the sense of Definition 2 at some value of the density parameter, and so one should expect to observe a folded geometry of isothermal vapor solution branches at subcritical temperatures.

4.1. Qualitatively correct convolution forms. The T1–T3 trichotomy again raises the central question of the choice of convolution form B in (3.8). There are advantages in defining B by

$$B(\mathbf{u}, \mathbf{v}) = F^T(F\mathbf{u} \otimes F\mathbf{v}) \quad \text{such that } FF^T = I \tag{4.9}$$

for some linear transformation F for the following reason. Such a choice for B allows one to introduce a *qualitatively correct* discrete compressibility functional, χ_d^{-1}, say, for (4.1)–(4.2) (with any nonlinearity G) by setting

$$\chi_d^{-1} = 1 - \rho \langle \mathbf{e}_0, F\mathbf{c} \rangle .$$

By the term *qualitatively correct*, we mean that unlike (4.1)–(4.2) which is realized on setting $B = B_n$, using instead the convolution form B which is based on such

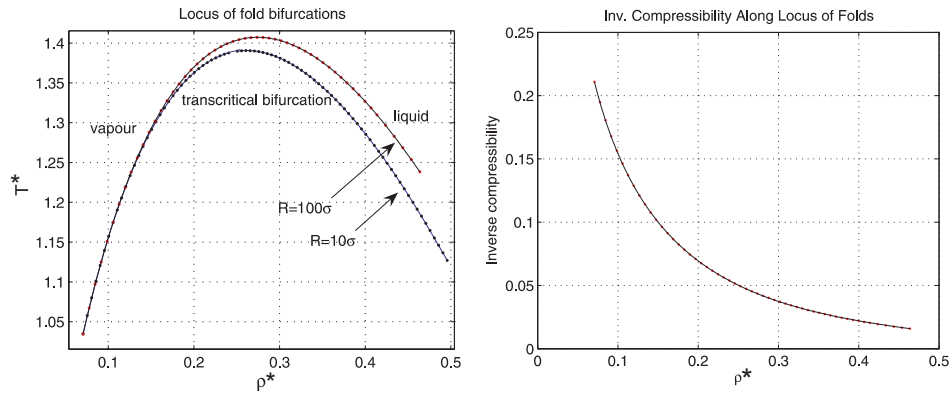


FIG. 5.1. *(Left) A comparison of two computed bifurcation loci for two values of R, namely,* 10σ *and* 100σ*. (Right) For the* $R = 100\sigma$ *case, inverse compressibility is evaluated along the liquid and vapor branches; note that this is never zero.*

a transformation F in (4.9) ensures that there is no solution of (4.1)–(4.2) that has zero inverse compressibility. This means that there is a version of Lemma 2.1 that holds when the convolution form B is based on (4.9), but Lemma 2.1 fails to hold when B_n is the chosen convolution form. We shall report the results of an analysis that results from the implemenation of a qualitatively correct convolution form in a separate publication.

5. Computations.

5.1. The LJ 6-12 potential. Throughout this section we use the Lennard–Jones potential and set $\sigma = 1, \epsilon = 1$, with the cut-off R chosen to be 100σ unless stated otherwise, and we have used a numerical mesh of $2^{13} + 1$ points.

1. Figure 5.1 (left) is a comparison of two codimension-two bifurcation loci in the (ρ, T)-plane that shows a significant discrepancy between the use of two choices of cut-off parameter R. The right-hand figure shows the values of inverse compressibility obtained along the loci from the left-hand curve for the $R = 100\sigma$ case. For *true spinodals*, which we gleaned from our previous analysis should not occur in the computations, this curve should be identically zero; clearly it is not.

One may notice that the curve in Figure 5.1 (left) lies slightly above that reported by others, such as [17, p. 833] and Figure 3 of [3]. The reasons for this are that the cut-off parameter R in the stated references is different from ours, and we have generally used a different number of points for our discretization than the cited articles. Our data may be slightly different also because we have computed the location of bifurcation points, rather than relying on the extrapolation of a numerical algorithm in order to estimate their location.

2. Figure 5.2 shows a computed unfolding of the transcritical bifurcation from the apex of Figure 5.1 (left), where nine different temperature values have been used to compute three subcritical isothermal solution branch pairs and six supercritical vapor isotherms. The appearance of several fold bifurcations can be seen on both liquid (high-density) and vapor (low-density) branches, and note the apparent existence of a locus of solutions of (3.6)–(3.7) with zero inverse compressibility.

3. Shown in Figure 5.3 (left) are the results of a computation undertaken with a *high* temperature at $\beta = 1$ and with $R = 120\sigma$; note the large number of fold bifurcations that seem to accumulate in the vicinity where inverse compressibility is

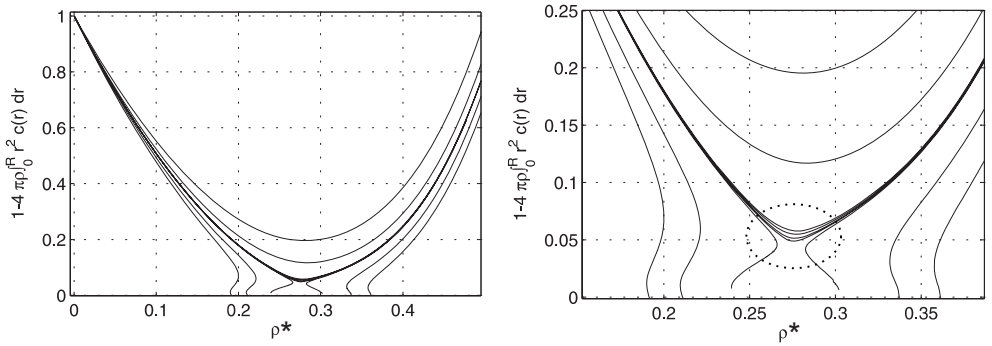


FIG. 5.2. *(Left) Inverse compressibility against density. (Right) Zoom of left-hand plot showing positive values of inverse compressibility at the fold points. Solutions with zero inverse compressibility are clearly seen but not* at *the fold bifurcations.*

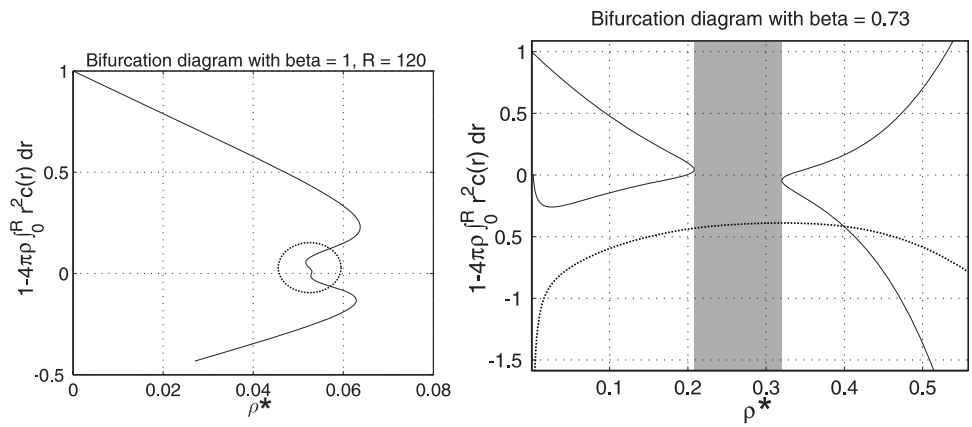


FIG. 5.3. *(Left) A solution branch with several bifurcations showing inverse compressibility against density with* $R = 120$ *and* $(1/T^* =)\beta = 1$. *(Right) Three solution branches and bifurcations with inverse compressibility plotted against density at* $R = 120$ *and* $\beta = 0.73$. *An apparent region (the shaded region) that contains no solutions is filled by a solution branch with negative compressibility.*

zero. In the right-hand figure we have located a branch of solutions at $\beta = 0.73$ that spans a shaded region where no solution is believed to exist, although this particular branch contains only solutions that have negative compressibility. Other branches of solutions have been found before within this region [21], but they were complex-valued and their presence was a simple consequence of the existence of fold bifurcations. We have not been able to locate any *real* solutions in this region with positive compressibility; that is not to say that none are there, but, unlike vapor isotherms, we have no systematic method for locating solutions in this region.

5.2. Comparison with the double Yukawa potential. For all of the following computations we used a numerical mesh with $2^{12} + 1$ points with the following double Yukawa (DY) intermolecular potential for various values of the cut-off R:

$$u_{DY}(r) = \frac{\epsilon}{r}\left(A_1 e^{-z_1(r-\sigma)} - A_2 e^{-z_2(r-\sigma)}\right),$$

where σ is particle diameter, and $A_1 = 1.6438\sigma$, $z_1 = 14.7\sigma^{-1}$, $A_2 = 2.03\sigma$, and $z_2 = 2.69\sigma^{-1}$ are the parameter values used in [8, 20]. This intermolecular potential

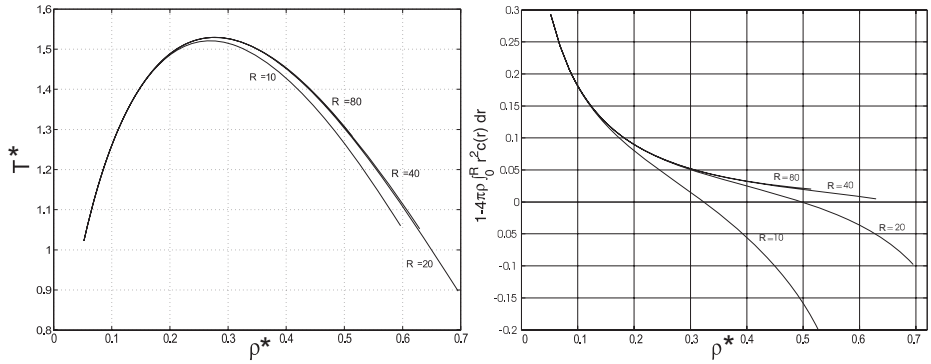


FIG. 5.4. *(Left) Locus of fold bifurcations for the DY potential with varied cut-offs and* 4096 *mesh points. (Right) Values of inverse compressibility along the locus for* $R = 10\sigma, \ldots, 80\sigma$.

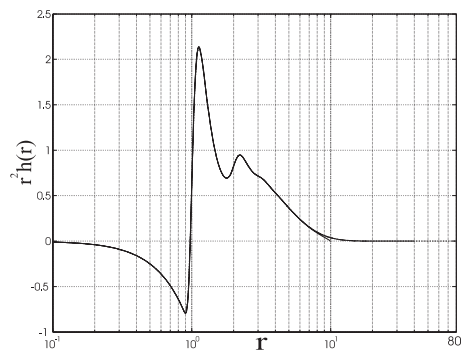


FIG. 5.5. *(Left) A plot of the low-density correlation function* $r^2h(r)$ *along the bifurcation locus from Figure* 5.4 *(left) at* $(\rho^*, T^*) = (0.085, 1.2)$ *for cut-off values* $R = 10\sigma, 20\sigma, 40\sigma$, *and* $R = 80\sigma$. *(Right) The high-density correlation function* $r^2h(r)$ *along the same bifurcation locus at* $(\rho^*, T^*) = (0.597, 1.06)$ *with* $R = 80\sigma$.

is commonly used as it has more rapid decay than LJ, with a repulsive soft core and attractive tail, which, it is believed, should lead to greater stability in computations. However, the property associated with the HNC closure of having L^1-divergence of the total correlation with L^2-boundedness at a spinodal is totally unaffected by the choice of potential and the decay rate of its tail.

1. Figure 5.4 is a repetition of Figure 5.1 for the DY potential for varying cut-offs: $R = 10\sigma, 20\sigma, 40\sigma$, and 80σ. The critical points at the apex of the locus in Figure 5.4 (left) are all close to $\rho_c = 0.27$ at $T_c = 1.52$. The curves corresponding to different cut-off points are in good agreement up to the apex but differ slightly beyond it; however, the values of inverse compressibility do vary somewhat with respect to the cut-off as shown in Figure 5.4 (right).

2. Figure 5.5 (left) illustrates a computed thermodynamic state with large compressibility that is located in the relatively low-density vapor Figure 5.4 (left), specifically at $(\rho^*, T^*) = (0.085, 1.2)$. Solutions have been computed with $R = 10\sigma, 20\sigma, 40\sigma$, and 80σ, and note that solutions at cut-offs higher than $R = 10\sigma$ are visually indistinguishable. A thermodynamic state from the high-density liquid region of Figure 5.4 (left) is shown in Figure 5.5 (right) for comparison.

Acknowledgments. Many thanks are due to Jim Henderson and Andrew Parry for their invaluable input to this work.

REFERENCES

[1] R. J. Baxter, *Percus-Yevick equation for hard spheres with surface adhesion*, J. Chem. Phys., 49 (1968), pp. 2770–2774.
[2] L. Belloni, *Inability of the hypernetted chain integral equation to exhibit a spinodal line*, J. Chem. Phys., 98 (1993), pp. 8080–8095.
[3] J. Brey and A. Santos, *Absence of criticality in the hypernetted chain equation for a truncated potential*, Mol. Phys., 57 (1986), pp. 149–160.
[4] J. J. Brey and A. Santos, *About the numerical solution of the Percus-Yevick equation in the critical region for nontruncated potentials*, J. Chem. Phys., 79 (1983), pp. 4652–4653.
[5] J. J. Brey, A. Santos, and F. Romero, *The critical region in the Percus-Yevick approximation. A numerical study for a Lennard-Jones potential*, J. Chem. Phys., 77 (1982), pp. 5058–5064.
[6] C. Caccamo, *Integral equation theory description of phase equilibria in classical fluids*, Phys. Rep., 274 (1996), pp. 1–105.
[7] P. T. Cummings and P. A. Monson, *Solution of the Ornstein-Zernike equation in the vicinity of the critical-point of a simple fluid*, J. Chem. Phys, 82 (1985), pp. 4303–4311.
[8] S. M. Foiles and N. W. Ashcroft, *Variational theory of phase separation in binary liquid mixtures*, J. Chem. Phys., 75 (1981), pp. 3594–3598.
[9] L. J. D. Frink and A. G. Salinger, *Rapid analysis of phase behavior with density functional theory.* II. *Capillary condensation in disordered porous media*, J. Chem. Phys., 118 (2003), pp. 7466–7476.
[10] L. J. D. Frink, A. G. Salinger, M. P. Sears, J. D. Weinhold, and A. L. Frischknecht, *Numerical challenges in the application of density functional theory to biology and nanotechnology*, J. Phys. Condens. Matter, 14 (2002), pp. 12167–12187.
[11] F. Gallerani, G. L. Vecchio, and L. Reatto, *Comment on the critical behavior in the Percus-Yevick equation*, Phys. Rev. A, 32 (1985), pp. 2526–2529.
[12] J. Hansen and I. Macdonald, *Theory of Simple Liquids*, Academic Press, London, 1976.
[13] J. R. Henderson and Z. A. Sabeur, *Liquid-state integral equations at high density: On the mathematical origin of infinite-range oscillatory solutions*, J. Chem. Phys., 97 (1992), pp. 6750–6758.
[14] H. B. Keller, *Numerical Solution of Bifurcation and Nonlinear Eigenvalue Problems*, P. Rabinowitz, ed., Academic Press, New York, 1977.
[15] C. T. Kelley and B. Montgomery-Pettitt, *A fast solver for the Ornstein–Zernike equations*, J. Comput. Phys., 197 (2004), pp. 491–501.
[16] R. J. F. Leote, R. Evans, D. C. Hoyle, and J. R. Henderson, *The decay of the pair correlation function in simple fluids: Long versus short-ranged potentials*, J. Phys. Condens. Matter, 6 (1994), pp. 9275–9294.
[17] E. Lomba and J. Lopez-Martin, *On the solutions of the hypernetted-chain equation inside the gas-liquid coexistence region*, J. Statist. Phys., 80 (1995), pp. 825–839.
[18] L. S. Ornstein and F. Zernike, *Accidental deviations of density and opalescence at the critical point of a single substance*, Proc. Akad. Sci. (Amsterdam), 17 (1914), pp. 793–806.
[19] J. K. Percus and G. J. Yevick, *Analysis of classical statistical mechanics by means of collective coordinates*, Phys. Rev. (2), 110 (1958), pp. 1–13.
[20] P. D. Poll and N. W. Ashcroft, *Critical behaviour of the hypernetted chain equation*, Phys. Rev. A, 35 (1987), pp. 5167–5173.
[21] G. Sarkisov and E. Lomba, *The gas-liquid phase-transition singularities in the framework of the liquid-state integral equation formalism*, Chem. Phys., 122 (2005), pp. 1–6.
[22] D. A. Tikhonov and G. N. Sarkisov, *Singularities of solution of the Ornstein–Zernike equation within a gas–liquid transition region*, Russian J. Phys. Chem., 74 (2000), pp. 470–476.
[23] J. M. J. van Leeuwen, J. Groeneveld, and J. D. Boer, *New method for the calculation of the pair correlation function*, Physica, 25 (1959), pp. 792–808.
[24] R. O. Watts, *Hypernetted-chain approximation applied to a modified Lennard-Jones fluid*, J. Chem. Phys., 50 (1969), pp. 1358–1365.
[25] E. Zeidler, *Nonlinear Functional Analysis and Its Applications* I: *Fixed Point Theorems*, Springer-Verlag, New York, 1986.

SIAM J. SCI. COMPUT.
Vol. 29, No. 6, pp. 2464–2493
© 2007 Society for Industrial and Applied Mathematics

FAST COMPUTATION OF FOURIER INTEGRAL OPERATORS*

EMMANUEL CANDÈS†, LAURENT DEMANET‡, AND LEXING YING§

Abstract. We introduce a general purpose algorithm for rapidly computing certain types of oscillatory integrals which frequently arise in problems connected to wave propagation, general hyperbolic equations, and curvilinear tomography. The problem is to numerically evaluate a so-called Fourier integral operator (FIO) of the form $\int e^{2\pi i\Phi(x,\xi)}a(x,\xi)\,\hat{f}(\xi)\mathrm{d}\xi$ at points given on a Cartesian grid. Here, ξ is a frequency variable, $\hat{f}(\xi)$ is the Fourier transform of the input f, $a(x,\xi)$ is an amplitude, and $\Phi(x,\xi)$ is a phase function, which is typically as large as $|\xi|$; hence the integral is highly oscillatory. Because a FIO is a dense matrix, a naive matrix vector product with an input given on a Cartesian grid of size N by N would require $O(N^4)$ operations. This paper develops a new numerical algorithm which requires $O(N^{2.5}\log N)$ operations and as low as $O(\sqrt{N})$ in storage space (the constants in front of these estimates are small). It operates by localizing the integral over polar wedges with small angular aperture in the frequency plane. On each wedge, the algorithm factorizes the kernel $e^{2\pi i\Phi(x,\xi)}a(x,\xi)$ into two components: (1) a diffeomorphism which is handled by means of a nonuniform FFT and (2) a residual factor which is handled by numerical separation of the spatial and frequency variables. The key to the complexity and accuracy estimates is the fact that the separation rank of the residual kernel is *provably independent of the problem size*. Several numerical examples demonstrate the numerical accuracy and low computational complexity of the proposed methodology. We also discuss the potential of our ideas for various applications such as reflection seismology.

Key words. Fourier integral operators, generalized Radon transform, separated representation, nonuniform fast Fourier transform, matrix approximation, operator compression, randomized algorithms, reflection seismology

AMS subject classifications. 35S30, 65F30, 86A15

DOI. 10.1137/060671139

1. Introduction.

This paper introduces a general purpose algorithm to compute the action of linear operators which are frequently encountered in analysis and scientific computing. These operators take the form

$$(Lf)(x) = \int_{\mathbb{R}^d} a(x,\xi)e^{2\pi i\Phi(x,\xi)}\hat{f}(\xi)\,\mathrm{d}\xi, \tag{1.1}$$

where $\Phi(x,\xi)$ is a phase function that is smooth in (x,ξ) for $\xi \neq 0$ and obeys the homogeneity relation $\Phi(x,\lambda\xi) = \lambda\Phi(x,\xi)$ for λ positive, and $a(x,\xi)$ is a smooth amplitude term. As is standard, $\hat{f}$ is the Fourier transform of f defined by

$$\hat{f}(\xi) = \int_{\mathbb{R}^d} f(x)e^{-2\pi ix\xi}\,\mathrm{d}x. \tag{1.2}$$

*Received by the editors September 30, 2006; accepted for publication (in revised form) April 9, 2007; published electronically October 17, 2007. This work was partially supported by an NSF grant CCF-0515362 and a DOE grant DE-FG03-02ER25529.

http://www.siam.org/journals/sisc/29-6/67113.html

†Applied and Computational Mathematics, California Institute of Technology, Pasadena, CA 91125 (ecandes@acm.caltech.edu).

‡Department of Mathematics, Stanford University, Stanford, CA 94305 (laurent@math.stanford.edu).

§Department of Mathematics, University of Texas at Austin, Austin, TX 78712 (lexing@math.utexas.edu).

With the proper regularity assumptions on the amplitude and the phase,[1] (1.1) defines a class of oscillatory integrals known as Fourier integral operators (FIOs). FIOs are the subject of considerable study for many of the operators encountered in physics and other fields. For instance, most differential and pseudodifferential operators are FIOs. Convolutions and multiplications by smooth functions are FIOs. Some "principal value" integrals are FIOs. And the list goes on.

An especially important example of a FIO is the solution operator to the free-space wave equation in $\mathbb{R}^d$, $d > 1$,

$$\frac{\partial^2 u}{\partial t^2}(x,t) = c^2 \Delta u(x,t), \tag{1.3}$$

with initial conditions $u(x,0) = u_0(x)$ and $\frac{\partial u}{\partial t}(x,0) = 0$, say. Everyone knows that for constant speeds, the Fourier transform decouples the different frequency components of u. Each Fourier component obeys an ordinary differential equation which can be solved explicitly. The solution $u(x,t)$ is the superposition of these Fourier modes and is given by

$$u(x,t) = \frac{1}{2}\left(\int e^{2\pi i(x\cdot\xi + c|\xi|t)} \hat{u_0}(\xi) \mathrm{d}\xi + \int e^{2\pi i(x\cdot\xi - c|\xi|t)} \hat{u_0}(\xi) \mathrm{d}\xi \right). \tag{1.4}$$

The connection is now clear: the solution operator is the sum of two FIOs with phase functions

$$\Phi_{\pm}(x,\xi) = x\cdot\xi \pm c|\xi|t.$$

For variable but reasonably smooth sound speeds $c(x)$, the solution operator is for small times a sum of two FIOs with more complicated phases and amplitudes. In particular, the phase can be constructed from the optical traveltime in a medium with index of refraction $1/c(x)$; see [11] for details.

An important property of FIOs is that they *displace* wavefront sets and singular supports (where the solution is singular), which is the favored mathematical way of formulating propagation of singularities along characteristic manifolds for hyperbolic equations [20]. For this reason, the solution operators for heat equations or Schrödinger equations are not FIOs, because these equations *diffuse* or *disperse* wavefronts. More precisely, their solution operators could potentially be put in the form (1.1), but the homogeneity relation $\Phi(x,\lambda\xi) = \lambda\Phi(x,\xi)$ would be lost.

In short, it is useful to think of FIOs as proxies for the solution operator to large classes of hyperbolic differential equations.

1.1. FIO computations. Numerical simulation of free wave propagation with constant sound speed is straightforward. As long as the solution $u(x,t)$ is sufficiently well localized both in space and frequency, it can be computed accurately and rapidly by applying the sequence of steps below.

1. Compute the fast Fourier transform (FFT) of u_0.
2. Multiply the result by $e^{\pm 2\pi i c|\xi|t}$, and sum as in (1.4).

[1]The amplitude should be C^∞ and obey (2.3). The phase should be C^∞ except at $\xi = 0$. In general, we do *not* require the nondegeneracy condition $\det(\frac{\partial^2 \Phi}{\partial x_i \partial \xi_j}) \neq 0$ if the only goal is to apply the FIO (note that such a FIO may not be invertible). With this being said, we can handle only canonical relations that are locally the graph of a function; i.e., we do not deal with multivalued phases.

3. Compute the inverse FFT.

Of course, this works only in the very special case where the amplitude a is independent of x, and where the phase is of the form $x \cdot \xi$ plus a function of ξ alone. Expressed differently, this works when the FIO is shift-invariant so that it is diagonal in the Fourier basis. Note that there is in general no formula for the eigenfunctions when Φ or a depend on x. Computing these eigenfunctions on the fly is out of the question when the objective is merely to compute the action of the operator. (Note that even if the spectral decomposition of the operator were available, it is not clear how one would use it to speed up computations.)

The object of this paper is to find an algorithm that is considerably faster than evaluating (1.1) by direct quadratures and is yet suited to handle large classes of phases and amplitudes. Most of the existing fast summation techniques rely on either the nonoscillatory behavior (such as wavelet-based techniques [8]) or the existence of a low rank approximation (fast multipole methods [26], hierarchical matrices [27], pseudodifferential separation [4]). The difficulty here is that the kernel $e^{2\pi i\Phi(x,\xi)}$ is highly oscillatory and does not have a low rank separated approximation. Therefore, all the modern techniques are not directly applicable.

The main claim of this paper, however, is that there is a way to decompose the operator into a sum of components for which the oscillations are well understood and low rank representations are available. In addition, the number of such components is reasonably small, which paves the way to faster algorithms. Before expanding on this idea, we first explain the discretization of the operator (1.1).

1.2. Discretization. For simplicity, we restrict our attention in this paper to the two-dimensional case $d = 2$. The main ideas apply readily to the higher dimensions, though the analysis and implementation would be more involved.

Just as the discrete Fourier transform is the digital analogue of the continuous Fourier transform, one can also introduce discrete FIOs. Given a function f defined on a Cartesian grid $X = \{x = (\frac{n_1}{N}, \frac{n_2}{N}), 0 \le n_1, n_2 < N, \text{ and } n_1, n_2 \in \mathbb{Z}\}$, we simply define the discrete FIO by

$$(Lf)(x) := \frac{1}{N} \sum_{\xi \in \Omega} a(x,\xi) e^{2\pi i \Phi(x,\xi)} \hat{f}(\xi) \tag{1.5}$$

for every $x \in X$. (We are sorry for overloading the symbol L to denote both the discrete and continuous objects but there will be no confusion in what follows.) The summation above is taken over all $\Omega = \{\xi = (n_1, n_2), -\frac{N}{2} \le n_1, n_2 < \frac{N}{2}, \text{ and } n_1, n_2 \in \mathbb{Z}\}$, and throughout this paper, we will assume that N is an even integer. Here and below, $\hat{f}$ is the discrete Fourier transform (DFT) of f and is defined as

$$\hat{f}(\xi) = \frac{1}{N} \sum_{x \in X} e^{-2\pi i x \cdot \xi} f(x). \tag{1.6}$$

The normalizing constant $\frac{1}{N}$ in (1.5) (resp., (1.6)) ensures that L (resp., the DFT) is a discrete isometry in the case where $\Phi(x,\xi) = x \cdot \xi$.

The formula (1.5) turns out to be an accurate discretization of (1.1) as soon as f obeys standard localization estimates both in space and frequency. A justification of this fact, however, would go beyond the scope of this paper and is omitted. In the remainder of the paper, we will take (1.5) as the quantity we wish to compute once we are given a phase and an amplitude function.

The parameter N measures the size and difficulty of the computational problem. In a nutshell, it corresponds to the number of points which are needed in each direction to accurately sample the continuous object $f(x)$. This is the reason why N will be a central quantity throughout the rest of paper.

As mentioned earlier, the straightforward method for computing (1.5) simply evaluates the summation independently for each x. Since each sum takes $O(N^2)$ operations and there are N^2 grid points in X, this strategy requires $O(N^4)$ operations. When N is moderately large, this can be prohibitive. This paper describes a novel algorithm which computes all the values of $Lf(x)$ for $x \in X$ with high accuracy in $O(N^{2.5} \log N)$ operations. The only requirement is that the amplitude and the phase obey mild smoothness conditions, which are in fact standard. As a matter of fact, our algorithm can be applied even to the cases where the nondegeneracy condition $\det\left(\frac{\partial^2 \Phi}{\partial x_i \partial \xi_j}\right) \neq 0$ does not hold.

1.3. Separation within angular wedges. This section outlines the main idea of the paper. Let $\arg \xi$ be the angle between ξ and the horizontal vector $(1, 0)$, and partition the frequency domain into a family of angular wedges $\{W_\ell\}$ defined by

$$W_\ell = \{\xi : (2\ell - 1)\pi/\sqrt{N} \leq \arg \xi < (2\ell + 1)\pi/\sqrt{N}\}$$

for $0 \leq \ell < \sqrt{N}$ (assume $\sqrt{N}$ is an integer). An important property of these wedges is that each W_ℓ satisfies the parabolic relationship

$$\textit{length} \simeq \textit{width}^2, \tag{1.7}$$

up to multiplicative constants independent of N. There are $O(\sqrt{N})$ such wedges, as illustrated in Figure 1.

For each wedge W_ℓ, we let χ_ℓ be the indicator function of W_ℓ. Similarly, we denote by $\hat{\xi}_\ell$ the unit vector pointing to the center direction of W_ℓ

$$\hat{\xi}_\ell = \left(\cos \frac{2\ell\pi}{\sqrt{N}}, \sin \frac{2\ell\pi}{\sqrt{N}}\right).$$

It follows from the identify $\sum_\ell \chi_\ell(\xi) = 1$ that one can decompose the operator L as $\sum_\ell L_\ell$, where

$$(L_\ell f)(x) = \frac{1}{N} \sum_\xi a(x, \xi) e^{2\pi i \Phi(x,\xi)} \chi_\ell(\xi) \hat{f}(\xi).$$

Within each wedge W_ℓ, we can perform a Taylor expansion of $\Phi(x, \xi)$ in the second variable, around the point $\hat{\xi}_\ell |\xi|$. There is a point $\xi^\star$ which belongs to the line segment $[\hat{\xi}_\ell |\xi|, \xi]$ such that

$$\Phi(x, \xi) = \Phi(x, \hat{\xi}_\ell |\xi|) + \nabla_\xi \Phi(x, \hat{\xi}_\ell |\xi|) \cdot (\xi - \hat{\xi}_\ell |\xi|) + \frac{1}{2} (\xi - \hat{\xi}_\ell |\xi|)^T \nabla_{\xi\xi} \Phi(x, \xi^\star)(\xi - \hat{\xi}_\ell |\xi|).$$

By homogeneity of the phase ($\Phi(x, \lambda\xi) = \lambda \Phi(x, \xi)$ for $\lambda > 0$), it holds that $\Phi(x, \xi) = \xi \cdot \nabla_\xi \Phi(x, \xi)$ and $\nabla_\xi \Phi(x, \xi) = \nabla_\xi \Phi(x, \hat{\xi})$. The first and third terms in the above expression cancel, and thus

$$\Phi(x, \xi) = \nabla_\xi \Phi(x, \hat{\xi}_\ell) \cdot \xi + \frac{1}{2} (\xi - \hat{\xi}_\ell |\xi|)^T \nabla_{\xi\xi} \Phi(x, \xi^\star)(\xi - \hat{\xi}_\ell |\xi|).$$

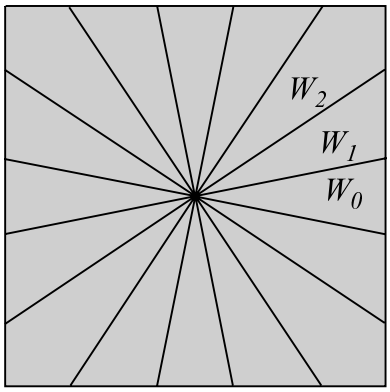


FIG. 1. *The frequency domain is partitioned into* $\sqrt{N}$ *equiangular wedges.*

The first term $\nabla_\xi \Phi(x, \hat{\xi}_\ell) \cdot \xi$, which is linear in ξ, is called *the linearized phase* and poses no problem as we will see later on. The rest, denoted as $\Phi_\ell(x, \xi) = \Phi(x, \xi) - \nabla_\xi \Phi(x, \hat{\xi}_\ell) \cdot \xi$ and called *the residual phase*, is of order $O(1)$ for $\xi \in W_\ell$, independently of N. This follows from

$$\nabla_{\xi\xi} \Phi(x, \xi^\star) = O(|\xi^\star|^{-1}) = O(|\xi|^{-1}),$$

since $\Phi(x, \xi)$ is homogeneous of degree 1 in ξ, together with

$$|\xi - \hat{\xi}_\ell |\xi||^2 = O(|\xi|^2/N) = O(|\xi|)$$

for all $|\xi| \leq N$, which uses the fact that the shape of W_ℓ obeys the parabolic relationship (1.7).

Because the residual phase $\Phi_\ell(x, \xi)$ is of order $O(1)$ independently of N, we say that the function $e^{2\pi i \Phi_\ell(x,\xi)}$ is *nonoscillatory*. Under mild assumptions, this observation guarantees the existence of a low rank separated representation which decouples the variables x and ξ and approximates the complex exponential very well. Define the ϵ-separation rank of a function $f(x, y)$ of two variables as the smallest integer r_ϵ for which there exist $c_n(x)$, $d_n(y)$ such that

$$\left| f(x, y) - \sum_{n=0}^{r_\epsilon - 1} c_n(x) d_n(y) \right| \leq \epsilon.$$

Then we prove the following theorem in section 2.

THEOREM. *For all* $0 < \epsilon \leq 1$, *there exist* $N^* > 0$ *and* $C > 0$ *such that for all* $N \geq N^*$, *the* ϵ*-separation rank of* $e^{2\pi i \Phi_\ell(x,\xi)}$ *for* $x \in [0, 1]^2$ *and* $\xi \in W_\ell$ *obeys*

$$r_\epsilon \leq \log_2(C\epsilon^{-1}). \tag{1.8}$$

In section 2 we make explicit the values of the constants N^* and C by relating them, among other things, to the smoothness of Φ and the angular span of W_ℓ. We will also provide results in the case where $N \leq N^*$ and explain why the separation rank for the amplitude is also under control.

The point of the theorem is that the bound on the ϵ-rank does not grow as a function of N—in fact, the threshold condition on N indicates that the ϵ-rank *decays*

as N grows. The logarithmic dependence on ϵ is the signature of what is usually called spectral accuracy.

Note that the decomposition into frequency wedges obeying the parabolic scaling has a long history in mathematics. A multiscale version of this partitioning, the second dyadic decomposition, was introduced by Fefferman in 1973 for the study of Bochner–Riesz multipliers [22] and used by Seeger, Sogge, and Stein in 1991 to prove a sharp L^p-boundedness result for FIOs [36]. More recently, it also served as the basis for the construction of curvelets, with applications to sparsity of FIOs and related results for wave equations [37, 10, 11].

1.4. Outline of the algorithm. The low rank separated representation provided by the theorem above offers us a way to compute (1.5) efficiently with high accuracy. Each term in the decomposition $Lf = \sum_\ell L_\ell f$ can be further simplified as follows:

$$\begin{aligned}
(L_\ell f)(x) &= \frac{1}{N} \sum_\xi a(x,\xi) e^{2\pi i \Phi(x,\xi)} \chi_\ell(\xi) \hat{f}(\xi) \\
&= \frac{1}{N} \sum_\xi e^{2\pi i \nabla_\xi \Phi(x,\hat{\xi}_\ell)\cdot\xi}\ a(x,\xi) e^{2\pi i \Phi_\ell(x,\xi)}\ \chi_\ell(\xi) \hat{f}(\xi) \\
&= \frac{1}{N} \sum_\xi e^{2\pi i \nabla_\xi \Phi(x,\hat{\xi}_\ell)\cdot\xi}\ \sum_{t=1}^{\infty} \gamma_{\ell t}^x(x) \gamma_{\ell t}^\xi(\xi)\ \chi_\ell(\xi) \hat{f}(\xi) \\
&= \frac{1}{N} \sum_{t=1}^{\infty} \gamma_{\ell t}^x(x) \sum_\xi e^{2\pi i \nabla_\xi \Phi(x,\hat{\xi}_\ell)\cdot\xi} \left[\gamma_{\ell t}^\xi(\xi) \chi_\ell(\xi) \hat{f}(\xi) \right]. \qquad (1.9)
\end{aligned}$$

Our analysis guarantees that the sum over t can be truncated to a fixed, hopefully small number of terms without significant loss of precision.

In order to carry out the final summation over t, we first need to construct the functions $\gamma_{\ell t}^x(x)$ and $\gamma_{\ell t}^\xi(\xi)$. Sections 3.1 and 3.2 discuss two different methods of finding these functions. In section 3.1 we present an elementary deterministic approach, while in section 3.2 we present a randomized approach that offers better efficiency both timewise and storagewise. Assuming that $\gamma_{\ell t}^x(x)$ and $\gamma_{\ell t}^\xi(\xi)$ are available for all values of ℓ and t, the computation of (Lf) for a given f consists of the following four steps:

1. Fourier transform f by means of the FFT to get $\hat{f}$.
2. Choose a bound q greater than the ε-rank r_ε. For each ℓ and $t \le q$, form $\hat{f}_{\ell t}(\xi) := \gamma_{\ell t}^\xi(\xi) \chi_\ell(\xi) \hat{f}(\xi)$.
3. For each ℓ and $t \le q$, compute $g_{\ell t}(x) := \sum_\xi e^{2\pi i \nabla_\xi \Phi(x,\hat{\xi}_\ell)\cdot\xi} \hat{f}_{\ell t}(\xi)$ by means of a nonuniform FFT algorithm.
4. Compute $(Lf)(x) \approx \frac{1}{N} \sum_\ell \sum_{t=1}^q \gamma_{\ell t}^x(x) g_{\ell t}(x)$.

The only step that requires further discussion is the computation of $g_{\ell,t}$. We defer the details to section 3.4.

It is instructive to understand why linearizing the phase is so important. If we disregard the error introduced by the discretization in ξ, we observe that $g_{\ell,t}(x)$ is simply

$$g_{\ell,t}(x) = f_{\ell,t}(\nabla_\xi \Phi(x, \hat{\xi}_\ell)).$$

The interpretation of an oscillatory integral in the Fourier domain as a diffeomorphism is possible only when the phase is linear in ξ. For each ℓ and t, the computation of $g_{\ell,t}$, which is an interpolation problem, is therefore a much simpler problem than applying the original operator. Admittedly, diffeomorphisms do not provide accurate approximations to FIOs over angular wedges, but the content of our analysis in section 2 shows that the computational budget to make up for the residual is safely under control.

1.5. Significance. Applying nontrivial FIOs repeatedly has proved to be the computational bottleneck in various inverse problems. There is serious scientific as well as industrial interest in speeding up FIO computations, and accordingly many resources have been invested over the past decades in engineering better codes.

We believe that the ideas introduced in this paper provide new directions. To explain and illustrate this contrast, let us consider an important example from the field of reflection seismology: Kirchhoff migration. The problem is to produce an image of the discontinuities in the Earth's upper crust from seismograms, i.e., wave measurements $f_s(t, x_r)$ parameterized by time t, receiver coordinate x_r, and source coordinate x_s. The core of Kirchhoff migration is a generalized Radon transform (GRT) which, (in its shot-gather version) consists in integrating several different functions $f_s(t, x_r)$ indexed by s over a fixed set of curves, determined as the level lines of a traveltime function $\tau(x, x_r) + \tau(x, x_s)$, and modulated by an amplitude $a(t, x_r, x_s)$:

$$g_s(x) = \int \delta(t - \tau(x, x_r) - \tau(x, x_s)) a(t, x_r, x_s) f_s(t, x_r)\, \mathrm{d}t\, \mathrm{d}x_r. \tag{1.10}$$

The functions $g_s(x)$ determine the model. A stack operation over the s index then allows us to recover the adequate physical parameters of the Earth, like the speed of sound. When x_r is one-dimensional, it is useful to think of the traveltime level curves as distorted hyperbolas in the (t, x_r) plane. A standard notation for Kirchhoff migration is $g_s(x) = (F^* f_s)(x)$, where F^* is called the imaging operator.

Equation (1.10) is in fact a backprojection strategy for approximately inverting a forward GRT—F in the above notation—which in turn comes from a linearization for the physical parameters about small perturbations, combined with a leading-order high-frequency approximation of the wave equation's Green's function. This view of migration as a GRT is now authoritative in the field of seismic imaging and was pioneered in work by Beylkin and also Miller, Tarantola, Lailly, and Rakesh. The original papers by Beylkin and collaborators are [6, 7, 33]. For more references, see the review article [39] and references therein.

A GRT like (1.10), with a proper cutoff amplitude, can be put in a FIO form suitable for our algorithm. For convenience, the appendix explains why integration along ellipses—a simple GRT—is a sum of two FIOs.

The standard algorithm for applying the imaging operator as in (1.10) is a simple quadrature of $f(t, x_r)$, interpolated and integrated along each curve $t = \tau(x, x_r) + \tau(x, x_s)$ (parameterized by x). Assume again that x_r is one-dimensional for simplicity. If the data $f(t, x_r)$ oscillates at a wavelength comparable to the grid spacing $1/N$, then an accurate quadrature on a smooth curve requires $O(N)$ points. Since x takes on $O(N^2)$ values, the curve integration results in a total complexity of $O(N^3)$ for applying the imaging operator (which is of course better than the $O(N^4)$ complexity of the naive summation).

The first claim of this paper is that a potentially more attractive computational strategy for computing (1.10) is to transform it into FIOs, by considering the data

$f_s(t, x_r)$ in the frequency domain (in both variables t and x_r). The asymptotic complexity is then reduced to $O(N^{2.5} \log N)$ for the algorithm presented in this paper.

The second claim is that since our algorithm is based on FIOs and not just GRTs, it can potentially handle more general migration and imaging operators. For instance, the true imaging operator F^* is almost never a GRT like (1.10). If more terms are kept in the geometric optics approximation leading to (1.10), then the resulting migration operator is no longer a GRT, although it is still a FIO under some very general assumptions; see [38] for a detailed exposition. This observation is akin to the fact that the retarded propagator of the wave equation in two dimensions is not a distribution strictly supported on the boundary of the light cone—only its singular support coincides with the boundary of the cone.

The advantages of the proposed algorithm should now be clear: quite general FIOs can be handled with an asymptotic computational complexity which is lower than that required for GRT summation, i.e., $O(N^{2.5} \log N)$ versus $O(N^3)$, and this without making any curvilinear approximation. In addition, we will show that the storage overhead (on top of storing the phase and amplitude) is negligible and scales like $O(\sqrt{N})$.

We have discussed only applications to reflection seismology, but there are many other areas where nontrivial FIOs are computed routinely, e.g., as part of solving an inverse problem. Examples in radar imaging, ultrasound imaging, and electron microscopy all come to mind. Some Hough transforms for feature detection in image processing can also be formulated as FIOs. In short, the ideas presented in this paper may enable the speedup of fundamental computations in a variety of problem areas.

1.6. Related work. In the case where $\Phi(x, \xi) = x \cdot \xi$, the operator is said to be pseudodifferential. In this simpler setting, it is known that separated variable expansions of the symbol $a(x, \xi)$ are good strategies for reducing complexity. For instance, Bao and Symes [4] propose an $O(N^2 \log N)$ numerical method based on a Fourier series expansion of the symbol in the angular variable $\arg \xi$, and a polyhomogeneous expansion in $|\xi|$, which is a particularly effective example of separation of variables.

Another popular approach for compressing operators is to decompose them in a well-chosen, possibly adaptive basis of L^2. Once a sparse representation is achieved, evaluation simply consists of applying a sparse matrix in the transformed domain. In the case of one-dimensional oscillatory integrals, this program was advocated and carried out by Bradie, Coifman, and Grossman [9] and Averbuch et al. [3]. In spite of these successes, the generalization to multiple dimensions has so far remained an open problem. We will come back to this question in section 5 and in particular will discuss the relationship with modern multiscale transformations such as curvelets [10, 11] and wave atoms [15, 16].

We would also like to acknowledge the line of research related to Filon-type quadratures for oscillatory integrals [29]. When the integrand is of the form $g(x)e^{ikx}$ with g smooth and k large, it is not always necessary to sample the integrand at the Nyquist rate. For instance, integration of a polynomial interpolant of g (Filon quadrature) provides an accurate approximation to $\int g(x)e^{ikx}\,\mathrm{d}x$ using fewer and fewer evaluations of the function g as $k \to \infty$. While these ideas are important, they are not directly applicable in the case of FIOs. The reasons are threefold. First, we make no notable assumption on the support of the function to which the operator is applied, meaning that the oscillations of $\hat{f}(\xi)$ may be on the same scale as those of the exponential $e^{2\pi i \Phi(x,\xi)}$. Second, the phase does not in general have a simple formula that would lend itself to precomputations. And third, Filon-type quadratures do not

address the problem of simplifying computations of *several* such oscillatory integrals at once (i.e., computing a family of integrals indexed by x in the case of FIOs).

Finally, we remark that FIOs are also interesting when the canonical relation is nontrivial—that is, multivalued phase—because they allow us to study propagation of singularities of hyperbolic equations in regimes of multipathing and caustics [28, 20]. To mathematicians taking this specialized viewpoint, the focus of the present paper may appear restrictive. Our outlook and ambition are different. We find FIOs to be interesting mathematical objects even when the canonical relation is a graph and degenerates to the gradient of a phase. Our concern is to understand their structure from an operational standpoint and exploit it to design efficient numerical algorithms. In fact, we expect this paper to be the first of a projected series which will eventually deal with more complex setups.

1.7. Contents. The rest of the paper is organized as follows. Section 2 proves all the analytical estimates which support our methodology. In section 3, we describe algorithms for constructing the low rank separated approximation, evaluating $(Lf)(x)$, as well as for evaluating its adjoint, namely, computing $(L^*f)(x)$. Numerical examples in section 4 illustrate the properties of our algorithms. Finally, section 5 discusses some related work and potential alternatives.

2. Analytical estimates. In this section, we return to a description of the problem in continuous variables x and ξ to prove estimates on the separation rank of $e^{2\pi i\Phi_\ell(x,\xi)}$, where $\Phi_\ell(x,\xi)$ is the residual phase after linearization about $\hat{\xi}_\ell$.

2.1. Background. We begin with a lemma which concerns the separation of the exponential function and whose variations play a central role in modern numerical analysis.

LEMMA 1. *Consider the domain defined by $x \in [-A, A]$ for some $A > 0$, and $y \in [-1, 1]$. For all $\epsilon > 0$ the ϵ-rank r_ϵ of e^{ixy} on $[-A, A] \times [-1, 1]$ obeys the bound $r_\epsilon \le r_\epsilon^*$, where*

$$r_\epsilon^* = 1 + \max\{2eA\ ,\ \log_2(2\epsilon^{-1})\}. \tag{2.1}$$

Furthermore, if $A \le \frac{1}{2e}$, then the stronger bound

$$r_\epsilon^* = 1 + \frac{\log(2\epsilon^{-1})}{\log \frac{1}{eA}} \tag{2.2}$$

holds as well. In both cases, the corresponding separated representation is the expansion

$$\left| e^{ixy} - \sum_{n=0}^{r_\epsilon^*-1} \frac{i^n}{n!} x^n y^n \right| \le \epsilon.$$

Proof. The proof is very simple. We start with

$$\left| e^{ixy} - \sum_{n=0}^{r-1} \frac{(ixy)^n}{n!} \right| \le \sum_{n\ge r} \frac{A^n}{n!} \le \sum_{n\ge r} \left(\frac{eA}{n}\right)^n \le \sum_{n\ge r} \left(\frac{eA}{r}\right)^n = \left(\frac{eA}{r}\right)^r \frac{1}{1-\frac{eA}{r}}.$$

If $eA \le 1/2$, then a fortiori $eA/r \le 1/2$, and the condition $r \ge \log(2\epsilon^{-1})/\log\frac{1}{eA}$ allows us to bound

$$\left(\frac{eA}{r}\right)^r \frac{1}{1-\frac{eA}{r}} \le 2 \cdot (eA)^r \le \epsilon.$$

On the other hand, if $eA \geq 1/2$, then we have to impose $eA/r \leq 1/2$ by hand, as one of the alternatives of the max in (2.1). Then the extra condition $r \geq \log_2(2\epsilon^{-1})$ allows us to bound

$$\left(\frac{eA}{r}\right)^r \frac{1}{1-\frac{eA}{r}} \leq 2 \cdot 2^{-r} \leq \epsilon.$$

Since the ϵ-rank r_ϵ is integer-valued, the estimate on r may need to be rounded up to the next integer—hence the precaution of incrementing the bounds in (2.1) and (2.2) by one. □

In the next section we will make use of Lemma 1 to prove that the nonoscillatory factor $e^{2\pi i \Phi_\ell(x,\xi)}$ has a separation rank which is independent of N. The other factor in the kernel $a(x,\xi)e^{2\pi i \Phi_\ell(x,\xi)}$, namely, the amplitude $a(x,\xi)$, is in general a simpler object to study. The standard assumption in the literature, and also in applications, is to assume that $a(x,\xi)$ is a smooth symbol of order zero and type $(1,0)$, meaning that for each pair of integers (α,β), there is a positive constant $C_{\alpha\beta}$ obeying

$$|\partial_\xi^\alpha \partial_x^\beta a(x,\xi)| \leq C_{\alpha\beta}(1+|\xi|^2)^{-|\alpha|/2}. \tag{2.3}$$

For simplicity, we will also assume that $a(x,\xi)$ is compactly supported in x.[2] The nice separation properties of a are simple consequences of its assumed smoothness.

LEMMA 2. *Assume $a(x,\xi)$ is a symbol of order zero. Then for all $M > 0$ there exists $C_M > 0$ such that for all $\epsilon > 0$, the ϵ-rank for the separation of x and ξ in $a(x,\xi)$ obeys*

$$r_\epsilon \leq C_M\, \epsilon^{-1/M}.$$

Proof. Perform a Fourier transform of the C^∞, compactly supported function $a(\cdot,\xi)$. It suffices to keep $O(\epsilon^{-1/M})$ Fourier modes to approximate $a(\cdot,\xi)$ to accuracy ϵ on its compact support. Each Fourier mode is of the form $\hat{a}(\omega,\xi)e^{i\omega x}$ and is hence separated. □

It goes without saying that the ϵ-rank of the product $a(x,\xi)e^{2\pi i\Phi_\ell(x,\xi)}$ is bounded by a constant times the product of the individual ϵ-ranks, and we now focus on the real object of interest, the factor $e^{2\pi i\Phi_\ell}$.

2.2. Large N asymptotics. In this section we assume that the phase $\Phi(x,\xi)$ is C^3 in ξ, measurable only in x, and we define

$$C_k = 2\pi \sup_{x\in[0,1]^2} \sup_{\xi:|\xi|=1} |\partial_\theta^k \Phi(x,\xi)| \qquad \text{for } 0 \leq k \leq 3,$$

where $\theta = \arg\xi$. These constants will enter our estimates only through the following combinations:

$$D_2 = C_0 + C_2 \qquad \text{and} \qquad D_3 = C_1 + C_3.$$

As before, we also require homogeneity of order one in ξ. Finally, we let the general angular opening of the cone W_ℓ be $\frac{2\alpha}{\sqrt{N}}$ radians for some constant α (the introduction section proposed $\alpha = \pi$).

[2]This assumption is equivalent to assuming that functions in the range of L are themselves compactly supported in situations of interest, which ought to be the case for accurate numerical computations.

The result below is a more precise version of the theorem we introduced in section 1.

THEOREM 1. *For all* $0 < \epsilon \le 1$ *and* $N \ge \frac{\alpha^6 D_3^2}{18\epsilon^2}$, *the* ϵ*-separation rank of* $e^{2\pi i \Phi_\ell(x,\xi)}$ *for* $x \in [0,1]^2$ *and* $\xi \in W_\ell$ *obeys*

$$r_\epsilon \le 1 + \max\left\{ \frac{e\sqrt{2}}{2}\alpha^2 D_2 \, , \, \log_2(4\epsilon^{-1}) \right\}. \tag{2.4}$$

Furthermore, if α *is admissible in the sense that* $\alpha \le \sqrt{\frac{\sqrt{2}}{eD_2}}$, *then*

$$r_\epsilon \le 1 + \frac{\log(4\epsilon^{-1})}{\log \frac{2\sqrt{2}}{e\alpha^2 D_2}}. \tag{2.5}$$

Proof. Put $r = |\xi|$ and θ as the angle measured from the vector ξ_ℓ. The phase Φ can be rewritten as $\Phi(x,\xi) = r\phi(x,\theta)$. Let ξ_1 be the frequency coordinate along ξ_ℓ and ξ_2 orthogonal to ξ_1, so that we can switch between polar and Cartesian coordinates using

$$\frac{\partial \Phi}{\partial \xi_1}(x,\xi_\ell) = \phi(x,0) \qquad \text{and} \qquad \frac{\partial \Phi}{\partial \xi_2}(x,\xi_\ell) = \phi'(x,0),$$

where the derivative of ϕ is taken in θ. The residual phase is

$$\begin{aligned} \Phi_\ell(x,\xi) &= \Phi(x,\xi) - \nabla_\xi \Phi(x,\xi_\ell)\cdot\xi \\ &= r\,\left(\phi(x,\theta) - \cos\theta\phi(x,0) - \sin\theta\phi'(x,0)\right). \end{aligned}$$

We can now expand $\phi(x,\theta)$, $\cos\theta$, and $\sin\theta$ in a Maclaurin series (around $\theta = 0$) to obtain

$$\Phi_\ell(x,\xi) = \frac{r\theta^2}{2}\left(\phi(x,0) + \phi''(x,0)\right) + \frac{r\theta^3}{6}\left[\phi'''(x,\overline{\theta}) - \cos(\tilde{\theta})\phi'(x,0)\right] \tag{2.6}$$

for some $\tilde{\theta}$ and $\overline{\theta}$ between 0 and θ (with $\overline{\theta}$ depending on x.)

The x and ξ variables are separated in the first term of (2.6), and so we write

$$f(x)g(\xi) \equiv 2\pi\left(\phi(x,0) + \phi''(x,0)\right)\frac{r\theta^2}{2}.$$

The term with square brackets is the remainder, and we write

$$R(x,\xi) = 2\pi\frac{r\theta^3}{6}\left[\phi'''(x,\overline{\theta}) - \cos(\tilde{\theta})\phi'(x,0)\right].$$

Our strategy will be to choose N large enough so that $R(x,\xi)$ becomes negligible; hence only the exponential of the first term needs to be separated.

Recall that in two dimensions the frequency domain is the square $[-\frac{N}{2}, \frac{N}{2} - 1]^2$. Since $|\theta| \le \frac{\alpha}{\sqrt{N}}$ in the wedge W_ℓ, and $r \le \frac{\sqrt{2}}{2}N$, we have the following bounds for the two terms in (2.6):

$$|f(x)g(\xi)| \le \frac{\sqrt{2}}{4}\alpha^2 D_2, \qquad |R(x,\xi)| \le \frac{\sqrt{2}}{12}\frac{\alpha^3}{\sqrt{N}}D_3.$$

It is instructive to notice that the bound on $|fg|$ is independent of N. That is the reason why we chose the angular opening of the cone W_ℓ proportional to $N^{-1/2}$ (parabolic scaling).

The first contribution to the separation remainder is given by

$$|e^{i(fg+R)} - e^{ifg}| = |e^{iR} - 1|$$

$$\leq |R| \leq \frac{\sqrt{2}}{12}\frac{\alpha^3}{\sqrt{N}} D_3.$$

The condition on N ensures precisely that this remainder be dominated by $\epsilon/2$.

The second contribution to the total error is due to the separation of e^{ifg} itself and needs to be made smaller than $\epsilon/2$ as well. We invoke Lemma 1 with $f(x) \times \sup|g(\xi)|$ in place of x, $g(\xi)/\sup|g(\xi)|$ in place of y, and $\epsilon/2$ in place of ϵ. With these choices, A becomes $\frac{\sqrt{2}}{4}\alpha^2 D_2$, and we obtain the desired result. □

2.3. Small ϵ asymptotics. Theorem 1 is a special asymptotic result in the case of large N (problem size)—or alternatively small α (cone's angular opening). This regime may not be attained in practice, and so we need another result, without restrictions on N and informative for arbitrarily small ϵ.

To this effect, we need stronger (yet still realistic) smoothness assumptions on the phase Φ: for each x, we require that $\Phi(x,\xi)$ be a real-analytic function of ξ. This condition implies the bound

$$2\pi \sup_{|\xi|=1} |\partial_\theta^k \Phi(x,\xi)| \leq Q\, k!\, R^{-k}$$

for some constants Q and R. For example, R can be taken as any number smaller than the uniform radius of convergence in θ, in which case Q will in general depend on R. Let us term such phases, or functions, (Q,R)-*analytic*. As before, we also require homogeneity in ξ.

THEOREM 2. *Assume $\Phi_\ell(x,\xi)$ is measurable in x and (Q,R)-analytic in ξ for some constants Q and R. Assume that α is admissible in the sense that*

$$\alpha < \min\left\{ \frac{R\sqrt{N}}{2}, \frac{R}{\sqrt{\sqrt{2}Q}} \right\}.$$

Then for all $0 < \epsilon \leq 1$, the ϵ-separation rank of $e^{2\pi i \Phi_\ell(x,\xi)}$ for $x \in [0,1]^2$ and $\xi \in W_\ell$ obeys

$$r_\varepsilon \leq C_p\, \epsilon^{-p} \quad \forall p : p > \frac{2}{\log_2\left(\frac{R\sqrt{N}}{\alpha}\right)}.$$

Proof. Throughout the proof, $x \in [0,1]^2$ and $\xi \in W_\ell$. Using the smoothness assumption on Φ_ℓ, we can repeat the reasoning of the proof of Theorem 1 and obtain the convergent series

$$2\pi\Phi_\ell(x,\xi) = \sum_{k=0}^{\infty} f_k(x) g_k(\xi),$$

where $f_k(x) = 2\pi\phi^{(k)}(x,0)$ (the differentiations are in θ) and

$$g_0(\xi) = r(1-\cos\theta), \qquad g_1(\xi) = r(\theta - \sin\theta), \qquad g_k(\xi) = \frac{r\theta^k}{k!}.$$

We denote the bound $|f_k(x)g_k(\xi)| \le A_k$ with

$$A_0 = \frac{\sqrt{2}}{4} Q\alpha^2, \qquad A_1 = \frac{\sqrt{2}}{12} Q \frac{\alpha^3}{R\sqrt{N}}, \qquad A_k = \frac{\sqrt{2}}{2} QN \left(\frac{\alpha}{R\sqrt{N}}\right)^k \quad \text{for } k > 2.$$

Our strategy will be to call upon Lemma 1 for the first few factors $e^{if_k g_k}$ in order to obtain a separation rank r_k and an error ϵ_k for each of them:

$$\left| e^{if_k g_k} - \sum_{n=0}^{r_k - 1} \frac{i^n}{n!} f_k^n(x) g_k^n(\xi) \right| \le \epsilon_k. \tag{2.7}$$

We will perform this operation for each $k < K$, with K large enough, to be determined. Once the separation of each factor is available, we can write

$$e^{i\sum_{k=0}^{K-1} f_k g_k} = \prod_{k=0}^{K-1} e^{if_k g_k}$$

and obtain the bound on the overall separation rank as the *product* $\prod_{k=0}^{K-1} r_k$.

There are two sources of errors we must contend with:

- *Truncation in k.* The factors $e^{if_k g_k}$ for $k \ge K$ will be deemed negligible if their combined contribution results in an overall error smaller than $\epsilon/2$, meaning

  $$|e^{2\pi i \Phi_\ell} - e^{i\sum_{k=0}^{K-1} f_k g_k}| \le \frac{\epsilon}{2}. \tag{2.8}$$

  The left-hand side is bounded by $|\sum_K^\infty f_k g_k|$. Using the bound we stated earlier on A_k and the admissibility condition on α, a bit of algebra shows that (2.8) is satisfied for

  $$K = \left\lceil \frac{\log\left(2\sqrt{2} QN \epsilon^{-1}\right)}{\log\left(\frac{R\sqrt{N}}{\alpha}\right)} \right\rceil \tag{2.9}$$

  (meaning the smallest integer greater than the quotient inside the brackets). This quantity in turn obeys $K \le \log_2(16\epsilon^{-1})$.
- *Truncation in n.* The truncation errors from (2.7) must be made sufficiently small so that their combined contribution also results in an overall error smaller than $\epsilon/2$, meaning

  $$\left| \prod_{k=0}^{K-1} e^{if_k g_k} - \prod_{k=0}^{K-1} \sum_{n=0}^{r_k - 1} \frac{i^n}{n!} f_k^n(x) g_k^n(\xi) \right| \le \frac{\epsilon}{2}. \tag{2.10}$$

  Easy manipulations[3] show that (2.10) follows from the bound

[3] To justify this step, put $E_k(x,\xi) = e^{if_k(x)g_k(\xi)}$ and start from the identity

$$\prod_{k=0}^{K-1} (E_k + \epsilon_k) - \prod_{k=0}^{K-1} E_k = \sum_j \epsilon_j \prod_{k \ne j} (E_k + \tau_{jk}\epsilon_k)$$

$$= \sum_j \epsilon_j E_j^{-1} \prod_{k=0}^{K-1} (E_k + \tau_{jk}\epsilon_k),$$

where $\tau_{jk} = 0$ if $j \le k$, and $\tau_{jk} = 1$ if $j > k$. Then make use of the bound $(1 + \frac{\epsilon}{3K})^K < e^{\epsilon/3} \le e^{1/3} < 3/2$.

$$\epsilon_k = \frac{\epsilon}{3K}.$$

(Recall that K is comparable to $\log(C\epsilon^{-1})$.)
Such a bound holds if, in turn, we take r_k large enough. The admissibility condition on α ensures, among others, that we can invoke the strong version of Lemma 1, namely, (2.2), and obtain

$$r_k \le 1 + \frac{\log(2\epsilon_k^{-1})}{\log \frac{1}{eA_k}}. \tag{2.11}$$

It now remains to estimate $\prod_{k=0}^{K-1} r_k$, where r_k is given by (2.11) and K by (2.9). We treat the first two factors independently. It follows from the bounds on A_0, A_1, and K and the fact that α and Q are constant in ϵ and N that

$$r_0 \le 1 + \frac{\log(6K\epsilon^{-1})}{\log \frac{4}{e\sqrt{2}Q\alpha^2}} \le C\log(6K\epsilon^{-1})$$

$$\le C\log(3\log(16\epsilon^{-1})) + C\log(2\epsilon^{-1}) \le C\log(2\epsilon^{-1})$$

(the constant C changes from expression to expression), and similarly

$$r_1 \le C\log(2\epsilon^{-1}).$$

The same logarithmic bound holds for r_k in the case $k \ge 2$ but will not suffice for our purpose. Instead, we write

$$r_k \le 1 + \frac{\log(6K\epsilon^{-1})}{\log\left[\frac{\sqrt{2}}{QN}\left(\frac{R\sqrt{N}}{\alpha}\right)^k\right]}$$

$$\le \frac{\log(C\epsilon^{-1}\log(2\epsilon^{-1})) + k\log\left(\frac{R\sqrt{N}}{\alpha}\right)}{\log(C) + k\log\left(\frac{R\sqrt{N}}{\alpha}\right)}$$

$$\equiv \frac{A+k}{B+k} \qquad (k \ge 2).$$

We simplified notations only in the last line. Notice that $A > B$ and that $B + k \ge 1$ when $k \ge 2$. We will assume without loss of generality that A and B are integers. The value of the product $\prod r_k$ can increase only if we replace the initial bound $0 \le k < K$ by the condition that the bound on r_k be greater than 2. So we certainly have

$$r_\epsilon \le \prod_{k\ge 2 : r_k \ge 2} r_k \le \frac{A+2}{B+2}\,\frac{A+3}{B+3}\,\cdots\,\frac{A+A}{B+A}$$

$$= \frac{(2A)!/(A+1)!}{(B+A)!/(B+1)!}.$$

We can now make use of the two-sided Stirling bound

$$\sqrt{2\pi}\, n^{n+1/2} e^{-n+\frac{12}{n+1}} \le n! \le \sqrt{2\pi}\, n^{n+1/2} e^{-n+\frac{12}{n}}$$

to obtain

$$\begin{aligned} r_\epsilon &\le C\,\frac{(2A)^{2A}(A+1)^{-(A+1)}}{(A+B)^{A+B}(B+1)^{-(B+1)}} \\ &\le C\,2^{2A}\,\frac{A^{2A}}{(A+1)^{(A+1)}(A+B)^{A-1}}\,\frac{(B+1)^{(B+1)}}{(A+B)^{B+1}} \\ &\le C\,2^{2A}. \end{aligned}$$

In turn,

$$2^{2A} \le \left(C\epsilon^{-1}\log(2\epsilon^{-1})\right)^{\frac{2}{\log_2\left(\frac{R\sqrt{N}}{\alpha}\right)}},$$

which concludes the proof. □

The lower the fractional exponent of ϵ^{-1} the faster the convergence of separated expansions. Theorem 2 shows exactly which factors can make this exponent arbitrarily small:

- large grid size N, or
- small angular opening constant α, or
- large radius of analyticity R of the phase in arg ξ (uniformly in x).

Observe that the rank bound *decreases* as N increases.

Theorem 2 assumes that the residual phase function $\Phi_\ell(x,\xi)$ is (Q,R)-analytic in ξ. The variation below follows the same path of reasoning and is useful when $\Phi_\ell(x,\xi)$ is only C^∞ in ξ for $\xi \neq 0$.

THEOREM 3. *Assume $\Phi_\ell(x,\xi)$ is C^∞ in ξ for $\xi \neq 0$. For any $p > 0$, there exist two constants C_p and C'_p such that for any N, the ϵ-separation rank with $\varepsilon = C_p\,N^{-p}$ is bounded by $C'_p\,\log N$.*

Proof. The structure of the proof is similar to that of Theorem 2. One needs only to keep the first $2p+2$ term of the series

$$2\pi\Phi_\ell(x,\xi) = \sum_{k=0}^{\infty} f_k(x)g_k(\xi)$$

in order to have $\varepsilon = C_p\,N^{-p}$ for some constant C_p which depends only on p and Φ_ℓ. The product $\prod_{k=0}^{2p+1} r_k$ upper bounds the overall separation rank and is less than $C'_p\,\log N$ for some constant C'_p which depends only on p. □

In many computational problems, the mesh size N^{-1} is linked directly to the desired accuracy ε, usually in the form of a power law, e.g., $\varepsilon = O(N^{-p})$ for some constant p. Therefore, Theorem 3 is interesting for practical reasons.

3. Algorithm. For notational convenience, we assume in this section that the amplitude is identically equal to one; that is, we focus on the so-called (discretized) Egorov operator

$$(Lf)(x) = \frac{1}{N}\sum_{\xi\in\Omega} e^{2\pi i\Phi(x,\xi)}\hat{f}(\xi). \tag{3.1}$$

Both in practice (section 4) and in theory (section 2), one can easily take care of general amplitude terms.

The algorithm for computing (3.1) has two main components:

- *Preprocessing step.* Given the residual phase $\Phi_\ell(x,\xi) \equiv \Phi(x,\xi) - \nabla_\xi \Phi(x,\hat{\xi}_\ell)\cdot\xi$, this step constructs, for each wedge W_ℓ, a low rank separated approximation

$$\left| e^{2\pi i \Phi_\ell(x,\xi)} - \sum_{t=1}^{q} \gamma_{\ell t}^x(x)\gamma_{\ell t}^\xi(\xi) \right| \le \varepsilon.$$

  The functions $\{\gamma_{\ell t}^x(x)\}$ and $\{\gamma_{\ell t}^\xi(\xi)\}$, or their compressed versions, are then stored for use in the next step.
- *Evaluation step.* Given a function f, this step computes $(Lf)(x)$ approximately by

$$(Lf)(x) \approx \frac{1}{N} \sum_\ell \sum_t \gamma_{\ell t}^x(x) \sum_\xi e^{2\pi i \nabla_\xi \Phi(x,\hat{\xi}_\ell)\cdot\xi} \left[\gamma_{\ell t}^\xi(\xi)\chi_\ell(\xi)\hat{f}(\xi) \right].$$

The preprocessing step is performed only once for a fixed phase function $\Phi(x,\xi)$. The family of functions $\{\gamma_{\ell t}^x(x)\}$ and $\{\gamma_{\ell t}^\xi(\xi)\}$ should of course be used again and again to compute $(Lf)(x)$ for different inputs f.

In sections 3.1 and 3.2, we propose two different approaches for constructing the families $\{\gamma_{\ell t}^x(x)\}$ and $\{\gamma_{\ell t}^\xi(\xi)\}$. Section 3.4 describes the details of the evaluation step. Finally, section 3.5 outlines the algorithm for rapidly applying the adjoint operator. In this section, we calculate time and storage complexity under the assumption of large grids, i.e., that of Theorem 2. For other kinds of asymptotics, one may need to adjust these estimates with a multiplicative $\log N$ factor, which is typically negligible.

3.1. Preprocessing step: Deterministic approach. We first describe a deterministic approach for constructing the low rank separated expansion, based on a Taylor expansion, exactly as in the proof of Lemma 1. For each wedge W_ℓ, the strategy consists of the following sequence of steps:

- First, construct a low rank separated approximation of $\Phi_\ell(x,\xi)$. This is done by truncating the polar coordinates Taylor expansion to the $(2p+1)$st term

$$\Phi_\ell(x,\xi) \approx |\xi| \sum_{k=1}^{2p+1} c_{\ell k}(x)(\theta - \theta_\ell)^k.$$

  Here p is a constant that determines the level of accuracy.
- Second, for each k construct a separated expansion of $e^{2\pi i c_{\ell k}(x)\,|\xi|(\theta-\theta_\ell)^k}$. This is done by truncating the Taylor expansion to the first $d_{\ell k}$ terms

$$e^{2\pi i c_{\ell k}(x)\,|\xi|(\theta-\theta_\ell)^k} \approx \sum_{m=0}^{d_{\ell k}-1} \beta_{\ell k m}^x(x)\beta_{\ell k m}^\xi(\xi).$$

  The value of each $d_{\ell k}$ is also chosen to obtain a good accuracy.
- Third, combine the separated expansions for $k = 1, \ldots, 2p+1$ into one separated representation for $e^{2\pi i \Phi_\ell(x,\xi)}$. Simply expanding the product of the expansions obtained in the previous step would be sufficient for proving a theorem like those presented in section 2, but in practice, though, the number of terms in the expansion is too large and far from optimal. We thus combine the product of separated expansions two-by-two with the compression procedure to be described next and repeat the process until there is only one separated expansion left. The final expansion provides us with the required functions $\{\gamma_{\ell t}^x(x)\}$ and $\{\gamma_{\ell t}^\xi(\xi)\}$.

The compression procedure used to combine the product of two separated expansions is quite standard. Suppose we have only two expansions (the subscript ℓ is implicit) and write their product as

$$\left(\sum_{m_1=0}^{d_1-1} \beta^x_{1m_1}(x)\beta^\xi_{1m_1}(\xi)\right)\left(\sum_{m_2=0}^{d_2-1} \beta^x_{2m_2}(x)\beta^\xi_{2m_2}(\xi)\right)$$
$$= \sum_{m_1,m_2} \left(\beta^x_{1m_1}(x)\beta^x_{2m_2}(x)\right)\left(\beta^\xi_{1m_1}(\xi)\beta^\xi_{2m_2}(\xi)\right) := \sum_m c^x_m(x)c^\xi_m(\xi).$$

We adopt the matrix notation and introduce

$$(A)_{x,m} = c^x_m(x), \quad (B^*)_{m,\xi} = c^\xi_m(\xi).$$

The problem is to find two matrices $\tilde{A}$ and $\tilde{B}$ which have far fewer columns than A and B yet obey $\tilde{A}\tilde{B}^* \approx AB^*$. This may be achieved by means of the QR factorization and the SVD:

1. Construct QR factorizations $A = Q_A R_A$ and $B = Q_B R_B$.
2. Compute the SVD of $R_A R_B^*$ and truncate the singular values below a threshold ε together with their associated left and right singular vectors, i.e., $R_A R_B^* \approx U_M S_M V_M^*$, where S_M is a truncated diagonal matrix of singular values.
3. Set $\tilde{A} = Q_A U_M S_M$ and $\tilde{B} = Q_B V_M$.

Suppose A is $m \times q$ and B is $n \times q$ with both m and n much larger than q. The computational complexity of the compression procedure is $O((m+n)q^2)$. In our setup, $m = |X| = N^2$, $n = |W_\ell| = O(N^{1.5})$, and q, the rank bound, is uniformly bounded in N (Theorem 2 shows that q is bounded by a small fractional power of ε, independently of N). Therefore, the complexity of a single compression procedure is $O(N^2)$. Since this needs to be carried out $2p-1$ times for each of the $\sqrt{N}$ wedges, the overall complexity of the deterministic preprocessing is $O(\sqrt{N} \times N^2) = O(N^{2.5})$, where the constant is directly related to the rank bounds of section 2.

Next, let us consider the storage requirement. For each wedge, the size of the final separated expansion is $O(N^2)$. Since there are $\sqrt{N}$ wedges, the total storage requirement is $O(N^{2.5})$, which can be costly when N is large. For example, in a typical problem with $N = 1024$ and $q = 20$, the total storage would be about 10 GB assuming double precision is used. Our second approach to solving the preprocessing step addresses this issue and requires dramatically less storage space.

3.2. Preprocessing step: Randomized approach. This section describes a randomized approach for computing the functions $\{\gamma^x_{\ell t}(x)\}$ and $\{\gamma^\xi_{\ell t}(\xi)\}$ for a fixed ℓ. The method is based on the work presented in Kapur and Long [30]. We use matrix notation and set A to be the matrix defined by

$$A_{x,\xi} := e^{2\pi i \Phi_\ell(x,\xi)}, \quad x \in X,\ \xi \in W_\ell. \tag{3.2}$$

The matrix A is m by n with $m = N^2$ and $n = O(N^{1.5})$. Assume the prescribed error ε is fixed. Theorem 2 tells us that there exists a low rank factorization of A with rank $r_\varepsilon = O(1)$ (again, by this we mean that r_ε is bounded by a constant independent of N, although not independent of ϵ). Using this knowledge, the following randomized method finds an approximate factorization

$$A \approx UT,$$

where U is of size $m \times q$, T is $q \times n$, and $q = O(1)$ in N.

- Select a set C of r columns taken from A uniformly at random, and define $A_{[C]}$ to be the submatrix formed by these columns. In practice, a safe choice is to take r about three times larger than the (unknown) r_ε.
- Compute the SVD $A_{[C]} \approx USV^*$, where the diagonal of S contains only the singular values greater than the threshold ε. Since A has a separation rank $r_\varepsilon = O(1)$, we expect U to be of size $m \times q$, where q is about r_ϵ.
- Select a set R of r rows taken from A uniformly at random, and define $A_{[R]}$ to be the submatrix formed by these rows. Similarly, let $U_{[R]}$ be the submatrix of U containing the same rows.
- Set $T = U^+_{[R]} A_{[R]}$, where $U^+_{[R]}$ is the pseudoinverse of $U_{[R]}$.
- The matrices U and T provide an approximate factorization, i.e., $A \approx UT$. We identify the columns of U with the family $\{\gamma^x_{\ell t}(x)\}$ and the rows of T with $\{\gamma^\xi_{\ell t}(\xi)\}$.

This randomized approach works well in practice, although we are not able to offer a rigorous proof of its accuracy and expect one to be nontrivial. We merely argue that the validity of this methodology hinges on the following observations:

- First, the columns of A are highly correlated. Following the arguments in section 2, it is not difficult to show that a pair of columns with nearby values of the frequency index $\xi \in W_\ell$ has a large inner product. Therefore, as we sample uniformly at random, we get good coverage of the set W_ℓ (leaving no large hole), and as a result, the sampled columns nearly span the space generated by the columns of A. Note that one could also use a deterministic regular sampling strategy; for instance, we could take a Cartesian subgrid as a subset of W_ℓ. We observed that, in practice, the probabilistic approach provides slightly better approximations.
- As the SVD routine is numerically stable, it allows us to extract an orthobasis of the column space of $A_{[C]}$ in a robust way.
- By construction, the columns of U are orthonormal. Results from random projection and the geometry of high-dimensional spaces imply that, as long as U does not correlate with the canonical orthobasis, the columns of $U_{[R]}$ are almost orthogonal as well. This allows us to recover the matrix T in a stable and robust fashion.

The computational complexity of this randomized approach is quite low. The SVD step has a complexity of $O(mr^2) = O(N^2)$, while the matrix product $T = U^+_{[R]} A_{[R]}$ takes $O(nrq) = O(N^{1.5})$ operations. Therefore, for each ℓ, the complexity of the randomized approach is $O(N^2)$. Since the same procedure needs to be carried out for all the $\sqrt{N}$ wedges, the overall complexity is $O(N^{2.5})$.

Often we do not know the exact value of r_ε. Instead of setting r conservatively to be an unnecessarily large number, this difficulty is addressed as follows: we begin with a small r, and check whether q is significantly smaller than r. If this is the case, we accept the factorization. Otherwise, we double r and restart the process. Geometrical increase guarantees that the work wasted (due to unsuccessful attempts) is bounded by the work of the final successful attempt. In practice, we accept the result when $q \leq r/3$, and this criterion seems to work well in our numerical experiments. A more conservative test certainly improves the reliability of the factorization but increases the running time.

We finally examine the storage requirement. A naive approach is to store the matrices U and T for each wedge W_ℓ. As T is much smaller than U in size, the storage requirement for each wedge is roughly the size of U, which is $N^2 q = O(N^2)$.

TABLE 1
Comparison of the deterministic and randomized approaches.

	Time	Storage
Randomized	$O(N^{2.5})$ (small constant)	$O(\sqrt{N})$
Deterministic	$O(N^{2.5})$ (large constant)	$O(N^{2.5})$

Multiplying this by the number of wedges gives a total storage requirement of $O(N^{2.5})$, which can be quite costly for large N, as already mentioned in the last section. We propose to store the matrices VS^{-1} and $U^+_{[R]}$ instead. Both matrices require only storage of size $O(rq) = O(1)$. Whenever we need U and T, we form the products $U = A_{[C]}VS^{-1}$ and $T = U^+_{[R]}A_{[R]}$. Note that the elements of the matrices $A_{[C]}$ and $A_{[R]}$ are given explicitly by the formula (3.2), and there is of course no need to store them at all. Putting it differently, we rewrite the computed factorization as

$$A \approx A_{[C]}\, VS^{-1} U^+_{[R]}\, A_{[R]} \tag{3.3}$$

and store only the matrices VS^{-1} and $U^+_{[R]}$.

We would like to point out that such a scheme is not likely to work for the deterministic approach. The main reason is that the deterministic approach involves multiple compression procedures which make use of QR factorizations and SVD decompositions. These numerical linear algebra routines are quite complicated, and therefore, it would be difficult to relate the resulting low rank factorization with the elements of the matrix A, which have the simple form (3.2).

3.3. Comparison. Table 1 compares the deterministic and the randomized approaches in view of the computational complexity and storage requirement. The deterministic approach has the advantage of guaranteeing an accurate low rank separation. However, the constant in the time complexity can be quite large, as for each wedge, it requires $2p$ compression procedures to combine multiple separated expansions into a single one. Moreover, since the compression step uses QR factorizations and SVDs, we are forced to store the final expansion, which can be quite costly for large N. In practice, the randomized approach constructs a near-optimal low rank expansion with very high probability, requires very low storage space, and enjoys a significantly lower constant in time complexity since it does not utilize repeated QR factorizations or SVDs.

3.4. Evaluation step. Once the families $\{\gamma^x_{\ell t}(x)\}$ and $\{\gamma^\xi_{\ell t}(\xi)\}$ are available, we use the approximation

$$(Lf)(x) \approx \frac{1}{N} \sum_\ell \sum_t \gamma^x_{\ell t}(x) \sum_\xi e^{2\pi i \nabla_\xi \Phi(x,\hat{\xi}_\ell)\cdot\xi} \left[\gamma^\xi_{\ell t}(\xi) \chi_\ell(\xi) \hat{f}(\xi) \right]$$

to evaluate $Lf(x)$. The algorithm simply carries out the evaluation step by step:

1. Compute $\hat{f}$, the Fourier transform of f.
2. For each ℓ and t, form $\hat{f}_{\ell t}(\xi) := \gamma^\xi_{\ell t}(\xi)\chi_\ell(\xi)\hat{f}(\xi)$.
3. For each ℓ and t, compute $g_{\ell t}(x) := \sum_\xi e^{2\pi i \nabla_\xi \Phi(x,\hat{\xi}_\ell)\cdot\xi} \hat{f}_{\ell t}(\xi)$.
4. Compute $(Lf)(x) \approx \frac{1}{N}\sum_\ell \sum_t \gamma^x_{\ell t}(x) g_{\ell t}(x)$.

The only step that requires attention is the third: it asks to evaluate the Fourier series $\sum_\xi e^{2\pi i \nabla_\xi \Phi(x,\hat{\xi}_\ell)\cdot\xi} \hat{f}_{\ell t}(\xi)$ at the N^2 points $\{\nabla_\xi \Phi(x,\hat{\xi}_\ell) : x \in X\}$. Even though X is a Cartesian grid, the warped grid $\{\nabla_\xi \Phi(x,\hat{\xi}_\ell) : x \in X\}$ is no longer so. In

fact, the formula for $g_{\ell t}$ is a nonuniform Fourier transform of the second kind, a subject of considerable attention [2, 5, 25, 34, 35] since the seminal paper of Dutt and Rokhlin [21]. We adopt the approach introduced in the latter paper and, following their notation, set

- $m = 4$, $q = 8$, and $b = 0.425$ for 6 digits of accuracy,
- $m = 4$, $q = 16$, and $b = 0.785$ for 11 digits of accuracy.

We specify these parameter values because they impact the numerical accuracies we will report in the next section, and because it will help anyone interested in reproducing our results.

The algorithm in [21] generally assumes that the Fourier coefficients are supported on the full grid Ω which is symmetric with respect to the origin. For each ℓ, the support of $\hat{f}_{\ell t}(\xi)$ is W_ℓ, which is to say that most of the values of the input on the grid Ω are zero. To speed up the nonuniform FFT, each wedge W_ℓ, which is close to either one of the diagonals, is sheared by 45 degrees so that it becomes approximately horizontal or vertical. Notice that 45 degree shearing of $\hat{f}_{\ell t}(\xi)$ is a simple relabeling of the array. In addition, all wedges are then translated so that their support fits in a rectangle of smaller volume centered around the origin. As the nonuniform FFT [21] asks to compute the FFT of the input data (and then finds a way of interpolating the result on an unstructured grid), we gain efficiency since the input array is now of smaller size. Mathematically, the shearing operation takes the form

$$\xi' = M\xi - \xi_c,$$

where M is either the identity or a 45-degree shear matrix and ξ_c is a translation parameter. Thus, we organize the computations as in

$$\begin{aligned}\sum_{\xi} e^{2\pi i \nabla_\xi \Phi(x,\hat{\xi}_\ell)\cdot\xi} \hat{f}_{\ell t}(\xi) &= \sum_{\xi'} e^{2\pi i \nabla_\xi \Phi(x,\hat{\xi}_\ell)\cdot M^{-1}(\xi'+\xi_c)} \hat{f}_{\ell t}(M^{-1}(\xi'+\xi_c)) \\ &= e^{2\pi i \nabla_\xi \Phi(x,\hat{\xi}_\ell)\cdot M^{-1}\xi_c} \sum_{\xi'} e^{2\pi i \nabla_\xi \Phi(x,\hat{\xi}_\ell)\cdot M^{-1}\xi'} \hat{f}_{\ell t}(M^{-1}(\xi'+\xi_c)),\end{aligned}$$

where the final summation is a nonuniform Fourier transform at the set of points $(M^*)^{-1}\nabla_\xi \Phi(x,\hat{\xi}_\ell)$. In condensed form, the oscillatory modes of the function we wish to evaluate are centered around a center frequency; we factor out this frequency, interpolate the residual, and add the factor back in; for the same accuracy, interpolating the smoother residual requires a smaller computational effort.

A two-dimensional nonuniform FFT takes $O(N^2 \log N)$ operations. This operation needs to be repeated $q = O(1)$ times for each one of the $\sqrt{N}$ wedges. Therefore, the overall complexity is $O(N^{2.5} \log N)$.

3.5. Evaluating the adjoint operator. We conclude this section by presenting how to rapidly apply the adjoint FIO. Begin by expanding the Fourier transform in (1.1) and write

$$(Lf)(x) = \int \left(\int e^{2\pi i(\Phi(x,\xi) - y\cdot\xi)} \mathrm{d}\xi \right) f(y) \mathrm{d}y$$

for $x, y, \xi \in \mathbb{R}^2$. The adjoint operator is then given by

$$\begin{aligned}(L^* f)(x) &= \int \left(\int e^{-2\pi i(\Phi(y,\xi) - x\cdot\xi)} \mathrm{d}\xi \right) f(y) \mathrm{d}y \\ &= \int \left(\int e^{-2\pi i \Phi(y,\xi)} f(y) \mathrm{d}y \right) e^{2\pi i x\cdot\xi} \mathrm{d}\xi\end{aligned}$$

or equivalently as

$$\widehat{(L^*f)}(\xi) = \int e^{-2\pi i\Phi(y,\xi)} f(y)\mathrm{d}y$$

in the Fourier domain. Similarly, one readily checks that the adjoint of the discrete-time FIO is given by the formula

$$\widehat{(L^*f)}(\xi) = \frac{1}{N}\sum_y e^{-2\pi i\Phi(y,\xi)} f(y),$$

where $\xi \in \Omega$ and $y \in X$.

Now follow the same set of ideas as in section 3.4, and decompose L^* as

$$\begin{aligned}
\widehat{(L^*f)}(\xi) &= \frac{1}{N}\sum_\ell \chi_\ell(\xi)\sum_y e^{-2\pi i\Phi(y,\xi)} f(y) \\
&= \frac{1}{N}\sum_\ell \chi_\ell(\xi)\sum_y e^{-2\pi i\Phi_\xi(y,\hat{\xi}_\ell)\cdot\xi} e^{-2\pi i\Phi_\ell(y,\xi)} f(y) \\
&= \frac{1}{N}\sum_\ell \chi_\ell(\xi)\sum_y e^{-2\pi i\Phi_\xi(y,\hat{\xi}_\ell)\cdot\xi}\sum_t \overline{\gamma_{\ell t}^x(y)\gamma_{\ell t}^\xi(\xi)} f(y) \\
&= \frac{1}{N}\sum_\ell \sum_t \chi_\ell(\xi)\overline{\gamma_{\ell t}^\xi(\xi)}\sum_y e^{-2\pi i\Phi_\xi(y,\hat{\xi}_\ell)\cdot\xi}\left(\overline{\gamma_{\ell t}^x(y)} f(y)\right).
\end{aligned}$$

The right-hand side of the last equation provides the key steps of the algorithm.

1. For each ℓ and $t \le q$, compute $f_{\ell t}(y) := \overline{\gamma_{\ell t}^x(y)} f(y)$.
2. For each ℓ and $t \le q$, compute $g_{\ell t}(\xi) := \sum_y e^{-2\pi i\Phi_\xi(y,\hat{\xi}_\ell)\cdot\xi} f_{\ell t}(y)$ using the nonuniform FFT of the first kind; see [21, 25] for details.
3. Compute $\widehat{(L^*f)}(\xi) \approx \frac{1}{N}\sum_\ell \sum_t \chi_\ell(\xi)\overline{\gamma_{\ell t}^\xi(\xi)} g_{\ell t}(\xi)$.
4. Finally, take an inverse two-dimensional FFT to get $(L^*f)(x)$.

Clearly, all the results and discussions concerning the matrix vector product Lf apply here as well.

4. Numerical results. This section presents several numerical examples to demonstrate the effectiveness of the algorithms introduced in section 3. Our implementation is in MATLAB and all the computational results we are about to report were obtained on a desktop computer with a 2.6 GHz CPU and 3 GB of memory. We have implemented both the deterministic and randomized approaches for the preprocessing step. We choose to report the timing and accuracy results of the randomized approach only since it requires less time and storage, as shown in section 3.2.

We first study the error of the separated approximation generated by the randomized preprocessing step. For $x = (x_1, x_2)$ and $\xi = (\xi_1, \xi_2)$, set the phase function to be

$$\Phi_\pm(x,\xi) = x\cdot\xi \pm \sqrt{r_1^2(x)\xi_1^2 + r_2^2(x)\xi_2^2}. \tag{4.1}$$

We show in the appendix that the transformation, which for each x integrates f along an ellipse centered at x and with axes of length $r_1(x)$ and $r_2(x)$, can be cast as a sum $L_+ + L_-$ of two FIOs given by

$$(L_\pm f)(x) = \int a_\pm(x,\xi) e^{2\pi i\Phi_\pm(x,\xi)}\hat{f}(\xi)\mathrm{d}\xi, \tag{4.2}$$

TABLE 2
Relative errors of the low rank separated representation constructed using the randomized approach.

	$\varepsilon =$ 1e-3	$\varepsilon =$ 1e-4	$\varepsilon =$ 1e-5	$\varepsilon =$ 1e-6
$N = 64$	3.57e-04	4.93e-05	3.21e-06	5.17e-07
$N = 128$	3.11e-04	2.28e-05	4.19e-06	5.81e-07
$N = 256$	2.85e-04	2.83e-05	2.94e-06	4.13e-07
$N = 512$	1.66e-04	2.82e-05	4.38e-06	6.80e-07

and with phases obeying (4.1).

In our numerical example, we consider the phase Φ_+ and choose

$$r_1(x) = \frac{1}{9}(2 + \sin(4\pi x_1))(2 + \sin(4\pi x_2)),$$

$$r_2(x) = \frac{1}{9}(2 + \cos(4\pi x_1))(2 + \cos(4\pi x_2)).$$

In each wedge W_ℓ, the phase is then linearized and a low rank separated approximation UT of the matrix

$$A = \left(e^{2\pi i \Phi_\ell(x,\xi)}\right)_{x \in X, \xi \in W_\ell}$$

is computed. To estimate the approximation error, we randomly select two sets Γ and Δ of s rows and s columns. Put $A_{\Gamma\Delta}$ to be the s by s submatrix of A with these rows and columns. The separated rank approximation to $A_{\Gamma\Delta}$ is then obtained by multiplying U_Γ and T_Δ, where U_Γ is the submatrix of U with rows in Γ and T_Δ is that of T with columns in Δ. The error is then estimated via

$$\frac{\|A_{\Gamma\Delta} - U_\Gamma T_\Delta\|_F}{\|A_{\Gamma\Delta}\|_F},$$

where $\|\cdot\|_F$ stands for the Frobenius norm. In our numerical test, we set s to be 200, and Table 2 displays approximation errors for different combinations of problem size N and accuracy ε. The results show that the randomized approach works quite well and that the estimated error is controlled well below the threshold ε.

Next, consider the relationship between the separation rank and the threshold ε. Theorem 3 shows that ε scales like N^{-p} for a fixed constant p provided that the separation rank grows gently like $p \log N$. In this experiment, we use the same phase function $\Phi(x, \xi)$ in (4.1) and show the separation rank for different values of N and p in Table 3. These results suggest that the separation rank is roughly proportional to both p and the logarithm of N, which is compatible with the theoretical estimate. Moreover, when N is fixed, the rank seems to grow linearly with respect to p, which possibly implies that the constant $C(p)$ in Theorem 3 in fact grows linearly with respect to p.

We would like to point out that the number of wedges affects the complexity of our algorithm in two ways. On the one hand, it is obvious from section 3.4 that the complexity grows with the number of wedges in a linear fashion. On the other hand, if we lower the number of wedges, the angular aperture of each wedge has to increase. This implies an increase in the separation rank, which results in a growing computational complexity. In our experiments (Tables 2 and 3), we set the number of wedges to be $\lceil \sqrt{2N} \rceil$, which, in practice, balances these two competing factors.

TABLE 3

Ranks of the separated representation generated by the randomized approach for different values of N and p. The prescribed error is equal to N^{-p}.

	$p=1$	$p=1.5$	$p=2$	$p=2.5$	$p=3$
$N=64$	7	10	14	18	22
$N=128$	9	12	17	21	24
$N=256$	9	12	17	21	25
$N=512$	10	15	19	24	27

TABLE 4

Numerical evaluation of $Lf(x)$ with f a two-dimensional white-noise array. The second and third columns give the number of seconds spent in the preprocessing and evaluation steps, respectively. The fourth column shows the speedup factor over the naive algorithm for computing $(Lf)(x)$ using the direct summation (4.3). The fifth column is the estimated relative error, and the last gives the amount of memory used in terms of megabytes.

(N,ε)	Preprocessing	Evaluation	Speedup	Error	Storage
(64,2.44e-03)	2.06e+00s	3.89e+00s	2.05e+00	2.08e-03	0.76M
(128,6.10e-04)	1.09e+01s	2.45e+01s	6.58e+00	8.02e-04	1.26M
(256,1.53e-04)	8.10e+01s	1.65e+02s	1.67e+01	1.00e-04	2.01M
(512,3.81e-05)	4.67e+02s	9.88e+02s	4.46e+01	4.22e-05	3.06M

We now turn to the numerical evaluation of $(Lf)(x)$,

$$(Lf)(x) = \frac{1}{N}\sum_{\xi} e^{2\pi i\Phi(x,\xi)}\hat{f}(\xi), \tag{4.3}$$

where the phase function Φ is the same as in (4.1). In this example, f is an array of independently and identically mean-zero normal random variables (Gaussian white noise), which in some ways is the most challenging input. The threshold ε is set to be $10\,N^{-2}$ (i.e., $p=2$). To estimate the error, we first pick s points $\{x_i : i=1,\dots,s\}$ from X and put $\{\widetilde{(Lf)}(x_i)\}$ for the output of our algorithm (section 3.4). We then compare the values of $\widetilde{(Lf)}(x_i)$ at these points with those of $\{(Lf)(x_i)\}$ obtained by evaluating (4.3) directly. Finally, we estimate the relative error with

$$\sqrt{\frac{\sum_i |(Lf)(x_i) - \widetilde{(Lf)}(x_i)|^2}{\sum_i |(Lf)(x_i)|^2}}.$$

Here, we choose $s=100$, and Table 4 summarizes our findings for various values of N. The results show that our algorithm performs well. The error is controlled well below threshold and the speedup over the naive algorithm is significant for large values of N.

We have considered only the evaluation of FIOs in "Egorov" form thus far (constant amplitude), but the algorithm described in section 3 can be easily extended to operate with general amplitudes provided that the term $a(x,\xi)$ also admits a low rank separated representation in the variables x and ξ.

To study the performance of our algorithm in the more general setup of variable amplitudes, we continue with the example where f is integrated along ellipses (4.2) (recall the phase (4.1)). The appendix shows that a possible choice for the amplitudes $a_\pm(x,\xi)$ and phases $\Phi_\pm(x,\xi)$ is

$$a_\pm(x,\xi) = \frac{1}{4\pi}(J_0(2\pi\rho(x,\xi)) \pm iY_0(2\pi\rho(x,\xi)))e^{\mp 2\pi i\rho(x,\xi)}, \tag{4.4}$$

$$\Phi_\pm(x,\xi) = x\cdot\xi \pm \rho(x,\xi) \tag{4.5}$$

TABLE 5
Numerical evaluation of $Lf(x)$ with f a two-dimensional white-noise array.

(N, ε)	Preprocessing	Evaluation	Speedup	Error	Storage
(64,2.44e-03)	2.18e+01s	3.67e+01s	4.54e+00	7.30e-04	0.37M
(128,6.10e-04)	1.09e+02s	1.65e+02s	1.49e+01	4.00e-04	0.59M
(256,1.53e-04)	6.62e+02s	8.46e+02s	4.49e+01	1.39e-04	0.89M
(512,3.81e-05)	3.42e+03s	4.43e+03s	1.62e+02	3.69e-05	1.38M

with

$$\rho(x,\xi) = \sqrt{r_1^2(x)\xi_1^2 + r_2^2(x)\xi_2^2}.$$

Here, J_0 and Y_0 are Bessel functions of the first and second kinds, respectively; see the appendix for details.

For the axes lengths, set

$$r_1(x) = r_2(x) \equiv r(x) = \frac{1}{16}(3 + \sin(4\pi x_1))(3 + \sin(4\pi x_2)) \tag{4.6}$$

(which means that our ellipses are circles). We compute $(L_+ f)(x)$ for different values of N and ε and provide the results in Table 5. The computational analysis shows that our algorithm performs equally well in the variable amplitude case. For $N = 512$, the speedup factor over the naive evaluation is about 162. It is clear from section 3.4 that the dominant part of the evaluation step is the $O(q\sqrt{N})$ calls to the nonuniform FFT. An improved implementation of the nonuniform FFT would certainly result in greater speedup factors.

An extremely important property of Fourier integral operators is that, under the nondegeneracy condition

$$\det\left(\frac{\partial^2 \Phi}{\partial x_i \partial \xi_j}\right) \neq 0,$$

the composition of a FIO with its adjoint preserves the singularities of the input function. Mathematically speaking, if $WF(f)$ is the wave front set of f [20, 39], then

$$WF(L^* L f) = WF(f).$$

This property serves as the foundation for most of the current imaging techniques in reflection seismology [39]. In the final example of this section, we verify this phenomenon numerically. We choose the phase function to be

$$\Phi(x,\xi) = x \cdot \xi + r(x)|\xi|,$$

where $r(x)$ is given by (4.6), and compute $(L^* L f)$ using the algorithm discussed in sections 3.4 and 3.5. Figure 2 displays results for three input functions with different kinds of singularities. Looking at the picture, we see that the "singularities" of Lf are of course different than those of f, but we also see that the "singularities" of $L^* L f$ coincide with those of f.

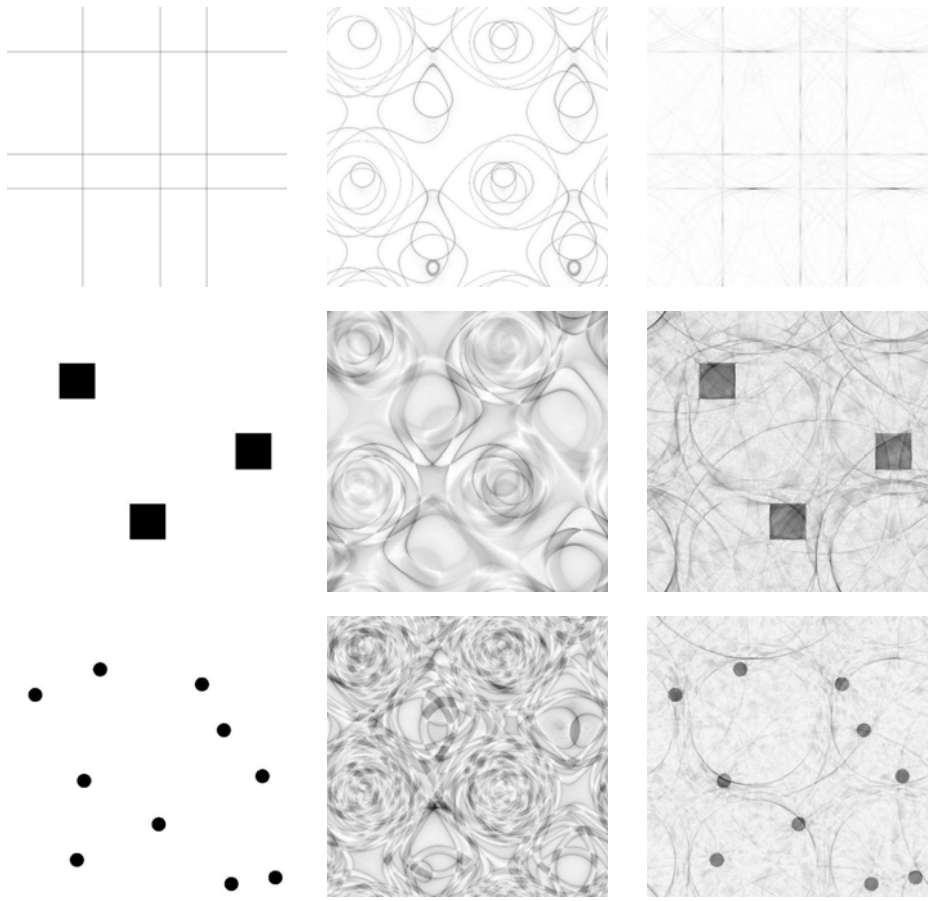


FIG. 2. *Numerical verification of the fact* $WF(L^*Lf) = WF(f)$. *Each row, from left to right, shows the magnitudes of* $f(x)$, $(Lf)(x)$, *and* $(L^*Lf)(x)$. *Notice that the wave front set of* $f(x)$ *and* $(L^*Lf)(x)$ *are numerically close.* Remark*: The images in the left column and in the right column are not supposed to be the same; only their "singularities" coincide. In other words, the adjoint* L^* *is not the inverse of* L. *The artifacts appear because the data to which we apply the FIO is not bandlimited.*

5. Discussion.

5.1. About randomized algorithms. The method used in the randomized preprocessing step was first introduced by Kapur and Long [30]. Lately, there has been a lot of research devoted to the development of randomized algorithms for generating low rank factorizations, and we would like to discuss some of this work.

Drineas, Kannan, and Mahoney [19] describe a randomized algorithm for computing a low rank approximation to a fixed matrix. The main idea is to form a submatrix by selecting columns with a probability proportional to their norm. Since this work is about unstructured general matrices, it does not guarantee a small approximation error. As an example, suppose all the columns of the matrix have the same norm and one of them is orthogonal to the span of the other columns. Unless this column is selected, the orthogonal component is lost and the resulting approximation is poor.

Our situation is different. Since each entry of our matrix

$$A = \left(e^{2\pi i \Phi_\ell(x,\xi)}\right)_{x\in X, \xi\in W_\ell}$$

has unitary magnitude, the uniform probability used in our algorithm is actually the same as that proposed above [19]. In some ways, then, our approach is a special case of that of Drineas, Kannan, and Mahoney. But the point is that our matrix has a special structure. As we argued earlier, the columns of A are often highly correlated, and we believe that this is the reason why the randomized subsampling performs well.

A recent article by Martinsson, Rokhlin, and Tygert [32] presents a new randomized solution to the same problem. The only inconvenience of this algorithm, probably inevitable for general matrices, is that one needs to visit all the entries of the matrix multiple times. This can be quite costly in our setup since there are $O(N^4)$ entries. This is why we adopt the method by Kapur and Long.

5.2. Storage compression. We would like to comment on the storage compression strategy discussed at the end of section 3.2. In fact, what we described there can be viewed as a way of compressing low rank matrices.

In a general context, the entries of a matrix can be viewed as interaction coefficients between a set of objects indexed by the rows and another set indexed by the columns. In our case, the first set contains the grid points x in X, while the second set consists of the frequencies ξ in W_ℓ. Call these two sets I and J, and the interaction matrix $A_{I,J}$. The standard practice for compressing $A_{I,J}$ is to find two sets I' and J' of smaller sizes and form an approximation

$$A_{I,J} \approx M_{I,I'} M_{I',J'} M_{J',J}.$$

Here I' is either a subset of I or a set which is close by in some sense, and likewise for J' and J. For example, in the fast multipole method of Greengard and Rokhlin [26], J' is the multipole representation at the center of the box containing J, while I' is the local representation at the center of the box containing I. The matrices $M_{I,I'}$, $M_{I',J'}$, and $M_{J',J}$ are implemented as the multipole-to-multipole, multipole-to-local, and local-to-local translations. This becomes even more obvious when one considers the newly proposed kernel independent fast multipole method by Ying, Biros, and Zorin [40]. There, I' and J' are the equivalent densities supported on the boxes containing I and J, while $M_{I,I'}$, $M_{I',J'}$, and $M_{J',J}$ can be computed directly from interaction matrices and their inverses. In both cases, we are fortunate in the sense that prior knowledge offers us efficient ways to multiply $M_{I,I'}$, $M_{I',J'}$, and $M_{J',J}$ with arbitrary vectors. Whenever this is not true, one might be forced to store these matrices, which could be quite costly.

What we have presented in (3.3) is a different factorization:

$$A_{IJ} \approx A_{IJ'} R_{J'I'} A_{I'J}.$$

This factorization can be viewed as a variant of the pseudoskeleton approximation proposed in [23, 24], which came to our attention after we had released the initial version of this paper. Notice that since $A_{IJ'}$ and $A_{I'J}$ are interaction matrices themselves, there is no need to store them as long as we can compute the interaction coefficients easily. The only thing we need to keep in storage is the matrix $R_{J'I'}$. However, as long as the interaction is low rank, I' and J' have far fewer objects than I and J, so that $R_{J'I'}$ uses only very little storage. Finally, we would like to point out

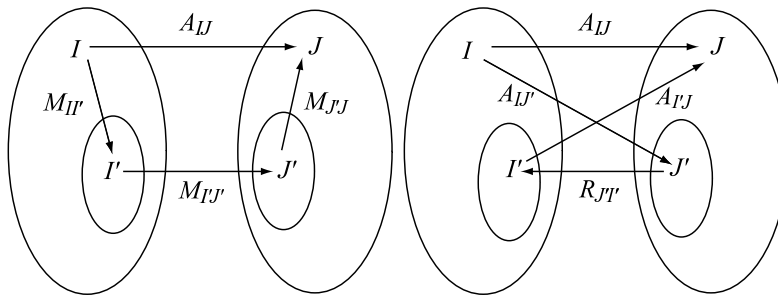


FIG. 3. *Factorization of interaction between A and B.* (a) *The standard scheme.* (b) *The scheme abstracted from the storage compression method used in* (3.3).

that, instead of representing the interaction from J' (a subset of J) to I' (a subset of I), $R_{J'I'}$ is a reverse interaction. Figure 3 shows conceptually how the current factorization differs from the standard one.

5.3. Curvelets, wave atoms, and beamlets. There might be other ways of evaluating FIOs, and we would like to discuss their relationships to the approach taken in this paper.

Curvelets, proposed by Candès and Donoho [12], are two-dimensional waveforms which are highly anisotropic in the fine scales. Each curvelet is identified with three numbers to indicate its scale, orientation, and position, and the set of all curvelets forms a tight frame. Recently, Candès and Demanet [10, 11] have shown that the curvelet representation of the FIOs is optimally sparse. More precisely, a FIO has only $O(N^2)$ nonnegligible entries in the curvelet domain. The wave atom frame, which was recently introduced by Demanet and Ying [16], has the same property. If we were able to find such a representation efficiently, we would hold an $O(N^2 \log N)$ algorithm for evaluating a FIO which would operate as follows:

1. Apply the forward curvelet transform to the input and get curvelet coefficients.
2. Apply the sparse FIO to the curvelet coefficient sequence.
3. Apply the inverse curvelet transform.

Both steps 1 and 3 require at most $O(N^2 \log N)$ operations [13].

Constructing the curvelet representation of a FIO from the phase function $\Phi(x, \xi)$ efficiently has, however, proved to be nontrivial. At the moment, we are able only to construct an approximation which is asymptotically accurate by studying the canonical relation embedded inside the phase function $\Phi(x, \xi)$. Such a construction would be adequate if we were interested in applying a FIO to input functions with only high frequency modes. However, one often wants a representation which is accurate for all frequency modes, and we are currently not aware of any efficient method for constructing such a representation.

Beamlets [17] were introduced by Donoho and Huo at roughly the same time as curvelets. Beamlets are small segments at different positions, scales, and orientations. As pointed out in section 4, curvilinear integrals make up an important subclass of FIOs, and beamlets may offer ways to efficiently compute such simpler integrals. One might think of something like this:

1. Compute the beamlet coefficient sequence of the input.
2. For each $x \in X$, figure out the integration curve and approximate it with a chain of beamlet segments. Sum up the beamlet coefficients along the chain.

Assuming the integration curves are twice differentiable, we would need about $\sqrt{N}$ beamlet segments to approximate each curve. Thus, the overall complexity of this algorithm might scale like $O(N^{2.5})$, which is the same scaling as that of our algorithm. The problem is that it is unclear how one would efficiently approximate the integration curve with beamlet segments without sacrificing accuracy. Situations in which the input function f is highly oscillatory or in which the integration curves have parts with a high curvature seem very problematic.

Our algorithms decompose the FIO in the frequency domain, whereas the beamlet based approach processes data in the spatial domain. Sandwiched right in the middle, curvelets and wave atoms operate in the phase-space—the product of the frequency and of the spatial domains. We believe that operating in phase-space by exploiting the microlocal properties of FIOs would be important in bringing down the complexity to the optimal value of about N^2 operations.

Appendix. Integration along ellipses. The material in this section is probably not new, but we expand on it for the convenience of the nonspecialist. Consider the generalization Radon transform that consists in integrating $f(x)$ along ellipses of axes lengths $r_1(x)$ and $r_2(x)$, and centered around x:

$$Gf(x) = \int f\left(x + \begin{pmatrix} r_1(x)\cos\theta \\ r_2(x)\sin\theta \end{pmatrix}\right) d\theta.$$

We want to recast it as a sum of FIOs. Let us start by writing

$$Gf(x) = \int K(x,\xi)\hat{f}(\xi)\, d\xi$$

with

$$K(x,\xi) = e^{2\pi i x\cdot\xi} \int \exp\left[2\pi i \begin{pmatrix} r_1(x)\cos\theta \\ r_2(x)\sin\theta \end{pmatrix} \cdot \xi\right] d\theta.$$

Put $\rho(x,\xi) = \sqrt{r_1^2(x)\xi_1^2 + r_2^2(x)\xi_2^2}$ and rewrite

$$K(x,\xi) = e^{2\pi i x\cdot\xi} \int \exp\left[2\pi i \rho(x,\xi) \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix} \cdot \begin{pmatrix} \alpha(x,\xi) \\ \beta(x,\xi) \end{pmatrix}\right] d\theta.$$

Here $\alpha^2 + \beta^2 = 1$, and both α and β depend on x and ξ, but the value of the integral is independent of their particular value. This is because any change of variables $\theta \to \theta + \phi(x,\xi)$, effectively corresponding to a rotation of the unit vector (α,β), keeps the integral invariant. So we may as well take $\alpha = 1$, $\beta = 0$ and obtain

$$K(x,\xi) = e^{2\pi i x\cdot\xi} \int e^{2\pi i \rho(x,\xi)\cos\theta}\, d\theta = \frac{e^{2\pi i x\cdot\xi}}{2\pi} J_0(2\pi\rho(x,\xi)).$$

Of course the Bessel function J_0 oscillates, and we need to extract the phase from its asymptotic behavior

$$J_0(2\pi\rho(x,\xi)) \sim \sqrt{\frac{1}{\pi^2\rho(x,\xi)}} \cos\left(2\pi\rho(x,\xi) - \frac{\pi}{4}\right).$$

The idea is now to express $J_0(2\pi\rho(x,\xi))$ as a sum of two terms, each of which is the product between a smooth amplitude (a demodulated version of J_0 or the envelope of

J_0 if you will) and the oscillatory exponential $e^{\pm 2\pi i \rho(x,\xi)}$. In effect, K is decomposed as a sum of two FIOs:

$$K(x,\xi) = a_+(x,\xi)e^{2\pi i \Phi_+(x,\xi)} + a_-(x,\xi)e^{2\pi i \Phi_-(x,\xi)}$$

with

$$\Phi_\pm(x,\xi) = x \cdot \xi \pm \rho(x,\xi).$$

There are different ways to choose the amplitudes. One way is to let Y_0 be the Bessel function of the second kind of order zero [1] and exploit the identity $2J_0 = (J_0 + iY_0) + (J_0 - iY_0)$, which allows us to write

$$a_\pm(x,\xi) = \frac{1}{4\pi}(J_0(2\pi\rho(x,\xi)) \pm iY_0(2\pi\rho(x,\xi)))e^{\mp 2\pi i \rho(x,\xi)}.$$

Both amplitudes behave asymptotically like $\sqrt{1/\pi^2\rho(x,\xi)}$ as $x \to \infty$, which incidentally shows that the order of the FIO is $-1/2$. The logarithmic singularity of Y_0 near the origin in ξ is mild and easily regularized with no loss of accuracy.

Acknowledgments. We are thankful to William Symes for stimulating discussions about Kirchhoff migration and related topics, to Mark Tygert for pointing our attention to the pseudoskeleton approximation, and to the referees for their scrutiny.

REFERENCES

[1] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, Dover, New York, 1972.
[2] C. Anderson and M. D. Dahleh, *Rapid computation of the discrete Fourier transform*, SIAM J. Sci. Comput., 17 (1996), pp. 913–919.
[3] A. Averbuch, E. Braverman, R. Coifman, M. Israeli, and A. Sidi, *Efficient computation of oscillatory integrals via adaptive multiscale local Fourier bases*, Appl. Comput. Harmon. Anal., 9 (2000), pp. 19–53.
[4] G. Bao and W. W. Symes, *Computation of pseudo-differential operators*, SIAM J. Sci. Comput., 17 (1996), pp. 416–429.
[5] G. Beylkin, *On the fast Fourier transform of functions with singularities*, Appl. Comput. Harmon. Anal., 2 (1995), pp. 363–381.
[6] G. Beylkin, *The inversion problem and application of the generalized Radon transform*, Comm. Pure Appl. Math., 37 (1984), pp. 579–599.
[7] G. Beylkin, *Imaging of discontinuities in the inverse scattering problem by inversion of a causal generalized Radon transform*, J. Math. Phys., 26 (1985), pp. 99–108.
[8] G. Beylkin, R. Coifman, and V. Rokhlin, *Fast wavelet transforms and numerical algorithms*. I, Comm. Pure Appl. Math., 44 (1991), pp. 14–183.
[9] B. Bradie, R. Coifman, and A. Grossman, *Fast numerical computation of oscillatory integrals related to acoustic scattering*, I, Appl. Comput. Harmon. Anal., 1 (1993), pp. 94–99.
[10] E. J. Candès and L. Demanet, *Curvelets and Fourier integral operators*, C. R. Math. Acad. Sci. Paris, 336 (2003), pp. 395–398.
[11] E. J. Candès and L. Demanet, *The curvelet representation of wave propagators is optimally sparse*, Comm. Pure Appl. Math., 58 (2005), pp. 1472–1528.
[12] E. J. Candès and D. L. Donoho, *New tight frames of curvelets and optimal representations of objects with piecewise C^2 singularities*, Comm. Pure Appl. Math., 57 (2004), pp. 219–266.
[13] E. J. Candès, L. Demanet, D. L. Donoho, and L. Ying, *Fast discrete curvelet transforms*, Multiscale Model. Simul., 5 (2006), pp. 861–899.
[14] W. Chang, P. Carrion, and G. Beylkin, *Wavefront sets of solutions to linearised inverse scattering problems*, Inverse Problems, 3 (1987), pp. 683–690.
[15] L. Demanet, *Curvelets, Wave Atoms, and Wave Equations*, Ph.D. thesis, California Institute of Technology, Pasadena, CA, 2006.

[16] L. Demanet and L. Ying, *Wave Atoms and Sparsity of Oscillatory Patterns*, Technical report, California Institute of Technology, Pasadena, CA, 2006. Appl. Comput. Harmon. Anal., to appear.
[17] D. L. Donoho and X. Huo, *Beamlets and multiscale image analysis*, in Multiscale and Multiresolution Methods, Lect. Notes Comput. Sci. Eng. 20, Springer-Verlag, Berlin, 2002, pp. 149–196.
[18] H. Douma and M. V. de Hoop, *Leading-order seismic imaging using curvelets*, in SEG Technical Program Expanded Abstracts, 2006, pp. 2411–2415.
[19] P. Drineas, R. Kannan, and M. W. Mahoney, *Fast Monte Carlo algorithms for matrices* II: *Computing a low-rank approximation to a matrix*, SIAM J. Comput., 36 (2006), pp. 158–183.
[20] J. Duistermaat, *Fourier Integral Operators*, Birkhäuser Boston, Boston, 1996.
[21] A. Dutt and V. Rokhlin, *Fast Fourier transforms for nonequispaced data*, SIAM J. Sci. Comput., 14 (1993), pp. 1368–1393.
[22] C. Fefferman, *A note on spherical summation multipliers*, Israel J. Math., 15 (1973), pp. 44–52.
[23] S. A. Goreinov, E. E. Tyrtyshnikov, and N. L. Zamarashkin, *A theory of pseudoskeleton approximations*, Linear Algebra Appl., 261 (1997), pp. 1–21.
[24] S. A. Goreinov, N. L. Zamarashkin, and E. E. Tyrtyshnikov, *Pseudoskeleton approximations by submatrices of greatest size*, Mat. Zametki, 62 (1997), pp. 619–623.
[25] L. Greengard and J.-Y. Lee, *Accelerating the nonuniform fast Fourier transform*, SIAM Rev., 46 (2004), pp. 443–454.
[26] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comput. Phys., 73 (1987), pp. 325–348.
[27] W. Hackbusch, *A sparse matrix arithmetic based on H-matrices.* I. *Introduction to H-matrices*, Computing, 62 (1999), pp. 89–108.
[28] L. Hörmander, *The Analysis of Linear Partial Differential Operators*, Springer-Verlag, New York, 1985.
[29] A. Iserles, *On the numerical quadrature of highly oscillating integrals* I: *Fourier transforms*, IMA J. Numer. Anal., 24 (2004), pp. 365–391.
[30] S. Kapur and D. E. Long, *Ies*3*: A fast integral equation solver for efficient* 3*-dimensional extraction*, in Proceedings of the 1997 IEEE/ACM International Conference on Computer-Aided Design, Washington, DC, 1997, pp. 448–455.
[31] M. P. Lamoureux and G. F. Margrave, *An introduction to numerical methods of pseudodifferential operators*, in Proceedings of the CIME Workshop on Pseudodifferential Operators, Quantization and Signals, Centraro, Italy, 2006.
[32] P.-G. Martinsson, V. Rokhlin, and M. Tygert, *A Randomized Algorithm for the Approximation of Matrices*, Technical report, Yale University, New Haven, CT, 2006.
[33] D. Miller, M. Oristaglio, and G. Beylkin, *A new slant on seismic imaging: Classical migration and integral geometry*, Geophysics, 52 (1987), pp. 943–964.
[34] N. Nguyen and Q. H. Liu, *The regular Fourier matrices and nonuniform fast Fourier transforms*, SIAM J. Sci. Comput., 21 (1999), pp. 283–293.
[35] D. Potts, G. Steidl, and M. Tasche, *Fast Fourier transforms for nonequispaced data: A tutorial*, in Modern Sampling Theory, Appl. Numer. Harmon. Anal., Birkhäuser Boston, Boston, 2001, pp. 247–270.
[36] A. Seeger, C. D. Sogge, and E. M. Stein, *Regularity properties of Fourier integral operators*, Ann. of Math (2), 134 (1991), pp. 231–251.
[37] H. Smith, *A Hardy space for Fourier integral operators*, J. Geom. Anal., 8 (1998), pp. 629–653.
[38] C. Stolk and M. de Hoop, *Microlocal analysis of seismic inverse scattering in anisotropic elastic media*, Comm. Pure Appl. Math., 55 (2002), pp. 261–301.
[39] W. Symes. *Mathematics of Reflection Seismology*, Technical report, Rice University, Houston, TX, 1998, available at http://www.trip.caam.rice.edu/txt/tripinfo/preamble.pdf.
[40] L. Ying, G. Biros, and D. Zorin, *A kernel-independent adaptive fast multipole algorithm in two and three dimensions*, J. Comput. Phys., 196 (2004), pp. 591–626.

SIAM J. SCI. COMPUT.
Vol. 29, No. 6, pp. 2494–2506
© 2007 Society for Industrial and Applied Mathematics

NUMERICAL SOLUTION TO THE SINE-GORDON EQUATION DEFINED ON THE WHOLE REAL AXIS*

CHUNXIONG ZHENG†

Abstract. Numerical simulation of the solution to the sine-Gordon equation on the whole real axis is considered in this paper. Based on nonlinear spectral analysis, exact nonreflecting boundary conditions are derived at two artificially introduced boundary points. Then a numerical scheme of second order is proposed to approximate the solution. In the end, some numerical examples are provided to demonstrate the effectiveness of the proposed scheme.

Key words. sine-Gordon equation, unbounded domains, nonreflecting boundary conditions, inverse scattering

AMS subject classifications. 35P25, 65M06

DOI. 10.1137/050640643

1. Introduction.

In this paper, we consider the numerical simulation of the solution to the sine-Gordon equation

$$q_{tt} - q_{xx} + \sin q = 0,\ x \in \mathbf{R},\ 0 < t \le T, \tag{1}$$

$$q(x,0) = q_0(x),\ q_t(x,0) = q_1(x),\ x \in \mathbf{R}. \tag{2}$$

The sine-Gordon equation has applications in various research areas, such as differential geometry and relativistic field theory, and it also appears in a number of other physical applications, including the propagation of fluxons in Josephson junctions, the motion of rigid pendula attached to a stretched wire, and dislocations in crystals [4, 5].

When numerically solving problems formulated on unbounded domains, such as the considered problem (1)–(2), the unboundedness of definition domains usually presents a great numerical difficulty. This is simply because most of the well-developed methods, such as finite element and finite difference, are domain-based and capable only of handling problems with finite domains. Thus, a natural resolution for these problems is to limit the computational domains by introducing artificial boundaries where suitable artificial boundary conditions (ABCs) can be imposed. As repeatedly shown by different authors both theoretically and experimentally, the overall accuracy and performance of numerical schemes strongly depend on the choice of ABCs. For more details, the readers are referred to the review papers by Givoli [8], Hagstrom [10, 11], and Tsynkov [18].

The ideal ABCs, which are usually referred to as the exact nonreflecting boundary conditions (NRBCs) for wave-like equations in the literature, make the solution to the "truncated" problem exactly the same as that of the untruncated one restricted to the computational domain. For some linear time-dependent problems, these ABCs

*Received by the editors September 19, 2005; accepted for publication (in revised form) April 12, 2007; published electronically October 17, 2007. This work was supported by the National Natural Science Foundation of China under grant 10401020, the Alexander von Humboldt Foundation, and the Key Project of China *High Performance Scientific Computation Research*.
http://www.siam.org/journals/sisc/29-6/64064.html

†Department of Mathematical Sciences, Tsinghua University, Beijing 100084, People's Republic of China, and Institute of Applied and Numerical Mathematics, University of Münster, D-48149 Münster, Germany (czheng@math.tsinghua.edu.cn).

can be derived with the tool of Laplace transformation or by other means; see [1, 3, 9, 13, 16, 17]. But for genuinely nonlinear time-dependent problems, this is by no means a trivial matter. Compared with the large number of papers on linear problems, there are only a few concerned with the nonlinear ones. For at least two special types of nonlinear problems, it is possible to derive the exact ABCs. The first one includes those nonlinear equations which can be transformed into linear equations by the Cole–Hopf transformation, such as the viscous Burgers' equation [12] and the deterministic Kardar–Parisi–Zhang equation [19], and the second one includes some integrable nonlinear evolution equations, such as the one-dimensional cubic nonlinear Schrödinger equation, the sine-Gordon equation, and the modified KdV equation [7]. Apart from these two types, how to derive the exact ABCs is still an open and complicated issue.

Instead of deriving the exact ABCs, some authors turn to the highly accurate approximate ABCs for nonlinear problems. Using pseudodifferential operator theory, Antoine, Besse, and Descombes [2] designed a series of NRBCs for the one-dimensional cubic nonlinear Schrödinger equation and proved their stability. With the tool of pseudodifferential and paradifferential calculus, Szeftel [14] constructed two families of NRBCs for the more general one-dimensional nonlinear Schrödinger equations. Some of these NRBCs were shown to present well-posed initial boundary value problems. Additionally, Szeftel also extended his method to the nonlinear scalar partial differential equations [15].

In [6, 7], Fokas derived a kind of exact generalized Dirichlet-to-Neumann (DtN) mapping for some integrable evolution equations by inverse scattering theory, which unfortunately cannot be used directly for the numerical treatment from the computational point of view. Based on his excellent work, Zheng derived new forms of these generalized DtN mappings for the Schrödinger equation [20] and the modified KdV equation [21] and designed numerical schemes for these two equations.

In this paper, we will derive a new generalized DtN mapping for the sine-Gordon equation following the main procedure of deduction in [7]. A function transformation will be made to avoid the singularity of tangent function involved in the generalized DtN mapping given in [7]. Following the idea involved in [20], a numerical scheme based on the five-point difference stencil will be proposed. In the end, some numerical tests will be presented to demonstrate the performance of this numerical scheme.

2. Exact nonreflecting boundary conditions. Assume there exists a finite interval $\Omega = (a, b)$, such that $q_0(x) = 2\pi m_1$ when $x > b$ and $q_0(x) = 2\pi m_2$ when $x < a$, and $q_1(x)$ is compactly supported in Ω. Here, m_1 and m_2 are two integers. We also assume that the solution of problem (1)–(2) is smooth, and that the functions q_x, q_t, $\sin q$, and $\cos q - 1$ decay to zero sufficiently fast when x goes to infinity. To seek the numerical solution in Ω, two boundary conditions are needed at two artificial boundary points $\{a, b\}$.

Let σ_j denote the usual Pauli matrices

$$\sigma_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \ \sigma_2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \ \sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix},$$

and let $\hat{\sigma}_3$ be the corresponding matrix commutator of σ_3, i.e.,

$$\hat{\sigma}_3 A \stackrel{def}{=} [\sigma_3, A] = \sigma_3 A - A\sigma_3.$$

Here, A is any 2-by-2 matrix. It is straightforward to verify that

$$e^{\hat{\sigma}_3} A = e^{\sigma_3} A e^{-\sigma_3}. \tag{3}$$

The sine-Gordon equation (1) admits a Lax pair in the form

$$\mu_x + if_1(k)\hat{\sigma}_3\mu = Q(x,t,k)\mu, \tag{4}$$

$$\mu_t + if_2(k)\hat{\sigma}_3\mu = \tilde{Q}(x,t,k)\mu,\ \ k \in \mathcal{C}, \tag{5}$$

where $\mathcal{C}$ is the complex field, $\mu = \mu(x,t,k)$ is a 2-by-2 matrix-valued function, and

$$f_1(k) = \frac{1}{4}\left(k - \frac{1}{k}\right),\ \ f_2(k) = \frac{1}{4}\left(k + \frac{1}{k}\right), \tag{6}$$

$$Q(x,t,k) = -\frac{i}{4}(q_x + q_t)\sigma_1 - \frac{i}{4k}(\sin q)\sigma_2 + \frac{i}{4k}(\cos q - 1)\sigma_3, \tag{7}$$

$$\tilde{Q}(x,t,k) = Q(x,t,-k). \tag{8}$$

The existence of a Lax pair in the form of (4)–(5) means that, if and only if function $q = q(x,t)$ satisfies the sine-Gordon equation (1), then system (4)–(5) is compatible and thus solvable. In this case, we have

$$d\left(e^{i(f_1(k)x + f_2(k)t)\hat{\sigma}_3}\mu\right) = e^{i(f_1(k)x + f_2(k)t)\hat{\sigma}_3}(Q\mu dx + \tilde{Q}\mu dt), \tag{9}$$

which implies the right-hand side of (9) is the exact differential 1-form of function $e^{i(f_1(k)x + f_2(k)t)\hat{\sigma}_3}\mu$.

Now we determine $e^{i(f_1(k)x + f_2(k)t)\hat{\sigma}_3}\mu$, and thus μ, by specifying its value at point $(+\infty, t^*)$ as the identity matrix $I = \mathrm{diag}(1,1)$. Here, $t^* \in [0,T]$ is a fixed time point. That is to say, we let μ be determined by

$$e^{i(f_1(k)x + f_2(k)t)\hat{\sigma}_3}\mu = I + \int_{(+\infty,t^*)}^{(x,t)} e^{i(f_1(k)\xi + f_2(k)\tau)\hat{\sigma}_3}(Q\mu d\xi + \tilde{Q}\mu d\tau). \tag{10}$$

The integration is performed on a smooth curve of the $\{x,t\}$-plane which connects $(+\infty, t^*)$ to (x,t). Several remarks should be made here. First, since the right-hand side of (9) is an exact differential 1-form, function μ is independent of the choice of integrating path. Second, the definition of μ is also independent of the choice of t^*. This is because the functions Q and $\tilde{Q}$ decay to zero sufficiently fast thanks to our assumption on the solution q. Third, function μ might not be well defined for some $k \in \mathcal{C}$, since an improper integral gets involved in (10).

We let the integrating path go straight from $(+\infty, t)$ to (x,t), and we rewrite the integral equation (10) as

$$e^{if_1(k)x\hat{\sigma}_3}\mu(x,t,k) = I - \int_x^{+\infty} e^{if_1(k)\xi\hat{\sigma}_3}Q(\xi,t,k)\mu(\xi,t,k)d\xi$$

or, equivalently,

$$\mu(x,t,k) = I - \int_x^{+\infty} e^{if_1(k)(\xi - x)\hat{\sigma}_3}Q(\xi,t,k)\mu(\xi,t,k)d\xi.$$

Since

$$e^{\alpha\hat{\sigma}_3}\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12}e^{2\alpha} \\ a_{21}e^{-2\alpha} & a_{22} \end{pmatrix},$$

the second column of μ, denoted by μ_2, satisfies an integral equation with exponential $e^{2if_1(k)(\xi - x)}$ involved. Since $\xi - x \geq 0$, μ_2 is at least well defined on the complex

domain with $\Im f_1(k) \geq 0$, i.e., $\Im k \geq 0$ except $|k| = 0$. Also, one can show that μ_2 is bounded and continuous and behaves like $(0,1)^T + O(1/k)$ as $k \to \infty$ on this domain. μ_2 is also analytic on the upper-half complex domain. On the other hand, μ_1, the first column of μ, is at least well defined, bounded, and continuous on the complex domain with $\Im k \leq 0$ except $|k| = 0$. It is also analytic on the lower-half complex plane, behaving like $(1,0)^T + O(1/k)$ as $k \to \infty$. More detailed results can be found in [7].

We next study $\mu(b,t,k)$, the value of $\mu(x,t,k)$ on the line $x = b$. We let the integrating path go straight from $(+\infty, 0)$ to $(b,0)$ first, and then straight to (b,t). Since $Q(x,0,k) = \tilde{Q}(x,0,k) = 0$, we have $\mu(x,0,k) = I$ from integral equation (10). Thus, $\mu(b,t,k)$ satisfies the integral equation

$$e^{if_2(k)t\hat{\sigma}_3}\mu(b,t,k) = I + \int_0^t e^{if_2(k)\tau\hat{\sigma}_3}\tilde{Q}(b,\tau,k)\mu(b,\tau,k)d\tau.$$

Setting $\Psi(t,k) = \mu(b,t,k)e^{-if_2(k)t\sigma_3}$ and recalling the identity (3), one knows that Ψ satisfies

$$e^{if_2(k)t\sigma_3}\Psi(t,k) = I + \int_0^t e^{if_2(k)\tau\sigma_3}\tilde{Q}(b,\tau,k)\Psi(\tau,k)d\tau$$

or, equivalently,

$$\Psi_t(t,k) + if_2(k)\sigma_3\Psi(t,k) = \tilde{Q}(b,t,k)\Psi(t,k),\ \Psi(0,k) = I.$$

Setting $g_0(t) = q(b,t)$ and $g_1(t) = q_x(b,t)$, from the definition (8) of $\tilde{Q}$, one has

$$\tilde{Q}(b,t,k) = -\frac{i}{4}(\dot{g}_0 + g_1)\sigma_1 + \frac{i}{4k}(\sin g_0)\sigma_2 - \frac{i}{4k}(\cos g_0 - 1)\sigma_3.$$

It can be verified directly that the second column of function $\Psi(t,k)$, denoted by $\Phi(t,k)$, can be expressed as

$$\Phi(t,k) = \begin{pmatrix} 0 \\ e^{if_2 t} \end{pmatrix} + \int_{-t}^{t} \begin{pmatrix} iL_1(t,s) - \frac{1}{2k}\left[\cos\left[\frac{g_0(t)}{2}\right]M_1(t,s) + \sin\left[\frac{g_0(t)}{2}\right]M_2(t,s)\right] \\ iL_2(t,s) + \frac{i}{2k}\left[-\cos\left[\frac{g_0(t)}{2}\right]M_2(t,s) + \sin\left[\frac{g_0(t)}{2}\right]M_1(t,s)\right] \end{pmatrix} e^{if_2 s}ds \tag{11}$$

with $\{L_j, M_j\}_{j=1,2}$ defined by the following equations:

$$L_1(t,t) = -\frac{g_1(t) + \dot{g}_0(t)}{8},\ M_1(t,t) = -\frac{1}{2}\sin\left[\frac{g_0(t)}{2}\right], \tag{12}$$

$$L_2(t,-t) = M_2(t,-t) = 0,$$

$$L_{1_t} - L_{1_s} = -\frac{g_1(t) + \dot{g}_0(t)}{4}L_2 + \frac{1}{4}\sin\left[\frac{g_0(t)}{2}\right]M_2, \tag{13}$$

$$L_{2_t} + L_{2_s} = \frac{g_1(t) + \dot{g}_0(t)}{4}L_1 - \frac{1}{4}\sin\left[\frac{g_0(t)}{2}\right]M_1, \tag{14}$$

$$M_{1_t} - M_{1_s} = -\sin\left[\frac{g_0(t)}{2}\right]L_2 + \frac{g_1(t) - \dot{g}_0(t)}{4}M_2, \tag{15}$$

$$M_{2_t} + M_{2_s} = \sin\left[\frac{g_0(t)}{2}\right]L_1 - \frac{g_1(t) - \dot{g}_0(t)}{4}M_1. \tag{16}$$

Since $\Psi(t,k)=\mu(b,t,k)e^{-if_2(k)t\sigma_3}$, and μ_2 and Φ are the second column of μ and Ψ, respectively, one has

$$e^{if_2(k)t}\Phi = e^{2if_2(k)t}\mu_2. \tag{17}$$

Let D denote the upper-half complex plane with oriented boundary ∂D, which originates from $-\infty$ to -1 on the real axis, then to 1 on the upper-half unit circle, and finally to $+\infty$ on the real axis. Multiplying (17) by $\left(1-\frac{1}{k^2}\right)e^{-2if_2(k)t'}$ with $0<t'<t$ and performing integration along ∂D, the first element of the right-hand side vanishes owing to Jordan's lemma. Thus,

$$\int_{\partial D}\left(1-\frac{1}{k^2}\right)e^{-2if_2(k)t'}e^{if_2(k)t}\Phi_1 dk = 0. \tag{18}$$

Here, Φ_1 is the first element of vector Φ. From the expression of Φ (11), one has

$$\begin{aligned}\Phi_1 &= \int_{-t}^{t}\left(iL_1(t,s)-\frac{1}{2k}\left[\cos\left[\frac{g_0(t)}{2}\right]M_1(t,s)+\sin\left[\frac{g_0(t)}{2}\right]M_2(t,s)\right]\right)e^{if_2(k)s}ds\\ &= 2\int_0^t\left(iL_1(t,2\tau-t)-\frac{1}{2k}\left[\cos\left[\frac{g_0(t)}{2}\right]M_1(t,2\tau-t)\right.\right.\\ &\qquad\left.\left.+\sin\left[\frac{g_0(t)}{2}\right]M_2(t,2\tau-t)\right]\right)e^{if_2(k)(2\tau-t)}d\tau.\end{aligned}$$

Substituting the above expression of Φ_1 into (18), using the identity

$$\int_{\partial D}\left(1-\frac{1}{k^2}\right)\left[\int_0^t e^{2if_2(\tau-t')}K(\tau,t)d\tau\right]dk = 4\pi K(t',t),$$

taking the limit $t'\to t$, and finally using boundary condition (12), one derives

$$\begin{aligned}g_1(t) &= -\dot g_0(t)-\frac{1}{i\pi}\int_{\partial D}\frac{1}{k}\left(1-\frac{1}{k^2}\right)\\ &\left[\int_0^t e^{-\frac{i}{2}\left(k+\frac{1}{k}\right)(t-\tau)}\left(\cos\left[\frac{g_0(t)}{2}\right]M_1(t,2\tau-t)+\sin\left[\frac{g_0(t)}{2}\right]M_2(t,2\tau-t)\right)d\tau\right]dk.\end{aligned}$$

Exchanging the integrating order of the double integral gives

$$\begin{aligned}g_1(t) = -\dot g_0(t)-\int_0^t \mathcal{Z}(t-\tau)&\left(\cos\left[\frac{g_0(t)}{2}\right]M_1(t,2\tau-t)\right.\\ &\left.+\sin\left[\frac{g_0(t)}{2}\right]M_2(t,2\tau-t)\right)d\tau,\end{aligned} \tag{19}$$

with

$$\begin{aligned}\mathcal{Z}(t) &\overset{def}{=} \frac{1}{i\pi}\int_{\partial D}\frac{1}{k}\left(1-\frac{1}{k^2}\right)e^{-\frac{i}{2}\left(k+\frac{1}{k}\right)t}dk\\ &= -\frac{4}{\pi}\int_1^{+\infty}(x-\sqrt{x^2-1})\sin(xt)dx-\frac{2J_1(t)}{t}+\frac{4}{\pi}\frac{t\cos t-\sin t}{t^2}.\end{aligned}$$

At any fixed time point t, $\mathcal{Z}(t)$ can be evaluated by a numerical quadrature scheme. We depict this function in Figure 1.

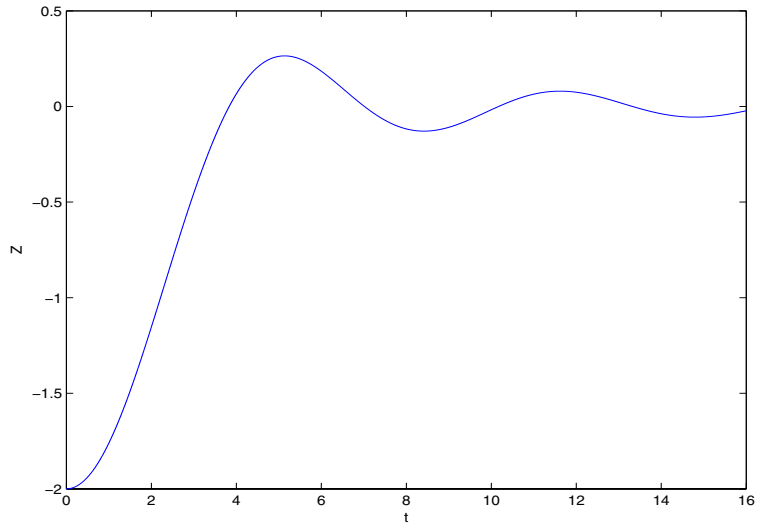


FIG. 1. *Plot of function* $\mathcal{Z}(t)$.

Given $g_0(t)$ and $g_1(t)$, $\{L_j, M_j\}_{j=1,2}$ can be determined from (12)–(16). Thus, the right-hand side of (19) defines a mapping from $g_0(t)$ and $g_1(t)$. We designate this mapping as $\mathcal{K}$ and rewrite (19) as

$$g_1(t) = \mathcal{K}(t; g_0(\cdot), g_1(\cdot)). \tag{20}$$

This equation reveals clearly how the Dirichlet datum $g_0(t)$ and the Neumann datum $g_1(t)$ are related to each other: given $g_0(t)$, $g_1(t)$ is just the fixed point function of mapping $\mathcal{K}$. In this sense, (19) or (20) can be taken as a generalized DtN mapping. We remark that this mapping is different from that given in [7]. A function transformation has been made to avoid the singularity of tangent function $\tan(\cdot)$, which appears in the generalized DtN mapping given in [7, pp. 649–650].

Now imposing the mapping (20) at $x = b$ as a boundary condition, and similarly at $x = a$, we can derive a new "truncated" problem, which is defined only on the finite interval $\Omega = (a, b)$:

$$q_{tt} - q_{xx} + \sin q = 0,\ x \in \Omega,\ t > 0, \tag{21}$$

$$q_\nu(a,t) = \mathcal{K}(t; q(a,\cdot), q_\nu(a,\cdot)),\ q_\nu(b,t) = \mathcal{K}(t; q(b,\cdot), q_\nu(b,\cdot)),\ t > 0, \tag{22}$$

$$q(x,0) = q_0(x),\ q_t(x,0) = q_1(x),\ x \in \Omega. \tag{23}$$

Here, ν denotes the outer normal direction. Based on our analysis, the solution to this problem is exactly the same as that of problem (1)–(2) restricted to Ω.

3. Numerical scheme. Let Δt be the time step size, and let $h = (b-a)/M$ be the spatial step size with M being the number of the grid points. Let

$$t_n = n\Delta t,\ x_j = a + jh,\ q_j^n \sim q(x_j, t_n).$$

We note that owing to the requirement of numerical stability, Δt must be less than h.

We start with numerically approximating the generalized DtN mapping $\mathcal{K}$ in (20) if $g_0(t)$ and $g_1(t)$ are prescribed at discrete time points t_n. We take two steps: seek the

numerical solutions of $\{L_j, M_j\}_{j=1,2}$, and then approximate the integral in (19) by a numerical quadrature scheme. The idea used is a direct adaptation of that involved in [20] for the generalized DtN mapping of the cubic nonlinear Schrödinger equation.

For any function $A(\cdot,\cdot)$ defined in $\{(t,s) : -t \le s \le t,\ 0 \le t \le N\Delta t = T\}$, we let $A_j^n \sim A(t_n, -t_n + 2j\Delta t)$, $j = 0, 1, \ldots, n$. Additionally, we let $g_0^n \sim g_0(t_n)$ and $g_1^n \sim g_1(t_n)$. For the simplicity of notation, we would like to rewrite (12)–(16) in a vector form. Setting $V_1 = (L_1, M_1)^T$ and $V_2 = (L_2, M_2)^T$, we have

$$V_1(t,t) = \left(-\frac{g_1(t) + \dot{g}_0(t)}{8}, -\frac{1}{2}\sin\left[\frac{g_0(t)}{2}\right]\right)^T,\ V_2(t,-t) = (0,0)^T, \tag{24}$$

$$V_{1_t} - V_{1_s} = R(t)V_2,\ V_{2_t} + V_{2_s} = -R(t)V_1, \tag{25}$$

with

$$R(t) = \begin{pmatrix} -\frac{g_1(t)+\dot{g}_0(t)}{4} & \frac{1}{4}\sin\left[\frac{g_0(t)}{2}\right] \\ -\sin\left[\frac{g_0(t)}{2}\right] & \frac{g_1(t)-\dot{g}_0(t)}{4} \end{pmatrix}.$$

By boundary conditions (24), we naturally set

$$V_{1,n}^n = \left(-\frac{1}{8}\left(\frac{g_0^{n+1} - g_0^{n-1}}{2\Delta t} + g_1^n\right), -\frac{1}{2}\sin\left[\frac{g_0^n}{2}\right]\right)^T,\ V_{2,0}^n = (0,0)^T. \tag{26}$$

Since $\frac{1}{\sqrt{2}}(A_t + A_s) = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}) \cdot \nabla A$ is the directional derivative of A with angle $\frac{\pi}{4}$, while $\frac{1}{\sqrt{2}}(A_t - A_s) = (\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}) \cdot \nabla A$ is that with angle $-\frac{\pi}{4}$, we can discretize the directional derivative with the central difference scheme in (25),

$$\frac{V_{1,j}^n - V_{1,j}^{n-1}}{\Delta t} = R_c\frac{V_{2,j}^n + V_{2,j}^{n-1}}{2},\ \frac{V_{2,j}^n - V_{2,j-1}^{n-1}}{\Delta t} = -R_c\frac{V_{1,j}^n + V_{1,j-1}^{n-1}}{2}, \tag{27}$$

with

$$R_c = \begin{pmatrix} -\frac{1}{4}\left[\frac{g_0^n - g_0^{n-1}}{\Delta t} + \frac{g_1^n + g_1^{n-1}}{2}\right] & \frac{1}{4}\sin\left[\frac{g_0^n + g_0^{n-1}}{4}\right] \\ -\sin\left[\frac{g_0^n + g_0^{n-1}}{4}\right] & -\frac{1}{4}\left[\frac{g_0^n - g_0^{n-1}}{\Delta t} - \frac{g_1^n + g_1^{n-1}}{2}\right] \end{pmatrix}.$$

Equation (27) is a linear system with $4(n-1)$ degrees of freedom, and solving this system turns costly when the number of time steps is large. For example, a direct solver needs $O(n^3)$ operations. Instead of the trapezoidal scheme, we can employ the modified Euler scheme, a technique similar to that used in [20]. That is to say, we first use the Euler scheme to derive the first-order approximations $\{V_{1,j,*}^n, V_{2,j,*}^n\}$ through

$$\frac{V_{1,j,*}^n - V_{1,j}^{n-1}}{\Delta t} = R_c V_{2,j}^{n-1},\ \frac{V_{2,j,*}^n - V_{2,j-1}^{n-1}}{\Delta t} = -R_c V_{1,j-1}^{n-1}, \tag{28}$$

and then we compute $\{V_{1,j}^n, V_{2,j}^n\}$ through

$$\frac{V_{1,j}^n - V_{1,j}^{n-1}}{\Delta t} = R_c\frac{V_{2,j,*}^n + V_{2,j}^{n-1}}{2},\ \frac{V_{2,j}^n - V_{2,j-1}^{n-1}}{\Delta t} = -R_c\frac{V_{1,j,*}^n + V_{1,j-1}^{n-1}}{2}. \tag{29}$$

The final scheme is still of second order, but needs only $O(n)$ operations, which is a significant improvement compared with any direct solver.

Next, we consider the approximation of the relation (20). This can be easily handled by a proper interpolation method. If the piecewise linear interpolation is used, and the integration is approximated with the midpoint scheme, one derives the relation

$$g_1^n = -\frac{g_0^{n+1} - g_0^{n-1}}{2\Delta t} - \frac{\Delta t}{4}\sum_{j=0}^{n-1}(\mathcal{Z}(t_{n-j}) + \mathcal{Z}(t_{n-j-1}))(f_{j+1}^n + f_j^n) \tag{30}$$

with

$$f_j^n \stackrel{def}{=} \cos\left[\frac{g_0^n}{2}\right] M_{1,j}^n + \sin\left[\frac{g_0^n}{2}\right] M_{2,j}^n.$$

Now we can see that if g_0^{n+1} and g_1^n are given, $\{V_{1,j}^n, V_{2,j}^n\}$, and thus $\{M_{1,j}^n, M_{2,j}^n\}$, can be computed through (26), (28), and (29). Then, the right-hand side of (30) defines a mapping. Designating this mapping as $\mathcal{K}_{\Delta t}$, (30) can be abbreviated to

$$g_1^n = \mathcal{K}_{\Delta t}(n; g_0^{n+1}, g_1^n). \tag{31}$$

This is a discrete analogy of the continuous generalized DtN mapping (20), and g_1^n can be determined by g_0^{n+1}.

Now we consider the discretization of the sine-Gordon equation in Ω. There are many choices which can serve this purpose. Here, we employ the standard five-point difference scheme. Discretizing the equation at the nth time step and utilizing the five-point stencil of second order, we obtain for $n \geq 1$ and $j = 0, 1, \ldots, M$

$$\frac{q_j^{n+1} - 2q_j^n + q_j^{n-1}}{\Delta t^2} - \frac{q_{j+1}^n - 2q_j^n + q_{j-1}^n}{h^2} + \sin q_j^n = 0, \tag{32}$$

$$\frac{q_{-1}^n - q_1^n}{2h} = \partial_\nu q_0^n, \ \frac{q_{M+1}^n - q_{M-1}^n}{2h} = \partial_\nu q_M^n, \tag{33}$$

$$q_j^0 = q_0(x_j), \ q_j^1 = q_j^0 + \Delta t q_1(x_j) + \frac{\Delta t^2}{2}\left(\frac{q_{j+1}^0 - 2q_j^0 + q_{j-1}^0}{h^2} - \sin q_j^0\right), \tag{34}$$

where $\partial_\nu q_0^n$ and $\partial_\nu q_M^n$ are determined by

$$\partial_\nu q_0^n = \mathcal{K}_{\Delta t}(n; q_0^{n+1}, \partial_\nu q_0^n), \ \partial_\nu q_M^n = \mathcal{K}_{\Delta t}(n; q_M^{n+1}, \partial_\nu q_M^n). \tag{35}$$

On the nth time step, $\{q_1^n, \ldots, q_{M-1}^n\}$ can be determined directly through (32). From (32) and (33), we have

$$q_0^{n+1} = 2q_0^n - q_0^{n-1} + \Delta t^2\left(\frac{2q_1^n - 2q_0^n + 2h\partial_\nu q_0^n}{h^2} - \sin q_0^n\right), \tag{36}$$

$$q_M^{n+1} = 2q_M^n - q_M^{n-1} + \Delta t^2\left(\frac{2q_{M-1}^n - 2q_M^n + 2h\partial_\nu q_M^n}{h^2} - \sin q_M^n\right). \tag{37}$$

Defining

$$\mathcal{L}_{\Delta t,h}(u, v, w, q) = 2u - v + \Delta t^2\left(\frac{2w - 2u + 2hq}{h^2} - \sin u\right),$$

we get

$$q_0^{n+1} = \mathcal{L}_{\Delta t,h}(q_0^n, q_0^{n-1}, q_1^n, \partial_\nu q_0^n), \ q_M^{n+1} = \mathcal{L}_{\Delta t,h}(q_M^n, q_M^{n-1}, q_{M-1}^n, \partial_\nu q_M^n).$$

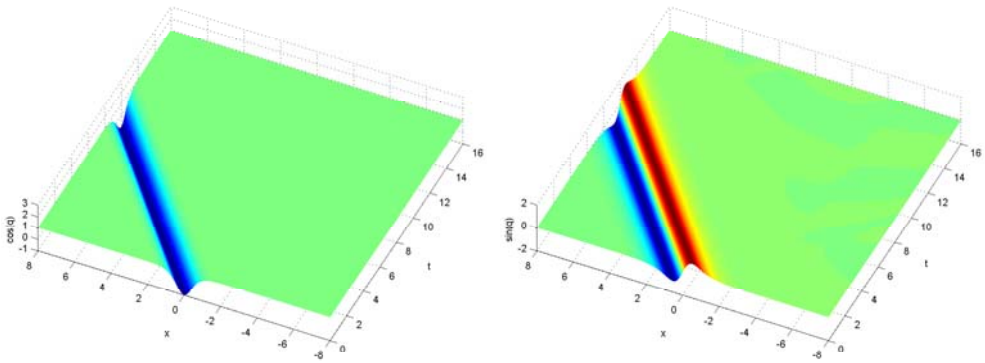


FIG. 2. *Evolution of the kink solution.* $\Delta t = \frac{1}{256}$. *Left:* $\cos q$. *Right:* $\sin q$.

Substituting the above formula into (35) gives

$$\partial_\nu q_0^n = \mathcal{K}_{\Delta t}(n; \mathcal{L}_{\Delta t,h}(q_0^n, q_0^{n-1}, q_1^n, \partial_\nu q_0^n), \partial_\nu q_0^n), \tag{38}$$

$$\partial_\nu q_M^n = \mathcal{K}_{\Delta t}(n; \mathcal{L}_{\Delta t,h}(q_M^n, q_M^{n-1}, q_{M-1}^n, \partial_\nu q_M^n), \partial_\nu q_M^n). \tag{39}$$

We can solve $\partial_\nu q_0^n$ and $\partial_\nu q_M^n$ with fixed point iterations. Once this is done, q_0^n and q_M^n are then obtained through (36)–(37).

4. Numerical tests. The sine-Gordon equation admits some traveling soliton solutions: the kink solution

$$q_{kink} = 4 \arctan \exp \frac{x - at}{\sqrt{1 - a^2}}$$

and the antikink solution

$$q_{antikink} = 4 \arctan \exp - \frac{x - at}{\sqrt{1 - a^2}},$$

with $a \in (0, 1)$ being the moving velocity. When $t = 0$, these solutions tend to some constants exponentially fast as x tends to infinity. For example, $q_{kink} \to 2\pi$ as $x \to +\infty$, and $q_{kink} \to 0$ as $x \to -\infty$. Thus we can take them as reference solutions to test the performance of our numerical scheme. In all computations, we let $h = 2\Delta t$. To evaluate the quality of the numerical solution, we define an error function as

$$E(t) = \frac{||q_{num}(\cdot, t) - q_{exa}(\cdot, t)||_{0,\Omega}}{||q_{exa}(\cdot, t)||_{0,\Omega}}.$$

First, we test the accuracy of our numerical scheme. Let $a = 0.8$, $T = 10$, and $(a, b) = (-8, 8)$, and take $q_0 = q_{kink}(x, 0)$ and $q_1 = \frac{\partial}{\partial t} q_{kink}(x, 0)$ as the initial data. Figure 2 shows the numerical solution with $\Delta t = \frac{1}{256}$ and $h = \frac{1}{128}$. No reflection can be observed. Figure 3 shows the evolution of the relative errors for different time step sizes, and Table 1 lists the errors and the convergence orders. It can be seen that our numerical scheme is of second order, just as we anticipated.

Next, we use our scheme to study the interaction of two solitons. The sine-Gordon equation admits the kink-antikink solution

$$q_{kink\text{-}antikink} = 4 \arctan \left(\frac{p \sinh(\sqrt{p^2 - 1}t)}{\sqrt{p^2 - 1} \cosh(px)} \right)$$

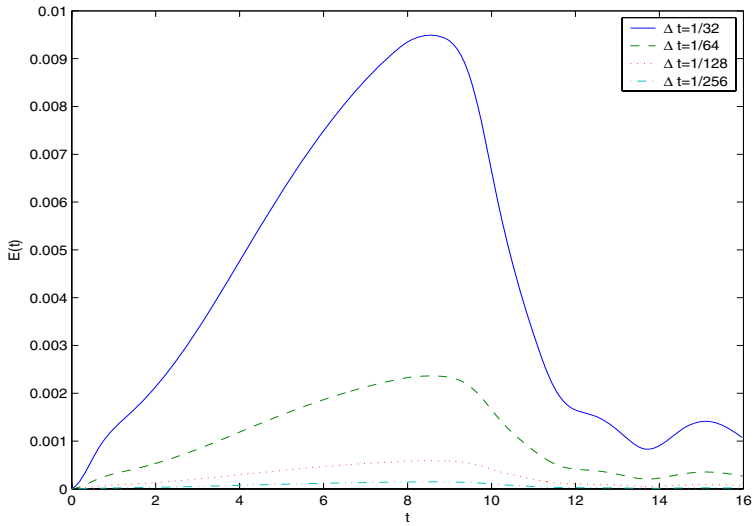


FIG. 3. *Evolution of E for different time step sizes.* $h = 2\Delta t$.

TABLE 1
Computation of $E(10)$ *and the convergence order.* $h = 2\Delta t$.

Time step size Δt	$\frac{1}{32}$	$\frac{1}{64}$	$\frac{1}{128}$	$\frac{1}{256}$
$E(10)$	6.659E-3	1.656E-3	4.134E-4	1.030E-4
Convergence order	-	2.008	2.002	2.005

and the kink-kink solution

$$q_{kink\text{-}kink} = 4\arctan\left(\frac{\sqrt{p^2-1}\sinh(px)}{p\cosh(\sqrt{p^2-1}t)}\right).$$

These two solitons move with the same speed $\sqrt{p^2-1}/p$ but different direction. In these two cases, we let $p = 2$, $\Omega = (-8, 8)$, and

$$q_0(x) = q_{exa}(x, -4),\ q_1(x) = \frac{\partial}{\partial t} q_{exa}(x, -4),$$

with q_{exa} being either $q_{kink\text{-}antikink}$ or $q_{kink\text{-}kink}$. Figures 4 and 5 show the evolution of numerical solutions. Again, no reflections can be observed.

We then consider a more complicated setting. The initial data are set to be four well-separated kink solitons. More precisely, we set

$$q_0(x) = q_{kink\text{-}kink}(x, -6) + q_{kink\text{-}kink}(x, -2),$$
$$q_1(x) = \frac{\partial}{\partial t}\left(q_{kink\text{-}kink}(x, -6) + q_{kink\text{-}kink}(x, -2)\right).$$

The exact solution for this setup is approximately

$$q_{exa} \approx q_{kink\text{-}kink}(x, t-6) + q_{kink\text{-}kink}(x, t-2),$$

which represents the interaction of four kink solitons. At $T = 20$, all solitons will move out of the computational domain $\Omega = (-8, 8)$. We depict the numerical solution in

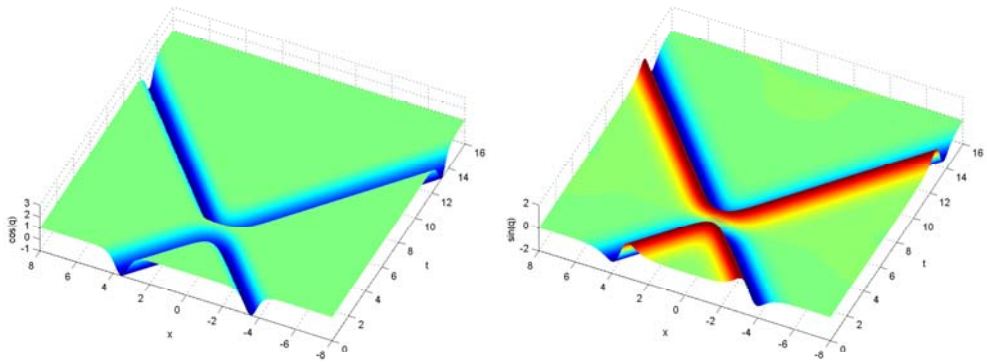


FIG. 4. *Interaction of a kink and an antikink with opposite direction.* $\Delta t = \frac{1}{256}$. *Left:* $\cos q$. *Right:* $\sin q$.

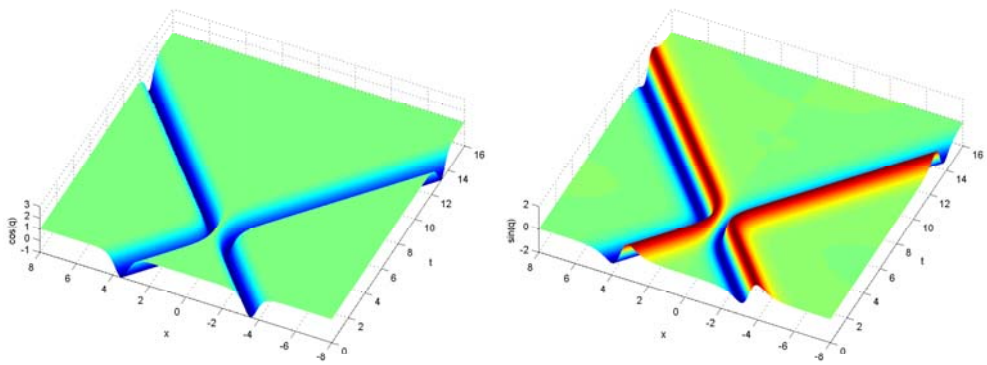


FIG. 5. *Interaction of two kinks with opposite direction.* $\Delta t = \frac{1}{256}$. *Left:* $\cos q$. *Right:* $\sin q$.

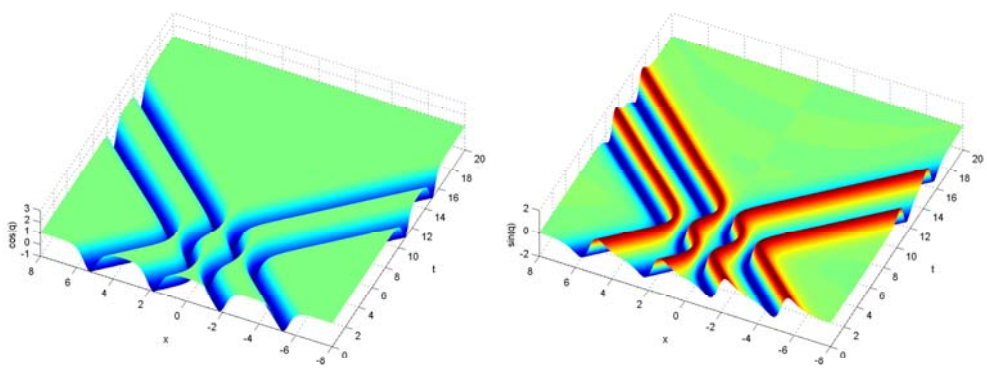


FIG. 6. *Interaction of four kinks.* $\Delta t = \frac{1}{256}$. *Left:* $\cos q$. *Right:* $\sin q$.

Figure 6. It is shown that the interaction of solitons is elastic and our boundary conditions work very well.

Finally, we consider the evolution of a Gaussian wave packet. The initial data are set to be

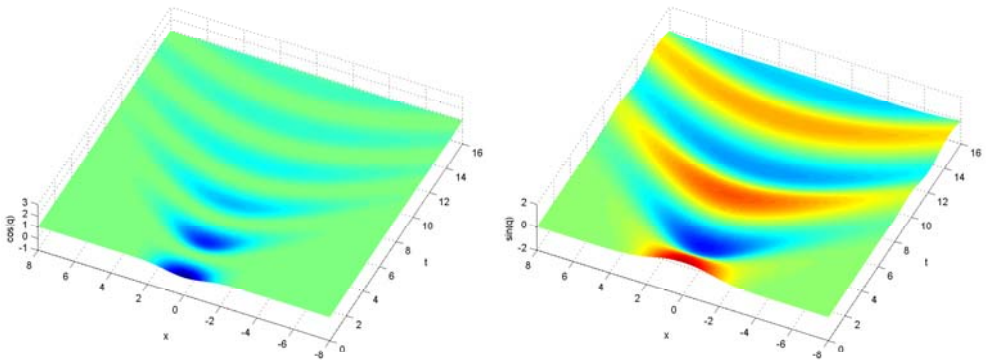


FIG. 7. *Evolution of a Gaussian wave packet.* $\Delta t = \frac{1}{256}$. *Left:* $\cos q$. *Right:* $\sin q$.

TABLE 2
Some statistics in the computation.

Time step size Δt	$\frac{1}{32}$	$\frac{1}{64}$	$\frac{1}{128}$	$\frac{1}{256}$
Max iterations	15	14	13	12
Average iterations	13.12	12.11	11.13	10.13
Time	2.2	7.52	27.25	99.33

$$q_0(x) = e^{-x^2/4},\ q_1(x) = 0.$$

From Figure 7, we see that soliton-like humps are generated successively and move away from the center of this wave packet. In Table 2, we list some numerical statistics. Remember that on the nth time step, we need to solve nonlinear equations (38)–(39) with fixed point iterations, and each iteration needs $O(n)$ operations. Iterations are terminated when the difference of two successive approximations is less than a threshold value, which is set to be 10^{-6} in all our computations. The second row of Table 2 lists the maximal number of iterations $\max_{n=1}^{N} i_n$ for different time step sizes. Here, i_n denotes the number of iterations on the nth time step. Since on different time steps, each iteration requires different operations, we define the average number of iterations as $\sum_{n=1}^{N} i_n n \,/\, \sum_{n=1}^{N} n$, listed in the third row of Table 2. We see that both the maximal and the average iteration numbers decrease with the time step size. The last row of Table 2 lists the time of computation. It increases nearly quadratically with the number of time steps. This statement will cease to hold when the number of time steps becomes much larger. In that case, the evaluation of discrete convolution involved in (30) becomes much more expensive, and the overall cost of computation would increase nearly cubically with the total number of time steps. That means that for long time computation, fast evaluation of convolution must be considered. This discussion is beyond the scope of this paper, and we leave it open here.

We should remark that no instability has been observed in any of our computations.

Acknowledgment. The author would like to thank the anonymous reviewers for their valuable suggestions, which have greatly improved the quality of this paper.

REFERENCES

[1] B. Alpert, L. Greengard, and T. Hagstrom, *Rapid evaluation of nonreflecting boundary kernels for time-domain wave propagation*, SIAM J. Numer. Anal., 37 (2000), pp. 1138–1164.

[2] X. Antoine, C. Besse, and S. Descombes, *Artificial boundary conditions for one-dimensional cubic nonlinear Schrödinger equations*, SIAM J. Numer. Anal., 43 (2006), pp. 2272–2293.

[3] A. Arnold, M. Ehrhardt, and I. Sofronov, *Approximation, stability and fast calculation of non-local boundary conditions for the Schrödinger equation*, Commun. Math. Sci., 1 (2003), pp. 501–556.

[4] R. K. Dodd, J. C. Eilbeck, J. D. Gibbon, and H. C. Morris, *Solitons in Nonlinear Wave Equations*, Academic Press, New York, 1982.

[5] P. J. Drazin and R. S. Johnson, *Solitons: An Introduction*, Cambridge University Press, Cambridge, UK, 1989.

[6] A. S. Fokas, *Integrable nonlinear evolution equations on the half-line*, Comm. Math. Phys., 230 (2002), pp. 1–39.

[7] A. S. Fokas, *The generalized Dirichlet-to-Neumann map for certain nonlinear evolution PDEs*, Comm. Pure Appl. Math., 58 (2005), pp. 639–670.

[8] D. Givoli, *High-order local non-reflecting boundary conditions: A review*, Wave Motion, 39 (2004), pp. 319–326.

[9] M. J. Grote and J. B. Keller, *Exact nonreflecting boundary conditions for the time dependent wave equation*, SIAM J. Appl. Math., 55 (1995), pp. 280–297.

[10] T. Hagstrom, *Radiation boundary conditions for the numerical simulation of waves*, in Acta Numerica 1999, Acta Numer. 8, Cambridge University Press, Cambridge, UK, 1999, pp. 47–106.

[11] T. Hagstrom, *New results on absorbing layers and radiation boundary conditions*, in Topics in Computational Wave Propagation, M. Ainsworth et al., eds., Springer-Verlag, New York, 2003, pp. 1–42.

[12] H. D. Han, X. N. Wu, and Z. L. Xu, *Artificial boundary method for Burgers' equation using nonlinear boundary conditions*, J. Comput. Math., 24 (2006), pp. 295–304.

[13] H. D. Han and C. Zheng, *Exact nonreflecting boundary conditions for an acoustic problem in three dimensions*, J. Comput. Math., 21 (2003), pp. 15–24.

[14] J. Szeftel, *A nonlinear approach to absorbing boundary conditions for the semilinear wave equation*, Math. Comp., 75 (2006), pp. 565–594.

[15] J. Szeftel, *Absorbing boundary conditions for nonlinear scalar partial differential equations*, Comput. Methods Appl. Mech. Engrg., 195 (2006), pp. 3760–3775.

[16] Z. H. Teng, *Exact boundary condition for time-dependent wave equation based on boundary integral*, J. Comput. Phys., 190 (2003), pp. 398–418.

[17] L. Ting and M. J. Miksis, *Exact boundary conditions for scattering problems*, J. Acoust. Soc. Amer., 80 (1986), pp. 1825–1827.

[18] S. V. Tsynkov, *Numerical solution of problems on unbounded domains. A review*, Appl. Numer. Math., 27 (1998), pp. 465–532.

[19] Z. L. Xu, H. D. Han, and X. N. Wu, *Numerical method for the deterministic Kardar-Parisi-Zhang equation in unbounded domains*, Commun. Comput. Phys., 1 (2006), pp. 479–493.

[20] C. Zheng, *Exact nonreflecting boundary conditions for one-dimensional cubic nonlinear Schrödinger equations*, J. Comput. Phys., 215 (2006), pp. 552–565.

[21] C. Zheng, *Numerical simulation of a modified KdV equation on the whole real axis*, Numer. Math., 105 (2006), pp. 315–335.

SIAM J. SCI. COMPUT.
Vol. 29, No. 6, pp. 2507–2530
© 2007 Society for Industrial and Applied Mathematics

IMPLEMENTING GENERATING SET SEARCH METHODS FOR LINEARLY CONSTRAINED MINIMIZATION*

ROBERT MICHAEL LEWIS†, ANNE SHEPHERD‡, AND VIRGINIA TORCZON‡

Abstract. We discuss an implementation of a derivative-free generating set search method for linearly constrained minimization with no assumption of nondegeneracy placed on the constraints. The convergence guarantees for generating set search methods require that the set of search directions possesses certain geometrical properties that allow it to approximate the feasible region near the current iterate. In the hard case, the calculation of the search directions corresponds to finding the extreme rays of a cone with a degenerate vertex at the origin, a difficult problem. We discuss here how state-of-the-art computational geometry methods make it tractable to solve this problem in connection with generating set search. We also discuss a number of other practical issues of implementation, such as the careful treatment of equality constraints and the desirability of augmenting the set of search directions beyond the theoretically minimal set. We illustrate the behavior of the implementation on several problems from the CUTEr test suite. We have found it to be successful on problems with several hundred variables and linear constraints.

Key words. nonlinear programming, nonlinear optimization, constrained optimization, linear constraints, degeneracy, direct search, generating set search, generalized pattern search, derivative-free methods, double description algorithm

AMS subject classifications. 90C30, 90C56, 65K05

DOI. 10.1137/050635432

1. Introduction.

We consider ways to implement direct search methods for solving linearly constrained nonlinear optimization problems of the form

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && \ell \le Ax \le u. \end{aligned} \tag{1.1}$$

The objective function is $f : \mathbb{R}^n \to \mathbb{R}$, with decision variables $x \in \mathbb{R}^n$. The constraint matrix is $A \in \mathbb{R}^{m \times n}$. If some constraints are unbounded below or above, the components of ℓ and u are allowed to take on the values $-\infty$ and $+\infty$, respectively. Our approach handles both equality and inequality constraints.

A key step in generating set search (GSS) algorithms [16] is the computation of the requisite search directions. Generating set search methods for linearly constrained problems achieve convergence without explicit recourse to the gradient or the directional derivative of the objective. They also do not attempt to estimate Lagrange multipliers. Instead, when the search is close to the boundary of the feasible region, the set of search directions must include directions that conform to the ge-

*Received by the editors July 7, 2005; accepted for publication (in revised form) May 31, 2007; published electronically October 17, 2007.

http://www.siam.org/journals/sisc/29-6/63543.html

†Department of Mathematics, College of William & Mary, P.O. Box 8795, Williamsburg, VA 23187-8795 (buckaroo@math.wm.edu). This research was supported by the National Aeronautics and Space Administration under grant NCC-1-02029, by the Computer Science Research Institute at Sandia National Laboratories, and by the National Science Foundation under grant DMS-0215444.

‡Department of Computer Science, College of William & Mary, P.O. Box 8795, Williamsburg, VA 23187-8795 (plshep@cs.wm.edu, va@cs.wm.edu). The research of the second author was conducted under the appointment of a Department of Energy High-Performance Computer Science Graduate Fellowship administered by the Krell Institute and funded by the Computer Science Research Institute at Sandia National Laboratories. The research of the third author was supported by the Computer Science Research Institute at Sandia National Laboratories.

ometry of the nearby boundary. For linearly constrained minimization, this requires identifying a working set of nearly binding constraints at each iteration. The search directions then are computed as the generators of the polar of the cone generated by the outward-pointing normals to the constraints in the working set [23, 29, 20, 21, 17]. When the working set defines a cone with a nondegenerate vertex at the origin, the case is straightforward; computational approaches for handling this case have been known for some time [23, 29, 20].

The hard case arises if the working set defines a cone with a degenerate vertex at the origin. Analysis of algorithms that allow degenerate working sets is well-established [20, 21]. On the other hand, the resulting calculation of the search directions quickly becomes too difficult for a naive approach. As noted in [20], sophisticated and efficient computational geometry algorithms are required.

In this paper we show that, by using state-of-the-art algorithms from computational geometry, the hard case can be dealt with in a computationally effective way and that the naive bounds on the combinatorial complexity involved in managing the hard case are unduly pessimistic. For example, in section 9 we illustrate a problem for which the estimate on the number of possible vectors to be considered is more than 2×10^{17} when, in fact, only 5 vectors are needed—and those vectors are correctly identified in less than one-fifth of a second on a standard desktop computer.

Our specific implementation makes use of the analysis of generating set search found in [17], which synthesizes the analytical results found in [20, 21]. A new feature of the algorithms presented in [17] is the way in which the working set of constraints is chosen, which we show here to be effective in practice.

We use Ω to denote the feasible region for problem (1.1):

$$\Omega = \{\, x \in \mathbb{R}^n \,:\, \ell \leq Ax \leq u \,\}.$$

The algorithm we present here, like all of the approaches referenced previously, is a *feasible iterates* method for solving (1.1): The initial iterate x_0 must be feasible, and all subsequent iterates x_k must satisfy $x_k \in \Omega$. We assume that f is continuously differentiable on Ω but that gradient information is not computationally available; i.e., no procedure exists for computing the gradient of the objective function, and it cannot be approximated accurately. We do not assume that the constraints are nondegenerate.

We discuss several topics regarding our implementation, though computing a sufficient set of search directions is the primary focus of this paper. Our goal is to dynamically identify a set of search directions sufficient to guarantee desirable convergence properties. The general idea is that the set must contain search directions that comprise all the generators for a particular cone of feasible directions, hence the name generating set search. An important question to be addressed in any implementation of this approach is what is meant by "close to the boundary"—i.e., which constraints belong in the current working set—since the current working set determines the requirements on the set of search directions needed for the iteration.

Once the current working set has been identified, the next question becomes how to obtain the desired set of search directions. If the current working set is not degenerate, then straightforward procedures are outlined in [23, 20]. In the presence of degeneracy, the situation becomes considerably more complex—a key issue we discuss here. Equalities in the working set were not treated explicitly in previous work, so here we elaborate on this case. We also discuss how augmenting the set of search directions can accelerate progress toward a solution.

Once a set of search directions has been identified, a step of appropriate length along each of the search directions must be found. The analysis makes this straightforward through the use of a scalar step-length control parameter Δ_k. In our implementation the choice of the working set is tied to Δ_k as proposed in [17].

Finally, there are the usual implementation details of optimization algorithms to consider. These include step acceptance criteria, stopping criteria, scaling of the variables, and data caching.

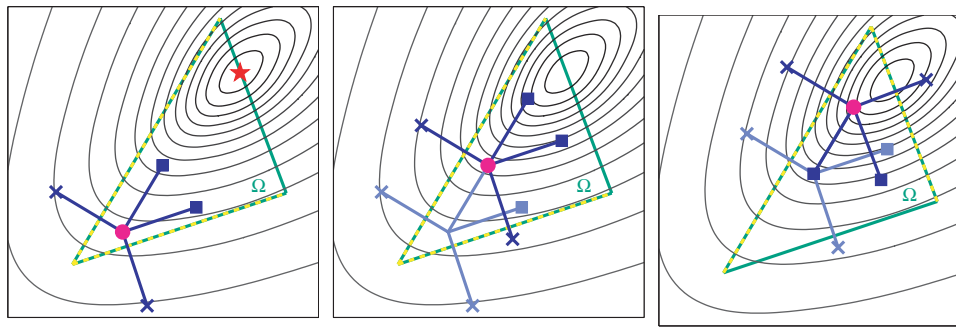
We begin in section 2 with an example and then give some preliminary definitions and notation in section 3. Section 4 discusses how to identify the working set of constraints, while section 5 discusses how to generate the core set of search directions once the working set has been identified. Section 6 discusses why it may be advantageous to augment the core set of search directions. Section 7 discusses how we determine the lengths of the steps to be taken. In section 8 we state the algorithm and discuss other details such as initial feasible iterates (section 8.1), step-acceptance criteria (section 8.2), stopping conditions (section 8.3), scaling (section 8.4), and caching data (section 8.5). Section 9 contains some illustrative numerical results. Concluding remarks are given in section 10.

2. An illustrative example. Figure 2.1 illustrates a few iterations of a greatly simplified linearly constrained GSS method applied to the two-dimensional modified Broyden tridiagonal function [4, 25], to which we have added three linear inequality constraints. Level curves of f are shown in the background. The three constraints form a triangle. In each figure, a dot denotes the current iterate x_k, which is the "best" point—the feasible point with the lowest value of f found so far.

We partition the indices of the iterations into two sets: $\mathcal{S}$ and $\mathcal{U}$. The set $\mathcal{S}$ corresponds to all *successful* iterations—those for which $x_k \neq x_{k+1}$ (i.e., at iteration k the search identified a feasible point with a lower value of the objective, and that point will become the next iterate). The set $\mathcal{U}$ corresponds to all *unsuccessful* iterations—those for which $x_{k+1} = x_k$ since the search was unable to identify a feasible point with a lower value of the objective.

Each of the six subfigures in Figure 2.1 represents one iteration of a linearly constrained GSS method. Taking a step of length Δ_k along each of the four search directions yields the four trial points. The squares represent the trial points under consideration (points at which the objective function is evaluated) at that particular iteration. The crosses represent the trial points that are not considered (points at which the objective function is not evaluated) at that particular iteration because they are infeasible. In subfigures (b)–(f), the trial points from the previous iteration are shown in the background for comparison. In subfigure (a), the solution to the problem is marked with a star.

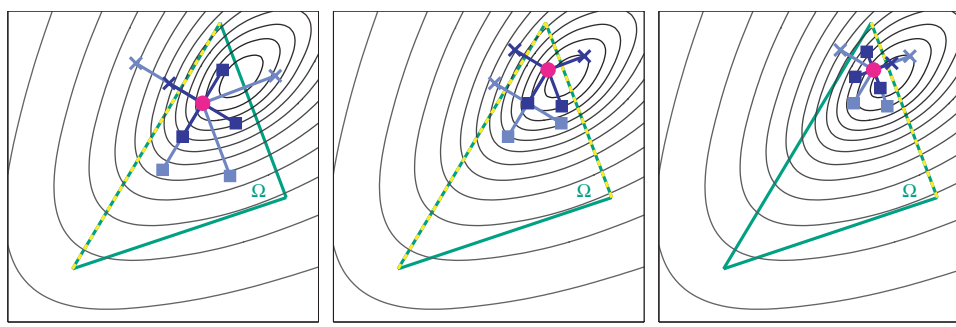
In subfigure (a), the initial iterate x_0 is near two constraints, which are highlighted using dashed lines. (The mechanism by which we identify nearby constraints and its connection to the value of Δ_k are discussed in more detail in section 4.) Two of the initial search directions are chosen so that they are parallel to these two nearby constraints. The other two search directions are the normals to the nearby constraints, translated to x_0. In this instance, there are two feasible descent directions; i.e., moving along two of the four search directions would allow the search to realize descent on the value of f at x_0 while remaining feasible. Either of the two trial points produced by these two feasible descent directions is acceptable; here we choose the one that gives the larger decrease in the value of the objective f to become x_1. Since the step of length Δ_0 from x_0 to x_1 was sufficient to produce decrease, we set Δ_1 to Δ_0.



(a) $k = 0$. The initial set of search directions. Move Northeast; $k \in \mathcal{S}$.

(b) $k = 1$. Keep the set of search directions. Move Northeast; $k \in \mathcal{S}$.

(c) $k = 2$. Change the set of search directions. Contract; $k \in \mathcal{U}$.



(d) $k = 3$. Change the set of search directions. Move Northeast; $k \in \mathcal{S}$.

(e) $k = 4$. Change the set of search directions. Contract; $k \in \mathcal{U}$.

(f) $k = 5$. Change the set of search directions. Move Southeast; $k \in \mathcal{S}$.

FIG. 2.1. *A version of linearly constrained GSS applied to the modified Broyden tridiagonal function augmented with three linear inequality constraints.*

In subfigure (b), the same two constraints remain nearby, so we leave the set of search directions unchanged. Again, there are two feasible descent directions, each of which leads to a trial point that decreases the value of f at the current iterate. We accept the trial point that yields the larger decrease as x_2 and set Δ_2 to Δ_1.

In subfigure (c), the set of nearby constraints changes. We drop one of the constraints that had existed in our working set of nearby constraints in favor of a newly identified constraint that yields a different working set. The set of search directions that results is shown. We do have one descent direction, but now the difficulty is that currently the step along that direction takes the search outside the feasible region. Thus, the best iterate is unchanged, so we set x_3 to x_2. We then *contract* by setting Δ_3 to half the value of Δ_2, which has the effect of halving the length of the steps taken at the next iteration.

The consequences of the contraction are illustrated in subfigure (d). In addition to reducing the length of the step allowed, the contraction reduced the working set to one constraint. Given both the new set of search directions along with the reduction in the length of the steps allowed, we now have a feasible trial point that gives a decrease in the value of the objective, which we make x_4, and set Δ_4 to Δ_3.

In subfigure (e) the search moves to the new iterate. Once again the search is near the same two constraints seen in subfigure (c), so we have the same working set and the same set of search directions, though now we search with a reduced step length (i.e., $\Delta_4 = \frac{1}{2}\Delta_2$). Again, the length of the steps along the two descent directions is too long. So we set x_5 to x_4 and Δ_5 to $\frac{1}{2}\Delta_4$.

The result of halving Δ is illustrated in subfigure (f). In addition to reducing the length of the steps allowed, we have changed the set of working constraints; once again, we have reduced the set to only one working constraint. A new feasible trial point that gives a decrease in the value of f is identified (to the Southeast), and at the next iteration the search will proceed from there.

This example illustrates the essential features the implementation must address: how to identify the nearby constraints, how to obtain a suitable set of feasible search directions, how to find a step of an appropriate length, and how to categorize the iteration as either successful ($k \in \mathcal{S}$) or unsuccessful ($k \in \mathcal{U}$).

3. Notation and definitions. Norms and inner products are assumed to be the Euclidean norm and inner product.

A *cone* K is a set that is closed under nonnegative scalar multiplication: K is a cone if $x \in K$ implies $ax \in K$ for all $a \geq 0$. A cone is *finitely generated* if there exists a (finite) set of vectors $v_1, \ldots, v_r$ such that

$$K = \{\ \lambda_1 v_1 + \cdots + \lambda_r v_r \ \mid\ \lambda_1, \ldots, \lambda_r \geq 0\ \}.$$

The vectors $v_1, \ldots, v_r$ are *generators* of K. Conversely, the cone generated by a set of vectors $v_1, \ldots, v_r$ is the set of all nonnegative combinations of these vectors.

The maximal linear subspace contained in a cone K is its *lineality space* [12]. The *polar* of a cone K, denoted by K°, is the set

$$K^\circ = \left\{\ v \ \mid\ w^T v \leq 0 \text{ for all } w \in K\ \right\}.$$

The polar K° is a convex cone, and if K is finitely generated, then so is K°.

Let K be a convex cone. Given a vector v, let v_K and v_{K° denote the projections of v onto K and its polar K°, respectively. The polar decomposition [24, 31] says that any vector v can be written as the sum of its projections onto K and K° and that the projections are orthogonal: $v = v_K + v_{K^\circ}$, and $v_K^T v_{K^\circ} = 0$. This means that $\mathbb{R}^n$ is positively spanned by a set of generators of K together with a set of generators of K°, a generalization of the direct sum decomposition given by a subspace and its orthogonal complement.

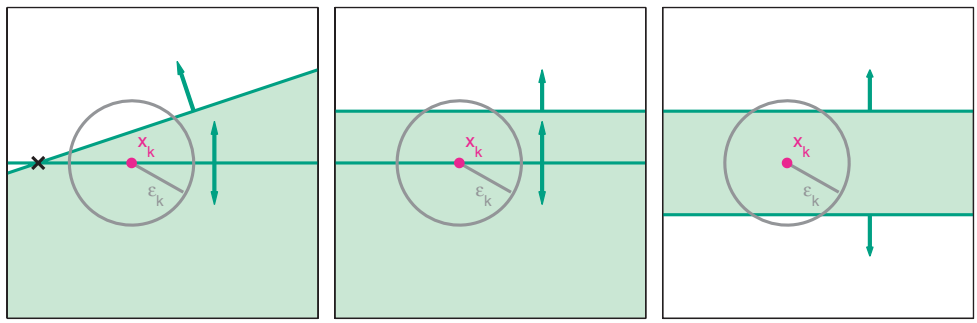
We borrow the following definition from [17]. For any finite set of vectors $\mathcal{G}$,

$$\kappa(\mathcal{G}) = \inf_{\substack{v \in \mathbb{R}^n \\ v_K \neq 0}} \max_{d \in \mathcal{G}} \frac{v^T d}{\| v_K \| \, \| d \|}, \quad \text{where } K \text{ is the cone generated by } \mathcal{G}. \tag{3.1}$$

The value $\kappa(\mathcal{G})$ is a property of the set $\mathcal{G}$, not of the cone K. The measure $\kappa(\mathcal{G})$ plays a critical role when choosing the set of search directions, as discussed further in section 5. Given (3.1), we then have the following.

PROPOSITION 3.1 (proposition 10.3 of [20]). *If $\mathcal{G} \neq \{\mathbf{0}\}$, then $\kappa(\mathcal{G}) > 0$.*

If K is a vector space, then a set of generators for K is called a *positive spanning set* [7]. A positive spanning set is like a linear spanning set but with the additional requirement that all of the coefficients be nonnegative.



(a) The inequality constraint is not included in the working set since its intersection with the equality constraint lies outside the ε_k-ball.

(b) The inequality constraint is redundant given the equality constraint.

(c) Both the upper and lower bounds are in the working set.

FIG. 4.1. *Special cases for linear equality constraints.*

4. Constructing the working set of constraints. We next discuss how we determine the working set of constraints that is used to compute the minimal set of necessary search directions. Let a_i^T denote the rows of A, indexed by i. We partition the constraints into the equalities and inequalities:

$$\mathcal{E} = \{\ i\ \mid\ \ell_i = u_i\ \} \quad \text{and} \quad \mathcal{I} = \{\ i\ \mid\ \ell_i < u_i\ \}.$$

We denote by $\mathcal{C}_{\ell,i}$ and $\mathcal{C}_{u,i}$ the sets where the bounds on constraint i are binding:

$$\mathcal{C}_{\ell,i} = \left\{\ y\ \mid\ a_i^T y = \ell_i\ \right\} \quad \text{and} \quad \mathcal{C}_{u,i} = \left\{\ y\ \mid\ a_i^T y = u_i\ \right\}.$$

These sets are faces of the feasible polyhedron. Of interest to us are the outward-pointing normals to the faces within a prescribed distance of x.

The presence of equality constraints slightly complicates the identification of inequality constraints near a given iterate. Figure 4.1(a) illustrates the desirability of measuring the distances to nearly binding inequality constraints only along directions that are feasible with respect to equality constraints. The horizontal line in Figure 4.1(a) represents an equality constraint, while the line at an oblique angle represents an inequality constraint. The shaded region represents the set of points feasible with respect to the inequality constraint. The inequality constraint is in close proximity to the equality constraint near x but only at points that are infeasible with respect to the equality constraint (a cross indicates where both constraints are active).

We compute the distance to nearby binding inequality constraints along directions that are feasible with respect to equality constraints as follows. Let $\mathcal{N}$ denote the nullspace of the equality constraints (i.e., the matrix whose rows are a_i^T for $i \in \mathcal{E}$), and let Z be an $n \times r$ orthogonal matrix whose columns are a basis for $\mathcal{N}$. Given $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$, let $S = \left\{\ y\ \mid\ a^T y = b\ \right\}$. We then define

$$\operatorname{dist}_{\mathcal{N}}(x, S) = \begin{cases} |\, a^T x - b\,| / \| \, Z^T a \, \| & \text{if } Z^T a \neq 0, \\ 0 & \text{if } Z^T a = 0 \text{ and } a^T x = b, \\ \infty & \text{if } Z^T a = 0 \text{ and } a^T x \neq b. \end{cases}$$

This is the distance from x to S inside the nullspace $\mathcal{N}$. If S is parallel to $\mathcal{N}$, as is the case for the inequality constraint in Figure 4.1(b), we set this distance to be ∞, unless $x \in S$, in which case the distance is 0, as is true for the equality constraint in Figure 4.1(b).

Using this specialized notion of distance, we define the sets of inequalities at their bounds that are within distance ε of x:

$$\mathcal{I}_\ell(x,\varepsilon) = \{\ i \in \mathcal{I}\ \mid\ \mathrm{dist}_{\mathcal{N}}(x, \mathcal{C}_{\ell,i}) \le \varepsilon\ \},$$
$$\mathcal{I}_u(x,\varepsilon) = \{\ i \in \mathcal{I}\ \mid\ \mathrm{dist}_{\mathcal{N}}(x, \mathcal{C}_{u,i}) \le \varepsilon\ \}.$$

Let $\mathcal{I}_E(x,\varepsilon) = \mathcal{I}_\ell(x,\varepsilon) \cap \mathcal{I}_u(x,\varepsilon)$.

We now define the working set of constraints. Given a feasible x and an $\varepsilon > 0$, the corresponding working set $\mathcal{W}(x,\varepsilon)$ is made up of two pieces, the first consisting of vectors we treat as defining equalities and the second consisting of vectors we treat as defining inequalities. The first piece is

$$\mathcal{W}_E(x,\varepsilon) = \{\ a_i\ \mid\ i \in \mathcal{E}\ \} \cup \{\ a_i\ \mid\ i \in \mathcal{I}_E(x,\varepsilon)\ \},$$

which corresponds to the equality constraints together with every inequality for which the faces for both its lower and upper bounds are within distance ε of x, as in Figure 4.1(c). The second piece is

$$\mathcal{W}_I(x,\varepsilon) = \{\ -a_i\ \mid\ i \in \mathcal{I}_\ell(x,\varepsilon) \setminus \mathcal{I}_E(x,\varepsilon)\ \} \cup \{\ a_i\ \mid\ i \in \mathcal{I}_u(x,\varepsilon) \setminus \mathcal{I}_E(x,\varepsilon)\ \},$$

which is the set of outward-pointing normals to the inequalities for which the faces of exactly one of their lower or upper bounds is within distance ε of x. Note that, for the three examples illustrated in Figure 4.1, $\mathcal{W}_I(x,\varepsilon) = \emptyset$. The distinction between $\mathcal{W}_E$, which consists of constraints that are treated as equalities, and $\mathcal{W}_I$, which consists of constraints that are treated as inequalities, figures in the computation of the requisite search directions. We return to this point in section 5.

Given $x \in \Omega$, the *ε-normal cone*, denoted $N(x,\varepsilon)$, is the cone generated by the linear span of the vectors in $\mathcal{W}_E(x,\varepsilon)$ together with the nonnegative span of the vectors in $\mathcal{W}_I(x,\varepsilon)$:

$$N(x,\varepsilon) = \left\{ \sum_{a_i \in \mathcal{W}_E(x,\varepsilon)} u_i a_i + \sum_{a_i \in \mathcal{W}_I(x,\varepsilon)} \xi_i a_i\ \mid\ \xi_i \ge 0 \right\}. \tag{4.1}$$

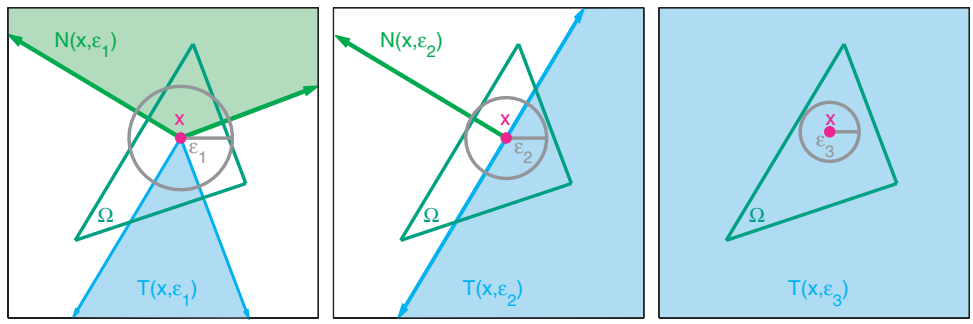
If both of the latter sets are empty, then $N(x,\varepsilon) = \{\mathbf{0}\}$. The *$\varepsilon$-tangent cone*, denoted by $T(x,\varepsilon)$, is the polar of $N(x,\varepsilon)$:

$$T(x,\varepsilon) = \left\{\ v\ \mid\ w^T v \le 0 \text{ for all } w \in N(x,\varepsilon)\ \right\}.$$

If $N(x,\varepsilon) = \{\mathbf{0}\}$, then $T(x,\varepsilon) = \mathbb{R}^n$. In addition to the examples illustrated in Figure 4.1, three examples involving only inequality constraints with three different values of ε are illustrated in Figure 4.2.

The set $x + T(x,\varepsilon)$ approximates the feasible region near x, where "near" is in terms of ε as measured using $\mathrm{dist}_{\mathcal{N}}(x,S)$. The cone $T(x,\varepsilon)$ is important because if $T(x,\varepsilon) \ne \{\mathbf{0}\}$, then one can proceed from x along all directions in $T(x,\varepsilon)$ for a distance of at least ε and still remain inside the feasible region [20, 17]. The examples in Figures 4.1–4.2 illustrate this point.

In the algorithm, we allow different values of ε_k at every iteration k. There is a lot of flexibility in determining the choice of ε_k, as discussed in [17]. We have chosen to yoke ε_k to Δ_k, for the reasons given in [17].



(a) Cones for ε_1. (b) Cones for ε_2. (c) Cones for ε_3.

FIG. 4.2. *The cones $N(x,\varepsilon)$ and $T(x,\varepsilon)$ for three values of ε. Note that for this example, as ε varies from ε_1 to 0, there are only the three distinct pairs of cones ($N(x,\varepsilon_3) = \{\mathbf{0}\}$).*

5. Generating the core set of search directions. Once the nearby constraints for a given iteration have been identified, we select the search directions. To ensure that the convergence results in [17] hold, we make the search conform to the local geometry of Ω in the sense that, at each iteration k, there is at least one direction in $\mathcal{D}_k$, the set of search directions at iteration k, along which the search can take an acceptably long step and still remain feasible.

We decompose $\mathcal{D}_k$ into $\mathcal{G}_k$ and $\mathcal{H}_k$, where $\mathcal{G}_k$ is made up of the directions necessary to ensure desirable convergence properties and $\mathcal{H}_k$ contains any additional directions. We sometimes refer to $\mathcal{G}_k$ as the *core set* of search directions. Several options for choosing $\mathcal{G}_k$ have been proposed:

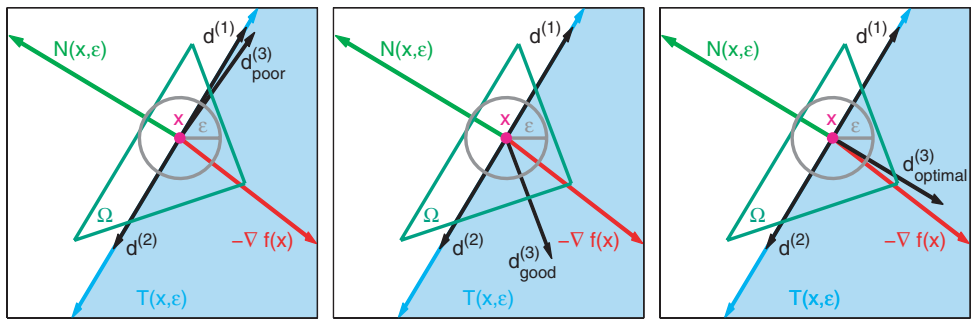
1. $\mathcal{G}_k$ contains generators for all of the cones $T(x_k,\varepsilon)$ for all $\varepsilon \in [0,\varepsilon_*]$, where $\varepsilon_* > 0$ is independent of k [23, 20].
2. $\mathcal{G}_k$ is exactly the set of generators for the cone $T(x_k,\varepsilon_k)$, where $\varepsilon_k \to 0$ is updated according to whether acceptable steps are found, and $\mathcal{H}_k = \emptyset$ [21].
3. $\mathcal{G}_k$ contains generators for the cone $T(x_k,\varepsilon_k)$, where $\varepsilon_k = \Delta_k \beta_{\max}$, and $\beta_{\max}$ is described in [17].

Here we use the third option, with its explicit link between ε_k and Δ_k, and choose $\beta_{\max} = 1$. As discussed further in section 6, the third option allows us to use a nonempty set $\mathcal{H}_k$ of additional directions chosen so as to accelerate the overall progress of the search. For any of the three options, the following is required of $\mathcal{G}_k$ [20, 17]:

> CONDITION 1. There exists a constant $\kappa_{\min} > 0$, independent of k, such that, for every k for which $T(x_k,\varepsilon_k) \neq \{\mathbf{0}\}$, the set $\mathcal{G}_k$ generates $T(x_k,\varepsilon_k)$ and satisfies $\kappa(\mathcal{G}_k) \geq \kappa_{\min}$.

Observe that the number of distinct ε-normal cones (and consequently the number of distinct ε-tangent cones) is finite.

The following simple technique, outlined in [17], ensures Condition 1 is satisfied. Let $k_2 > k_1$. If $\mathcal{W}_E(x_{k_2},\varepsilon_{k_2}) = \mathcal{W}_E(x_{k_1},\varepsilon_{k_1})$ and $\mathcal{W}_I(x_{k_2},\varepsilon_{k_2}) = \mathcal{W}_I(x_{k_1},\varepsilon_{k_1})$, then use the same generators for $T(x_{k_2},\varepsilon_{k_2})$ as were used for $T(x_{k_1},\varepsilon_{k_1})$. There are at most 2^m distinct sets $\mathcal{G}$ if the same set of generators is always used to generate a particular ε-tangent cone. Recall that m is the number of linear constraints. Since by Proposition 3.1 each $\mathcal{G}_k \neq \{\mathbf{0}\}$ has a strictly positive value for $\kappa(\mathcal{G}_k)$, and since this



(a) Poor choice (b) Reasonable choice (c) "Optimal" choice (w.r.t. $\kappa(\mathcal{G}_k)$)

FIG. 5.1. *Condition* 1 *is needed to avoid a poor choice when choosing* $\mathcal{G}_k$.

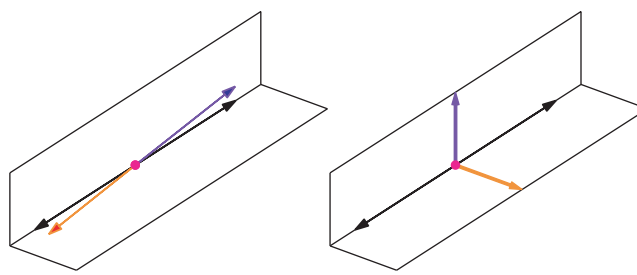


FIG. 5.2. *On the right, a set of generators for the cone (which includes a lineality space along the fold) that satisfies Condition* 1 *with a relatively large value of* $\kappa_{\min}$. *On the left, a different set of generators for which* $\kappa_{\min}$ *is much closer to zero.*

technique ensures there are only finitely many $\mathcal{G}_k$'s, we can set $\kappa_{\min} = \min\{\kappa(\mathcal{G}_k) : T(x_k, \varepsilon_k) \neq \{\mathbf{0}\}\}$. Thus, Condition 1 is satisfied.

The need for Condition 1 arises when there is a lineality space present in $T(x_k, \varepsilon_k)$. In this case the directions of the generators $\mathcal{G}_k$ are not uniquely determined. Figure 5.1 depicts the situation where $T(x_k, \varepsilon_k)$ contains a halfspace. Condition 1 enforces a lower bound on $\kappa(\mathcal{G}_k)$ to avoid the situation illustrated in Figure 5.1(a), in which the search directions can be almost orthogonal to the direction of steepest descent. Figure 5.1(c), which uses a minimal number of vectors but ensures as large a value for $\kappa(\mathcal{G}_k)$ as possible, represents an optimal choice in the absence of explicit knowledge of the gradient.

In $\mathbb{R}^2$ the presence of a lineality space in $T(x, \varepsilon)$ means that $T(x, \varepsilon)$ is either a halfspace, as illustrated in Figures 4.2(b) and 5.1, or $\mathbb{R}^2$, as illustrated in Figure 4.2(c). In more than two dimensions there are other geometrical possibilities that cause the directions of the generators of $T(x, \varepsilon)$ not to be uniquely determined, as illustrated in Figure 5.2.

As discussed in [17], we need to enforce lower and upper bounds on the lengths of the search directions. Here we accomplish this by simply normalizing all of the search directions.

The necessary conditions on the set of search directions all involve the calculation of generators for the polars of cones determined by the current working set. We treat

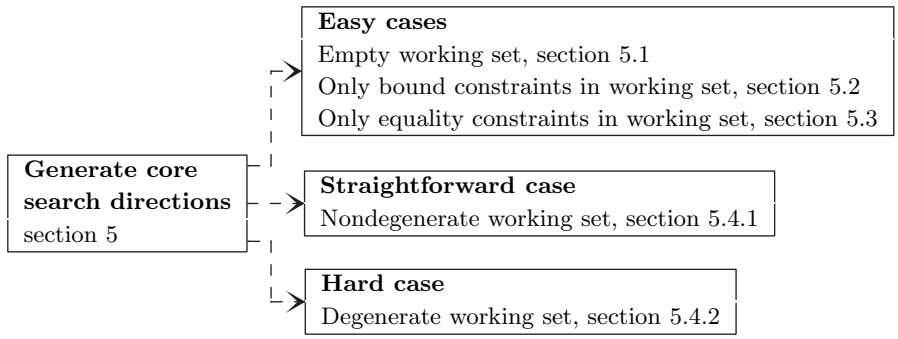


FIG. 5.3. *The components involved in the generation of the search directions.*

the determination of the search directions in increasing order of difficulty. Figure 5.3 shows the various cases to consider.

5.1. When the working set is empty. If the working set is empty, then $N(x_k, \varepsilon_k) = \{\mathbf{0}\}$ and $T(x_k, \varepsilon_k) = \mathbb{R}^n$, and any positive spanning set of $\mathbb{R}^n$ suffices as $\mathcal{G}_k$ [30, 18]. We use the unit coordinate vectors $\pm e_i$, $i = 1, \ldots, n$, to form $\mathcal{G}_k$ both because they are apt for many applications and because the orthonormality of the vectors ensures a reasonable value for $\kappa_{\min}$ in Condition 1. In our testing we have found that the coordinate directions are a consistently effective set of search directions for scientific and engineering problems. The coordinate directions have physical meaning for such applications; thus using them to form the core set of search directions $\mathcal{G}_k$ both guarantees a positive basis and seems to lead to better overall performance.

5.2. When the working set contains only bound constraints. A similar situation holds if only bound constraints are present. This case is also simple since we know a priori that the generators for the ε-normal and ε-tangent cones can be drawn from the set of coordinate directions [19]. Thus we include in $\mathcal{G}_k$ the unit coordinate vectors $\pm e_i$, $i = 1, \ldots, n$. As noted in [20, section 8.3], if not all of the variables are bounded, then one can make a choice of $\mathcal{G}_k$ that is more parsimonious in the number of directions, but we did not choose to do so in our implementation.

5.3. When the working set consists only of equality constraints. If the working set consists only of equality constraints, then the generators of $T(x_k, \varepsilon_k)$ correspond to a positive spanning set for the nullspace of the vectors in $\mathcal{W}_E(x_k, \varepsilon_k)$. In this situation by default we compute an orthonormal basis Z for this nullspace and take as our generators the "coordinatelike" set consisting of the elements of Z and their negatives.

5.4. When the working set contains general linear constraints. Once the nearby constraints have been identified using x_k and ε_k, their outward-pointing normals are extracted from the rows of the constraint matrix A to form the working set of constraints. So, for instance, in the case shown in Figure 5.4 the outward-pointing normals a_1 and a_2 constitute the working set $\mathcal{W}_I(x_k, \varepsilon_k) = \{a_1, a_2\}$ (there are no equality constraints in this example). There are then two cases to consider: when the working set is known to be nondegenerate and when the working set may be degenerate.

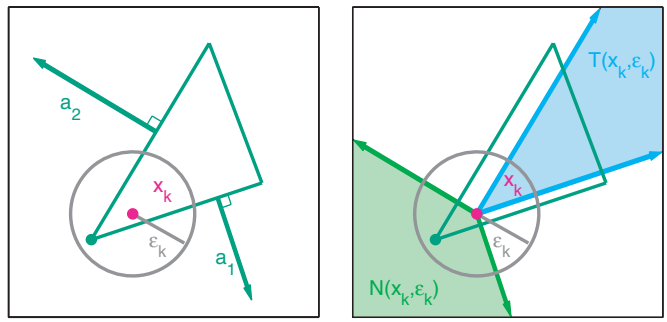


FIG. 5.4. *The outward-pointing normals a_1 and a_2 of the nearby constraints are translated to x_k to form the ε-normal cone $N(x_k, \varepsilon_k)$ and thus define its polar the ε-tangent cone $T(x_k, \varepsilon_k)$. The normalized generators for $N(x_k, \varepsilon_k)$ and $T(x_k, \varepsilon_k)$ form the set of search directions.*

5.4.1. When the working set is known to be nondegenerate. Figure 5.4 depicts the computationally straightforward case.

We begin with two general propositions concerning the polar of a finitely generated cone K. Proposition 5.1 is an elaboration of the construction given in [20, section 8.2] and shows how one can construct generators for K° if the generators of K are linearly independent. In particular, this means that K cannot contain a linear subspace. In those situations where K does contain a linear subspace, for instance, as illustrated in Figures 5.1–5.2, Proposition 5.2 enables us to reduce the situation to one where we can apply Proposition 5.1.

PROPOSITION 5.1. *Suppose K is generated by the positive span of the columns of the matrix $Q = [q_1 \dots q_r]$. Let N be a matrix whose columns are a positive spanning set for the nullspace of Q^T. Finally, suppose Q^T has a right inverse R.*

Then K° is the positive span of the columns of $-R$ together with the linear span of the columns of N:

$$K^\circ = \{\, w \mid w = -Ru + N\xi, \quad u \geq 0 \,\}.$$

Proof. Let $C = \{\, w \mid w = -Ru + N\xi, \quad u \geq 0 \,\}$. We first show that $C \subset K^\circ$. Suppose $d \in C$. To show that $d \in K^\circ$, we must show that $q_i^T d \leq 0$ for all $i = 1, \dots, r$, i.e., $Q^T d \leq 0$. Since $Q^T R = I$ and $Q^T N = 0$,

$$Q^T d = Q^T(-Ru + N\xi) = -u \leq 0.$$

Therefore $d \in K^\circ$, so $C \subset K^\circ$.

We next show that $C^\circ \subset K$, whence $K^\circ \subset C$, and so $K^\circ = C$. If $d \in C^\circ$, then $d^T(-Ru + N\xi) \leq 0$ for $u \geq 0$ and for all ξ. In particular, $d^T N\xi = 0$ for all ξ, so d is orthogonal to the positive span of the columns of N, which is the nullspace of Q^T. The latter means that d lies in the range of Q, so $d = Qs$ for some s. At the same time, we know that $-d^T Ru \geq 0$ for all $u \geq 0$, whence $-s^T Q^T Ru = -s^T u \leq 0$ for all $u \geq 0$, so $s \geq 0$. Thus $d = Qs$ for $s \geq 0$, meaning $d \in K$ and $C^\circ \subset K$. ∎

Proposition 5.2 says that if the cone K contains a linear subspace L, then computing K° is a matter of looking at the polar of $K \cap L^\perp$.

PROPOSITION 5.2. *Suppose that the cone K can be written as the linear span of the vectors $v_1, \dots, v_q$ together with the nonnegative span of the vectors $p_1, \dots, p_r$:*

$$K = \{\, v \mid v = \alpha_1 v_1 + \cdots + \alpha_q v_q + \lambda_1 p_1 + \cdots + \lambda_r p_r, \ \lambda_i \geq 0 \text{ for } i = 1, \dots, r \,\}.$$

Step 1. Let L be the linear subspace spanned by the vectors in $\mathcal{W}_E(x_k, \varepsilon_k)$. Compute a basis Z for $L^\perp$.

Step 2. Check whether the set of vectors $\{ Z^T p \mid p \in \mathcal{W}_I(x_k, \varepsilon_k) \}$ is linearly independent. If so, proceed to Step 3. If not, proceed to the more complicated construction described in section 5.4.2.

Step 3. Let Q be the matrix whose columns are the vectors $Z^T p$ for $p \in \mathcal{W}_I(x_k, \varepsilon_k)$. Compute a right inverse R for Q^T and a matrix N whose columns are a positive spanning set for the nullspace of Q^T.

Step 4. $\mathcal{G}_k$ is then the columns of $-ZR$ together the columns of ZN.

FIG. 5.5. *The process for attempting the straightforward construction of* $\mathcal{G}_k$.

Let L be the linear subspace spanned by $v_1, \ldots, v_q$. Then

$$K^\circ = \{ w \in L^\perp \mid w^T p_i \le 0 \text{ for all } i = 1, \ldots, r \}.$$

Proof. Let $P = \{ w \in L^\perp \mid w^T p_i \le 0 \text{ for all } i = 1, \ldots, r \}$. Clearly, $P \subset K^\circ$. Conversely, if $w \in K^\circ$, then we have $w^T p_i \le 0$ for all $i = 1, \ldots, r$. In addition, since L is contained in K we have $v_i \in K$ and $-v_i \in K$ for all $i = 1, \ldots, q$. Thus we have $w^T v_i \le 0$ and $-w^T v_i \le 0$, so $w^T v_i = 0$ and $w \in L^\perp$, whence $K^\circ \subset P$. ∎

We use Propositions 5.1 and 5.2 to compute generators for $T(x_k, \varepsilon_k) = N(x_k, \varepsilon_k)^\circ$ as follows. Recall the definition of $N(x_k, \varepsilon_k)$ in (4.1). Let L denote the linear span of the vectors in $\mathcal{W}_E(x_k, \varepsilon_k)$. Let Z denote a matrix whose columns span $L^\perp$. Applying Proposition 5.2 with $K = N(x_k, \varepsilon_k)$ then says

$$\begin{aligned} T(x_k, \varepsilon_k) &= \{ w \in L^\perp \mid w^T p \le 0 \text{ for all } p \in \mathcal{W}_I(x_k, \varepsilon_k) \} \\ &= \{ Zc \mid c^T Z^T p \le 0 \text{ for all } p \in \mathcal{W}_I(x_k, \varepsilon_k) \}. \end{aligned}$$

The latter condition says that the set of c of interest to us is actually the polar of the cone $\tilde{K}$ generated by the vectors $Z^T p$ for all $p \in \mathcal{W}_I(x_k, \varepsilon_k)$. If this latter set of vectors is linearly independent, we can apply Proposition 5.1 to obtain generators for $\tilde{K}^\circ$ and then map them back to the original space under the action of Z. If they are not linearly independent, we must use the more sophisticated approach discussed in section 5.4.2. The procedure is summarized in Figure 5.5.

In our implementation we compute an orthonormal basis Z in Step 1. In Step 3 we use the pseudoinverse of Q^T as the right inverse. This choice is attractive because the columns of the pseudoinverse are orthogonal to the lineality space of $T(x_k, \varepsilon_k)$ (as in the examples on the right in Figures 5.1–5.2), which helps improve the conditioning of the resulting set of search directions (Condition 1).

Also in Step 3 our implementation computes the positive spanning set N by computing an orthonormal basis B for the nullspace of Q^T and setting $N = [B \ -B]$. This corresponds to a coordinatelike search in the nullspace of Q^T.

5.4.2. When the working set may be degenerate. If the construction described in Figure 5.5 cannot be applied, because linear dependence of the set of vectors $\{ Z^T p \mid p \in \mathcal{W}_I(x_k, \varepsilon_k) \}$ is detected, the computation of the search directions

is more complicated. In this case $N(x_k, \varepsilon_k)$ is a cone with a degenerate vertex at the origin, so we refer to this situation as the degenerate case. (The easiest example to envision is the apex defined by the four triangular faces of a pyramid.)

The problem of finding a set of generators for $T(x_k, \varepsilon_k)$ is a special case of the problem of finding the vertices and extreme rays of a polyhedron (the V-representation of the polyhedron) given the description of the polyhedron as an intersection of halfspaces (the H-representation). This is a well-studied problem in computational geometry (see [2] for a survey) for which practical algorithms have been developed.

The two main classes of algorithms for solving this type of problem are pivoting algorithms and insertion algorithms. Pivoting algorithms move from vertex to vertex along the edges of the polyhedron by generating feasible bases, as in the simplex method. The reverse search algorithm [1, 3] is an example of such a method.

Insertion algorithms compute the V-description by working with the intersection of an increasing number of halfspaces. Typically, for a polyhedral cone these algorithms start with a maximal linearly independent set of normals defining the halfspaces and construct an initial set of extreme rays as in Figure 5.5. Additional halfspaces are then introduced, and the extreme rays are recomputed as positive combinations of existing extreme rays; this process continues until the entire set of halfspaces has been introduced. The double description method [26] is an example of an insertion algorithm. A discussion of the double description method, along with a description of the techniques used in a practical implementation, can be found in [9].

As observed in [2], pivoting algorithms can be inefficient if they encounter a highly degenerate vertex. For this reason we chose the double description method. We use Fukuda's `cddlib` package [11], which is based on an efficient implementation of the double description algorithm that directly addresses known—and serious—pitfalls of a naive implementation of the double description method and that has been shown to be effective in dealing with highly degenerate vertices [9]. We conducted our own comparison of Avis's `lrs` reverse search code [3] and Fukuda's `cdd` double description code [11] on problems we knew to be highly degenerate. Our experience confirmed the observation made in [10]: `cdd` tends to be efficient for highly degenerate inputs, while `lrs` tends to be efficient for nondegenerate or slightly degenerate problems.

Another observation made in [10] that bears repeating is that the number of inequalities defining the polyhedron does not by itself indicate the degree of difficulty of a vertex/ray enumeration problem. It is noted in [10] that `cdd` can handle a highly degenerate problem with 711 inequalities in 19 dimensions quite easily, while a problem with 729 inequalities in 8 dimensions is extremely difficult to solve. Our own experience confirms this, as discussed further in section 9.

During our testing with `cdd`, one useful lesson we learned is that it is more efficient to retain equalities in the problem rather than to eliminate the equalities by reparameterizing the problem in terms of a basis for the nullspace of the equality constraints. We initially thought that using the equalities to eliminate variables and reduce the dimension of the problem would be an advantage. But this did not prove to be the case. In several instances, elimination of equalities led to a significant deterioration in the performance of `cdd`. This is due to the fact that the arrays of constraint coefficients went from being sparse—and frequently integral—to being dense real arrays, which greatly hindered the progress of the double description method.

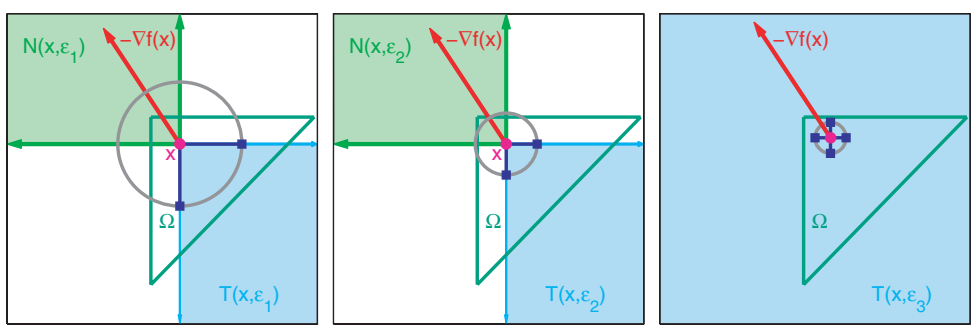
We close by noting that only a subset of the generators for $T(x_k, \varepsilon_k)$ may suffice at some iterations. For example, if an iteration is successful, the iteration can be terminated as soon as at least one of the search directions yields a trial point that

satisfies the acceptance criterion. Thus, the generators could be computed on an "as needed" basis (though, for an unsuccessful iteration, ultimately a complete set of generators for $T(x_k, \varepsilon_k)$ will be required). Such an approach would be akin to that used by the pivoting algorithms. But, as already noted, the pivoting algorithms can be inefficient if they encounter a highly degenerate vertex. Since `cdd` has proven efficient on all but one of the problems we have tested, we currently compute all of the generators for $T(x_k, \varepsilon_k)$ at each iteration for which the working set changes.

6. Augmenting the core set of search directions. While generators of $T(x_k, \varepsilon_k)$ are theoretically sufficient for the convergence analysis, in practice there can be appreciable benefits to including additional search directions. The situation illustrated in Figure 6.1 shows the desirability of additional directions, since the generators for $T(x_k, \varepsilon_k)$ necessarily point into the relative interior of the feasible region, while it may be preferable to move toward the boundary.

At the very least, we would like the total set of search directions to be a positive basis for $\mathbb{R}^n$ or, if equality constraints are present, a positive basis for the nullspace of the equality constraints. If $\mathcal{W}_E(x_k, \varepsilon_k)$ is empty, one simple way to accomplish this is to include in $\mathcal{H}_k$ the set $\mathcal{W}_I(x_k, \varepsilon_k)$, the outward-pointing normals to the working set of constraints. Since these vectors generate $N(x_k, \varepsilon_k)$, and since $T(x_k, \varepsilon_k)$ is polar to $N(x_k, \varepsilon_k)$, by the polar decomposition, $\mathbb{R}^n = N(x_k, \varepsilon_k) \oplus T(x_k, \varepsilon_k)$. Thus, including the vectors that generate $N(x_k, \varepsilon_k)$ in $\mathcal{H}_k$ means that $\mathcal{D}_k$ constitutes a positive spanning set for $\mathbb{R}^n$, as illustrated in Figure 6.2. See section 9 for a computational illustration of the advantage of including these directions.

The preceding discussion needs to be modified if equality constraints are present. Typically, in this case the outward-pointing normals a_i to constraints in the working set are infeasible directions—they do not lie in the nullspace of the equality constraints. Thus, a step along a direction a_i will lead to violation of the equality constraints. In this situation we replace the outward-pointing normals by their projection into the nullspace of the equality constraints.

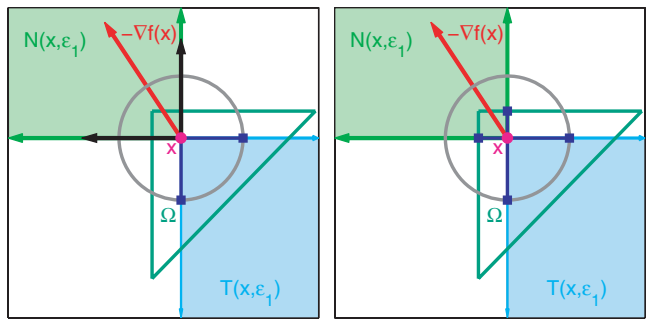


(a) First iteration is unsuccessful and so Δ_1 is halved for the next iteration.

(b) The second iteration is also unsuccessful and so Δ_2 is halved again.

(c) Now, locally, the problem looks unconstrained and a successful step toward the boundary is possible.

FIG. 6.1. *Minimizing the number of function evaluations at a single iteration might require more total function evaluations since in this instance it will take at least five function evaluations to find decrease.*



(a) Both generators of $N(x, \varepsilon_1)$ are descent directions.

(b) Limiting the steps along the generators of $N(x, \varepsilon_1)$ ensures feasible steps.

FIG. 6.2. *Why including the generators for the ε-normal cone $N(x, \varepsilon_1)$ might be a good idea since decrease may be found after as few as one function evaluation.*

7. Determining the lengths of the steps. The *step-length control parameter* Δ_k determines the length of a step along each direction for a given set of search directions. In the presence of constraints, we must consider the possibility that some of the trial points $x_k + \Delta_k d_k^{(i)}$, $d_k^{(i)} \in \mathcal{D}_k$, may be infeasible. With this in mind, we choose the maximum $\tilde{\Delta}_k^{(i)} \in [0, \Delta_k]$ such that $x_k + \tilde{\Delta}_k^{(i)} d_k^{(i)} \in \Omega$, thereby ensuring that a full step will be taken, if feasible.

It is possible to choose $\tilde{\Delta}_k^{(i)} = \Delta_k$ if the result will be feasible and $\tilde{\Delta}_k^{(i)} = 0$ otherwise, thereby rejecting infeasible points. This simple strategy was illustrated in the initial example given in Figure 2.1. We have found it much more efficient, however, to allow steps that stop at the boundary, as suggested in [22, 21]. The latter strategy is illustrated in Figure 6.2(b).

For the search directions in $\mathcal{H}_k$ we actually choose $\tilde{\Delta}_k^{(i)}$ to be the maximum value in $[\sigma_{\text{tol}} \Delta_k, \Delta_k]$ satisfying $x_k + \tilde{\Delta}_k^{(i)} d_k^{(i)} \in \Omega$, where $\sigma_{\text{tol}} \geq 0$. The factor σ_{tol} prevents excessively short steps to the boundary, relative to the current value of Δ_k. Our experience has been that such steps typically yield little improvement and have the overall effect of retarding decrease in Δ_k.

8. Linearly constrained GSS. The variant of GSS that we have implemented for problems with linear constraints is outlined in Figure 8.1. See [17] for discussions of some of the many other possible variations on the specification of GSS algorithms for linearly constrained optimization. The remainder of this section is devoted to further explanation of the algorithm given in Figure 8.1.

8.1. Finding an initial feasible iterate. GSS methods for linearly constrained problems are feasible iterates methods—all iterates x_k, $k = 0, 1, 2, \ldots$, satisfy $x_k \in \Omega$. If an infeasible initial iterate is given, then we compute its Euclidean projection onto the feasible polyhedron; this is a quadratic program. For some of the test problems discussed in section 9 the starting value provided is infeasible, and for these we used this projection strategy. Alternatively, one could use phase one of the simplex method to obtain a starting point at a vertex of the feasible region.

ALGORITHM 8.1 (LINEARLY CONSTRAINED GSS USING A SUFFICIENT DECREASE GLOBALIZATION STRATEGY)

INITIALIZATION.

Let $f : \mathbb{R}^n \to \mathbb{R}$ be given. Let Ω be the feasible region.

Let $x_0 \in \Omega$ be the initial guess.

Let $\Delta_{\text{tol}} > 0$ be the step-length convergence tolerance.

Let $\Delta_0 > \Delta_{\text{tol}}$ be the initial value of the step-length control parameter.

Let $\sigma_{\text{tol}} \geq 0$ control the lower bound on the length of successful steps.

Let $\varepsilon_{\max} > \Delta_{\text{tol}}$ be the maximum distance used to identify nearby constraints ($\varepsilon_{\max} = +\infty$ is permissible).

Let $\rho(\Delta) = \alpha\Delta^2$ be the forcing function, with $\alpha > 0$.

ALGORITHM. For each iteration $k = 0, 1, 2, \ldots$

Step 1. Let $\varepsilon_k = \min\{\varepsilon_{\max}, \Delta_k\}$. Choose a set of normalized search directions $\mathcal{D}_k = \mathcal{G}_k \cup \mathcal{H}_k$ satisfying Condition 1.

Step 2. If there exists $d_k^{(i)} \in \mathcal{D}_k$, together with the maximum $\tilde{\Delta}_k^{(i)} \in [\sigma_{\text{tol}}\Delta_k, \Delta_k]$ such that $x_k + \tilde{\Delta}_k d_k \in \Omega$, which satisfy

$$f\left(x_k + \tilde{\Delta}_k^{(i)} d_k^{(i)}\right) < f(x_k) - \rho(\Delta_k),$$

then:

- Set $x_{k+1} = x_k + \tilde{\Delta}_k^{(i)} d_k^{(i)}$ (a successful step).
- Set $\Delta_{k+1} = \Delta_k$.

Step 3. Otherwise, for every $d \in \mathcal{G}_k$,

$$f(x_k + \Delta_k d) \geq f(x_k) - \rho(\Delta_k).$$

In this case:

- Set $x_{k+1} = x_k$ (an unsuccessful step).
- Set $\Delta_{k+1} = \frac{1}{2}\Delta_k$.

If $\Delta_{k+1} < \Delta_{\text{tol}}$, then terminate.

FIG. 8.1. *Linearly constrained GSS using a sufficient decrease globalization strategy.*

8.2. Accepting steps. We opted to impose the sufficient decrease condition

$$f(x_k + \tilde{\Delta}_k d_k) < f(x_k) - \rho(\Delta_k),$$

with $\rho(\Delta) = \alpha\Delta^2$. For any given problem the value of α should be chosen to reflect the relative magnitude of f and Δ. Our primary reason for using a sufficient decrease condition was that, as discussed in section 7 (and illustrated in Figure 6.2(b)), it allows

us to choose $\tilde{\Delta}_k^{(i)}$ to be the maximum value in $[\sigma_{\text{tol}}\Delta_k, \Delta_k]$ satisfying $x_k + \tilde{\Delta}_k^{(i)} d_k^{(i)} \in \Omega$. Furthermore, as shown in [17, section 6], this particular choice of $\rho(\Delta)$ ensures the useful estimate (8.1), which we discuss in the next section.

For the results reported here we used the classical update for Δ_k. If $k \in \mathcal{S}$, then $\Delta_{k+1} = \phi_k \Delta_k$, with $\phi_k = 1$ for all k; otherwise, $\Delta_{k+1} = \theta_k \Delta_k$, with $\theta_k = \frac{1}{2}$ for all k.

8.3. Stopping criteria. We stop the algorithm once the step-length parameter Δ_k falls below a specified tolerance Δ_{tol}. This stopping criterion is motivated by the relationship between Δ_k and $\chi(x_k)$, a particular measure of progress toward a Karush–Kuhn–Tucker (KKT) point of (1.1). For $x \in \Omega$, let

$$\chi(x) \equiv \max_{\substack{x+w \in \Omega \\ \| w \| \leq 1}} -\nabla f(x)^T w.$$

Of interest to us here are the following properties [6, 5]: (1) $\chi(x)$ is continuous on Ω, (2) $\chi(x) \geq 0$, and (3) $\chi(x) = 0$ if and only if x is a KKT point for (1.1). We therefore wish to see $\chi(x_k)$ tend to zero.

In [17, Theorem 6.4] it is shown that once Δ_k is small enough, so that $\varepsilon_k = \Delta_k$, then

$$\chi(x_k) = O(\Delta_k) \text{ for } k \in \mathcal{U}. \tag{8.1}$$

Theorem 6.4 is most simply stated under the assumptions that the set $\mathcal{F} = \{ x \in \Omega \mid f(x) \leq f(x_0) \}$ is bounded and that the gradient of f is Lipschitz continuous with constant M on $\mathcal{F}$. Relaxations of these assumptions, along with a full statement and proof of the theorem, are found in [17].

The big-O relationship between $\chi(x_k)$ and Δ_k means that as Δ_k is reduced, which happens only at unsuccessful iterations, the upper bound on the value of the measure of stationarity is also reduced. Relationship (8.1) is analogous to the unconstrained minimization result (see [8, section 3] or [16, section 3.6]):

$$\| \nabla f(x_k) \| = O(\Delta_k) \text{ for } k \in \mathcal{U}. \tag{8.2}$$

Results (8.1) and (8.2) support using the magnitude of Δ_k as a test for termination.

Obviously, one can also stop after a specified number of objective evaluations, an option we exercise in our testing in section 9.

8.4. Scaling. The relative scaling of variables can have a profound effect on the efficiency of optimization algorithms. We allow as an option the common technique of shifting and rescaling the variables so that they have a similar range and size. We work in a computational space whose variables w are related to the original variables x via $x = Dw + c$, where D is a diagonal matrix with positive diagonal entries and c is a constant vector, both provided by the user. We solve the transformed problem

$$\begin{aligned} &\text{minimize} && f(Dw + c) \\ &\text{subject to} && \ell - Ac \;\leq\; ADw \;\leq\; u - Ac. \end{aligned}$$

The scaling scheme we use in the numerical tests presented in section 9 is based on that described in [13, section 7.5.1]. The latter scaling is based on the range of the variables and transforms the variables to lie between -1 and 1.

One advantage of scaling is that it makes it easier to define a reasonable default value for Δ_0. For the results reported in section 9, whenever scaling is applied to a problem, the default value for Δ_0 is 2 since the longest possible feasible step along a unit coordinate direction is 2 (the variables in the rescaled problem lie between -1 and 1).

8.5. Caching data. The potentially regular pattern to the steps the search takes means that the search may revisit points. This is particularly true for unconstrained and bound constrained problems. To avoid reevaluating known values of f, we store values of x at which we evaluate the objective and the corresponding objective values. Before evaluating f at a trial point w_t in the rescaled space, we examine the data cache to see whether we have already evaluated f at a point within some relative distance of w_t. We do not evaluate f at w_t if we have already evaluated f at a point $\bar{w}$ for which $\| w_t - \bar{w} \| < r_{\text{tol}} \| w_t \|$. By default, $r_{\text{tol}} = 10^{-8}$. This strategy allows us to take advantage of the vectorization within Matlab when searching the cache. For an alternative strategy based on splay trees, see [15].

Caching yields only minor improvements in efficiency if general linear inequality constraints are present. If there are a considerable number of general linear constraints (i.e., constraints other than bounds) in the working set, then there is only a small chance of reevaluating f near a previous iterate. Still, if each evaluation of the objective function is expensive, then it is worth avoiding reevaluations.

9. Some numerical illustrations. We present some illustrations of the behavior of our implementation of the algorithm given in Figure 8.1 using five test problems from the CUTEr test suite [14], listed in Table 9.1. All five problems have nonlinear objectives. We chose these specific problems for illustration since the algorithm must deal repeatedly with highly degenerate instances of the search direction calculation discussed in section 5.4.2. As the results in section 9.2 show, the solution of such problems is usually quite tractable given a good implementation of the double description algorithm.

9.1. Starting values and tolerances for the results reported here. We started with the value of x_0 specified by CUTEr, though we often had to project into the feasible region, as discussed in section 8.1.

We terminated the program either when the value of Δ_k fell below $\Delta_{\text{tol}} = \Delta_0/2^{20}$ or when the number of evaluations of the objective reached that which would have been needed for 20 centered difference evaluations of the gradient if one had used the equality constraints to eliminate variables. See Table 9.2 for the maximum number of evaluations allowed for each problem.

When the problem was scaled (as it was for all but LOADBAL), we used $\Delta_0 = 2$, as discussed in section 8.4. We used the default value $\sigma_{\text{tol}} = 10^{-3}$. We set $\varepsilon_{\max} = 2^5\Delta_0$ so that it played no role in the results reported here. We set $\alpha = 10^{-4}$ in the forcing function $\rho(\Delta) = \alpha\Delta^2$.

The core set of search directions $\mathcal{G}_k$ consisted of generators for the ε-tangent cone $T(x_k, \varepsilon_k)$. With one exception (for the purposes of illustrating the potential advantage of allowing a nonempty set $\mathcal{H}_k$) $\mathcal{H}_k$ consisted of generators for the ε-normal cone $N(x_k, \varepsilon_k)$ (projected into the nullspace of any equality constraints in the working set, as described in section 6). Recall that, by definition, $\mathcal{D}_k = \mathcal{G}_k \cup \mathcal{H}_k$.

TABLE 9.1
Problems used from the CUTEr test problem set.

		# of linear			# of bounds	
Problem	**n**	**Network equalities**	**Equalities**	**Inequalities**	**Lower**	**Upper**
AVION2	49	0	15	0	49	49
DALLASM	197	151	0	0	197	197
DALLASS	46	31	0	0	46	46
LOADBAL	31	11	0	20	31	11
SPANHYD	97	33	16	0	97	97

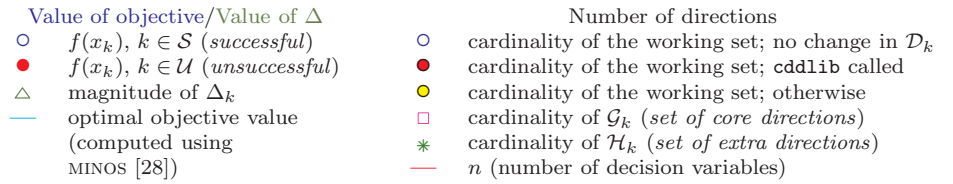


FIG. 9.1. *Legends for the remaining figures.*

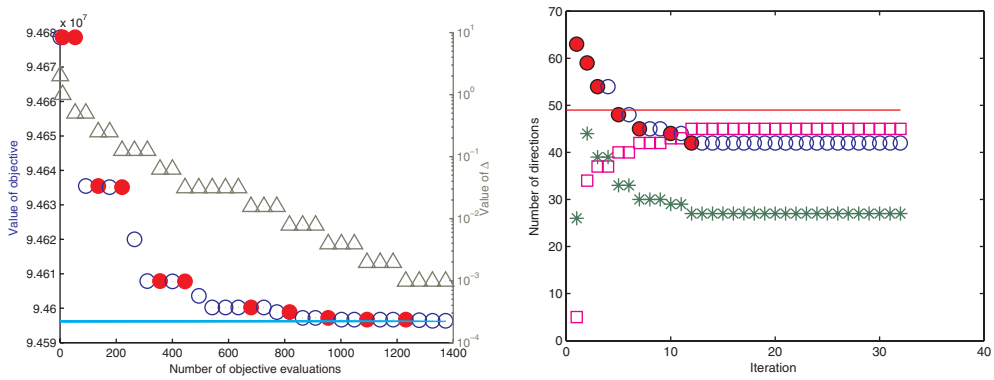


FIG. 9.2. *Results for the* AVION2 *test problem. Legends are given in Figure* 9.1.

The tests were run on a 1 processor, 3 GHz Pentium 4 workstation with 2 GB memory running Linux and using Matlab R2006b.

9.2. The results. Below we include a summary, along with illustrations, of the the iteration history for each problem. See Figure 9.1 for a legend.

The first problem is AVION2, an aeronautical design problem. In this case the equality constraints have rank 15, leaving 34 degrees of freedom in the problem. The plot on the left in Figure 9.2 gives the iteration history for the value of the objective and the value of Δ, both plotted against the number of evaluations of the objective. Solid circles indicate where unsuccessful iterations were encountered. The plot on the right in Figure 9.2 shows the size of the working set at each iteration, the number of vectors in $\mathcal{G}_k$, and the number of vectors in $\mathcal{H}_k$. Solid circles in this plot indicate iterations where the simple polar construction in section 5.4.1 was not applicable and consequently `cddlib` was called, as described in section 5.4.2. Open circles indicate iterations where the working set was unchanged from the previous iteration so no new calculation of the search directions was needed. The set $\mathcal{H}_k$ has fewer vectors than the working set because we do not include outward-pointing normals for equality constraints in the working set. The solid line indicates the number of decision variables in the problem; when the size of the working set is above this line, the algorithm must deal with the degeneracy discussed in section 5.4.2. Observe that degeneracy is also encountered when the size of the working set is smaller than the number of decision variables; this occurs because of redundant constraints.

The next test problem is SPANHYD, a model of a hydroelectric system. This problem includes 33 network equality constraints of rank 32 and 16 fixed variables, leaving 49 degrees of freedom in the problem. We modify the scaling scheme slightly by not scaling variables whose original range is narrow; specifically, we scale only the variables that vary over a range of more than 0.1. The plot on the left in Figure 9.3

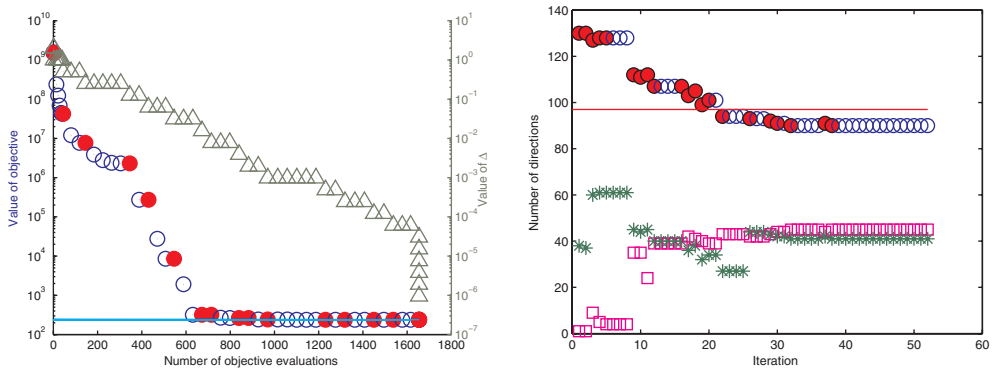


FIG. 9.3. *Results for the* SPANHYD *test problem. Legends are given in Figure* 9.1.

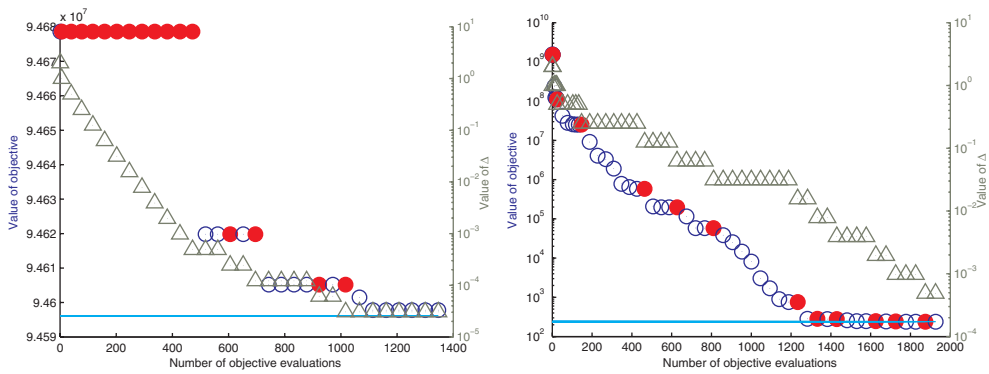


FIG. 9.4. *Progress versus cost history for the* AVION2 *(left) and* SPANHYD *(right) test problems if only the theoretically minimal set of search directions $\mathcal{D}_k = \mathcal{G}_k$ is used at each iteration k. Legend is given in Figure* 9.1.

shows that the algorithm makes good progress toward the optimal objective value. For this problem the test terminated because $\Delta_k \leq \Delta_{\text{tol}}$.

More interesting is the plot on the right, which shows that the degenerate case arises multiple times. Moreover, the plot on the right in Figure 9.3 reveals how badly one can be fooled by the naive bound on the potential number of search directions. For instance, at iteration 4 there are 128 constraints in the working set. Of these, 67 are equality constraints, with rank 66. Eliminating these degrees of freedom from the 97 variables still leaves brute force enumeration of

$$\begin{pmatrix} 128 - 67 \\ 97 - 66 \end{pmatrix} = \frac{61!}{31!\ 30!} \approx 2.33 \times 10^{17}$$

possible generators for $T(x_k, \varepsilon_k)$. There are, in fact, only 5 generators, which `cddlib` finds in less than 0.17 seconds.

Figure 9.4 illustrates the point made in section 6 about the desirability of using more than the theoretically minimal set of search directions. In this test we used only the generators of $T(x_k, \varepsilon_k)$ (the core set of search directions), setting $\mathcal{H}_k = \emptyset$. Comparisons with Figures 9.2 and 9.3 show that using the smaller set of search directions can appreciably retard the progress of the search.

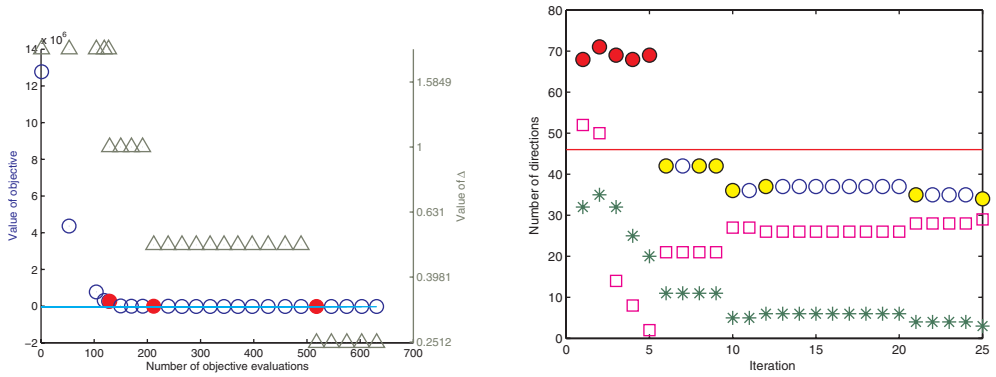


FIG. 9.5. *Results for the* DALLASS *test problem. Legends are given in Figure* 9.1.

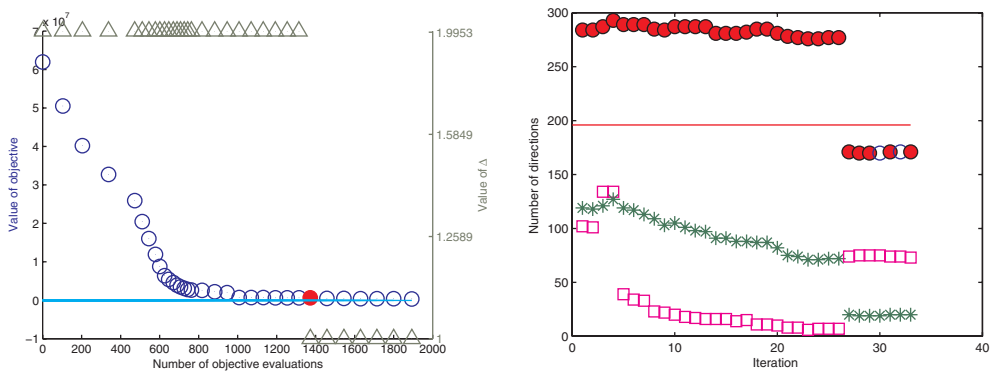


FIG. 9.6. *Results for the* DALLASM *test problem. Legends are given in Figure* 9.1.

Figure 9.5 gives results for the DALLASS test problem, a small model of the Dallas water system. The plot on the left shows that the problem is solved quickly. In the plot on the right, lighter solid circles indicate iterations where the simpler construction of the search directions given in section 5.4.1 was used. The degenerate case occurs only in the first five iterations.

In Figure 9.6 we give the results for the DALLASM test problem, a medium size model of the Dallas water system. While the algorithm makes progress towards the solution, reducing the value of the objective by a factor of 160, it does not solve the problem within the budget of objective evaluations allowed. The plot on the right shows that in the first 26 iterations there is considerable degeneracy in the working set. In the most extreme case, at iteration 4 there are 293 constraints in the working set, of which 151 are equalities with rank 150. Eliminating these degrees of freedom from the 197 variables leaves

$$\begin{pmatrix} 293 - 151 \\ 197 - 150 \end{pmatrix} = \frac{142!}{47!\ 95!} \approx 1.01 \times 10^{38}$$

possible generators for $T(x_k, \varepsilon_k)$. There are actually only 134 generators, which `cddlib` finds in about 4.3 seconds.

Figure 9.7 gives results for the LOADBAL problem, a load-balancing problem from computer science. Absent upper bounds on all of the variables, and with no prior

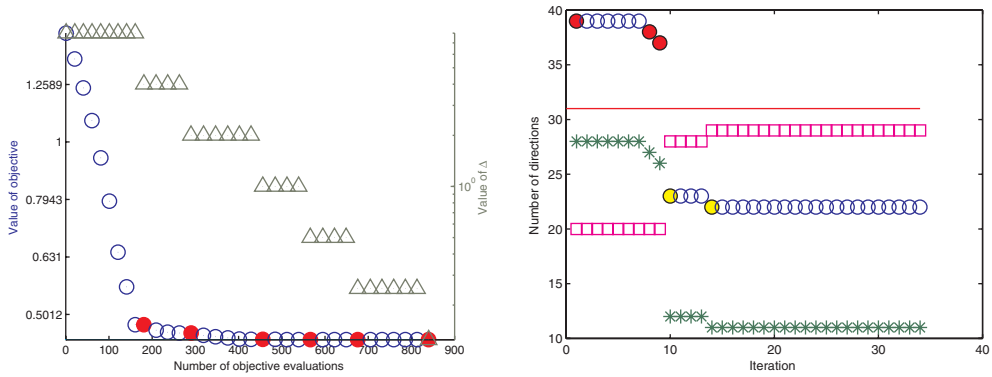


FIG. 9.7. *Results for the* LOADBAL *test problem. Legends are given in Figure* 9.1.

TABLE 9.2
Summary of test results.

CUTEr problem	Degrees of freedom	Stopping conditions		Total time (seconds)	
		Max evals.	Reason	Elapsed	In `cddlib`
AVION2	34	1380	Max evals.	2.47	0.15
DALLASM	47	1900	Max evals.	92.91	75.86
DALLASS	16	660	Max evals.	1.32	0.09
LOADBAL	21	860	Max evals.	1.56	0.02
SPANHYD	49	1980	$\Delta < \Delta_{\text{tol}}$	6.68	2.43

knowledge of reasonable ranges for the variables, we do not scale. We choose $\Delta_0 = 8$ based on the magnitude of the starting point. There is rapid progress toward the solution.

We summarize the test results in Table 9.2. In evaluating the timings, one should bear in mind that the cost of an objective evaluation was insignificant since the objectives are explicitly defined functions. In more realistic settings the cost of an objective evaluation would likely be much higher, so the relative cost of treating degenerate working sets should be appreciably less.

We tested our implementation on other problems from the CUTEr test set (e.g., HIMMELBI) and for most of them saw results comparable to the ones reported above. Two exceptions were QPCBOEI1 and QPCBOEI2. In both cases we encountered highly degenerate cases of the working set that exercised `cddlib`. For instance, in QPCBOEI1, a 384 variable problem, we encountered working sets with over 700 constraints for which `cddlib` took about 40 seconds to compute the core search directions. In QPCBOEI2, a 143 variable problem, we encountered a working set consisting of 30 equalities and 189 inequalities for which `cddlib` did not find a set of core search directions in an acceptable amount of time; we terminated the job after allowing `cddlib` to work for 12 hours.

10. Concluding remarks. Our numerical results illustrate the value of the condition defining the working set for generating set search introduced in [17]. This condition reduces the number of core search directions required in [20] while allowing the flexibility of introducing extra search directions. As the discussion in section 6 would suggest, we have found that augmenting the set of search directions beyond the theoretically minimal set typically leads to greater overall efficiency of the algorithm. We also have shown that state-of-the-art computational geometry methods make the

calculation of the search directions needed by generating set search tractable even in the degenerate case. In addition, the presentation here addresses the various details that arise when handling linear equality constraints. Finally, the introduction of linear constraints has allowed us to solve larger dimensional problems than is usually possible for the unconstrained case.

Our implementation gives us a basis for implementing a generating set search version of the augmented Lagrangian algorithm in [6]. In this approach, linear constraints are treated explicitly rather than being included in the augmented Lagrangian. The ability to treat degeneracy explicitly, as we do here, is important because the combination of general linear constraints and simple bounds frequently leads to degeneracy, and treatment of degeneracy by an augmented Lagrangian approach can be slow.

Future work includes using information collected during the progress of the algorithm to estimate the set of constraints binding at a solution as well as developing strategies for moving more quickly to solutions on the boundary.

Acknowledgments. Komei Fukuda has made his `cdd` package publicly available and graciously answered our questions about `cddlib`. Conversations with Rakesh Kumar about his experience implementing and testing the algorithm in [20] for the MathWorks Genetic Algorithm and Direct Search Toolbox [28] provided many useful insights when we tackled degenerate constraints. Josh Griffin read an early version of the paper and reported back numerous bugs and inconsistencies that we had overlooked. Tammy Kolda, Associate Editor, and two anonymous referees carefully read the paper and responded with numerous suggestions that appreciably improved the presentation. Stephen Nash also sent along several helpful observations concerning the first version of the paper. We extend our sincere thanks to all of these people.

REFERENCES

[1] D. M. Avis and K. Fukuda, *A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra*, Discrete Comput. Geom., 8 (1992), pp. 295–313.

[2] D. Avis, D. Bremner, and R. Seidel, *How good are convex hull algorithms?*, Comput. Geom., 7 (1997), pp. 265–301.

[3] D. Avis, *lrs: A revised implementation of the reverse search vertex enumeration algorithm*, in Polytopes - Combinatorics and Computation, G. Kalai and G. Ziegler, eds., Birkhauser-Verlag, Basel, Switzerland, 2000, pp. 177–198.

[4] C. G. Broyden, *A class of methods for solving nonlinear simultaneous equations*, Math. Comput., 19 (1965), pp. 577–593.

[5] A. R. Conn, N. I. M. Gould, and P. L. Toint, *Trust-Region Methods*, MPS/SIAM Series on Optimization 1, SIAM, Philadelphia, 2000.

[6] A. R. Conn, N. Gould, A. Sartenaer, and P. L. Toint, *Convergence properties of an augmented Lagrangian algorithm for optimization with a combination of general equality and linear constraints*, SIAM J. Optim., 6 (1996), pp. 674–703.

[7] C. Davis, *Theory of positive linear dependence*, Amer. J. Math., 76 (1954), pp. 733–746.

[8] E. D. Dolan, R. M. Lewis, and V. Torczon, *On the local convergence properties of pattern search*, SIAM J. Optim., 14 (2003), pp. 567–583.

[9] K. Fukuda and A. Prodon, *Double description method revisited*, in Combinatorics and Computer Science, M. Deza, R. Euler, and I. Manoussakis, eds., Lecture Notes in Comput. Sci. 1120, Springer-Verlag, Berlin, 1997, pp. 91–111.

[10] K. Fukuda, *cdd/cdd+ Reference Manual*, The Institute for Operations Research, ETH-Zentrum, Zurich, Switzerland, 2005. Available at ftp://ftp.ifor.math.ethz.ch/pub/fukuda/cdd/cddman/cddman.

[11] K. Fukuda, *cddlib*, http://www.ifor.math.ethz.ch/~fukuda/cdd_home/, 2005.

[12] M. Gerstenhaber, *Theory of convex polyhedral cones*, in Activity Analysis of Production and Allocation, T. C. Koopmans, ed., Cowles Commission Monograph No. 13, John Wiley & Sons, New York, 1951, pp. 298–316.

[13] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, London, 1981.

[14] N. I. M. Gould, D. Orban, and P. L. Toint, *CUTEr (and SifDec), a constrained and unconstrained testing environment, revisited*, ACM Transactions on Mathematical Software, 29 (2003), pp. 373–394. See also http://cuter.rl.ac.uk/cuter-www/.

[15] P. D. Hough, T. G. Kolda, and H. A. Patrick, *Usage manual for APPSPACK* 2.0, Technical report SAND2000–8843, Sandia National Laboratories, Livermore, CA 94550, 2000.

[16] T. G. Kolda, R. M. Lewis, and V. Torczon, *Optimization by direct search: New perspectives on some classical and modern methods*, SIAM Rev., 45 (2003), pp. 385–482.

[17] T. G. Kolda, R. M. Lewis, and V. Torczon, *Stationarity results for generating set search for linearly constrained optimization*, SIAM J. Optim., 17 (2006), pp. 943–968.

[18] R. M. Lewis and V. Torczon, *Rank ordering and positive bases in pattern search algorithms*, Technical report 96–71, Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, VA, 1996.

[19] R. M. Lewis and V. Torczon, *Pattern search algorithms for bound constrained minimization*, SIAM J. Optim., 9 (1999), pp. 1082–1099.

[20] R. M. Lewis and V. Torczon, *Pattern search methods for linearly constrained minimization*, SIAM J. Optim., 10 (2000), pp. 917–941.

[21] S. Lucidi, M. Sciandrone, and P. Tseng, *Objective-derivative-free methods for constrained optimization*, Math. Program., 92 (2002), pp. 37–59.

[22] S. Lucidi and M. Sciandrone, *A derivative-free algorithm for bound constrained optimization*, Comput. Optim. Appl., 21 (2002), pp. 119–142.

[23] J. H. May, *Linearly Constrained Nonlinear Programming: A Solution Method That Does Not Require Analytic Derivatives*, Ph.D. thesis, Yale University, New Haven, CT, 1974.

[24] J.-J. Moreau, *Décomposition orthgonale d'un espace hilbertien selon deux cônes mutuellement polaires*, C. R. Acad. Sci. Paris, 255 (1962), pp. 238–240.

[25] J. J. Moré, B. S. Garbow, and K. E. Hillstrom, *Testing unconstrained optimization software*, ACM Trans. Math. Software, 7 (1981), pp. 17–41.

[26] T. S. Motzkin, H. Raiffa, G. Thompson, and R. M. Thrall, *The double description method*, in Contributions to Theory of Games 2, H. Kuhn and A. W.Tucker, eds., Princeton University Press, Princeton, NJ, 1953.

[27] B. A. Murtagh and M. A. Saunders, *MINOS* 5.5 *user's guide*, Technical report SOL 83–20R, Department of Operations Research, Stanford University, Palo Alto, CA, 1983. Revised 1998.

[28] The Mathworks, Inc., *Genetic algorithm and direct search toolbox user's guide, version* 2.0, http://www.mathworks.com/products/gads/, 2005.

[29] W. Yu and Y. Li, *A direct search method by the local positive basis for linearly constrained optimization*, Chinese Ann. Math., 2 (1981), pp. 139–146.

[30] W. Yu, *Positive basis and a class of direct search techniques*, Scientia Sinica, Special Issue of Mathematics, 1 (1979), pp. 53–67.

[31] E. H. Zarantonello, *Projections on convex sets in Hilbert space and spectral theory*, in Contributions to Nonlinear Functional Analysis, E. H. Zarantonello, ed., Academic Press, London and New York, 1971, pp. 237–424.

SIAM J. SCI. COMPUT.
Vol. 29, No. 6, pp. 2531–2561
© 2007 Society for Industrial and Applied Mathematics

A DUAL OPTIMIZATION APPROACH TO INVERSE QUADRATIC EIGENVALUE PROBLEMS WITH PARTIAL EIGENSTRUCTURE*

ZHENG-JIAN BAI†, DELIN CHU‡, AND DEFENG SUN‡

Abstract. The inverse quadratic eigenvalue problem (IQEP) arises in the field of structural dynamics. It aims to find three symmetric matrices, known as the mass, the damping, and the stiffness matrices, such that they are closest to the given analytical matrices and satisfy the measured data. The difficulty of this problem lies in the fact that in applications the mass matrix should be positive definite and the stiffness matrix positive semidefinite. Based on an equivalent dual optimization version of the IQEP, we present a quadratically convergent Newton-type method. Our numerical experiments confirm the high efficiency of the proposed method.

Key words. nonlinear optimization, quadratic eigenvalue problem, inverse eigenvalue problem, partial eigenstructure

AMS subject classifications. 65F15, 65H17, 15A22, 90C30

DOI. 10.1137/060656346

1. Introduction.

Throughout this paper the following notation will be used:

- $A \succeq 0$ ($A \succ 0$, respectively) means that the $n \times n$ real matrix A is symmetric and positive semidefinite (symmetric and positive definite, respectively).
- A^T, $\sigma(A)$, and $\|A\|$ denote the transpose, the spectrum, and the Frobenius norm of $A \in \mathbb{R}^{n\times n}$, respectively.
- I denotes the identity matrix of an appropriate dimension.
- $\mathcal{S}^n$ and $\mathcal{S}^n_+$ denote the set of all real $n \times n$ symmetric matrices and the cone of positive semidefinite matrices in this set, respectively. Let

$$\Omega_0 := \mathcal{S}^n \times \mathcal{S}^n \times \mathcal{S}^n \quad \text{and} \quad \Omega := \mathcal{S}^n_+ \times \mathcal{S}^n \times \mathcal{S}^n_+.$$

For given matrices $M, C, K \in \mathbb{R}^{n\times n}$, let

$$Q(\lambda) := \lambda^2 M + \lambda C + K.$$

Then the quadratic eigenvalue problem (QEP) is to find scalars $\lambda \in \mathbb{C}$ and nonzero vectors x such that

$$Q(\lambda)x = 0, \tag{1.1}$$

where λ and x are called the eigenvalue and the eigenvector, respectively. For various applications, mathematical properties, and numerical solution techniques of the QEP,

*Received by the editors April 4, 2006; accepted for publication (in revised form) March 14, 2007; published electronically October 24, 2007.

http://www.siam.org/journals/sisc/29-6/65634.html

†Department of Information and Computational Mathematics, Xiamen University, Xiamen 361005, People's Republic of China (zjbai@xmu.edu.cn). This author's research was partially supported by the National Natural Science Foundation of China grant 10601043 and by Xiamen University grant 0000-X07152.

‡Department of Mathematics, National University of Singapore, 2 Science Drive 2, Republic of Singapore (matchudl@nus.edu.sg, matsundf@nus.edu.sg). The work of the second author was partially supported by grant R146-000-047-112 of the National University of Singapore. The work of the third author was partially supported by grant R146-00-061-112 of the National University of Singapore.

we refer the reader to the survey paper [48]. In many practical applications, the matrices M, C, and K are required to be symmetric. In particular, in one of the most popular numerical models, the finite element model (FEM) [19, 48]

$$M\ddot{x}(t) + C\dot{x}(t) + Kx(t) = 0, \tag{1.2}$$

the mass matrix M, the damping matrix C, and the stiffness matrix K are required to be symmetric with M and K being positive definite and positive semidefinite, respectively. It is shown in [48] that the general solution to the homogeneous equation (1.2) is given in terms of the solution of the QEP (1.1). Unfortunately, the natural frequencies and model shapes (eigenvalues and eigenvectors) predicted by the FEM often disagree with the physical experimental results. The FEM updating aims to improve the numerical model (1.2) by the measured data. Various model updating methods have been presented in the literature [19, 27, 29, 33]. Mathematically, the model updating problem is an inverse quadratic eigenvalue problem (IQEP) which aims to find matrices M, C, and K such that they are closest to the given estimated analytical matrices and satisfy the partially measured eigendata. The general IQEP can be defined as follows: Given a measured partial eigenpair $(\Lambda, X) \in \mathbb{R}^{k\times k} \times \mathbb{R}^{n\times k}$ with

$$1 \le k \le n, \quad \mathrm{rank}(X) = k, \quad \text{and} \quad \Lambda = \mathrm{diag}\{\Lambda_1, \ldots, \Lambda_\mu, \Lambda_{\mu+1}, \ldots, \Lambda_\nu\}, \tag{1.3}$$

where

$$\Lambda_i = \mathrm{diag}\{\overbrace{\lambda_i^{[2]}, \ldots, \lambda_i^{[2]}}^{s_i}\} \quad \text{for} \quad 1 \le i \le \mu \quad \text{and} \quad \Lambda_i = \lambda_i I_{s_i} \quad \text{for} \quad \mu+1 \le i \le \nu,$$

$$\lambda_i^{[2]} = \begin{bmatrix} \alpha_i & \beta_i \\ -\beta_i & \alpha_i \end{bmatrix} \in \mathbb{R}^{2\times 2}, \quad \beta_i \ne 0, \quad \sigma(\Lambda_i) \cap \sigma(\Lambda_j) = \emptyset, \quad \forall\, 1 \le i \ne j \le \mu,$$

$$\lambda_i \in \mathbb{R}, \quad \lambda_i \ne \lambda_j, \quad \forall\, \mu+1 \le i \ne j \le \nu,$$

$$k = \sum_{i=1}^{\mu} 2s_i + \sum_{i=\mu+1}^{\nu} s_i,$$

find $M, C, K \in \mathcal{S}^n$ with $M \succ 0$ and $K \succeq 0$ such that

$$MX\Lambda^2 + CX\Lambda + KX = 0. \tag{1.4}$$

The model updating problem for given $M_a, C_a, K_a \in \mathcal{S}^n$, which are called the estimated analytic mass, damping, and stiffness matrices, respectively, in the FEM updating [19], is then to find $M, C, K \in \mathcal{S}^n$ such that (M, C, K) is a solution to the following optimization problem:

$$\begin{array}{ll} \inf & \dfrac{c_1}{2}\|M - M_a\|^2 + \dfrac{c_2}{2}\|C - C_a\|^2 + \dfrac{1}{2}\|K - K_a\|^2 \\ \text{subject to (s.t.)} & MX\Lambda^2 + CX\Lambda + KX = 0, \\ & M \succ 0, \quad C = C^T, \quad K \succeq 0, \end{array}$$

where c_1 and c_2 are two positive weighting parameters. It is noted that in numerical computation it is better to replace the condition $M \succ 0$ by $M \succeq 0$. This consideration leads us to focus on the following optimization problem:

$$\begin{array}{ll} \min & \dfrac{c_1}{2}\|M - M_a\|^2 + \dfrac{c_2}{2}\|C - C_a\|^2 + \dfrac{1}{2}\|K - K_a\|^2 \\ \text{s.t.} & MX\Lambda^2 + CX\Lambda + KX = 0, \\ & M \succeq 0, \quad C = C^T, \quad K \succeq 0. \end{array} \tag{1.5}$$

To avoid confusion, we refer to the optimization problem (1.5) as the IQEP.

As in [25], let the QR factorization of X be given by

$$X = Q \begin{bmatrix} R \\ 0 \end{bmatrix},$$

where $Q \in \mathbb{R}^{n\times n}$ is orthogonal and $R \in \mathbb{R}^{k\times k}$ is nonsingular and upper triangular. By renaming $M := \sqrt{c_1}Q^T M Q$, $C := \sqrt{c_2}Q^T C Q$, $K := Q^T K Q$, $M_a := \sqrt{c_1}Q^T M_a Q$, $C_a := \sqrt{c_2}Q^T C_a Q$, and $K_a := Q^T K_a Q$, without loss of generality, we may assume from now on that problem (1.5) takes the following form:

$$\begin{aligned} \min \quad & \frac{1}{2}\|M - M_a\|^2 + \frac{1}{2}\|C - C_a\|^2 + \frac{1}{2}\|K - K_a\|^2 \\ \text{s.t.} \quad & \frac{1}{\sqrt{c_1}} M \begin{bmatrix} R \\ 0 \end{bmatrix} \Lambda^2 + \frac{1}{\sqrt{c_2}} C \begin{bmatrix} R \\ 0 \end{bmatrix} \Lambda + K \begin{bmatrix} R \\ 0 \end{bmatrix} = 0, \\ & M \succeq 0, \quad C = C^T, \quad K \succeq 0. \end{aligned} \tag{1.6}$$

THEOREM 1.1. *Problem* (1.6) *admits a strictly feasible solution if and only if* Λ *is nonsingular.*

Proof. First we show the *necessity* by contradiction. Assume that Λ is singular. Then there exists a nonzero vector y such that

$$\Lambda y = 0.$$

As a result, for any feasible solution (M, C, K) to problem (1.6), we have

$$\left(\frac{1}{\sqrt{c_1}} M \begin{bmatrix} R \\ 0 \end{bmatrix} \Lambda^2 + \frac{1}{\sqrt{c_2}} C \begin{bmatrix} R \\ 0 \end{bmatrix} \Lambda + K \begin{bmatrix} R \\ 0 \end{bmatrix} \right) y = 0; \quad \text{i.e.,} \quad K \begin{bmatrix} Ry \\ 0 \end{bmatrix} = 0.$$

Note that R is nonsingular, which yields that $Ry \neq 0$. Hence K must be singular. This is a contradiction to the existence of a strictly feasible solution to problem (1.6). Therefore, problem (1.6) can admit a strictly feasible solution only if Λ is nonsingular.

Next we show the *sufficiency*. Following the idea of [12], we can obtain a special feasible solution to (1.6) as follows:

$$\begin{cases} \widehat{M} = \sqrt{c_1} \begin{bmatrix} (RR^T)^{-1} & 0 \\ 0 & I \end{bmatrix}, \\ \widehat{C} = \sqrt{c_2} \begin{bmatrix} -R^{-T}(\Lambda + \Lambda^T)R^{-1} & 0 \\ 0 & 0 \end{bmatrix}, \\ \widehat{K} = \begin{bmatrix} R^{-T}\Lambda^T \Lambda R^{-1} & 0 \\ 0 & I \end{bmatrix}. \end{cases} \tag{1.7}$$

Since both R and Λ are nonsingular, $\widehat{M}$ and $\widehat{K}$ are symmetric and positive definite. This implies that $(\widehat{M}, \widehat{C}, \widehat{K})$ above is a strictly feasible solution to problem (1.6). ∎

Because of Theorem 1.1 and the generalized Slater condition (2.4) (see section 2) used in our dual approach, in the following development we make a blanket assumption, unless stated otherwise, that the following condition holds.

Assumption 1.2. Λ is nonsingular.

Remark 1.3. Problem (1.6) fails to possess a strictly feasible solution if Λ is singular. As a remedy, we can reduce problem (1.6) equivalently to a similar problem which admits a strictly feasible solution. See Appendix B for discussions on this.

Due to complicated structures of the IQEP and a lack of tools for dealing with it, many authors considered various simplified versions of the IQEP by dropping or relaxing the requirement on the positive semidefiniteness of the matrices M and K [3, 4, 5, 6, 14, 18, 19, 26, 51, 52, 53]. As observed in the literature on structural dynamics [18, 19], however, these approaches may fail to guarantee the positive semidefiniteness of M and K, which is vital in applications.

In this paper, we deal directly with the IQEP with the required positive semidefiniteness condition on the matrices M and K. We shall introduce a generalized Newton method to the dual of problem (1.6), whose objective function is continuously differentiable but not twice differentiable. This approach is motivated by two recent papers due to Chu, Kuo, and Lin [12] and Qi and Sun [37]. In [12], Chu, Kuo, and Lin made an important step by showing for the first time that the general IQEP admits a nontrivial solution; i.e., there exist symmetric matrices M, C, and K with $M \succ 0$ and $K \succeq 0$ satisfying (1.4). In [37], based on recent developments in strongly semismooth matrix-valued functions [47], Qi and Sun presented a highly efficient quadratically convergent Newton method for solving the nearest correlation matrix problem proposed by Higham in [22] (see Boyd and Xiao [10], Malick [30], and Toh, Tütüncü, and Todd [50] for more discussions on this problem and its extensions). Theoretically, the Newton method introduced in [37] can be used to solve a much broader class of problems that includes the IQEP. However, there exist several practical difficulties in applying the approach by Qi and Sun [37] to the IQEP. In particular, for the IQEP we must first circumvent the incomputability of an element in Clarke's generalized Jacobian of the gradient mapping of the dual function needed in the approach by Qi and Sun [37]. We shall address this important issue under a more general setting in this paper by introducing a modified version of Clarke's generalized Jacobian. Consequently, the convergence analysis must be modified, too. Note that the IQEP is a special constrained least-squares problem, which can be solved by applying conventional approaches such as first order methods or a quasi-Newton method (e.g., Malick [30] suggested using the BFGS update) to its dual problem. Here, we show that a generalized Newton method can be applied to the dual of the IQEP with guaranteed quadratic convergence and high efficiency.

The organization of this paper is as follows. In section 2, we discuss some basic convex optimization theory used in this paper and present some results regarding Clarke's generalized Jacobian of certain locally Lipschitz functions. In section 3, we present the dual forms of the IQEP. In section 4, we give the details of our generalized Newton method and its convergence analysis. We report our numerical results in section 5 and make some final conclusions in section 6.

2. Preliminaries. Let $\mathcal{X}$ and $\mathcal{Y}$ be two finite dimensional real vector spaces, each equipped with a scalar inner product $\langle \cdot, \cdot \rangle$ and its induced norm $\| \cdot \|$. Let $\mathcal{A} : \mathcal{X} \to \mathcal{Y}$ be a linear operator and $\mathcal{A}^* : \mathcal{Y} \to \mathcal{X}$ be the adjoint of $\mathcal{A}$. Let x^0 be a given vector in $\mathcal{X}$. Consider the following constrained least-squares problem:

$$\begin{array}{ll} \min & \dfrac{1}{2}\langle x - x^0, x - x^0 \rangle \\ \text{s.t.} & \mathcal{A}x = b, \\ & x \in \mathcal{Q}, \end{array} \tag{2.1}$$

where $b \in \mathcal{Y}$ and $\mathcal{Q}$ is a closed convex cone in $\mathcal{X}$. In the literature, problem (2.1) is also known as the best approximation problem in Hilbert space. For a survey on the latter, see [36]. Let $\mathcal{D} \subseteq \mathcal{X}$ be a closed convex set. For any $x \in \mathcal{X}$, let $\Pi_{\mathcal{D}}(x)$ denote

the metric projection of x onto $\mathcal{D}$; i.e., $\Pi_{\mathcal{D}}(x)$ is the unique optimal solution to the following convex programming problem:

$$\begin{array}{ll} \min & \dfrac{1}{2}\langle z-x, z-x\rangle \\ \text{s.t.} & z \in \mathcal{D}. \end{array}$$

Then, the dual of problem (2.1) (cf. Rockafellar and Wets [44]) has the following form:

$$\begin{array}{ll} \max & -\theta(y) \\ \text{s.t.} & y \in \mathcal{Y}, \end{array} \tag{2.2}$$

where $\theta : \mathcal{Y} \to \mathbb{R}$ is given by

$$\theta(y) := \frac{1}{2}\|x^0 + \mathcal{A}^* y\|^2 - \frac{1}{2}\|x^0 + \mathcal{A}^* y - \Pi_{\mathcal{Q}}(x^0 + \mathcal{A}^* y)\|^2 - \langle b, y\rangle - \frac{1}{2}\|x^0\|^2. \tag{2.3}$$

Recall that the generalized Slater condition is said to hold for the convex optimization problem (2.1) if

$$\begin{cases} \mathcal{A} : \mathcal{X} \to \mathcal{Y} \text{ is onto,} \\ \exists\ \bar{x} \in \mathcal{X} \text{ such that } \mathcal{A}\bar{x} = b,\ \bar{x} \in \operatorname{int}(\mathcal{Q}), \end{cases} \tag{2.4}$$

where "int" denotes the topological interior of a given set. The following result is well known in the classical duality theory for convex programming [43, Theorems 17 and 18].

PROPOSITION 2.1. *Under the generalized Slater condition* (2.4)*, the following hold:*

(i) *There exists at least one $\bar{y} \in \mathcal{Y}$ that solves the dual problem* (2.2)*. The unique solution to the original problem* (2.1) *is given by*

$$\bar{x} = \Pi_{\mathcal{Q}}(x^0 + \mathcal{A}^* \bar{y}). \tag{2.5}$$

(ii) *For every real number τ, the level set $\{y \in \mathcal{Y} : \theta(y) \le \tau\}$ is closed, bounded, and convex.*

For more recent discussions about the above duality theory for the constrained least-squares problem (2.1) and its extensions, see [10, 30, 37].

Since $\mathcal{Q}$ is assumed to be a closed convex cone, by [54], we know that the function θ defined by (2.3) takes the following form:

$$\theta(y) = \frac{1}{2}\|\Pi_{\mathcal{Q}}(x^0 + \mathcal{A}^* y)\|^2 - \langle b, y\rangle - \frac{1}{2}\|x^0\|^2, \quad y \in \mathcal{Y}. \tag{2.6}$$

From [54] we also know that θ is a continuously differentiable convex function and that its gradient at $y \in \mathcal{Y}$ is given by

$$\nabla\theta(y) = \mathcal{A}\,\Pi_{\mathcal{Q}}(x^0 + \mathcal{A}^* y) - b.$$

Furthermore, since $\Pi_{\mathcal{Q}}(\cdot)$ is globally Lipschitz continuous with modulus 1, $\nabla\theta(\cdot)$ is globally Lipschitz continuous. Therefore, if the Slater condition (2.4) holds, then one may use any gradient-based method, such as the steepest descent method or quasi-Newton methods (e.g., the BFGS method suggested by Malick [30]), to find an optimal solution y^* to the dual problem (2.2) first and then use (2.5) to get an optimal

solution to problem (2.1). This is exactly the dual approach outlined by Rockafellar in [43, page 4] for general convex optimization problems and tailored to the constrained least-squares problem (2.1). Since in general $\Pi_{\mathcal{Q}}(\cdot)$ is not continuously differentiable, $\theta(\cdot)$ is not twice differentiable. So we cannot directly use the Newton method to solve (2.2); i.e.,

$$\begin{array}{ll} \min & \theta(y) \\ \text{s.t.} & y \in \mathcal{Y}. \end{array} \tag{2.7}$$

Fortunately, we may apply Clarke's Jacobian-based Newton method developed in the last two decades for solving locally Lipschitz equations [24, 39]. The success of using Clarke's Jacobian-based Newton method to solve the nearest correlation matrix problem in Qi and Sun [37] inspires us to explore this idea further.

Let $\mathcal{Z}$ be an arbitrary finite dimensional real vector space. Let $\mathcal{O}$ be an open set in $\mathcal{Y}$ and let $\Xi : \mathcal{O} \subseteq \mathcal{Y} \to \mathcal{Z}$ be a locally Lipschitz continuous function on the open set $\mathcal{O}$. Rademacher's theorem [44, Chapter 9.J] says that Ξ is almost everywhere Fréchet differentiable in $\mathcal{O}$. We denote by $\mathcal{O}_\Xi$ the set of points in $\mathcal{O}$ where Ξ is Fréchet differentiable. Let $\Xi'(y)$ denote the Jacobian of Ξ at $y \in \mathcal{O}_\Xi$. Then Clarke's generalized Jacobian of Ξ at $y \in \mathcal{O}$ is defined by [13]

$$\partial \Xi(y) := \mathrm{conv}\{\partial_B \Xi(y)\},$$

where "conv" denotes the convex hull and the Bouligand subdifferential $\partial_B \Xi(y)$ is defined by Qi in [38] as

$$\partial_B \Xi(y) := \left\{ V \,:\, V = \lim_{j \to \infty} \Xi'(y^j),\, y^j \to y,\, y^j \in \mathcal{O}_\Xi \right\}.$$

Suppose that $F : \mathcal{O} \subseteq \mathcal{Y} \to \mathcal{Y}$ is a locally Lipschitz continuous function on the open set $\mathcal{O}$. Let y^0 be an arbitrary point in $\mathcal{O}$. Then Clarke's Jacobian-based Newton method for solving the locally Lipschitz equation $F(y) = 0$ can be described by

$$y^{j+1} = y^j - V_j^{-1} F(y^j), \qquad V_j \in \partial F(y^j), \quad j = 0, 1, 2, \ldots. \tag{2.8}$$

In general, the above iterative method does not converge even if $\{\|V_j^{-1}\|\}$ is uniformly bounded and y^0 is arbitrarily close to a given solution of $F(y) = 0$ (see [24]). Kummer in [24] proposed a general condition guaranteeing the superlinear convergence of (2.8), which generalized Kojima and Shindo's condition for superlinear (quadratic) convergence of the Newton method for piecewise smooth equations [23]. Kummer's result was largely unnoticed until Qi and Sun in [39] published their now well-known work by showing that the iteration sequence generated by (2.8) can converge superlinearly if F is a *semismooth function*.

The concept of semismoothness was introduced by Mifflin [32] for functionals. In order to study the convergence of the iterative method (2.8), Qi and Sun [39] extended the definition of semismoothness to vector-valued functions. The following definition of semismoothness is convenient to use.

DEFINITION 2.2. *Let $\Xi : \mathcal{O} \subseteq \mathcal{Y} \to \mathcal{Z}$ be a locally Lipschitz continuous function on the open set $\mathcal{O}$. We say that Ξ is semismooth at a point $y \in \mathcal{O}$ if*

(i) *Ξ is directionally differentiable at y, and*

(ii) *for any $x \to y$ and $V \in \partial \Xi(x)$,*

$$\Xi(x) - \Xi(y) - V(x - y) = o(\|x - y\|). \tag{2.9}$$

The function $\Xi : \mathcal{O} \subseteq \mathcal{Y} \to \mathcal{Z}$ is said to be strongly semismooth at a point $y \in \mathcal{O}$ if F is semismooth at y, and for any $x \to y$ and $V \in \partial\Xi(x)$,

$$\Xi(x) - \Xi(y) - V(x-y) = O(\|x-y\|^2). \tag{2.10}$$

The following convergence result is established in [39, Theorem 3.2], while the result on superlinear convergence can also be found in [24, Proposition 3].

PROPOSITION 2.3. *Let $F : \mathcal{O} \subseteq \mathcal{Y} \to \mathcal{Y}$ be a locally Lipschitz continuous function on the open set $\mathcal{O}$. Let $\bar{y} \in \mathcal{O}$ be such that $F(\bar{y}) = 0$. Assume that F is semismooth at $\bar{y}$ and that every element in $\partial F(\bar{y})$ is nonsingular. Then every sequence generated by (2.8) converges to $\bar{y}$ superlinearly provided that the starting point y^0 is sufficiently close to $\bar{y}$. Moreover, if F is strongly semismooth at $\bar{y}$, the rate of convergence is quadratic.*

Now, we consider the following equation:

$$F(y) := \nabla\theta(y) = \mathcal{A}\,\Pi_{\mathcal{Q}}(x^0 + \mathcal{A}^* y) - b = 0, \quad y \in \mathcal{Y}. \tag{2.11}$$

We have already shown that F is a globally Lipschitz continuous mapping, and thus $\partial F(y)$ is well defined everywhere in $\mathcal{Y}$. In order to apply Clarke's Jacobian-based Newton method (2.8) to find a solution of (2.11), we need to assume that the two conditions imposed in Proposition 2.3 are valid. Certainly, the (strong) semismoothness of F at $\bar{y}$ follows easily if the metric projector $\Pi_{\mathcal{Q}}(\cdot)$ is (strongly) semismooth at $x^0 + \mathcal{A}^*\bar{y}$. This means that if one knows in advance that $\Pi_{\mathcal{Q}}(\cdot)$ is (strongly) semismooth everywhere in $\mathcal{Y}$ (as is the case for the IQEP discussed in this paper), then there is no need to be concerned about the semismoothness assumption on F. However, the second condition, i.e., the nonsingularity assumption on the generalized Jacobian $\partial F(\bar{y})$, needs more elaboration. We shall discuss this issue in the remaining part of this section.

Let $F : \mathcal{Y} \to \mathcal{Y}$ be defined by (2.11). The next proposition on Clarke's Jacobian of F follows easily from [13, page 75].

PROPOSITION 2.4. *Let $\bar{y} \in \mathcal{Y}$ and $\bar{x} := x^0 + \mathcal{A}^*\bar{y}$. Then, for any $d \in \mathcal{Y}$, it holds that*

$$\partial F(\bar{y})d \subseteq \mathcal{A}\,\partial\Pi_{\mathcal{Q}}(\bar{x})\mathcal{A}^* d. \tag{2.12}$$

Proposition 2.4 relates the Jacobian of F to the Jacobian of the metric projector $\Pi_{\mathcal{Q}}(\cdot)$. Note, however, that (2.12) does not mean that $\partial F(\bar{y}) \subseteq \mathcal{A}\,\partial\Pi_{\mathcal{Q}}(\bar{x})\mathcal{A}^*$. For the metric projector, we have the following result from [31, Proposition 1].

LEMMA 2.5. *Let $\mathcal{D} \subseteq \mathcal{X}$ be a closed convex set. Then, for any $x \in \mathcal{X}$ and $V \in \partial\Pi_{\mathcal{D}}(x)$,*

(i) *V is self-adjoint,*
(ii) *$\langle d, Vd\rangle \geq 0$ for all $d \in \mathcal{X}$,*
(iii) *$\langle Vd, d - Vd\rangle \geq 0$ for all $d \in \mathcal{X}$.*

Let $\mathcal{D} \subseteq \mathcal{X}$ be a closed convex set. For any $x \in \mathcal{X}$, denote

$$\mathrm{dist}(x, \mathcal{D}) := \inf\{\|x - d\| \,:\, d \in \mathcal{D}\}.$$

The *tangent cone* of $\mathcal{D}$ at a point $x \in \mathcal{D}$, which we denote by $\mathcal{T}_{\mathcal{D}}(x)$, consisting of all tangent vectors of $\mathcal{D}$ at x, is defined by (cf. [9, section 2.2.4])

$$\mathcal{T}_{\mathcal{D}}(x) := \{d \in \mathcal{X} \,:\, \mathrm{dist}(x + td, \mathcal{D}) = o(t),\ t \geq 0\}. \tag{2.13}$$

For any $x \in \mathcal{D}$, we denote by $\operatorname{lin}(\mathcal{T}_{\mathcal{D}}(x))$ the lineality space of $\mathcal{T}_{\mathcal{D}}(x)$, i.e., the largest linear space in $\mathcal{T}_{\mathcal{D}}(x)$.

The following definition of constraint nondegeneracy is adapted from the works of Robinson, who defined it for general optimization problems and variational inequalities [40, 41, 42].

DEFINITION 2.6. *We say that a feasible point x to problem* (2.1) *is constraint nondegenerate if*

$$\begin{pmatrix} \mathcal{A} \\ I \end{pmatrix} \mathcal{X} + \begin{pmatrix} 0 \\ \operatorname{lin}(\mathcal{T}_{\mathcal{Q}}(x)) \end{pmatrix} = \begin{pmatrix} \mathcal{Y} \\ \mathcal{X} \end{pmatrix} \tag{2.14}$$

or, equivalently, if

$$\mathcal{A} \operatorname{lin}(\mathcal{T}_{\mathcal{Q}}(x)) = \mathcal{Y}. \tag{2.15}$$

The constraint nondegenerate condition (2.14) in its general form is extensively used by Bonnans and Shapiro in [9] for perturbation analysis of optimization problems. In the context of semidefinite programming, condition (2.14) reduces to the primal nondegenerate condition proposed by Alizadeh, Haeberly, and Overton [1]. For the conventional nonlinear programming problem with finitely many equality and inequality constraints, the constraint nondegenerate condition is equivalent to the well-known linear independence constraint qualification [40, 46].

DEFINITION 2.7. *Let $\mathcal{D} \subseteq \mathcal{X}$ be a closed convex set. We say that the metric projector $\Pi_{\mathcal{D}}(\cdot)$ is Jacobian amicable at $x \in \mathcal{X}$ if, for any $V \in \partial\Pi_{\mathcal{D}}(x)$ and $d \in \mathcal{X}$ such that $Vd = 0$, it holds that*

$$d \in \left(\operatorname{lin}\left(\mathcal{T}_{\mathcal{D}}(x_+)\right)\right)^{\perp},$$

where $x_+ := \Pi_{\mathcal{D}}(x)$ and $\left(\operatorname{lin}\left(\mathcal{T}_{\mathcal{D}}(x_+)\right)\right)^{\perp}$ is defined by

$$\left(\operatorname{lin}\left(\mathcal{T}_{\mathcal{D}}(x_+)\right)\right)^{\perp} := \{d \in \mathcal{X} \,:\, \langle d, h\rangle = 0 \quad \forall\, h \in \operatorname{lin}\left(\mathcal{T}_{\mathcal{D}}(x_+)\right)\}.$$

We say that $\Pi_{\mathcal{D}}(\cdot)$ is Jacobian amicable if it is Jacobian amicable at every point in $\mathcal{X}$.

Now, we are ready to state our result on the nonsingularity of Clarke's Jacobian of the mapping F defined by (2.11).

PROPOSITION 2.8. *Let $\bar{y} \in \mathcal{Y}$ be such that $F(\bar{y}) = 0$. Let $\bar{x} := x^0 + \mathcal{A}^*\bar{y}$ and $\bar{x}_+ := \Pi_{\mathcal{Q}}(\bar{x})$. Assume that the constraint nondegenerate condition* (2.15) *holds at $\bar{x}_+$ and that $\Pi_{\mathcal{Q}}(\cdot)$ is Jacobian amicable at $\bar{x}$. Then every element in $\mathcal{A}\partial\Pi_{\mathcal{Q}}(\bar{x})\mathcal{A}^*$ is self-adjoint and positive definite.*

Proof. Let V be an arbitrary element in $\mathcal{A}\partial\Pi_{\mathcal{Q}}(\bar{x})\mathcal{A}^*$. Then, there exists an element $W \in \partial\Pi_{\mathcal{Q}}(\bar{x})$ such that

$$V = \mathcal{A}W\mathcal{A}^*.$$

Since, by Lemma 2.5, W is self-adjoint and positive semidefinite, we know that V is also self-adjoint and positive semidefinite.

Next, we show the positive definiteness of V. Let $d \in \mathcal{Y}$ be such that $Vd = 0$. Then, by (iii) of Lemma 2.5, we obtain that

$$0 = \langle d, Vd\rangle = \langle d, \mathcal{A}W\mathcal{A}^*d\rangle = \langle \mathcal{A}^*d, W\mathcal{A}^*d\rangle \geq \langle W\mathcal{A}^*d, W\mathcal{A}^*d\rangle,$$

which implies

$$W\mathcal{A}^*d = 0.$$

Therefore, by the assumption that $\Pi_{\mathcal{Q}}(\cdot)$ is Jacobian amicable at $\bar{x}$, we know that

$$\mathcal{A}^* d \in (\mathrm{lin}\,(\mathcal{T}_{\mathcal{Q}}(\bar{x}_+)))^{\perp}.$$

Since the constraint nondegenerate condition (2.15) holds at $\bar{x}_+$, there exists a vector $h \in \mathrm{lin}\,(\mathcal{T}_{\mathcal{Q}}(\bar{x}_+))$ such that $\mathcal{A}h = d$. Hence, since $\mathcal{A}^* d \in (\mathrm{lin}\,(\mathcal{T}_{\mathcal{Q}}(\bar{x}_+)))^{\perp}$ and $h \in \mathrm{lin}\,(\mathcal{T}_{\mathcal{Q}}(\bar{x}_+))$, it holds that

$$\langle d, d\rangle = \langle d, \mathcal{A}h\rangle = \langle \mathcal{A}^* d, h\rangle = 0.$$

Thus $d = 0$. This, together with the fact that V is self-adjoint and positive semidefinite, shows that V is self-adjoint and positive definite. □

Proposition 2.8 motivates us to define the following modified version of the Jacobian-based Newton method (2.8) to solve (2.11):

$$y^{j+1} = y^j - V_j^{-1} F(y^j), \quad V_j := \mathcal{A} W_j \mathcal{A}^*, \quad W_j \in \partial \Pi_{\mathcal{Q}}(x^0 + \mathcal{A}^* y^j), \quad j = 0, 1, 2, \ldots, \tag{2.16}$$

where $y^0 \in \mathcal{Y}$ is an initial point.

By Proposition 2.8, we can obtain the following convergence results for the iterative method (2.16) in a way similar to that of Proposition 2.3. We include a brief proof for the sake of completeness.

PROPOSITION 2.9. *Let $\bar{y} \in \mathcal{Y}$ be such that $F(\bar{y}) = 0$. Let $\bar{x} := x^0 + \mathcal{A}^* \bar{y}$ and $\bar{x}_+ := \Pi_{\mathcal{Q}}(\bar{x})$. Assume that the constraint nondegenerate condition* (2.15) *holds at $\bar{x}_+$, $\Pi_{\mathcal{Q}}(\cdot)$ is semismooth at $\bar{x}$, and $\Pi_{\mathcal{Q}}(\cdot)$ is Jacobian amicable at $\bar{x}$. Then every sequence generated by* (2.16) *converges to $\bar{y}$ superlinearly provided that the starting point y^0 is sufficiently close to $\bar{y}$. Moreover, if $\Pi_{\mathcal{Q}}(\cdot)$ is strongly semismooth at $\bar{x}$, the rate of convergence is quadratic.*

Proof. Since, by Proposition 2.8, every element in $\mathcal{A}\partial\Pi_{\mathcal{Q}}(\bar{x})\mathcal{A}^*$ is positive definite, we know from the basic properties of $\partial\Pi_{\mathcal{Q}}(\cdot)$ (cf. [13, Proposition 2.6.2]) that there exist a constant $c > 0$ and an open neighborhood $\mathcal{N}$ of $\bar{y}$ such that for any $y \in \mathcal{N}$ and $W \in \partial\Pi_{\mathcal{Q}}(x^0 + \mathcal{A}^* y)$, $\mathcal{A}W\mathcal{A}^*$ is positive definite and satisfies

$$\| (\mathcal{A} W \mathcal{A}^*)^{-1} \| \le c. \tag{2.17}$$

By using (2.9) in Definition 2.2, we obtain for any $y \in \mathcal{N}$ and $W \in \partial\Pi_{\mathcal{Q}}(x^0 + \mathcal{A}^* y)$ with $y \to \bar{y}$ that

$$\|\Pi_{\mathcal{Q}}(x^0 + \mathcal{A}^* y) - \Pi_{\mathcal{Q}}(\bar{x}) - W(\mathcal{A}^* y - \mathcal{A}^* \bar{y})\| = o(\|\mathcal{A}^*(y - \bar{y})\|) = o(\|(y - \bar{y})\|), \tag{2.18}$$

which implies that for any $y \in \mathcal{N}$ (shrinking $\mathcal{N}$ if necessary) and $W \in \partial\Pi_{\mathcal{Q}}(x^0 + \mathcal{A}^* y)$ we have

$$\|\Pi_{\mathcal{Q}}(x^0 + \mathcal{A}^* y) - \Pi_{\mathcal{Q}}(\bar{x}) - W(\mathcal{A}^* y - \mathcal{A}^* \bar{y})\| \le (2c\|\mathcal{A}\|)^{-1} \|y - \bar{y}\|. \tag{2.19}$$

Then from (2.16), (2.17), and (2.19), we know that for any $y^j \in \mathcal{N}$, $W_j \in \partial\Pi_{\mathcal{Q}}(x^0 + \mathcal{A}^* y^j)$, and $V_j = \mathcal{A} W_j \mathcal{A}^*$,

$$\begin{aligned}
\|y^{j+1} - \bar{y}\| &= \|y^j - \bar{y} - V_j^{-1} F(y^j)\| \\
&= \|V_j^{-1}[V_j(y^j - \bar{y}) - F(y^j) + F(\bar{y})]\| \\
&\le c\|F(y^j) - F(\bar{y}) - V_j(y^j - \bar{y})\| \\
&\le c\|\mathcal{A}\|\|\Pi_{\mathcal{Q}}(x^0 + \mathcal{A}^* y^j) - \Pi_{\mathcal{Q}}(\bar{x}) - W_j(\mathcal{A}^* y^j - \mathcal{A}^* \bar{y})\| \\
&\le \frac{1}{2}\|y^j - \bar{y}\|.
\end{aligned} \tag{2.20}$$

This shows that once $y^0 \in \mathcal{N}$, the whole iteration sequence $\{y^j\}$ converges to $\bar{y}$ linearly. The superlinear convergence of $\{y^j\}$ follows from (2.18) and the second inequality in (2.20).

If $\Pi_{\mathcal{Q}}(\cdot)$ is strongly semismooth at $\bar{x}$, then by repeating the above process and using (2.10) in Definition 2.2 we can easily prove the quadratic convergence of $\{y^j\}$. □

In Proposition 2.9, besides the well-studied constraint nondegenerate condition (2.15) at $\bar{x}_+$ in the literature, we also assume that $\Pi_{\mathcal{Q}}(\cdot)$ is Jacobian amicable at $\bar{x}$, which may hold for many interesting closed convex cones. Here we discuss the Jacobian amicability of $\Pi_{\mathcal{Q}}(\cdot)$ only for the case that $\mathcal{Q} = \mathcal{S}^n_+$, the cone of symmetric and positive semidefinite matrices in $\mathcal{S}^n$.

Let $A \in \mathcal{S}^n$. Then A admits the following spectral decomposition:

$$A = P\Sigma P^T, \tag{2.21}$$

where

$$\Sigma = \begin{bmatrix} \sigma_1 & & & & & & & & \\ & \ddots & & & & & & & \\ & & \sigma_{k_1} & & & & & & \\ & & & 0 & & & & & \\ & & & & \ddots & & & & \\ & & & & & 0 & & & \\ & & & & & & \sigma_{k_1+k_2+1} & & \\ & & & & & & & \ddots & \\ & & & & & & & & \sigma_n \end{bmatrix}$$

with $\sigma_i > 0$ $(i = 1, \ldots, k_1)$ and $\sigma_i < 0$ $(i = k_1 + k_2 + 1, \ldots, n)$ is the diagonal matrix of eigenvalues of A, and

$$P = \begin{bmatrix} P_1 & P_2 & P_3 \end{bmatrix}$$

with $P_1 \in \mathbb{R}^{n\times k_1}$, $P_2 \in \mathbb{R}^{n\times k_2}$, and $P_3 \in \mathbb{R}^{n\times(n-k_1-k_2)}$ is a corresponding orthogonal matrix of orthonormal eigenvectors. Under the Frobenius inner product, the projection $A_+ := \Pi_{\mathcal{S}^n_+}(A)$ of a matrix $A \in \mathcal{S}^n$ onto the cone $\mathcal{S}^n_+$ has an analytic formula [21, 49]

$$A_+ = P \begin{bmatrix} \sigma_1 & & & & & \\ & \ddots & & & & \\ & & \sigma_{k_1} & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{bmatrix} P^T.$$

Define the matrix $U \in \mathbb{R}^{k_1\times(n-k_1-k_2)}$ with entries

$$U_{ij} := \frac{\max\{\sigma_i, 0\} + \max\{\sigma_{k_1+k_2+j}, 0\}}{|\sigma_i| + |\sigma_{k_1+k_2+j}|}, \quad i = 1, \ldots, k_1, \quad j = 1, \ldots, n - k_1 - k_2.$$

Bonnans, Cominetti, and Shapiro [8] showed that $\Pi_{\mathcal{S}^n_+}(\cdot)$ is directionally differentiable everywhere in $\mathcal{S}^n$. For any $H \in \mathcal{S}^n$, D. Sun and J. Sun [47] gave the following explicit

formula for $\Pi'_{\mathcal{S}^n_+}(A;H)$:

$$\Pi'_{\mathcal{S}^n_+}(A;H) = P \begin{bmatrix} P_1^T H P_1 & P_1^T H P_2 & U \circ P_1^T H P_3 \\ P_2^T H P_1 & \Pi_{\mathcal{S}^{k_2}_+}\left(P_2^T H P_2\right) & 0 \\ P_3^T H P_1 \circ U^T & 0 & 0 \end{bmatrix} P^T, \tag{2.22}$$

where $\circ$ denotes the Hadamard product. More importantly, Sun and Sun proved that $\Pi_{\mathcal{S}^n_+}$ is a strongly semismooth matrix-valued function [47]. When A is nonsingular, i.e., $k_2 = 0$, $\Pi_{\mathcal{S}^n_+}(\cdot)$ is continuously differentiable around A, and the formula (2.22) reduces to the classical result of Löwner [28]. See Donoghue [16] and Bhatia [7] for detailed discussions on the latter and Chen, Qi, and Tseng [11] for a generalization.

From (2.13) and (2.22), one can obtain Arnold's well-known characterization of the tangent cone of $\mathcal{S}^n_+$ at A_+ [2],

$$\begin{aligned} \mathcal{T}_{\mathcal{S}^n_+}(A_+) &= \{B \in \mathcal{S}^n \,:\, B = \Pi'_{\mathcal{S}^n_+}(A_+;B)\} \\ &= \left\{B \in \mathcal{S}^n \,:\, \begin{bmatrix} P_2 & P_3 \end{bmatrix}^T B \begin{bmatrix} P_2 & P_3 \end{bmatrix} \succeq 0\right\}, \end{aligned}$$

and its lineality space,

$$\operatorname{lin}\left(\mathcal{T}_{\mathcal{S}^n_+}(A_+)\right) = \left\{B \in \mathcal{S}^n \,:\, \begin{bmatrix} P_2 & P_3 \end{bmatrix}^T B \begin{bmatrix} P_2 & P_3 \end{bmatrix} = 0\right\}. \tag{2.23}$$

Let $\Phi(\cdot) := \Pi'_{\mathcal{S}^n_+}(A;\cdot)$. It is proved in [35, Lemma 11] that

$$\partial_B \Pi_{\mathcal{S}^n_+}(A) = \partial_B \Phi(0) \tag{2.24}$$

and that the operator $W : \mathcal{S}^n \to \mathcal{S}^n$ defined by

$$W(H) = P \begin{bmatrix} P_1^T H P_1 & P_1^T H P_2 & U \circ P_1^T H P_3 \\ P_2^T H P_1 & 0 & 0 \\ P_3^T H P_1 \circ U^T & 0 & 0 \end{bmatrix} P^T \quad \forall\, H \in \mathcal{S}^n \tag{2.25}$$

is an element in $\partial_B \Pi_{\mathcal{S}^n_+}(A)$.

The next proposition shows that $\Pi_{\mathcal{S}^n_+}(\cdot)$ is Jacobian amicable at A.

PROPOSITION 2.10. *The projection operator $\Pi_{\mathcal{S}^n_+}(\cdot)$ is Jacobian amicable everywhere in $\mathcal{S}^n$.*

Proof. Let $A \in \mathcal{S}^n$ admit the spectral decomposition (2.21). Let $W \in \partial\Pi_{\mathcal{S}^n_+}(A)$ and $H \in \mathcal{S}^n$ be such that $W(H) = 0$. Then from (2.22), (2.24), and the definition of the matrix U, we obtain

$$P_1^T H P_1 = 0, \quad P_1^T H P_2 = 0, \quad \text{and} \quad P_1^T H P_3 = 0,$$

which, together with (2.23), implies

$$H \in \left(\operatorname{lin}\left(\mathcal{T}_{\mathcal{S}^n_+}(A_+)\right)\right)^{\perp}.$$

Since A is chosen arbitrarily, we know from the definition that $\Pi_{\mathcal{S}^n_+}(\cdot)$ is Jacobian amicable everywhere in $\mathcal{S}^n$. ∎

3. Dual reformulations. In this section, we will consider the dual problem of the IQEP, i.e., problem (1.6). As mentioned in section 1, we need only focus on the IQEP under Assumption 1.2, i.e., Λ is nonsingular. Denote

$$M := \begin{bmatrix} M_1 & M_2 \\ M_2^T & M_4 \end{bmatrix}, \quad C := \begin{bmatrix} C_1 & C_2 \\ C_2^T & C_4 \end{bmatrix}, \quad K := \begin{bmatrix} K_1 & K_2 \\ K_2^T & K_4 \end{bmatrix},$$

where $M_1, C_1, K_1 \in \mathcal{S}^k$, $M_2, C_2, K_2 \in \mathbb{R}^{k\times(n-k)}$, and $M_4, C_4, K_4 \in \mathcal{S}^{(n-k)}$. Let $S := R\Lambda R^{-1}$. Then problem (1.6) can be written equivalently as

$$\begin{array}{ll} \min & \frac{1}{2}\|M - M_a\|^2 + \frac{1}{2}\|C - C_a\|^2 + \frac{1}{2}\|K - K_a\|^2 \\ \text{s.t.} & \frac{1}{\sqrt{c_1}}(\Lambda^2)^T(R^T M_1 R) + \frac{1}{\sqrt{c_2}}\Lambda^T(R^T C_1 R) + (R^T K_1 R) = 0, \\ & \frac{1}{\sqrt{c_1}}(S^2)^T M_2 + \frac{1}{\sqrt{c_2}}S^T C_2 + K_2 = 0, \\ & (M, C, K) \in \Omega. \end{array} \tag{3.1}$$

Let Ω_0 be equipped with the natural inner product

$$\left\langle (M, C, K), (\widetilde{M}, \widetilde{C}, \widetilde{K}) \right\rangle := \langle M, \widetilde{M}\rangle + \langle C, \widetilde{C}\rangle + \langle K, \widetilde{K}\rangle, \quad (M, C, K), (\widetilde{M}, \widetilde{C}, \widetilde{K}) \in \Omega_0,$$

and its induced norm $\|\cdot\|$. Define

$$\begin{aligned} &\mathcal{H}(M, C, K) \\ &:= \frac{1}{\sqrt{c_1}}(\Lambda^2)^T\left(R^T M_1 R\right) + \frac{1}{\sqrt{c_2}}\Lambda^T\left(R^T C_1 R\right) + \left(R^T K_1 R\right), \quad (M, C, K) \in \Omega_0, \end{aligned} \tag{3.2}$$

$$\mathcal{G}(M, C, K) := \frac{1}{\sqrt{c_1}}(S^2)^T M_2 + \frac{1}{\sqrt{c_2}}S^T C_2 + K_2, \quad (M, C, K) \in \Omega_0, \tag{3.3}$$

and

$$\mathcal{A}(M, C, K) := (\mathcal{H}(M, C, K), \mathcal{G}(M, C, K)), \quad (M, C, K) \in \Omega_0. \tag{3.4}$$

Then, problem (3.1), which is a special case of problem (2.1), becomes

$$\begin{array}{ll} \min & \frac{1}{2}\|(M, C, K) - (M_a, C_a, K_a)\|^2 \\ \text{s.t.} & \mathcal{A}(M, C, K) = 0, \\ & (M, C, K) \in \Omega. \end{array} \tag{3.5}$$

Let

$$\mathcal{R}(\mathcal{H}) := \{\mathcal{H}(M, C, K) \,:\, (M, C, K) \in \Omega_0\}. \tag{3.6}$$

Obviously, the linear operator $\mathcal{H} : \Omega_0 \to \mathcal{R}(\mathcal{H})$ defined in (3.2) is surjective. The adjoint $\mathcal{H}^* : \mathcal{R}(\mathcal{H}) \to \Omega_0$ of $\mathcal{H}$ is given by

$$\mathcal{H}^*(Y) := (\mathcal{H}_1^*(Y), \mathcal{H}_2^*(Y), \mathcal{H}_3^*(Y)), \quad Y \in \mathcal{R}(\mathcal{H}), \tag{3.7}$$

where for each $Y \in \mathcal{R}(\mathcal{H})$,

$$\mathcal{H}_1^*(Y) = \frac{1}{2\sqrt{c_1}} \begin{bmatrix} R\Lambda^2 Y R^T + (R\Lambda^2 Y R^T)^T & 0 \\ 0 & 0 \end{bmatrix},$$
$$\mathcal{H}_2^*(Y) = \frac{1}{2\sqrt{c_2}} \begin{bmatrix} R\Lambda Y R^T + (R\Lambda Y R^T)^T & 0 \\ 0 & 0 \end{bmatrix},$$
$$\mathcal{H}_3^*(Y) = \frac{1}{2} \begin{bmatrix} RYR^T + (RYR^T)^T & 0 \\ 0 & 0 \end{bmatrix}.$$

Similarly, $\mathcal{G} : \Omega_0 \to \mathbb{R}^{k\times(n-k)}$ defined in (3.3) is also surjective, and its adjoint $\mathcal{G}^* : \mathbb{R}^{k\times(n-k)} \to \Omega_0$ is given by

$$\mathcal{G}^*(Z) := (\mathcal{G}_1^*(Z), \mathcal{G}_2^*(Z), \mathcal{G}_3^*(Z)), \tag{3.8}$$

where for each $Z \in \mathbb{R}^{k\times(n-k)}$,

$$\mathcal{G}_1^*(Z) = \frac{1}{2\sqrt{c_1}} \begin{bmatrix} 0 & S^2 Z \\ (S^2 Z)^T & 0 \end{bmatrix},$$
$$\mathcal{G}_2^*(Z) = \frac{1}{2\sqrt{c_2}} \begin{bmatrix} 0 & SZ \\ (SZ)^T & 0 \end{bmatrix},$$
$$\mathcal{G}_3^*(Z) = \frac{1}{2} \begin{bmatrix} 0 & Z \\ Z^T & 0 \end{bmatrix}.$$

Since both $\mathcal{H} : \Omega_0 \to \mathcal{R}(\mathcal{H})$ and $\mathcal{G} : \Omega_0 \to \mathbb{R}^{k\times(n-k)}$ are surjective, the linear operator $\mathcal{A} : \Omega_0 \to \mathcal{R}(\mathcal{H}) \times \mathbb{R}^{k\times(n-k)}$ defined by (3.4) is also surjective. The adjoint $\mathcal{A}^* : \mathcal{R}(\mathcal{H}) \times \mathbb{R}^{k\times(n-k)} \to \Omega_0$ of $\mathcal{A}$ takes the following form:

$$\begin{aligned} \mathcal{A}^*(Y,Z) &:= (\mathcal{A}_1^*(Y,Z), \mathcal{A}_2^*(Y,Z), \mathcal{A}_3^*(Y,Z)) \\ &= \mathcal{H}^*(Y) + \mathcal{G}^*(Z), \quad (Y,Z) \in \mathcal{R}(\mathcal{H}) \times \mathbb{R}^{k\times(n-k)}, \end{aligned} \tag{3.9}$$

where for each $(Y,Z) \in \mathcal{R}(\mathcal{H}) \times \mathbb{R}^{k\times(n-k)}$, $\mathcal{H}^*(Y)$ and $\mathcal{G}^*(Z)$ are defined by (3.7) and (3.8), respectively. Define $\theta : \mathcal{R}(\mathcal{H}) \times \mathbb{R}^{k\times(n-k)} \to \mathbb{R}$ by

$$\theta(Y,Z) := \frac{1}{2}\|\Pi_\Omega((M_a, C_a, K_a) + \mathcal{A}^*(Y,Z))\|^2 - \frac{1}{2}\|(M_a, C_a, K_a)\|^2. \tag{3.10}$$

By (2.6) and (2.7), the dual of problem (3.5) is

$$\begin{aligned} &\min \quad \theta(Y,Z) \\ &\text{s.t.} \quad (Y,Z) \in \mathcal{R}(\mathcal{H}) \times \mathbb{R}^{k\times(n-k)}. \end{aligned} \tag{3.11}$$

Since Λ is nonsingular, by Theorem 1.1, problem (1.6), or, equivalently, problem (3.5), admits a strictly feasible solution; i.e., there exists a triplet $(\widehat{M}, \widehat{C}, \widehat{K}) \in \Omega_0$ such that $\mathcal{A}(\widehat{M}, \widehat{C}, \widehat{K}) = 0$ and $(\widehat{M}, \widehat{C}, \widehat{K}) \in \operatorname{int}(\Omega)$. Moreover, the linear operator $\mathcal{A} : \Omega_0 \to \mathcal{R}(\mathcal{H}) \times \mathbb{R}^{k\times(n-k)}$ defined by problem (3.5) is surjective. Thus, for the convex optimization problem (3.5), the generalized Slater condition (2.4) is satisfied. Consequently, the following result follows directly from Proposition 2.1.

PROPOSITION 3.1. *There exists at least one pair* $(\overline{Y}, \overline{Z}) \in \mathbb{R}^{k\times k} \times \mathbb{R}^{k\times(n-k)}$ *that solves the dual problem* (3.11). *The unique solution to the original problem* (1.6) *is given by*

$$(\overline{M}, \overline{C}, \overline{K}) = \Pi_\Omega\left((M_a, C_a, K_a) + \mathcal{A}^*(\overline{Y}, \overline{Z})\right).$$

Furthermore, for every real number τ, the level set

$$\{(Y,Z)\in\mathbb{R}^{k\times k}\times\mathbb{R}^{k\times(n-k)}:\ \theta(Y,Z)\le\tau\}$$

is closed, bounded, and convex.

It is clear from Proposition 3.1 that we need only find a solution of the dual problem (3.11) in order to obtain the unique solution of the original problem (1.6).

Remark 3.2. The operator $\mathcal{H}$ defined above is not necessarily surjective from Ω_0 to $\mathbb{R}^{k\times k}$. To see this, we refer the reader to Proposition A.1 in Appendix A.

Next, we discuss the following simplified version of the IQEP:

$$\begin{aligned}\min\quad & \frac{1}{2}\|(M,C,K)-(M_a,C_a,K_a)\|^2\\ \text{s.t.}\quad & \mathcal{A}(M,C,K)=0,\\ & (M,C,K)\in\Omega_0,\end{aligned}\tag{3.12}$$

which admits an explicit optimal solution and has been considered by many authors; for example, see [14, 19]. The dual of problem (3.12) is

$$\begin{aligned}\min\quad & \theta_0(Y,Z)\\ \text{s.t.}\quad & (Y,Z)\in\mathcal{R}(\mathcal{H})\times\mathbb{R}^{k\times(n-k)},\end{aligned}\tag{3.13}$$

where $\theta_0:\mathcal{R}(\mathcal{H})\times\mathbb{R}^{k\times(n-k)}\to\mathbb{R}$ is defined by

$$\theta_0(Y,Z):=\frac{1}{2}\|\Pi_{\Omega_0}((M_a,C_a,K_a)+\mathcal{A}^*(Y,Z))\|^2-\frac{1}{2}\|(M_a,C_a,K_a)\|^2.$$

Let $(Y^0,Z^0)\in\mathcal{R}(\mathcal{H})\times\mathbb{R}^{k\times(n-k)}$ be a solution to the dual problem (3.13). By section 2, the solution (M^0,C^0,K^0) to problem (3.12) is given by

$$(M^0,C^0,K^0)=\Pi_{\Omega_0}((M_a,C_a,K_a)+\mathcal{A}^*(Y^0,Z^0))=(M_a,C_a,K_a)+\mathcal{A}^*(Y^0,Z^0).\tag{3.14}$$

Although (Y^0,Z^0) may not be a solution to the dual problem (3.11) as M^0 and K^0 above may fail to be positive semidefinite, it may be a good guess at the solution of problem (3.11). Hence, we will use it as the starting point to our Newton method to be proposed in the next section.

Now we discuss the computation of (Y^0,Z^0) based on (3.13). Since $(Y^0,Z^0)\in\mathcal{R}(\mathcal{H})\times\mathbb{R}^{k\times(n-k)}$ satisfies

$$\nabla\theta_0(Y^0,Z^0)=\mathcal{A}\left((M_a,C_a,K_a)+\mathcal{A}^*(Y^0,Z^0)\right)=0,$$

thus, by (3.4), $(Y^0,Z^0)\in\mathcal{R}(\mathcal{H})\times\mathbb{R}^{k\times(n-k)}$ is a solution of

$$\left(\mathcal{H}\left((M_a,C_a,K_a)+\mathcal{A}^*(Y,Z)\right),\mathcal{G}\left((M_a,C_a,K_a)+\mathcal{A}^*(Y,Z)\right)\right)=0.\tag{3.15}$$

Therefore, we can obtain (Y^0,Z^0) by solving (3.15). Since the linear operator $\mathcal{A}:\Omega_0\to\mathcal{R}(\mathcal{H})\times\mathbb{R}^{k\times(n-k)}$ defined by (3.4) is surjective, $\mathcal{A}\mathcal{A}^*$ is self-adjoint and positive definite. This implies that (Y^0,Z^0) is the unique solution to (3.15).

By (3.7), (3.8), and (3.9), we obtain that for any $(Y,Z)\in\mathcal{R}(\mathcal{H})\times\mathbb{R}^{k\times(n-k)}$,

$$\begin{aligned}&\mathcal{H}((M_a,C_a,K_a)+\mathcal{A}^*(Y,Z))=\mathcal{H}(M_a,C_a,K_a)+\mathcal{H}(\mathcal{H}^*(Y))\\ &\quad=\mathcal{H}(M_a,C_a,K_a)+U_1YR^TR\\ &\qquad+\frac{1}{2}\left(\frac{1}{c_1}(\Lambda^2)^TR^TRY^T(\Lambda^2)^TR^TR+\frac{1}{c_2}\Lambda^TR^TRY^T\Lambda^TR^TR+R^TRY^TR^TR\right)\end{aligned}$$

and

$$\mathcal{G}((M_a, C_a, K_a) + \mathcal{A}^*(Y, Z)) = \mathcal{G}(M_a, C_a, K_a) + \mathcal{G}(\mathcal{G}^*(Z)) = \mathcal{G}(M_a, C_a, K_a) + U_2 Z,$$

where

$$U_1 := \frac{1}{2}\left(\frac{1}{c_1}(\Lambda^2)^T R^T R \Lambda^2 + \frac{1}{c_2}\Lambda^T R^T R \Lambda + R^T R\right) \tag{3.16}$$

and

$$U_2 := \frac{1}{2}\left(\frac{1}{c_1}(S^2)^T S^2 + \frac{1}{c_2} S^T S + I\right). \tag{3.17}$$

Since the linear operator $\mathcal{H} : \Omega_0 \to \mathcal{R}(\mathcal{H})$ defined by (3.2) is surjective, $\mathcal{H}\mathcal{H}^*$ is self-adjoint and positive definite. So we may use the conjugate gradient (CG) method[1] [20, Algorithm 10.2.1] to solve the first equation of (3.15), i.e.,

(3.18)
$$-\mathcal{H}(M_a, C_a, K_a) = \mathcal{H}(\mathcal{H}^*(Y))$$
$$= U_1 Y R^T R + \frac{1}{2}\left(\frac{1}{c_1}(\Lambda^2)^T R^T R Y^T (\Lambda^2)^T R^T R + \frac{1}{c_2}\Lambda^T R^T R Y^T \Lambda^T R^T R + R^T R Y^T R^T R\right),$$

to get $Y^0 \in \mathcal{R}(\mathcal{H})$. The required total computational cost of computing Y^0 is $O(k^5)$ flops (the CG method needs at most k^2 steps with each step costing $O(k^3)$ flops). The solution to the second equation of (3.15), i.e,

$$-\mathcal{G}(M_a, C_a, K_a) = \mathcal{G}(\mathcal{G}^*(Z)) = U_2 Z,$$

is given explicitly by

$$Z^0 = -U_2^{-1}\mathcal{G}(M_a, C_a, K_a).$$

We need $O(nk^2)$ flops to compute Z^0. The optimal solution to problem (3.12) can then be obtained via (3.14).

Remark 3.3. The above arguments imply that if we drop the positive semidefiniteness requirement on M and K, we can obtain the solution to problem (3.12) with a total cost of $O(nk^2 + k^5)$. This cost amount is small as in practice $k \ll n$. It is noted that in our approach the presence of the weighting parameters c_1 and c_2 presents no difficulty at all.

4. Algorithm and convergence analysis. In this section, we shall first present a globalized version of Clarke's Jacobian-based Newton method (2.16) for solving the dual problem (3.11) with $\theta : \mathcal{R}(\mathcal{H}) \times \mathbb{R}^{k\times(n-k)} \to \mathbb{R}$ being defined by (3.10).

Let $F : \mathcal{R}(\mathcal{H}) \times \mathbb{R}^{k\times(n-k)} \to \mathcal{R}(\mathcal{H}) \times \mathbb{R}^{k\times(n-k)}$ be defined by

$$F(Y, Z) = \nabla\theta(Y, Z) = \mathcal{A}\,\Pi_\Omega\left((M_a, C_a, K_a) + \mathcal{A}^*(Y, Z)\right), \quad (Y, Z) \in \mathcal{R}(\mathcal{H})\times\mathbb{R}^{k\times(n-k)},$$

where $\mathcal{A} : \Omega_0 \to \mathcal{R}(\mathcal{H}) \times \mathbb{R}^{k\times(n-k)}$ is defined by (3.4) and $\mathcal{A}^*$ is the adjoint of $\mathcal{A}$.

[1]Strictly speaking, we should apply the CG method to the equivalent equation form $Ax = b$ of (3.18), where $A \in \mathcal{S}^m$ is positive definite, $b \in \mathbb{R}^m$, and m is the dimension of $\mathcal{R}(\mathcal{H})$. In the implementation of CG methods there is no need to form A and b explicitly. One may also consider the preconditioned CG method to solve (3.18). However, caution should be taken as the preconditioned CG method may not guarantee that the generated matrices Y stay in $\mathcal{R}(\mathcal{H})$.

ALGORITHM 4.1 (Newton method).

Step 0. Given $(Y^0, Z^0) \in \mathcal{R}(\mathcal{H}) \times \mathbb{R}^{k\times(n-k)}$, $\eta \in (0,1)$, $\rho, \delta \in (0, 1/2)$. $j := 0$.

Step 1. Select an element $W_j \in \partial\Pi_\Omega\left((M_a, C_a, K_a) + \mathcal{A}^*(Y^j, Z^j)\right)$ and let $V_j := \mathcal{A}W_j\mathcal{A}^*$. Apply the conjugate gradient method [20, Algorithm 10.2.1] to find an approximate solution $(\Delta Y^j, \Delta Z^j) \in \mathcal{R}(\mathcal{H}) \times \mathbb{R}^{k\times(n-k)}$ to

$$F(Y^j, Z^j) + V_j(\Delta Y, \Delta Z) = 0 \tag{4.1}$$

such that

$$\|F(Y^j, Z^j) + V_j(\Delta Y^j, \Delta Z^j)\| \le \eta_j \|F(Y^j, Z^j)\| \tag{4.2}$$

and

$$\left\langle F(Y^j, Z^j), (\Delta Y^j, \Delta Z^j)\right\rangle \le -\eta_j \left\langle (\Delta Y^j, \Delta Z^j), (\Delta Y^j, \Delta Z^j)\right\rangle, \tag{4.3}$$

where $\eta_j := \min\{\eta, \|F(Y^j, Z^j)\|\}$. If (4.2) and (4.3) are not achievable, let

$$(\Delta Y^j, \Delta Z^j) := -F(Y^j, Z^j) = -\mathcal{A}\,\Pi_\Omega\left((M_a, C_a, K_a) + \mathcal{A}^*(Y^j, Z^j)\right).$$

Step 2. Let m_j be the smallest nonnegative integer m such that

$$\theta\left((Y^j, Z^j) + \rho^m(\Delta Y^j, \Delta Z^j)\right) - \theta(Y^j, Z^j) \le \delta\rho^m \left\langle F(Y^j, Z^j), (\Delta Y^j, \Delta Z^j)\right\rangle.$$

Set

$$(Y^{j+1}, Z^{j+1}) := (Y^j, Z^j) + \rho^{m_j}(\Delta Y^j, \Delta Z^j).$$

Step 3. Replace j by $j+1$ and go to Step 1.

Next we make several comments on the above algorithm before we present our convergence analysis.

- *The starting point.* One may choose an arbitrary starting point (Y^0, Z^0) as long as $(Y^0, Z^0) \in \mathcal{R}(\mathcal{H}) \times \mathbb{R}^{k\times(n-k)}$. In numerical computation, we recommend choosing the solution to (3.15) as (Y^0, Z^0).
- *The generalized Jacobian.* In Step 1 of Algorithm 4.1, we need to select an element

$$W_j \in \partial\Pi_\Omega\left((M_a, C_a, K_a) + \mathcal{A}^*(Y^j, Z^j)\right)$$

at the jth iteration. This can be computed according to (2.25) once the spectral decompositions of $M_a + \mathcal{A}_1^*(Y^j, Z^j)$ and $K_a + \mathcal{A}_3^*(Y^j, Z^j)$ are available.
- *The conjugate gradient method.* In Step 1 of Algorithm 4.1, if we apply the CG method [20, Algorithm 10.2.1] to (4.1), we can always keep $(\Delta Y^j, \Delta Z^j) \in \mathcal{R}(\mathcal{H}) \times \mathbb{R}^{k\times(n-k)}$ when the initial guess is in $\mathcal{R}(\mathcal{H}) \times \mathbb{R}^{k\times(n-k)}$. One may also consider the preconditioned CG method to solve (3.18) as long as the preconditioned CG method can guarantee that $(\Delta Y^j, \Delta Z^j) \in \mathcal{R}(\mathcal{H}) \times \mathbb{R}^{k\times(n-k)}$.

The next theorem is our global convergence result.

THEOREM 4.2. *For any $(Y^0, Z^0) \in \mathcal{R}(\mathcal{H}) \times \mathbb{R}^{k\times(n-k)}$, Algorithm 4.1 generates an infinite sequence $\{(Y^j, Z^j)\}$ with the properties that for each $j \ge 0$, $(Y^j, Z^j) \in \mathcal{R}(\mathcal{H}) \times \mathbb{R}^{k\times(n-k)}$, $\{(Y^j, Z^j)\}$ is bounded, and any accumulation point of $\{(Y^j, Z^j)\}$ is a solution to problem (3.11).*

Proof. Since, by Step 1 of Algorithm 4.1, for any $j \geq 0$, $(\Delta Y^j, \Delta Z^j)$ is always a descent direction of $\theta(\cdot)$ at (Y^j, Z^j) and $(\Delta Y^j, \Delta Z^j) \in \mathcal{R}(\mathcal{H}) \times \mathbb{R}^{k\times(n-k)}$, Algorithm 4.1 is well defined. Then an infinite sequence $\{(Y^j, Z^j)\}$ is generated and for each $j \geq 0$, $(Y^j, Z^j) \in \mathcal{R}(\mathcal{H}) \times \mathbb{R}^{k\times(n-k)}$.

From Theorem 1.1 we know that problem (3.1) has a strictly feasible solution. This, together with the surjectivity of $\mathcal{A} : \Omega_0 \to \mathcal{R}(\mathcal{H}) \times \mathbb{R}^{k\times(n-k)}$, implies that the generalized Slater condition (2.4) holds for problem (3.1). Thus, by Proposition 3.1, the level set

$$\{(Y, Z) \in \mathcal{R}(\mathcal{H}) \times \mathbb{R}^{k\times(n-k)} \,:\, \theta(Y, Z) \leq \theta(Y^0, Z^0)\}$$

is bounded. Therefore, the sequence $\{(Y^j, Z^j)\}$ is bounded. Furthermore, by employing standard convergence analysis of optimization methods (cf. [15, Theorem 6.3.3]), we can conclude that

$$\lim_{j\to\infty} \nabla\theta(Y^j, Z^j) = 0,$$

which, together with the convexity of $\theta(\cdot)$ and the boundedness of $\{(Y^j, Z^j)\}$, implies that any accumulation point of $\{(Y^j, Z^j)\}$ is a solution to (3.11). ∎

Theorem 4.2 shows that our algorithm converges globally. For discussions on the rate of convergence, we need the constraint nondegenerate condition (2.15) at the solution $(\overline{M}, \overline{C}, \overline{K})$ of problem (1.6). In our case, this condition can be written as

$$\mathcal{A}\left(\operatorname{lin}\left(\mathcal{T}_{\mathcal{S}^n_+}(\overline{M})\right), \mathcal{S}^n, \operatorname{lin}\left(\mathcal{T}_{\mathcal{S}^n_+}(\overline{K})\right)\right) = \mathcal{R}(\mathcal{H}) \times \mathbb{R}^{k\times(n-k)}. \tag{4.4}$$

PROPOSITION 4.3. *Condition* (4.4) *holds if the solution* $(\overline{M}, \overline{C}, \overline{K}) \in \Omega_0$ *to problem* (1.6) *satisfies* $\overline{K} \succ 0$ *or* $\overline{M} \succ 0$.

Proof. Since

$$\operatorname{lin}\left(\mathcal{T}_{\mathcal{S}^n_+}(\overline{K})\right) = \mathcal{S}^n \qquad \text{if } \overline{K} \succ 0,$$
$$\operatorname{lin}\left(\mathcal{T}_{\mathcal{S}^n_+}(\overline{M})\right) = \mathcal{S}^n \qquad \text{if } \overline{M} \succ 0,$$

the proof of Proposition A.1 in Appendix A gives that

$$\mathcal{H}\left(\operatorname{lin}\left(\mathcal{T}_{\mathcal{S}^n_+}(\overline{M})\right), \mathcal{S}^n, \mathcal{S}^n\right) = \mathcal{R}(\mathcal{H})$$

and

$$\mathcal{G}\left(\operatorname{lin}\left(\mathcal{T}_{\mathcal{S}^n_+}(\overline{M})\right), \mathcal{S}^n, \mathcal{S}^n\right) = \mathbb{R}^{k\times(n-k)},$$

and a similar discussion also yields that

$$\mathcal{H}\left(\mathcal{S}^n, \mathcal{S}^n, \operatorname{lin}\left(\mathcal{T}_{\mathcal{S}^n_+}(\overline{K})\right)\right) = \mathcal{R}(\mathcal{H})$$

and

$$\mathcal{G}\left(\mathcal{S}^n, \mathcal{S}^n, \operatorname{lin}\left(\mathcal{T}_{\mathcal{S}^n_+}(\overline{M})\right)\right) = \mathbb{R}^{k\times(n-k)},$$

Proposition 4.3 follows readily. ∎

Another interesting special case that guarantees condition (4.4) is $\Lambda = \tau I$ for some $\tau \neq 0$. This can be seen easily because in this case, for the solution $(\overline{M}, \overline{C}, \overline{K})$ to problem (1.6), we have

$$\mathcal{H}\left(\mathrm{lin}\left(\mathcal{T}_{\mathcal{S}^n_+}(\overline{M})\right), \mathcal{S}^n, \mathrm{lin}\left(\mathcal{T}_{\mathcal{S}^n_+}(\overline{K})\right)\right) = \mathcal{S}^k = \mathcal{R}(\mathcal{H})$$

and

$$\mathcal{G}\left(\mathrm{lin}\left(\mathcal{T}_{\mathcal{S}^n_+}(\overline{M})\right), \mathcal{S}^n, \mathrm{lin}\left(\mathcal{T}_{\mathcal{S}^n_+}(\overline{K})\right)\right) = \mathbb{R}^{k\times(n-k)} = \mathcal{R}(\mathcal{G}),$$

where $\mathcal{R}(\mathcal{G})$ denotes the range of the linear operator $\mathcal{G} : \Omega_0 \to \mathbb{R}^{k\times(n-k)}$ defined by (3.3).

We are now ready to show the quadratic convergence of Algorithm 4.1.

THEOREM 4.4. *Let $(\overline{Y}, \overline{Z})$ be an accumulation point of the infinite sequence $\{(Y^j, Z^j)\}$ generated by Algorithm* 4.1 *for solving the dual problem* (3.11). *Then let $(\overline{M}, \overline{C}, \overline{K})$ be the solution to problem* (1.6). *Assume that the constraint nondegenerate condition* (4.4) *holds at $(\overline{M}, \overline{C}, \overline{K})$. Then the whole sequence $\{(Y^j, Z^j)\}$ converges to $(\overline{Y}, \overline{Z})$ quadratically.*

Proof. By Proposition 3.1, we know

$$(\overline{M}, \overline{C}, \overline{K}) := \Pi_\Omega\left((M_a, C_a, K_a) + \mathcal{A}^*(\overline{Y}, \overline{Z})\right).$$

By Theorem 4.2, the whole sequence $\{(Y^j, Z^j)\}$ is bounded and $(\overline{Y}, \overline{Z})$ is a solution of the dual problem (3.11) such that

$$F(\overline{Y}, \overline{Z}) = \nabla\theta(\overline{Y}, \overline{Z}) = 0.$$

From the constraint nondegenerate condition (4.4) and Propositions 2.8 and 2.10, we know that any element $V := \mathcal{A}W\mathcal{A}^*$ with $W \in \partial\Pi_\Omega(\overline{M}, \overline{C}, \overline{K}))$ is self-adjoint and positive definite. Then, by Proposition 2.4, any element in $\partial F(\overline{Y}, \overline{Z})$ is also positive definite. This, together with the convexity of θ, implies that $(\overline{Y}, \overline{Z})$ is the unique solution of (3.11). Consequently, by Theorem 4.2, the whole sequence $\{(Y^j, Z^j)\}$ converges to $(\overline{Y}, \overline{Z})$. Thus, for all j sufficiently large, V_j is positive definite and $\{\|V_j^{-1}\|\}$ is uniformly bounded. Hence, for all j sufficiently large, the CG method can find $(\Delta Y^j, \Delta Z^j) \in \mathcal{R}(\mathcal{H}) \times \mathbb{R}^{k\times(n-k)}$ such that both (4.2) and (4.3) are satisfied. Then, by Propositions 2.9 and 2.10, for all j sufficiently large,

(4.5)
$$\begin{aligned}\|(Y^j, Z^j) + (\Delta Y^j, \Delta Z^j) - (\overline{Y}, \overline{Z})\| &\le O(\|(Y^j, Z^j) - (\overline{Y}, \overline{Z})\|^2) + \eta_j\|V_j^{-1}\|\|F(Y^j, Z^j)\| \\ &= O(\|(Y^j, Z^j) - (\overline{Y}, \overline{Z})\|^2) + O(\|F(Y^j, Z^j)\|^2) \\ &= O(\|(Y^j, Z^j) - (\overline{Y}, \overline{Z})\|^2),\end{aligned}$$

where in the last inequality the global Lipschitz continuity of F was used. By using (4.5) and the fact that $\{(Y^j, Z^j)\}$ converges to $(\overline{Y}, \overline{Z})$, we have for all j sufficiently large that

$$(Y^j, Z^j) - (\overline{Y}, \overline{Z}) = -(\Delta Y^j, \Delta Z^j) + O(\|(\Delta Y^j, \Delta Z^j)\|^2) \text{ and } (\Delta Y^j, \Delta Z^j) \to 0. \tag{4.6}$$

For each $j \ge 0$, let

$$r_j := F(Y^j, Z^j) + V_j(\Delta Y^j, \Delta Z^j).$$

Then for all j sufficiently large,

$$
\begin{aligned}
&-\left\langle F(Y^j,Z^j),(\Delta Y^j,\Delta Z^j)\right\rangle\\
&=\left\langle(\Delta Y^j,\Delta Z^j),V_j(\Delta Y^j,\Delta Z^j)\right\rangle-\left\langle(\Delta Y^j,\Delta Z^j),r_j\right\rangle\\
&\geq\left\langle(\Delta Y^j,\Delta Z^j),V_j(\Delta Y^j,\Delta Z^j)\right\rangle-\|(\Delta Y^j,\Delta Z^j)\|\|r_j\|\\
&\geq\left\langle(\Delta Y^j,\Delta Z^j),V_j(\Delta Y^j,\Delta Z^j)\right\rangle-\eta_j\|(\Delta Y^j,\Delta Z^j)\|\|F(Y^j,Z^j)\|\\
&\geq\left\langle(\Delta Y^j,\Delta Z^j),V_j(\Delta Y^j,\Delta Z^j)\right\rangle-\|(\Delta Y^j,\Delta Z^j)\|\|F(Y^j,Z^j)\|^2\\
&=\left\langle(\Delta Y^j,\Delta Z^j),V_j(\Delta Y^j,\Delta Z^j)\right\rangle-\|(\Delta Y^j,\Delta Z^j)\|\|F(Y^j,Z^j)-F(\overline{Y},\overline{Z})\|^2\\
&\geq\left\langle(\Delta Y^j,\Delta Z^j),V_j(\Delta Y^j,\Delta Z^j)\right\rangle-\|(\Delta Y^j,\Delta Z^j)\|\|\mathcal{A}\|^2\|\mathcal{A}^*\|^2\|(Y^j,Z^j)-(\overline{Y},\overline{Z})\|^2,
\end{aligned}
$$

which, together with (4.6) and the uniform positive definiteness of V_j, implies that there exists a positive constant $\hat{\rho}>0$ such that for all j sufficiently large we have

$$-\left\langle F(Y^j,Z^j),(\Delta Y^j,\Delta Z^j)\right\rangle\geq\hat{\rho}\|(\Delta Y^j,\Delta Z^j)\|^2.$$

Then, since F is strongly semismooth at $(\overline{Y},\overline{Z})$ as $\Pi_{\mathcal{S}^n_+}(\cdot)$ is strongly semismooth [47], we know from [17, Theorem 3.3 and Remark 3.4] that for all j sufficiently large,

$$\theta\left((Y^j,Z^j)+(\Delta Y^j,\Delta Z^j)\right)-\theta(Y^j,Z^j)\leq\delta\left\langle F(Y^j,Z^j),(\Delta Y^j,\Delta Z^j)\right\rangle,$$

which implies that for all j sufficiently large,

$$(Y^{j+1},Z^{j+1})=(Y^j,Z^j)+(\Delta Y^j,\Delta Z^j).$$

This, together with (4.5), completes the proof. □

Theorems 4.2 and 4.4 present global and local convergence analysis of Algorithm 4.1 for solving the IQEP under Assumption 1.2. In fact, our algorithm can also be used to solve the IQEP of general matrix weights even if Assumption 1.2 fails to hold. Without giving details, we briefly outline the way of doing so in Appendix B.

5. Numerical experiments. In this section, we report our numerical experiments of Algorithm 4.1 for solving the IQEP (1.6) carried out in `MATLAB 7.0.1` running on a PC Intel Pentium IV of 2.40 GHz CPU. In both of the following two numerical examples, we randomly generate $\Lambda\in\mathbb{R}^{k\times k}$ by using the built-in function `randn` in `MATLAB 7.0.1`. The total number of complex-valued eigenvalues is chosen to be around $k/2$. The matrix $R\in\mathbb{R}^{k\times k}$ is obtained by applying the QR factorization to a random generated $n\times k$ matrix X by using `randn`.

Example 5.1. Let $\widehat{M}$, $\widehat{C}$, and $\widehat{K}$ be given by (1.7). Set

$$M_a:=\widehat{M}+\tau R_M,\quad C_a:=\widehat{C}+\tau R_C,\quad K_a:=\widehat{K}+\tau R_K,$$

where R_M, R_C, and R_K are $n\times n$ symmetric matrices with random entries uniformly distributed between -1.0 and 1.0, and $\tau\in\mathbb{R}$ is a perturbed parameter. We report our numerical results for (a) $k=30$, $n=100,200,500,1000,1500,2000$, and $\tau=0.1,1.0$ and (b) $k\approx n/3$, $n=100,200,300,400,450$, and $\tau=0.1,1.0$.

Example 5.2. The matrices G_M and G_K are random $n\times n$ correlation matrices generated by `MATLAB 7.0.1`'s `gallery ('randcorr',n)`, and the matrix G_C is a random $n\times n$ symmetric matrix with entries $(G_C)_{ij}\in[-1,1]$ and $(G_C)_{ii}=1.0$ for $i,j=1,2,\ldots,n$. Then M_a, C_a, and K_a are, respectively, obtained via perturbing $\sqrt{c_1}G_M$, $\sqrt{c_2}G_C$, and G_K by a random $n\times n$ symmetric matrix with entries in $[-\tau,\tau]$, where $\tau=0.1,1.0$. We report our numerical results for (a) $k=30$,

TABLE 1
Numerical results of Example 5.1(a).

$k = 30$, $c_1 = c_2 = 1$							
τ	n	cputime	`It.`	`Func.`	`Res0.`	`Res*.`	`Tol.`
0.1	100	01 m 26 s	18	24	1.2×10^{2}	5.4×10^{-10}	3.9×10^{-11}
	200	04 m 39 s	14	15	3.2×10^{2}	9.0×10^{-10}	3.9×10^{-11}
	500	21 m 16 s	11	12	1.8×10^{3}	6.1×10^{-9}	1.3×10^{-10}
	1,000	44 m 13 s	9	10	4.2×10^{3}	9.3×10^{-8}	1.1×10^{-9}
	1,500	08 h 49 m 11 s	7	8	9.0×10^{3}	1.9×10^{-6}	1.6×10^{-8}
	2,000	05 h 24 m 37 s	9	10	1.4×10^{4}	5.0×10^{-6}	3.3×10^{-8}
1.0	100	40.6 s	10	11	1.2×10^{3}	1.9×10^{-9}	2.7×10^{-11}
	200	01 m 37 s	9	11	3.7×10^{3}	1.8×10^{-6}	1.3×10^{-8}
	500	09 m 03 s	10	12	1.4×10^{4}	7.0×10^{-7}	2.0×10^{-9}
	1,000	01 h 09 m 50 s	10	11	4.5×10^{4}	1.7×10^{-5}	2.4×10^{-8}
	1,500	08 h 12 m 53 s	14	18	6.7×10^{4}	3.9×10^{-5}	3.6×10^{-8}
	2,000	08 h 54 m 57 s	10	12	1.0×10^{5}	8.0×10^{-5}	5.7×10^{-8}
$k = 30$, $c_1 = 10$, and $c_2 = 0.1$							
τ	n	cputime	`It.`	`Func.`	`Res0.`	`Res*.`	`Tol.`
0.1	100	35.5 s	9	10	7.7×10^{1}	5.1×10^{-9}	2.8×10^{-10}
	200	04 m 18 s	9	10	2.8×10^{2}	1.7×10^{-6}	5.1×10^{-8}
	500	20 m 14 s	7	8	1.4×10^{3}	5.4×10^{-7}	7.3×10^{-9}
	1,000	53 m 30 s	8	9	3.7×10^{3}	1.8×10^{-6}	1.3×10^{-8}
	1,500	05 h 50 m 22 s	10	11	6.0×10^{3}	7.9×10^{-6}	3.7×10^{-8}
	2,000	05 h 22 m 25 s	9	10	8.1×10^{3}	1.3×10^{-6}	4.8×10^{-9}
1.0	100	36.4 s	6	7	1.3×10^{3}	1.3×10^{-6}	9.4×10^{-9}
	200	03 m 10 s	7	8	3.7×10^{3}	5.1×10^{-8}	1.9×10^{-10}
	500	19 m 33 s	8	9	1.2×10^{4}	5.3×10^{-6}	7.8×10^{-9}
	1,000	02 h 02 m 35 s	10	11	2.8×10^{4}	8.9×10^{-7}	6.6×10^{-10}
	1,500	04 h 52 m 15 s	11	14	6.9×10^{4}	6.7×10^{-7}	3.3×10^{-10}
	2,000	12 h 30 m 15 s	11	12	8.9×10^{4}	3.2×10^{-5}	1.2×10^{-8}

$n = 100, 200, 500, 1000, 1500, 2000$, and $\tau = 0.1, 1.0$ and (b) $k \approx n/3$, $n = 100, 200, 300, 400, 450$, and $\tau = 0.1, 1.0$.

In our numerical experiments, the initial point (Y^0, Z^0) is chosen to be the solution to (3.15), and the stopping criterion is

$$\texttt{Tol.} := \frac{\|\nabla\theta(Y_k, Z_k)\|}{\max\left\{1, \left\|\left(\frac{1}{\sqrt{c_1}}M_a, \frac{1}{\sqrt{c_2}}C_a, K_a\right)\right\|\right\}} \le 10^{-7}.$$

We set other parameters used in our algorithm as $\eta = 10^{-6}$, $\rho = 0.5$, and $\delta = 10^{-4}$. Our numerical results are given in Tables 1–4, where `It.`, `Func.`, `Res0.`, and `Res*.` stand for the number of iterations, the number of function evaluations, and the residuals $\|\nabla\theta(\cdot)\|$ at the starting point (Y^0, Z^0) and at the final iterate of our algorithm (the largest number of iterations in CG is set to be $\max(5000, nk)$), respectively. In Algorithm 4.1, the major cost at the jth iteration is solving the linear system (4.1), i.e.,

$$\begin{pmatrix} \mathcal{H}\left(\Pi_\Omega((M_a, C_a, K_a) + \mathcal{A}^*(Y^j, Z^j)) + W_j\mathcal{A}^*(\Delta Y^j, \Delta Z^j)\right) \\ \mathcal{G}\left(\Pi_\Omega((M_a, C_a, K_a) + \mathcal{A}^*(Y^j, Z^j)) + W_j\mathcal{A}^*(\Delta Y^j, \Delta Z^j)\right) \end{pmatrix} = 0,$$

where $W_j \in \partial\Pi_\Omega\left((M_a, C_a, K_a) + \mathcal{A}^*(Y^j, Z^j)\right)$. For better numerical performance, instead of solving the above linear system directly, we apply the CG method to the following preconditioned linear system [45, Chapter 9]:

TABLE 2
Numerical results of Example 5.1(b).

$k \approx n/3$, $c_1 = c_2 = 1$								
τ	n	k	cputime	It.	Func.	Res0.	Res*.	Tol.
0.1	100	33	02 m 31 s	18	23	1.0×10^{2}	9.8×10^{-7}	7.2×10^{-8}
	200	66	01 h 05 m 26 s	40	69	5.3×10^{2}	1.4×10^{-8}	6.3×10^{-10}
	300	100	01 h 37 m 32 s	20	21	1.0×10^{3}	4.8×10^{-7}	1.6×10^{-8}
	400	133	10 h 23 m 23 s	15	16	2.0×10^{3}	1.6×10^{-8}	4.5×10^{-10}
	450	150	10 h 24 m 04 s	14	15	2.7×10^{3}	3.0×10^{-8}	7.4×10^{-10}
1.0	100	33	01 m 43 s	10	12	1.4×10^{3}	1.7×10^{-7}	2.3×10^{-9}
	200	66	22 m 02 s	19	35	5.5×10^{3}	1.4×10^{-5}	9.5×10^{-8}
	300	100	01 h 32 m 58 s	12	14	1.2×10^{4}	1.3×10^{-5}	5.9×10^{-8}
	400	133	02 h 46 m 54 s	14	16	2.4×10^{4}	2.4×10^{-6}	8.3×10^{-9}
	450	150	09 h 38 m 15 s	18	21	2.9×10^{4}	4.9×10^{-7}	1.5×10^{-9}
$k \approx n/3$, $c_1 = 10$, and $c_2 = 0.1$								
τ	n	k	cputime	It.	Func.	Res0.	Res*.	Tol.
0.1	100	33	46.1 s	9	11	7.5×10^{1}	2.6×10^{-8}	1.4×10^{-9}
	200	66	42 m 42 s	13	15	3.6×10^{2}	1.8×10^{-6}	5.8×10^{-8}
	300	100	02 h 24 m 23 s	17	20	1.1×10^{3}	2.9×10^{-7}	6.5×10^{-9}
	400	133	04 h 38 m 42 s	10	11	1.8×10^{3}	2.4×10^{-6}	4.0×10^{-8}
	450	150	12 h 23 m 44 s	13	14	2.2×10^{3}	5.8×10^{-7}	8.8×10^{-9}
1.0	100	33	02 m 29 s	10	11	1.5×10^{3}	2.5×10^{-8}	1.8×10^{-10}
	200	66	23 m 26 s	10	11	5.0×10^{3}	5.5×10^{-8}	2.0×10^{-10}
	300	100	01 h 17 m 47 s	10	11	1.0×10^{4}	2.9×10^{-7}	7.1×10^{-10}
	400	133	06 h 24 m 49 s	17	22	1.9×10^{4}	8.4×10^{-6}	1.5×10^{-8}
	450	150	06 h 39 m 33 s	11	12	2.6×10^{4}	6.7×10^{-7}	1.1×10^{-9}

TABLE 3
Numerical results of Example 5.2(a).

$k = 30$, $c_1 = c_2 = 1$							
τ	n	cputime	It.	Func.	Res0.	Res*.	Tol.
0.1	100	25.1 s	9	10	6.2×10^{2}	1.2×10^{-8}	2.6×10^{-10}
	200	02 m 17 s	8	9	2.4×10^{3}	8.9×10^{-9}	1.0×10^{-10}
	500	16 m 55 s	10	53	9.9×10^{3}	6.5×10^{-8}	3.1×10^{-10}
	1,000	01 h 39 m 23 s	11	12	2.8×10^{4}	1.9×10^{-5}	4.6×10^{-8}
	1,500	04 h 48 m 57 s	11	27	5.5×10^{4}	3.9×10^{-5}	6.2×10^{-8}
	2,000	05 h 01 m 57 s	7	8	8.7×10^{4}	1.9×10^{-5}	2.2×10^{-8}
1.0	100	40.3 s	13	15	1.7×10^{3}	1.3×10^{-7}	1.5×10^{-9}
	200	01 m 30 s	11	13	4.2×10^{3}	1.4×10^{-6}	8.4×10^{-9}
	500	31 m 46 s	15	45	1.3×10^{4}	2.5×10^{-8}	6.1×10^{-11}
	1,000	03 h 04 m 28 s	12	15	4.7×10^{4}	1.8×10^{-7}	2.2×10^{-10}
	1,500	05 h 09 m 11 s	10	11	8.7×10^{4}	1.6×10^{-5}	1.3×10^{-8}
	2,000	12 h 47 m 19 s	9	10	1.2×10^{5}	2.3×10^{-5}	1.4×10^{-8}
$k = 30$, $c_1 = 10$, and $c_2 = 0.1$							
τ	n	cputime	It.	Func.	Res0.	Res*.	Tol.
0.1	100	31.4 s	8	9	1.7×10^{2}	9.5×10^{-9}	2.0×10^{-10}
	200	04 m 30 s	11	12	3.0×10^{2}	5.3×10^{-6}	5.9×10^{-8}
	500	17 m 31 s	8	9	1.8×10^{3}	1.8×10^{-6}	8.2×10^{-9}
	1,000	02 h 17 m 39 s	9	10	2.8×10^{3}	1.9×10^{-6}	4.4×10^{-9}
	1,500	04 h 32 m 10 s	7	8	6.4×10^{3}	3.5×10^{-7}	5.4×10^{-10}
	2,000	Failed	7	8	1.0×10^{4}	5.6×10^{-4}	6.5×10^{-7}
1.0	100	02 m 36 s	13	14	1.2×10^{3}	7.0×10^{-7}	4.8×10^{-9}
	200	03 m 30 s	7	8	2.8×10^{3}	3.0×10^{-8}	1.0×10^{-10}
	500	45 m 44 s	11	14	1.2×10^{4}	7.0×10^{-6}	9.8×10^{-9}
	1,000	54 m 36 s	9	11	2.9×10^{4}	1.4×10^{-5}	9.6×10^{-9}
	1,500	04 h 53 m 19 s	8	9	6.1×10^{4}	9.8×10^{-5}	4.6×10^{-8}
	2,000	10 h 53 m 54 s	6	7	9.3×10^{4}	2.2×10^{-4}	7.7×10^{-8}

TABLE 4
Numerical results of Example 5.2(b).

$k \approx n/3$, $c_1 = c_2 = 1$								
τ	n	k	cputime	It.	Func.	Res0.	Res*.	Tol.
0.1	100	33	47.8 s	12	13	8.3×10^2	1.5×10^{-6}	3.3×10^{-8}
	200	66	12 m 01 s	16	22	3.3×10^3	1.6×10^{-8}	1.8×10^{-10}
	300	100	01 h 58 m 22 s	14	15	7.7×10^3	2.2×10^{-7}	1.7×10^{-9}
	400	133	05 h 39 m 37 s	17	18	1.6×10^4	6.2×10^{-8}	3.7×10^{-10}
	450	150	05 h 28 m 26 s	20	21	1.7×10^4	5.1×10^{-8}	2.7×10^{-10}
1.0	100	33	37.4 s	9	10	1.6×10^3	2.1×10^{-6}	2.5×10^{-8}
	200	66	01 h 24 m 43 s	45	75	5.2×10^3	1.8×10^{-7}	1.1×10^{-9}
	300	100	01 h 38 m 58 s	29	33	1.5×10^4	6.3×10^{-7}	2.6×10^{-9}
	400	133	04 h 36 m 30 s	16	18	2.5×10^4	2.6×10^{-6}	8.0×10^{-9}
	450	150	04 h 55 m 35 s	18	20	3.1×10^4	1.0×10^{-5}	2.7×10^{-8}
$k \approx n/3$, $c_1 = 10$, and $c_2 = 0.1$								
τ	n	k	cputime	It.	Func.	Res0.	Res*.	Tol.
0.1	100	33	52.0 s	8	9	1.8×10^2	1.0×10^{-7}	2.2×10^{-9}
	200	66	11 m 01 s	9	10	4.5×10^2	5.6×10^{-7}	6.3×10^{-9}
	300	100	02 h 20 m 24 s	13	14	1.1×10^3	1.4×10^{-7}	1.1×10^{-9}
	400	133	03 h 06 m 49 s	10	11	3.0×10^3	1.3×10^{-7}	7.5×10^{-10}
	450	150	06 h 01 m 45 s	15	16	2.4×10^3	1.3×10^{-6}	6.6×10^{-9}
1.0	100	33	01 m 54 s	9	10	9.8×10^2	1.7×10^{-8}	1.2×10^{-10}
	200	66	22 m 27 s	8	9	4.7×10^3	2.2×10^{-5}	7.6×10^{-8}
	300	100	01 h 21 m 14 s	11	13	9.8×10^3	4.3×10^{-6}	1.0×10^{-8}
	400	133	03 h 22 m 32 s	12	14	1.9×10^4	2.5×10^{-5}	4.4×10^{-8}
	450	150	06 h 59 m 29 s	16	20	2.3×10^4	1.7×10^{-5}	2.5×10^{-8}

$$\begin{pmatrix} U_1^{-1}\mathcal{H}\left(\Pi_\Omega((M_a, C_a, K_a) + \mathcal{A}^*(Y^j, Z^j)) + W_j\mathcal{A}^*(\Delta Y^j, \Delta Z^j)\right)(R^T R)^{-1} \\ U_2^{-1}\mathcal{G}\left(\Pi_\Omega((M_a, C_a, K_a) + \mathcal{A}^*(Y^j, Z^j)) + W_j\mathcal{A}^*(\Delta Y^j, \Delta Z^j)\right) \end{pmatrix} = 0,$$

where U_1 and U_2 are given by (3.16) and (3.17), respectively. That is, we apply a preconditioned CG method to the linear system (4.1).

Based on our numerical experiments, we make the following observations.

- Our algorithm converges to the required accuracy at a relatively small number of iterations for all but one case ($n = 2,000$, $k = 30$, $c_1 = 10$, $c_2 = 0.1$, $\tau = 1$) in Table 3. Here, Failed means that the CG method fails to converge within the required maximum number of iterations. Therefore, it is not contradictive with the global convergence result as in Theorem 4.2. Quadratic convergence was observed for all convergent cases. This confirms our theoretical result on quadratic convergence.
- The largest numerical examples that we tested in this paper are (i) $n = 2{,}000$ and $k = 30$ and (ii) $n = 450$ and $k = 150$. For case (i), there are roughly 6,000,000 unknowns in the primal problem and 60,000 unknowns in the dual problem, while for case (ii), these numbers are roughly 300,000 and 67,000, respectively. In consideration of the scales of problems solved, our algorithm is very effective.
- We also tested the cases where $k \approx n/4$, which have better numerical performance than the cases where $k \approx n/3$. Generally speaking, the smaller the ratio $\frac{k}{n}$ is, the better our algorithm performs. This is no surprise as a smaller ratio $\frac{k}{n}$ implies that the constraint nondegenerate condition (4.4) is more likely to hold.
- Finally, note that the numbers of iterations for some small cases (see, for instance, $n = 100, 200$, $\tau = 0.1$, $c_1 = c_2 = 1$ in Table 1 and $n = 300$, $\tau = 0.1$,

TABLE 5
Numerical results of Example 5.1(I).

$k = 10$, $c_1 = c_2 = 1$, and $\tau = 0.1$					
Alg.	n	cputime	`It.`	`Func.`	`Res*.`
Newton	50	1.0 s	6	7	3.1×10^{-5}
	100	2.0 s	8	9	1.0×10^{-5}
	150	3.0 s	8	9	2.9×10^{-5}
	200	3.9 s	7	8	1.0×10^{-6}
BFGS	50	14.3 s	163	328	4.9×10^{-5}
	100	51.4 s	197	395	4.8×10^{-5}
	150	01 m 59 s	202	406	4.7×10^{-5}
	200	LS failed	214	447	5.1×10^{-5}
$k = 10$, $c_1 = 10$, $c_2 = 0.1$, and $\tau = 0.1$					
Alg.	n	cputime	`It.`	`Func.`	`Res*.`
Newton	50	0.9 s	5	6	4.7×10^{-5}
	100	1.6 s	6	7	9.0×10^{-6}
	150	2.2 s	6	7	4.5×10^{-5}
	200	4.1 s	7	8	3.2×10^{-6}
BFGS	50	10.9 s	128	257	3.1×10^{-5}
	100	34.6 s	135	271	3.7×10^{-5}
	150	01 m 20 s	134	269	4.1×10^{-5}
	200	02 m 29 s	135	270	4.4×10^{-5}

$c_1 = c_2 = 1$ in Table 2) are larger than those of some large-scale cases. This is normal, as these problems are randomly generated so that the resulting linear system (4.1) may have different condition numbers and the total number of iterations may be different.

To further illustrate the effectiveness of our method, we compared the performance of our method with that of the well-known BFGS method with the Wolfe line search, which was described in detail in [34, Chapter 8] and was suggested by Malick [30] to solve some semidefinite least-squares problems.

In our experiments for the comparison, the starting point is the solution of (3.15), and the stopping tolerance is

$$\|\nabla\theta(Y^j, Z^j)\| \le 5.0 \times 10^{-5}.$$

The reason that we did not choose a higher accuracy is because the BFGS method may run into difficulties for an accuracy greater than 5.0×10^{-5}. In Tables 5–8, we report the numerical results of Examples 5.1 and 5.2 for the following two cases: (I) $k = 10$, $n = 50, 100, 150, 200$, and $\tau = 0.1$ and (II) $k \approx n/3$, $n = 50, 75, 100$, and $\tau = 0.1$. Table 9 includes the numerical results of Example 5.2 for the fixed $n = 100$ and $k = 10$ with the varying τ: $\tau = 0.01, 0.1, 1.0$. Here, `Res*.` stands for the residuals $\|\nabla\theta(\cdot)\|$ at the final iterate of an algorithm (the maximal number of iterations is set to be 5,000), while `It.` and `Func.` still denote the number of iterations and the number of function evaluations, respectively. LS failed means that the line search failed (i.e., the step length is too small to proceed) during the computation.

We see from the numerical performances that our method needs fewer than 15 iterations for all the problems to obtain the required accuracy, while the BFGS method needs more iterations. In terms of the cputime, our method also performs much better than the BFGS method.

6. Conclusions. In this paper, we considered the inverse quadratic eigenvalue problem (IQEP) with the positive semidefiniteness condition on matrices M and K.

TABLE 6
Numerical results of Example 5.1(II).

$k \approx n/3$, $c_1 = c_2 = 1$, and $\tau = 0.1$						
Alg.	n	k	cputime	`It.`	`Func.`	`Res*.`
Newton	50	16	2.3 s	10	12	2.6×10^{-6}
	75	25	4.8 s	10	11	2.3×10^{-6}
	100	33	18.1 s	13	15	1.1×10^{-5}
BFGS	50	16	48.2 s	326	653	3.3×10^{-5}
	75	25	07 m 40 s	714	1429	4.3×10^{-5}
	100	33	43 m 51 s	1416	2833	4.8×10^{-5}
$k \approx n/3$, $c_1 = 10$, $c_2 = 0.1$, and $\tau = 0.1$						
Alg.	n	k	cputime	`It.`	`Func.`	`Res*.`
Newton	50	16	2.3 s	8	9	1.1×10^{-5}
	75	25	4.9 s	8	9	2.1×10^{-5}
	100	33	13.3 s	10	11	2.0×10^{-5}
BFGS	50	16	40.6 s	284	569	4.3×10^{-5}
	75	25	06 m 39 s	624	1249	4.7×10^{-5}
	100	33	38 m 09 s	1223	2448	4.9×10^{-5}

TABLE 7
Numerical results of Example 5.2(I).

$k = 10$, $c_1 = c_2 = 1$, and $\tau = 0.1$					
Alg.	n	cputime	`It.`	`Func.`	`Res*.`
Newton	50	2.6 s	13	15	5.1×10^{-7}
	100	5.3 s	14	16	1.0×10^{-6}
	150	8.7 s	13	14	2.1×10^{-5}
	200	13.4 s	12	13	2.1×10^{-5}
BFGS	50	22.3 s	294	471	4.2×10^{-5}
	100	01 m 14 s	316	420	3.2×10^{-5}
	150	02 m 57 s	344	449	3.7×10^{-5}
	200	05 m 50 s	372	474	4.3×10^{-5}
$k = 10$, $c_1 = 10$, $c_2 = 0.1$, and $\tau = 0.1$					
Alg.	n	cputime	`It.`	`Func.`	`Res*.`
Newton	50	1.4 s	8	9	6.1×10^{-7}
	100	2.3 s	7	8	2.0×10^{-5}
	150	5.7 s	8	9	3.4×10^{-5}
	200	9.6 s	10	11	1.3×10^{-6}
BFGS	50	16.3 s	203	387	4.6×10^{-5}
	100	56.9 s	229	419	3.8×10^{-5}
	150	03 m 00 s	274	507	2.4×10^{-5}
	200	05 m 15 s	300	551	3.9×10^{-5}

We expressed the IQEP as a semidefinite constraint nonlinear optimization problem and introduced a quadratically convergent Newton method for solving the problem. Our numerical experiments show that our method is very efficient. We also observed that the BFGS method converges much more slowly than the Newton method and, moreover, that the Newton method is capable of solving problems of large scales. Since the vast majority of our computer cputime is spent on the preconditioned conjugate gradient method for solving the linear equation (4.1), we would save much computing time by finding a better preconditioner for (4.1). We leave this as our future research topic. Another interesting topic would be to study the IQEP with M, C, and K being of various special structures.

Appendix A. In this appendix, we study the linear space $\mathcal{R}(\mathcal{H})$ defined by (3.6), which varies according to the distribution of the prescribed eigenvalues. For the sake

TABLE 8
Numerical results of Example 5.2(II).

$k \approx n/3$, $c_1 = c_2 = 1$, and $\tau = 0.1$						
Alg.	n	k	cputime	`It.`	`Func.`	`Res*.`
Newton	50	16	2.2 s	7	8	5.6×10^{-6}
	75	25	4.0 s	7	8	1.7×10^{-5}
	100	33	13.1 s	8	9	2.1×10^{-6}
BFGS	50	16	47.0 s	331	664	4.0×10^{-5}
	75	25	08 m 23 s	789	1580	4.5×10^{-5}
	100	33	39 m 10 s	1260	2523	4.9×10^{-5}
$k \approx n/3$, $c_1 = 10$, $c_2 = 0.1$, and $\tau = 0.1$						
Alg.	n	k	cputime	`It.`	`Func.`	`Res*.`
Newton	50	16	1.9 s	6	7	3.1×10^{-6}
	75	25	5.3 s	6	7	4.3×10^{-5}
	100	33	10.5 s	7	8	6.0×10^{-6}
BFGS	50	16	42.8 s	298	597	4.9×10^{-5}
	75	25	07 m 39 s	717	1435	4.7×10^{-5}
	100	33	37 m 34 s	1206	2413	3.3×10^{-5}

TABLE 9
Numerical results of Example 5.2.

$n = 100$, $k = 10$, and $c_1 = c_2 = 1$					
Alg.	τ	cputime	`It.`	`Func.`	`Res*.`
Newton	0.01	2.8 s	10	11	8.0×10^{-7}
	0.1	2.8 s	10	11	7.6×10^{-7}
	1.0	2.7 s	9	10	8.1×10^{-6}
BFGS	0.01	01 m 10 s	273	538	2.9×10^{-5}
	0.1	01 m 18 s	279	559	4.0×10^{-5}
	1.0	LS failed	472	997	8.7×10^{-5}
$n = 100$, $k = 10$, $c_1 = 10$, and $c_2 = 0.1$					
Alg.	τ	cputime	`It.`	`Func.`	`Res*.`
Newton	0.01	1.4 s	6	7	1.3×10^{-5}
	0.1	1.6 s	6	7	7.2×10^{-6}
	1.0	1.9 s	7	8	2.4×10^{-5}
BFGS	0.01	48.3 s	185	369	4.9×10^{-5}
	0.1	53.9 s	209	417	2.0×10^{-5}
	1.0	01 m 29 s	348	694	4.7×10^{-5}

of convenience, we define a new linear operator $\mathcal{J} : \mathcal{S}^k \times \mathcal{S}^k \times \mathcal{S}^k \to \mathbb{R}^{k\times k}$ by

$$\mathcal{J}(B, D, E) := \frac{1}{\sqrt{c_1}}(\Lambda^2)^T B + \frac{1}{\sqrt{c_2}}\Lambda^T D + E, \quad (B, D, E) \in \mathcal{S}^k \times \mathcal{S}^k \times \mathcal{S}^k,$$

where Λ is defined by (1.3).

Denote

$$\mathcal{R}(\mathcal{J}) := \{\mathcal{J}(B, D, E) \,:\, (B, D, E) \in \mathcal{S}^k \times \mathcal{S}^k \times \mathcal{S}^k\}.$$

By the definitions of $\mathcal{R}(\mathcal{H})$ and $\mathcal{R}(\mathcal{J})$, we can see that $\mathcal{R}(\mathcal{H}) = \mathcal{R}(\mathcal{J})$. The following result gives the exact number of $\mathrm{Dim}\,(\mathcal{R}(\mathcal{H}))$, the dimension of $\mathcal{R}(\mathcal{H})$.

PROPOSITION A.1. *Suppose that the prescribed eigenmatrix* Λ *is defined by* (1.3). *Then we have*

$$\mathrm{Dim}\,(\mathcal{R}(\mathcal{H})) = \mathrm{Dim}\,(\mathcal{R}(\mathcal{J})) = k^2 - \sum_{i=1}^{\mu} s_i(s_i - 1) - \frac{1}{2}\sum_{i=\mu+1}^{\nu} s_i(s_i - 1).$$

In particular, if $s_1 = \cdots = s_\mu = s_{\mu+1} = \cdots = s_\nu = 1$, $\mathrm{Dim}\,(\mathcal{R}(\mathcal{H})) = \mathrm{Dim}\,(\mathcal{R}(\mathcal{J})) = k^2$.

Proof. For any $\widetilde{D} \in \mathcal{S}^k$ and $Y \in \mathbb{R}^{k\times k}$, denote

$$\widetilde{D} = \begin{bmatrix} \widetilde{D}_{1,1} & \cdots & \widetilde{D}_{1,\mu} & \widetilde{D}_{1,\mu+1} & \cdots & \widetilde{D}_{1,\nu} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \widetilde{D}_{1,\mu}^T & \cdots & \widetilde{D}_{\mu,\mu} & \widetilde{D}_{\mu,\mu+1} & \cdots & \widetilde{D}_{\mu,\nu} \\ \widetilde{D}_{1,\mu+1}^T & \cdots & \widetilde{D}_{\mu,\mu+1}^T & \widetilde{D}_{\mu+1,\mu+1} & \cdots & \widetilde{D}_{\mu+1,\nu} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \widetilde{D}_{1,\nu}^T & \cdots & \widetilde{D}_{\mu,\nu}^T & \widetilde{D}_{\mu+1,\nu}^T & \cdots & \widetilde{D}_{\nu,\nu} \end{bmatrix}$$

and

$$Y = \begin{bmatrix} Y_{1,1} & \cdots & Y_{1,\mu} & Y_{1,\mu+1} & \cdots & Y_{1,\nu} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ Y_{\mu,1} & \cdots & Y_{\mu,\mu} & Y_{\mu,\mu+1} & \cdots & Y_{\mu,\nu} \\ Y_{\mu+1,1} & \cdots & Y_{\mu+1,\mu} & Y_{\mu+1,\mu+1} & \cdots & Y_{\mu+1,\nu} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ Y_{\nu,1} & \cdots & Y_{\nu,\mu} & Y_{\nu,\mu+1} & \cdots & Y_{\nu,\nu} \end{bmatrix}, \tag{A.1}$$

where $\widetilde{D}_{i,j}, Y_{i,j} \in \mathbb{R}^{s_i\times s_j}$ and $Y_{j,i} \in \mathbb{R}^{s_j\times s_i}$ with $1 \le i \le j \le \nu$.

First, we note that for any $Y \in \mathbb{R}^{k\times k}$,

(A.2)

$$\begin{aligned}
& Y \in \mathcal{R}(\mathcal{J}) \\
\Leftrightarrow\ & \tfrac{1}{\sqrt{c_1}}(\Lambda^2)^T B + \tfrac{1}{\sqrt{c_2}}\Lambda^T D + E = Y && \text{for some } (B,D,E) \in \mathcal{S}^k\times\mathcal{S}^k\times\mathcal{S}^k \\
\Leftrightarrow\ & E = Y - (\Lambda^2)^T\left(\tfrac{1}{\sqrt{c_1}}B\right) - \Lambda^T\left(\tfrac{1}{\sqrt{c_2}}D\right) && \text{for some } (B,D,E) \in \mathcal{S}^k\times\mathcal{S}^k\times\mathcal{S}^k \\
\Leftrightarrow\ & Y - (\Lambda^2)^T\left(\tfrac{1}{\sqrt{c_1}}B\right) - \Lambda^T\left(\tfrac{1}{\sqrt{c_2}}D\right) = E = E^T \\
& \quad = Y^T - \left(\tfrac{1}{\sqrt{c_1}}B\right)\Lambda^2 - \left(\tfrac{1}{\sqrt{c_2}}D\right)\Lambda && \text{for some } (B,D,E) \in \mathcal{S}^k\times\mathcal{S}^k\times\mathcal{S}^k \\
\Leftrightarrow\ & \left(\tfrac{1}{\sqrt{c_2}}D\right)\Lambda - \Lambda^T\left(\tfrac{1}{\sqrt{c_2}}D\right) \\
& \quad = Y^T - Y + (\Lambda^2)^T\left(\tfrac{1}{\sqrt{c_1}}B\right) - \left(\tfrac{1}{\sqrt{c_1}}B\right)\Lambda^2 && \text{for some } (B,D) \in \mathcal{S}^k\times\mathcal{S}^k \\
\Leftrightarrow\ & \widetilde{D}\Lambda - \Lambda^T\widetilde{D} = Y^T - Y && \text{for some } \widetilde{D} \in \mathcal{S}^k,
\end{aligned}$$

where $\widetilde{D} = \frac{1}{\sqrt{c_2}}D + (\frac{1}{\sqrt{c_1}}B)\Lambda + \Lambda^T(\frac{1}{\sqrt{c_1}}B)$. Then it is easy to verify that the last equation in (A.2) holds for some $\widetilde{D} \in \mathcal{S}^k$ if and only if

$$\widetilde{D}_{i,i}\Lambda_i - \Lambda_i^T\widetilde{D}_{i,i} = Y_{i,i}^T - Y_{i,i}, \quad i = 1,\ldots,\nu, \tag{A.3}$$

$$\widetilde{D}_{i,j}\Lambda_j - \Lambda_i^T\widetilde{D}_{i,j} = Y_{j,i}^T - Y_{i,j}, \quad 1 \le i < j \le \nu. \tag{A.4}$$

- Since $\sigma(\Lambda_i) \cap \sigma(\Lambda_j) = \emptyset$, for any $Y_{i,j} \in \mathbb{R}^{s_i\times s_j}$, there always exists a unique $\widetilde{D}_{i,j} \in \mathbb{R}^{s_i\times s_j}$ such that (A.4) holds.
- For $\mu+1 \le i \le \nu$, since $\Lambda_i = \lambda_i I_{s_i}$, (A.3) holds if and only if

$$Y_{i,i}^T = Y_{i,i}. \tag{A.5}$$

- For $1 \le i \le \mu$,

$$\Lambda_i = \alpha_i I + \beta_i J, \quad J = \operatorname{diag}(\overbrace{J_2, \ldots, J_2}^{s_i}), \quad J_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}.$$

Thus, (A.3) holds if and only if

(A.6)
$$(Y_{i,i}(2q+1:2q+2, 2p+1:2p+2))^T - Y_{i,i}(2p+1:2p+2, 2q+1:2q+2) \in \left\{ \begin{bmatrix} y_1 & y_2 \\ -y_2 & y_1 \end{bmatrix} : y_1, y_2 \in \mathbb{R} \right\},$$

where $0 \le p < q \le s_i - 1$ and $Y_{i,i}(u_1 : u_2,\ v_1 : v_2)$ denotes the submatrix obtained by extracting rows u_1 through u_2 and columns v_1 through v_2 from a matrix $Y_{i,i}$.

Hence, we have obtained that $Y \in \mathcal{R}(\mathcal{J})$ if and only if Y is of form (A.1) satisfying (A.5) and (A.6). Therefore,

$$\operatorname{Dim}(\mathcal{R}(\mathcal{H}))\operatorname{Dim}(\mathcal{R}(\mathcal{J})) = k^2 - \sum_{i=1}^{\mu} s_i(s_i-1) - \frac{1}{2}\sum_{i=\mu+1}^{\nu} s_i(s_i-1),$$

which implies that if $s_1 = \cdots = s_\mu = s_{\mu+1} = \cdots = s_\nu = 1$, $\operatorname{Dim}(\mathcal{R}(\mathcal{H})) = \operatorname{Dim}(\mathcal{R}(\mathcal{J})) = k^2$. □

Remark A.2. From the proof of Proposition A.1 on the dimension of $\mathcal{R}(\mathcal{J})$ (and also $\mathcal{R}(\mathcal{H})$), we actually get the general solution of $\mathcal{J}(B, D, E) = 0$ for any fixed $B \in \mathcal{S}^k$, and thus the general solution of $\mathcal{A}(M, C, K) = 0$ for any fixed $M \in \mathcal{S}^n$. This generalizes a corresponding result of [25] for Λ with simple eigenvalues only.

Appendix B. Let $\Phi_M, \Phi_C, \Phi_K \in \mathcal{S}^n$ be three positive definite matrices. The IQEP of matrix weights can be defined as

(B.1)
$$\begin{aligned} \min \quad & \tfrac{1}{2}\|\Phi_M^{-1}(M - M_a)\Phi_M^{-1}\|^2 + \tfrac{1}{2}\|\Phi_C^{-1}(C - C_a)\Phi_C^{-1}\|^2 + \tfrac{1}{2}\|\Phi_K^{-1}(K - K_a)\Phi_K^{-1}\|^2 \\ \text{s.t.} \quad & M \begin{bmatrix} R \\ 0 \end{bmatrix} \Lambda^2 + C \begin{bmatrix} R \\ 0 \end{bmatrix} \Lambda + K \begin{bmatrix} R \\ 0 \end{bmatrix} = 0, \\ & (M, C, K) \in \Omega. \end{aligned}$$

By renaming $M := \Phi_M^{-1} M \Phi_M^{-1}$, $C := \Phi_C^{-1} C \Phi_C^{-1}$, $K := \Phi_K^{-1} K \Phi_K^{-1}$, $M_a := \phi_M^{-1} M_a \Phi_M^{-1}$, $C_a := \Phi_C^{-1} C_a \Phi_C^{-1}$, and $K_a := \Phi_K^{-1} K_a \Phi_K^{-1}$, we can write problem (B.1) equivalently as

(B.2)
$$\begin{aligned} \min \quad & \tfrac{1}{2}\|M - M_a\|^2 + \tfrac{1}{2}\|C - C_a\|^2 + \tfrac{1}{2}\|K - K_a\|^2 \\ \text{s.t.} \quad & (\Phi_M M \Phi_M) \begin{bmatrix} R \\ 0 \end{bmatrix} \Lambda^2 + (\Phi_C C \Phi_C) \begin{bmatrix} R \\ 0 \end{bmatrix} \Lambda + (\Phi_K K \Phi_K) \begin{bmatrix} R \\ 0 \end{bmatrix} = 0, \\ & (M, C, K) \in \Omega. \end{aligned}$$

Then, from the proof of Theorem 1.1, we can see that problem (B.2) admits a strictly feasible solution if and only if Λ is nonsingular.

Next, we shall show how problem (B.1) can be reduced to a similar problem which has a strictly feasible solution if Λ is singular. Without loss of generality, we assume that

$$\Lambda = \mathrm{diag}\{O_t, \Gamma\},$$

where $O_t = \mathrm{diag}\{\overbrace{0, \ldots, 0}^{t}\}$ and Γ is nonsingular and has the same structure as Λ defined in (1.3). Partition M, C, K, and $\left[\begin{smallmatrix} R \\ 0 \end{smallmatrix}\right]$ by

$$M = \begin{bmatrix} M_1 & M_2 \\ M_2^T & M_4 \end{bmatrix}, \quad C = \begin{bmatrix} C_1 & C_2 \\ C_2^T & C_4 \end{bmatrix}, \quad K = \begin{bmatrix} K_1 & K_2 \\ K_2^T & K_4 \end{bmatrix},$$

$$\text{and} \quad \begin{bmatrix} R \\ 0 \end{bmatrix} = \begin{bmatrix} R_1 & R_2 \\ 0 & R_3 \end{bmatrix},$$

where $M_1, C_1, K_1 \in \mathcal{S}^t$, $M_2, K_2, C_2, \in \mathbb{R}^{t\times(n-t)}$, $M_4, C_4, K_4 \in \mathcal{S}^{(n-t)}$, and $R_1 \in \mathbb{R}^{t\times t}$, $R_2 \in \mathbb{R}^{t\times(k-t)}$, $R_3 \in \mathbb{R}^{(n-t)\times(k-t)}$. The matrix R_3 can be further written as $R_3 = \left[\begin{smallmatrix} R_4 \\ 0 \end{smallmatrix}\right]$, where $R_4 \in \mathbb{R}^{(k-t)\times(k-t)}$ is an invertible upper triangular matrix. Then the first equation in problem (B.1) takes the following form:

$$\text{(B.3)} \qquad \begin{cases} K_1R_1 = 0, \quad K_2^TR_1 = 0, \\ (M_1R_2\Gamma^2 + C_1R_2\Gamma) + (M_2R_3\Gamma^2 + C_2R_3\Gamma) + (K_1R_2 + K_2R_3) = 0, \\ (M_2^TR_2\Gamma^2 + C_2^TR_2\Gamma) + (M_4R_3\Gamma^2 + C_4R_3\Gamma + K_4R_3) + K_2^TR_2 = 0. \end{cases}$$

Since R_1 is invertible, we have

$$K_1 = 0 \quad \text{and} \quad K_2 = 0.$$

Let

$$\widehat{\mathcal{A}}(M, C, K_4) := \begin{pmatrix} (M_1R_2\Gamma^2 + C_1R_2\Gamma) + (M_2R_3\Gamma^2 + C_2R_3\Gamma) \\ (M_2^TR_2\Gamma^2 + C_2^TR_2\Gamma) + (M_4R_3\Gamma^2 + C_4R_3\Gamma + K_4R_3) \end{pmatrix}.$$

Then the last two equations in (B.3) reduce to $\widehat{\mathcal{A}}(M, C, K_4) = 0$. Partition K_a by

$$K_a = \begin{bmatrix} (K_a)_1 & (K_a)_2 \\ (K_a)_2^T & (K_a)_4 \end{bmatrix}$$

with $(K_a)_1 \in \mathcal{S}^t$, $(K_a)_2 \in \mathbb{R}^{t\times(n-t)}$, and $(K_a)_4 \in \mathcal{S}^{(n-t)}$. Let

$$\Psi := \Phi_K^{-1} = \begin{bmatrix} L_1 & L_2 \\ L_2^T & L_4 \end{bmatrix} \quad \text{and} \quad \Psi K_a \Psi := \begin{bmatrix} (H_a)_1 & (H_a)_2 \\ (H_a)_2^T & (H_a)_4 \end{bmatrix},$$

where $L_1, (H_a)_1 \in \mathcal{S}^t$, $L_2, (H_a)_2 \in \mathbb{R}^{t\times(n-t)}$, and $L_4, (H_a)_4 \in \mathcal{S}^{(n-t)}$. By using the fact that

$$\begin{aligned} \|\Psi(K - K_a)\Psi\|^2 &= \left\| \begin{bmatrix} L_2K_4L_2^T - (H_a)_1 & L_2K_4L_4 - (H_a)_2 \\ L_4K_4L_2^T - (H_a)_2^T & L_4K_4L_4 - (H_a)_4 \end{bmatrix} \right\|^2 \\ &= \left\|\Psi_K^{-1}(K_4 - \Upsilon_a)\Psi_K^{-1}\right\|^2 + \kappa, \end{aligned}$$

where

$$\begin{aligned}\Psi_K &= (L_2^T L_2 + L_4^2)^{-\frac{1}{2}},\\ \Upsilon_a &= \Psi_K^2\left[(L_2^T(H_a)_1 L_2 + 2L_2^T(H_a)_2 L_4 + L_4(H_a)_4 L_4\right]\Psi_K^2,\\ \kappa &= -\|\Psi_K^{-1}\Upsilon_a\Psi_K^{-1}\|^2 + \|\Psi K_a \Psi\|^2,\end{aligned}$$

we can see that problem (B.1) is equivalent to

(B.4)
$$\begin{aligned}\min\quad & \tfrac{1}{2}\|\Phi_M^{-1}(M-M_a)\Phi_M^{-1}\|^2 + \tfrac{1}{2}\|\Phi_C^{-1}(C-C_a)\Phi_C^{-1}\|^2 + \tfrac{1}{2}\|\Psi_K^{-1}(K_4-\Upsilon_a)\Psi_K^{-1}\|^2\\ \text{s.t.}\quad & \widehat{\mathcal{A}}(M,C,K_4)=0,\\ & M\succeq 0,\quad C^T=C,\quad K_4\succeq 0.\end{aligned}$$

By renaming $M := \Phi_M^{-1}M\Phi_M^{-1}$, $C := \Phi_C^{-1}C\Phi_C^{-1}$, $K_4 := \Psi_K^{-1}K_4\Psi_K^{-1}$, $M_a := \Phi_M^{-1}M_a\Phi_M^{-1}$, $C_a := \Phi_C^{-1}C_a\Phi_C^{-1}$, and $\Upsilon_a := \Psi_K^{-1}\Upsilon_a\Psi_K^{-1}$, we see that problem (B.4) takes the following form:

(B.5)
$$\begin{aligned}\min\quad & \tfrac{1}{2}\|M-M_a\|^2 + \tfrac{1}{2}\|C-C_a\|^2 + \tfrac{1}{2}\|K_4-\Upsilon_a\|^2\\ \text{s.t.}\quad & \widehat{\mathcal{A}}\left(\Phi_M M\Phi_M, \Phi_C C\Phi_C, \Psi_K K_4\Psi_K\right)=0,\\ & M\succeq 0,\quad C^T=C,\quad K_4\succeq 0.\end{aligned}$$

Let $\widehat{M}$, $\widehat{C}$, and $\widehat{K}_4$ be defined by

$$\widehat{M} := \begin{bmatrix}\widehat{M}_1 & \widehat{M}_2\\ \widehat{M}_2^T & \widehat{M}_4\end{bmatrix},\quad \widehat{C} := \begin{bmatrix}\widehat{C}_1 & \widehat{C}_2\\ \widehat{C}_2^T & \widehat{C}_4\end{bmatrix},\quad \text{and}\quad \widehat{K}_4 := \begin{bmatrix}R_4^{-T}\Gamma^T\Gamma R_4^{-1} & 0\\ 0 & I\end{bmatrix},$$

where

$$\widehat{M}_1 = R_1^{-T}R_1^{-1},\quad \widehat{M}_2 = -\begin{bmatrix}R_1^{-T}R_1^{-1}R_2R_4^{-1} & 0\end{bmatrix},\quad \widehat{C}_1=0,\quad \widehat{C}_2=0,$$

and

$$\widehat{M}_4 = \begin{bmatrix}R_4^{-T}R_2^TR_1^{-T}R_1^{-1}R_2R_4^{-1} + R_4^{-T}R_4^{-1} & 0\\ 0 & I\end{bmatrix},$$

$$\widehat{C}_4 = -\begin{bmatrix}R_4^{-T}(\Gamma+\Gamma^T)R_4^{-1} & 0\\ 0 & 0\end{bmatrix}.$$

Clearly,

$$\left(\Phi_M^{-1}\widehat{M}\Phi_M^{-1}, \Phi_C^{-1}\widehat{C}\Phi_C^{-1}, \Psi_K^{-1}\widehat{K}_4\Psi_K^{-1}\right)$$

is a strictly feasible solution to problem (B.5). Therefore, we can apply Algorithm 4.1 to the dual of problem (B.5).

Acknowledgments. The authors would like to thank the associate editor and the referees for their valuable comments.

REFERENCES

[1] F. Alizadeh, J.-P. A. Haeberly, and M. L. Overton, *Complementarity and nondegeneracy in semidefinite programming*, Math. Programming, 77 (1997), pp. 111–128.

[2] V. I. Arnold, *On matrices depending on parameters*, Russian Math. Surveys, 26 (1971), pp. 29–43.

[3] M. Baruch, *Optimization procedure to correct stiffness and flexibility matrices using vibration data*, AIAA J., 16 (1978), pp. 1208–1210.

[4] M. Baruch and I. Y. Bar-Itzhack, *Optimal weighted orthogonalization of measured modes*, AIAA J., 16 (1978), pp. 346–351.

[5] A. Berman, *Comment on "Optimal weighted orthogonalization of measured modes,"* AIAA J., 17 (1979), pp. 927–928.

[6] A. Berman and E. J. Nagy, *Improvement of a large analytical model using test data*, AIAA J., 21 (1983), pp. 1168–1173.

[7] R. Bhatia, *Matrix Analysis*, Springer, New York, 1997.

[8] J. F. Bonnans, R. Cominetti, and A. Shapiro, *Sensitivity analysis of optimization problems under second order regularity constraints*, Math. Oper. Res., 23 (1998), pp. 803–832.

[9] J. F. Bonnans and A. Shapiro, *Perturbation Analysis of Optimization Problems*, Springer, New York, 2000.

[10] S. Boyd and L. Xiao, *Least-squares covariance matrix adjustment*, SIAM J. Matrix Anal. Appl., 27 (2005), pp. 532–546.

[11] X. Chen, H. Qi, and P. Tseng, *Analysis of nonsmooth symmetric-matrix-valued functions with applications to semidefinite complementarity problems*, SIAM J. Optim., 13 (2003), pp. 960–985.

[12] M. T. Chu, Y.-C. Kuo, and W.-W. Lin, *On inverse quadratic eigenvalue problems with partially prescribed eigenstructure*, SIAM J. Matrix Anal. Appl., 25 (2004), pp. 995–1020.

[13] F. H. Clarke, *Optimization and Nonsmooth Analysis*, John Wiley & Sons, New York, 1983.

[14] B. N. Datta, *Finite-element model updating, eigenstructure assignment and eigenvalue embedding techniques for vibrating systems*, Mech. Systems Signal Process., 16 (2002), pp. 83–96.

[15] J. E. Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice–Hall, Englewood Cliffs, NJ, 1983.

[16] W. F. Donoghue, *Monotone Matrix Functions and Analytic Continuation*, Springer, New York, 1974.

[17] F. Facchinei, *Minimization of SC^1 functions and the Maratos effect*, Oper. Res. Lett., 17 (1995), pp. 131–137.

[18] M. I. Friswell, D. J. Inman, and D. F. Pilkeyand, *The direct updating of damping and stiffness matrices*, AIAA J., 36 (1998), pp. 491–493.

[19] M. I. Friswell and J. E. Mottershead, *Finite Element Model Updating in Structural Dynamics*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1995.

[20] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, London, 1996.

[21] N. J. Higham, *Computing a nearest symmetric positive semidefinite matrix*, Linear Algebra Appl., 103 (1988), pp. 103–118.

[22] N. J. Higham, *Computing the nearest correlation matrix—A problem from finance*, IMA J. Numer. Anal., 22 (2002), pp. 329–343.

[23] M. Kojima and S. Shindo, *Extensions of Newton and quasi-Newton methods to systems of PC^1 equations*, J. Oper. Res. Soc. Japan, 29 (1986), pp. 352–374.

[24] B. Kummer, *Newton's method for nondifferentiable functions*, in Advances in Mathematical Optimization, Math. Res. 45, Akademie-Verlag, Berlin, 1988, pp. 114–125.

[25] Y.-C. Kuo, W.-W. Lin, and S.-F. Xu, *Solutions of the partially described inverse quadratic eigenvalue problem*, SIAM J. Matrix Anal. Appl., 29 (2006), pp. 33–53.

[26] Y. C. Kuo, W. W. Lin, and S. F. Xu, *A New Model Updating Method for Quadratic Eigenvalue Problems.* II, Preprint 2005-2-001, National Center for Theoretical Sciences, National Tsing Hua University, Taiwan, 2005.

[27] M. Link, *Updating of analytical models—Review of numerical procedures and application aspects*, in Structural Dynamics 2000: Current Status and Future Directions, D. J. Ewins and D. J. Inman, eds., Research Studies Press, Baldock, UK, 2001, pp. 193–223.

[28] K. Löwner, *Über monotone Matrixfunktionen*, Math. Z., 38 (1934), pp. 177–216.

[29] N. M. M. Maia and J. M. M. Silva, *Theoretical and Experimental Modal Analysis*, Research Studies Press, Somerset, UK, 1997.

[30] J. Malick, *A dual approach to semidefinite least-squares problems*, SIAM J. Matrix Anal. Appl., 26 (2004), pp. 272–284.

[31] F. Meng, D. Sun, and G. Zhao, *Semismoothness of solutions to generalized equations and the Moreau–Yosida regularization*, Math. Program., 104 (2005), pp. 561–581.
[32] R. Mifflin, *Semismooth and semiconvex functions in constrained optimization*, SIAM J. Control Optim., 15 (1977), pp. 959–972.
[33] J. E. Mottershead and M. I. Friswell, *Model updating in structural dynamics: A survey*, J. Sound Vibration, 167 (1993), pp. 347–375.
[34] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, New York, 1999.
[35] J. S. Pang, D. Sun, and J. Sun, *Semismooth homeomorphisms and strong stability of semidefinite and Lorentz complementarity problems*, Math. Oper. Res., 28 (2003), pp. 39–63.
[36] H.-D. Qi, *Some theoretical aspects of Newton's method for constrained best interpolation*, in Continuous Optimization: Current Trends and Modern Applications, Springer, New York, 2005, pp. 23–49.
[37] H. D. Qi and D. Sun, *A quadratically convergent Newton method for computing the nearest correlation matrix*, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 360–385.
[38] L. Qi, *Convergence analysis of some algorithms for solving nonsmooth equations*, Math. Oper. Res., 18 (1993), pp. 227–244.
[39] L. Qi and J. Sun, *A nonsmooth version of Newton's method*, Math. Programming, 58 (1993), pp. 353–367.
[40] S. M. Robinson, *Local structure of feasible sets in nonlinear programming.* II. *Nondegeneracy*, Math. Programming Stud., 22 (1984), pp. 217–230.
[41] S. M. Robinson, *Local structure of feasible sets in nonlinear programming.* III. *Stability and sensitivity*, Math. Programming Stud., 30 (1987), pp. 45–66.
[42] S. M. Robinson, *Constraint nondegeneracy in variational analysis*, Math. Oper. Res., 28 (2003), pp. 201–232.
[43] R. T. Rockafellar, *Conjugate Duality and Optimization*, CBMS-NSF Regional Conf. Ser. in Appl. Math. 16, SIAM, Philadelphia, 1974.
[44] R. T. Rockafellar and R. J. B. Wets, *Variational Analysis*, Springer, Berlin, 1998.
[45] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.
[46] A. Shapiro, *Sensitivity analysis of generalized equations*, J. Math. Sci. (N.Y.), 115 (2003), pp. 2554–2565.
[47] D. Sun and J. Sun, *Semismooth matrix valued functions*, Math. Oper. Res., 27 (2002), pp. 150–169.
[48] F. Tisseur and K. Meerbergen, *The quadratic eigenvalue problem*, SIAM Rev., 43 (2001), pp. 235–286.
[49] P. Tseng, *Merit functions for semi-definite complementarity problems*, Math. Programming, 83 (1998), pp. 159–185.
[50] K. C. Toh, R. H. Tütüncü, and M. J. Todd, *Inexact primal-dual path-following algorithms for a special class of convex quadratic SDP and related problems*, Pac. J. Optim., 3 (2007), pp. 135–164.
[51] F.-S. Wei, *Structural dynamic model identification using vibration test data*, in Proceedings of the 7th International Modal Analysis Conference (New York), Society for Experimental Mechanics, Inc., Las Vegas, NV, 1989, pp. 562–567.
[52] F.-S. Wei, *Structural dynamic model improvement using vibration test data*, AIAA J., 28 (1990), pp. 175–177.
[53] F.-S. Wei, *Mass and stiffness interaction effects in analytical model modification*, AIAA J., 28 (1990), pp. 1686–1688.
[54] E. H. Zarantonello, *Projections on convex sets in Hilbert space and spectral theory.* I *and* II, in Contributions to Nonlinear Functional Analysis, E. H. Zarantonello, ed., Academic Press, New York, 1971, pp. 237–424.

SIAM J. SCI. COMPUT.
Vol. 29, No. 6, pp. 2562–2580
© 2007 Society for Industrial and Applied Mathematics

INEXACT UNIFORMIZATION METHOD FOR COMPUTING TRANSIENT DISTRIBUTIONS OF MARKOV CHAINS*

ROGER B. SIDJE†, KEVIN BURRAGE†, AND SHEV MACNAMARA†

Abstract. The uniformization method (also known as randomization) is a numerically stable algorithm for computing transient distributions of a continuous time Markov chain. When the solution is needed after a long run or when the convergence is slow, the uniformization method involves a large number of matrix-vector products. Despite this, the method remains very popular due to its ease of implementation and its reliability in many practical circumstances. Because calculating the matrix-vector product is the most time-consuming part of the method, overall efficiency in solving large-scale problems can be significantly enhanced if the matrix-vector product is made more economical. In this paper, we incorporate a new relaxation strategy into the uniformization method to compute the matrix-vector products only approximately. We analyze the error introduced by these inexact matrix-vector products and discuss strategies for refining the accuracy of the relaxation while reducing the execution cost. Numerical experiments drawn from computer systems and biological systems are given to show that significant computational savings are achieved in practical applications.

Key words. Markov chains, uniformization, inexact methods, relaxed matrix-vector

AMS subject classifications. 65C40, 65F50

DOI. 10.1137/060662629

1. Introduction.

Computing transient probabilities is an important problem in Markov chain modeling. Numerical techniques are based on solving the Chapman–Kolmogorov system of differential equations:

$$\begin{cases} \dfrac{d\boldsymbol{w}(t)}{dt} = \boldsymbol{A}\boldsymbol{w}(t), & t \in [0,\ T], \\ \boldsymbol{w}(0) = \boldsymbol{v}, & \text{an initial probability distribution.} \end{cases} \tag{1}$$

The coefficient matrix $\boldsymbol{A}$ is an *infinitesimal generator* of order n, where n is the number of states in the Markov chain. Thus $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, with elements $a_{ij} \geq 0$ when $i \neq j$, and $a_{jj} = -\sum_{i=1, i\neq j}^{n} a_{ij}$, which means that $\boldsymbol{A}$ is a minus Z-matrix [2] with zero column sum (our notation uses column vectors and transposes the transition matrix; see, e.g., [18, section 8.5.2]). Of interest is the *transient solution*, $\boldsymbol{w}(t)$, which is given by the solution of (1) and is known to be

$$\boldsymbol{w}(t) = e^{t\boldsymbol{A}}\boldsymbol{v}. \tag{2}$$

Since the matrix exponential can be full even when the original matrix is sparse, the practical computation of $e^{t\boldsymbol{A}}$ in full remains possible only when $\boldsymbol{A}$ is relatively small, i.e., when the number of states in the Markov chain does not exceed a few hundreds. Examples of such methods can be found in Moler and Van Loan [11]. In general, aspects such as size, stiffness, and accuracy are the main concerns when solving

*Received by the editors June 9, 2006; accepted for publication (in revised form) March 26, 2007; published electronically October 24, 2007. This work was supported by the National Facility of the Australian Partnership for Advanced Computing (APAC).

http://www.siam.org/journals/sisc/29-6/66262.html

†Department of Mathematics, Advanced Computational Modelling Centre, University of Queensland, Brisbane QLD 4072, Australia (rbs@maths.uq.edu.au, kb@maths.uq.edu.au, shev@maths.uq.edu.au). The second author thanks the Australian Research Council (ARC) for supporting him via a Federation Fellowship.

Markov chains numerically. Solution techniques for large-scale problems include general ODE solvers and Krylov-based methods [13, 14, 18]. However, Markovian analysts have traditionally used the uniformization method, which still remains very popular. Evidence suggests that it can work very well, especially on nonstiff problems [15], and this is one of the reasons why it remains one of the most widely used methods for computing transient solutions.

Because the matrix-vector product is the most time-consuming part of the uniformization method, the overall efficiency of the method in solving large-scale problems can be significantly enhanced if the matrix-vector product is made more economical. In this paper, we incorporate a new relaxation strategy into the method so as to perform matrix-vector products only approximately. We analyze the error introduced by these inexact matrix-vector products, and discuss strategies for refining the accuracy of the relaxation while reducing the execution cost. Numerical experiments drawn from computer systems and biological systems show that significant computational savings are achieved in practical applications.

The paper is structured as follows. Section 2 provides a brief overview of the uniformization method. Section 3 describes our proposed variant of the method based on matrix-vector products that are inexact. An analysis of the error is given, and strategies to refine the accuracy while reducing the execution cost are discussed. Section 4 presents some numerical results. Section 5 provides some concluding remarks.

2. The uniformization method. The uniformization (or randomization) algorithm is based on the evaluation of the ℓth partial Taylor series expansion of the matrix exponential [6, 7]. The length ℓ is determined so that the prescribed tolerance on the approximation is fulfilled. Since $\boldsymbol{A}$ is a minus Z-matrix (i.e., the diagonal elements of $\boldsymbol{A}$ are negative and the off-diagonal elements are nonnegative), a naive use of the expression

$$\boldsymbol{w}(t) = e^{t\boldsymbol{A}}v \approx \boldsymbol{v} + \frac{t\boldsymbol{A}}{1!}\boldsymbol{v} + \cdots + \frac{(t\boldsymbol{A})^{\ell}}{\ell!}\boldsymbol{v} + \cdots$$

is subject to severe roundoff errors due to terms of alternating signs. Thus the uniformization technique instead uses the modified formulation

$$\boldsymbol{w}(t) = e^{\alpha t(\boldsymbol{P}-\boldsymbol{I})}\boldsymbol{v} = e^{-\alpha t}e^{\alpha t\boldsymbol{P}}\boldsymbol{v}, \quad \boldsymbol{P} = \boldsymbol{I} + \frac{1}{\alpha}\boldsymbol{A}, \quad \alpha = \max_i |a_{ii}|,$$

where $0 \le \boldsymbol{P} \le 1$ componentwise and $\|\boldsymbol{P}\|_1 = 1$. The resulting truncated approximation

$$\overline{\boldsymbol{w}}(t) = \sum_{k=0}^{\ell} e^{-\alpha t}\frac{(\alpha t)^k}{k!}\boldsymbol{P}^k\boldsymbol{v} \tag{3}$$

involves only nonnegative terms and becomes numerically stable. If ε_{tol} denotes the prescribed error tolerance, the condition

$$\|\boldsymbol{w}(t) - \overline{\boldsymbol{w}}(t)\|_1 \le \varepsilon_{\text{tol}}$$

leads to a choice of ℓ such that

$$\sum_{k=0}^{\ell} e^{-\alpha t}\frac{(\alpha t)^k}{k!} \ge 1 - \varepsilon_{\text{tol}}. \tag{4}$$

The length ℓ may therefore be determined simply by adding up the above series until the inequality is met. In fact, practical implementations will subdivide the integration domain to prevent overflow issues. Specifically, choose θ in advance (say, $\theta = 100$), and set

$$m = \left\lceil \frac{\alpha t}{\theta} \right\rceil, \quad \bar{t} = \frac{t}{m};$$

then we have

$$\overline{\boldsymbol{w}}(t) = \left(e^{-\alpha \bar{t}} e^{\alpha \bar{t} \boldsymbol{P}} \right)^m \boldsymbol{v} = e^{-\alpha \bar{t}} e^{\alpha \bar{t} \boldsymbol{P}} \cdots e^{-\alpha \bar{t}} e^{\alpha \bar{t} \boldsymbol{P}} \boldsymbol{v}, \tag{5}$$

which is evaluated from left to right, and each step amounts essentially to our earlier discussion with different $\boldsymbol{v}$'s and a reduced $\bar{t}$ such that $\alpha \bar{t} \leq \theta$. The overall scheme is summarized in the pseudocode below.

ALGORITHM 1: Uniformization$(t, A, v, \varepsilon_{\text{tol}})$.
{Compute $\boldsymbol{w} \approx \exp(t\boldsymbol{A})\boldsymbol{v}$}
1. $\alpha = \max_i |a_{ii}|$; $\boldsymbol{P} := \boldsymbol{I} + \boldsymbol{A}/\alpha$;
2. $\theta := 100$; $m := \lceil \alpha t/\theta \rceil$; $t := t/m$;
3. $s := \alpha t$; $r := e^{-s}$;
4. Choose ℓ to satisfy (4) ;
5. $\boldsymbol{w} := \boldsymbol{v}$;
6. **for** $i := 1 : m$ **do**
7. $\quad \boldsymbol{f} := \boldsymbol{w}$;
8. $\quad$ **for** $k := 1 : \ell$ **do**
9. $\quad\quad \boldsymbol{f} := \frac{s}{k} \boldsymbol{P} \boldsymbol{f}$;
10. $\quad\quad \boldsymbol{w} := \boldsymbol{w} + \boldsymbol{f}$;
11. $\quad$ **end**
12. $\quad \boldsymbol{w} := r\boldsymbol{w}$;
13. **end**

The popularity of uniformization is related to three main facts. First, its simplicity and malleability facilitate its implementation—only a matrix-vector product is needed per Horner-like iteration, with absolutely minimal extra storage (the matrix $\boldsymbol{P}$ need not be formed explicitly). Second, it works surprisingly well in a great variety of circumstances. Third, and perhaps most significantly, the transformation from $\boldsymbol{A}$ to $\boldsymbol{P}$ has a concrete interpretation. The matrix $\boldsymbol{P}$ is stochastic and is the transition matrix of a discrete time Markov chain (DTMC) that emulates the behavior of the continuous time Markov chain (CTMC), whose infinitesimal generator is $\boldsymbol{A}$. As described in Gross and Miller [7], this probabilistic interpretation allows deriving other parameters of interest. A shortcoming of uniformization is that ℓ is likely to be large, and thus (3) involves many matrix-vector multiplications, for large values of αt. Therefore, making the matrix-vector product economical, as we are proposing, can substantially decrease the execution time.

3. Inexact uniformization. Over recent years, there has been a growing interest in using inexact (or relaxed) matrix-products that are computed only to sufficient accuracy as needed [3, 16, 19]. Such interest has been motivated by applications where the matrix is not known exactly or is too expensive to apply. Most of the studies have focused, however, on Krylov subspace methods because by construction they are matrix-free and heavily dependent on matrix-vector products. The results obtained there have so far been quite promising [4, 10].

3.1. The inexact uniformization algorithm. Based on this evidence, we are interested in developing an inexact uniformization, where the matrix in statement 9 of Algorithm 1 is allowed to be $\boldsymbol{P} + \boldsymbol{E}_k$, where $\boldsymbol{E}_k$ is an error matrix at the kth step, and can change at each iteration. Thus, at each step, rather than computing $\boldsymbol{f}_k = \frac{\alpha t}{k}\boldsymbol{P}\boldsymbol{f}_{k-1}$, we compute an approximation $\tilde{\boldsymbol{f}}_k = \frac{\alpha t}{k}(\boldsymbol{P}+\boldsymbol{E}_k)\tilde{\boldsymbol{f}}_{k-1}$. Directly doing such a substitution is, however, fraught with the risk of corrupting the computations. To understand why, observe that if we are to compute $\tilde{\boldsymbol{f}}_k = \frac{\alpha t}{k}(\boldsymbol{P} + \boldsymbol{E}_k)\tilde{\boldsymbol{f}}_{k-1}$, the sequence of $\|\tilde{\boldsymbol{f}}_k\|$ grows before it decays, amplifying any error introduced and/or making it difficult to monitor the error. In general, this illustrates a possible pitfall when converting an existing method to using an inexact matrix-vector product. Thus we draw the attention of the reader to the fact that it may sometimes be necessary to recast an algorithm for it to fully benefit from the inexact matrix-vector idea.

In our present case, a small yet vital change is needed to make the approach successful. In this formulation, we split the computations using

$$\gamma_k = \frac{(\alpha t)^k}{k!}$$

and

$$\widetilde{\boldsymbol{f}}_k = (\boldsymbol{P} + \boldsymbol{E}_k)\widetilde{\boldsymbol{f}}_{k-1},$$

where $\boldsymbol{E}_k$ is the error matrix at step k. The updated algorithm is given below, showing the change (statement 11 of Algorithm 2).

ALGORITHM 2: InexactUniformization$(t, A, v, \varepsilon_{\text{tol}})$.

{Compute $\boldsymbol{w} \approx \exp(t\boldsymbol{A})\boldsymbol{v}$}

```
 1.  α = max_i |a_ii| ;   P := I + A/α ;
 2.  θ := 100 ;   m := ⌈αt/θ⌉ ;   t := t/m ;
 3.  s := αt ;   r := e^{-s} ;
 4.  Choose ℓ to satisfy (4) ;
 5.  w := v ;
 6.  for i := 1 : m do
 7.      γ := 1 ;
 8.      f := w ;
 9.      for k := 1 : ℓ do
10.          γ := γ s/k ;
11.          f :≈ Pf ;
12.          w := w + γf ;
13.      end
14.      w := rw ;
15.  end
```

If we were using *exact* arithmetic with $\boldsymbol{E}_k = 0$, we would generate the sequence of vectors $\boldsymbol{f}_k = \boldsymbol{P}\boldsymbol{f}_{k-1}$, which themselves would remain probability vectors, but with our relaxation approach, we compute

$$e^{\alpha t}\widetilde{\boldsymbol{w}} = \boldsymbol{v} + \sum_{k=1}^{\ell} \frac{(\alpha t)^k}{k!}(\boldsymbol{P} + \boldsymbol{E}_k)\cdots(\boldsymbol{P} + \boldsymbol{E}_1)\boldsymbol{v}$$

$$= \boldsymbol{v} + \sum_{k=1}^{\ell} \frac{(\alpha t)^k}{k!}(\boldsymbol{P}^k + \boldsymbol{\mathcal{E}}_k)\boldsymbol{v},$$

where

$$\boldsymbol{\mathcal{E}}_k = (\boldsymbol{P} + \boldsymbol{E}_k) \cdots (\boldsymbol{P} + \boldsymbol{E}_1) - \boldsymbol{P}^k.$$

Therefore the global error due to the inexact matrix-vector products is

$$\overline{\boldsymbol{w}} - \widetilde{\boldsymbol{w}} = -\sum_{k=1}^{\ell} e^{-\alpha t} \frac{(\alpha t)^k}{k!} \boldsymbol{\mathcal{E}}_k \boldsymbol{v}, \tag{6}$$

and the total error is

$$\|\boldsymbol{w} - \widetilde{\boldsymbol{w}}\|_1 = \|(\boldsymbol{w} - \overline{\boldsymbol{w}}) + (\overline{\boldsymbol{w}} - \widetilde{\boldsymbol{w}})\|_1 \le \varepsilon_{\text{tol}} + \|\overline{\boldsymbol{w}} - \widetilde{\boldsymbol{w}}\|_1. \tag{7}$$

In order to show that our inexact scheme is a viable approach, we have to show that the gap between the true solution and the computed solution, $\|\overline{\boldsymbol{w}} - \widetilde{\boldsymbol{w}}\|_1$, can be bounded in a practically useful way. This correlates with [16, 19], where a bound is established for the gap between the true residual and the computed residual in the context of linear systems. To obtain a meaningful bound in our context, it is helpful to consider the relationship between the global error and the local errors. This relationship is derived in the following result.

THEOREM 3.1. *Let $\widetilde{\boldsymbol{f}}_0 \equiv \boldsymbol{v}$ be the given starting vector, and for $k = 1, \ldots, \ell$ let $\widetilde{\boldsymbol{f}}_k$ be the approximate vector computed by the inexact matrix-vector product at step k, and let $\boldsymbol{\varepsilon}_k$ be the corresponding local error, that is,*

$$\boldsymbol{P}\widetilde{\boldsymbol{f}}_{k-1} = \widetilde{\boldsymbol{f}}_k + \boldsymbol{\varepsilon}_k, \quad k = 1, \ldots, \ell.$$

Then the global error of the inexact uniformization method satisfies

$$\overline{\boldsymbol{w}} - \widetilde{\boldsymbol{w}} = \sum_{k=1}^{\ell} e^{-\alpha t} \frac{(\alpha t)^k}{k!} \sum_{j=1}^{k} \boldsymbol{P}^{k-j} \boldsymbol{\varepsilon}_j \tag{8}$$

and therefore

$$\|\overline{\boldsymbol{w}} - \widetilde{\boldsymbol{w}}\|_1 \le \sum_{k=1}^{\ell} \|\boldsymbol{\varepsilon}_k\|_1. \tag{9}$$

Proof. We have

$$\begin{aligned} e^{\alpha t}\widetilde{\boldsymbol{w}} &= \widetilde{\boldsymbol{f}}_0 + \sum_{k=1}^{\ell} \frac{(\alpha t)^k}{k!} \widetilde{\boldsymbol{f}}_k \\ &= \boldsymbol{v} + \sum_{k=1}^{\ell} \frac{(\alpha t)^k}{k!} (\boldsymbol{P}\widetilde{\boldsymbol{f}}_{k-1} - \boldsymbol{\varepsilon}_k) \\ &= \boldsymbol{v} + \sum_{k=1}^{\ell} \frac{(\alpha t)^k}{k!} (\boldsymbol{P}^k \boldsymbol{v} - \boldsymbol{P}^{k-1}\boldsymbol{\varepsilon}_1 - \boldsymbol{P}^{k-2}\boldsymbol{\varepsilon}_2 - \cdots - \boldsymbol{\varepsilon}_k). \end{aligned}$$

Hence (8) follows immediately. And, using the fact that $\|\boldsymbol{P}\|_1 = 1$, we obtain

$$\|\overline{\boldsymbol{w}} - \widetilde{\boldsymbol{w}}\|_1 \le \sum_{k=1}^{\ell} e^{-\alpha t} \frac{(\alpha t)^k}{k!} \sum_{j=1}^{k} \|\boldsymbol{\varepsilon}_j\|_1 \le \left(\sum_{k=1}^{\ell} e^{-\alpha t} \frac{(\alpha t)^k}{k!} \right) \sum_{k=1}^{\ell} \|\boldsymbol{\varepsilon}_k\|_1 \le \sum_{k=1}^{\ell} \|\boldsymbol{\varepsilon}_k\|_1. \qquad \square$$

COROLLARY 3.2. *With the same notation as in Theoreom* 3.1, *the accumulated error due to the inexact matrix-vector products satisfies*

$$\boldsymbol{f}_k - \widetilde{\boldsymbol{f}}_k = -\boldsymbol{\mathcal{E}}_k \boldsymbol{v} = -\sum_{j=1}^{k} \boldsymbol{P}^{k-j} \boldsymbol{\varepsilon}_j$$

and therefore

$$\|\boldsymbol{f}_k - \widetilde{\boldsymbol{f}}_k\|_1 = \|\boldsymbol{\mathcal{E}}_k \boldsymbol{v}\|_1 \leq \sum_{j=1}^{k} \|\boldsymbol{\varepsilon}_j\|_1.$$

Proof. The result can be inferred from (6) and (8) and the uniqueness of the Taylor expansion. Another way is to expand $\boldsymbol{f}_k - \widetilde{\boldsymbol{f}}_k = \boldsymbol{P}(\boldsymbol{f}_{k-1} - \widetilde{\boldsymbol{f}}_{k-1}) - \boldsymbol{\varepsilon}_k$. ∎

The above analysis shows that the local errors in the matrix-vector products provide a useful mechanism for estimating the accumulated errors. And more significantly, (9) shows that the global error grows only linearly with the local errors, which implies that the algorithm is robust. When the domain is split as shown in (5), we do not have $\widetilde{\boldsymbol{f}}_0 = \boldsymbol{v}$ anymore, but the analysis can be easily extended to this case by setting $\widetilde{\boldsymbol{f}}_0 = \boldsymbol{v} + \boldsymbol{\varepsilon}_0$, with $\boldsymbol{v}$ understood as the current operand and $\boldsymbol{\varepsilon}_0$ as the error so far. We now discuss some ways of relaxing the matrix-vector product while controlling the local error.

3.2. Monitoring the inexact matrix-vector product. For very large problems, the probability sum condition $\|\boldsymbol{f}\|_1 = \mathbb{1}^T \boldsymbol{f} = 1$ implies that some of the components of $\boldsymbol{f}$ must necessarily be zero or at least very small. In other words, the most significant part of the support of the distribution at a given time is limited to a few components corresponding to the subset of most likely reachable states at that time. This subset changes over time, and hence the most significant part of the support changes over time too, but the size of the current subset remains small compared to the size of the full state space. While this may not always be the case, for example, if the distribution is uniform, there is a large class of important problems that exhibit this property, reflecting the fact that not all states are reached instantaneously. This observation is the cornerstone of our strategy for reducing the cost of the matrix-vector product. Let

$$\mathrm{Supp}(\boldsymbol{f}) = \{1 \leq j \leq n \mid f_j > 0\};$$

we can write the matrix-vector product in a column-oriented manner as

$$\boldsymbol{P}\boldsymbol{f} = \Big(\boldsymbol{I} + \frac{1}{\alpha}\boldsymbol{A}\Big)\boldsymbol{f} = \boldsymbol{f} + \frac{1}{\alpha}\begin{bmatrix}\boldsymbol{a}_1 & \boldsymbol{a}_2 & \dots & \boldsymbol{a}_n\end{bmatrix}\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} = \boldsymbol{f} + \sum_{j \in \mathrm{Supp}(\boldsymbol{f})} \frac{f_j}{\alpha}\boldsymbol{a}_j.$$

The key to efficiency resides therefore in tracking only a suitable approximation of this support. To this end, define the ε-support of $\boldsymbol{f}$ as

$$\mathrm{Supp}_\varepsilon(\boldsymbol{f}) = \{1 \leq j \leq n \mid f_j > \varepsilon\},$$

and let $\mathrm{Supp}_\varepsilon^C(\boldsymbol{f})$ be its complement, that is,

$$\mathrm{Supp}(\boldsymbol{f}) = \mathrm{Supp}_\varepsilon(\boldsymbol{f}) \cup \mathrm{Supp}_\varepsilon^C(\boldsymbol{f}).$$

We now use

$$\boldsymbol{P}\boldsymbol{f} \approx \boldsymbol{f} + \sum_{j \in \operatorname{Supp}_\varepsilon(\boldsymbol{f})} \frac{f_j}{\alpha} \boldsymbol{a}_j.$$

Since the essential part of the support is captured in this way, the inexact matrix-vector product may be quite a good approximation. Furthermore, the larger the problem size, the fewer (comparatively) the number of components of significance in the support, making the approach even more effective. The corresponding local error vector is therefore

$$\boldsymbol{\varepsilon} = \sum_{j \notin \operatorname{Supp}_\varepsilon(\boldsymbol{f})} \frac{f_j}{\alpha} \boldsymbol{a}_j = \sum_{j \in \operatorname{Supp}_\varepsilon^C(\boldsymbol{f})} \frac{f_j}{\alpha} \boldsymbol{a}_j.$$

Noting that $\|\boldsymbol{a}_j\|_1 = 2|a_{jj}|$ and $\|\boldsymbol{a}_j\|_\infty = |a_{jj}|$ since $a_{jj} = -\sum_{i=1, i\neq j}^n a_{ij}$, we have

$$\|\boldsymbol{\varepsilon}\|_1 \leq \sum_{j \in \operatorname{Supp}_\varepsilon^C(\boldsymbol{f})} \frac{2\varepsilon}{\alpha} |a_{jj}|, \qquad \|\boldsymbol{\varepsilon}\|_\infty \leq \sum_{j \in \operatorname{Supp}_\varepsilon^C(\boldsymbol{f})} \frac{\varepsilon}{\alpha} |a_{jj}|,$$

and since $\alpha \geq |a_{jj}|$, we get

$$\|\boldsymbol{\varepsilon}\|_1 \leq 2\varepsilon(|\operatorname{Supp}(\boldsymbol{f})| - |\operatorname{Supp}_\varepsilon(\boldsymbol{f})|), \qquad \|\boldsymbol{\varepsilon}\|_\infty \leq \varepsilon(|\operatorname{Supp}(\boldsymbol{f})| - |\operatorname{Supp}_\varepsilon(\boldsymbol{f})|),$$

where $|S|$ denotes the cardinality of the set S. Note that $|\operatorname{Supp}(\boldsymbol{f})| - |\operatorname{Supp}_\varepsilon(\boldsymbol{f})|$ is anticipated to be less than n, although we shall later use n as an upper bound for simplicity. Combining this analysis with Theorem 3.1, we obtain the following result.

THEOREM 3.3. *For $k = 1, \ldots, \ell$ let $\widetilde{\boldsymbol{f}}_k$ be the approximate vector computed by the inexact matrix-vector product at step k using the ε-support criteria. Then the global error of the inexact uniformization method satisfies*

$$\|\overline{\boldsymbol{w}} - \widetilde{\boldsymbol{w}}\|_1 \leq 2\varepsilon \sum_{k=1}^{\ell} (|\operatorname{Supp}(\widetilde{\boldsymbol{f}}_k)| - |\operatorname{Supp}_\varepsilon(\widetilde{\boldsymbol{f}}_k)|) \leq 2\varepsilon \sum_{k=1}^{\ell} (n - |\operatorname{Supp}_\varepsilon(\widetilde{\boldsymbol{f}}_k)|) \leq 2n\ell\varepsilon.$$

Remark 3.4. Our description has focused on applying a threshold on the support of a probability vector $\boldsymbol{f}$. It has the severity of not weighting the components with the actual matrix. Indeed since $|a_{jj}|/\alpha \leq 1$, it is possible to have an $f_j > \varepsilon$, and yet $f_j|a_{jj}|/\alpha \leq \varepsilon$, in which case we may ignore the column $\boldsymbol{a}_j$ as well. Thus we can trade the conciseness of the earlier formulation for a little extra saving in the matrix-vector product. In fact, in our experiments, our column-oriented matrix-vector product $\boldsymbol{A}\boldsymbol{f}/\alpha$ accumulates columns only where $f_j|a_{jj}|/\alpha > \varepsilon$. Thus our inexact matrix-vector product is effectively

$$\boldsymbol{P}\boldsymbol{f} \approx \boldsymbol{f} + \sum_{j \text{ s.t. } f_j \frac{|a_{jj}|}{\alpha} > \varepsilon} \frac{f_j}{\alpha} \boldsymbol{a}_j, \tag{10}$$

for which the corresponding local error vector is

$$\boldsymbol{\varepsilon} = \sum_{j \text{ s.t. } f_j \frac{|a_{jj}|}{\alpha} \leq \varepsilon} \frac{f_j}{\alpha} \boldsymbol{a}_j \leq \varepsilon \left| \left\{ j \text{ s.t. } f_j \frac{|a_{jj}|}{\alpha} \leq \varepsilon \right\} \right|.$$

We shall refer to this scheme as the *scaled* ε-support.

Remark 3.5. Another way to identify the most relevant part of the support is to determine the components of $\boldsymbol{f}$ that quickly sum to one, by, for example, sorting $\boldsymbol{f}$ in decreasing order and selecting the first few probabilities that quickly sum to one. However, for large problems, the sort, in $O(n \log n)$ operations, is potentially more expensive than the matrix-vector product, in $O(nz)$ operations, where nz is the number of nonzero entries of the matrix. A partial, heuristic sort is possible, but this is more involved.

Remark 3.6. Sparse matrices are usually stored using compact storage formats, of which a great variety exist [1]. Since our approach is column-oriented, it is crucial to use a format that fits the approach; otherwise the overhead of looking up and accessing the relevant columns of the matrix can outweigh any benefit of the relaxation algorithm. We recommend therefore the compressed column storage (CCS) format (cf. [1]), as it is ideally suited to our approach. Unfortunately, if the algorithm is implemented in MATLAB, which has its native sparse data type [5], referencing the relevant columns at each matrix-vector product hampers the vectorization and introduces a severe overhead. We observed that when performing the matrix-vector product $\boldsymbol{f} \leftarrow \boldsymbol{f} + \boldsymbol{A}\boldsymbol{f}/\alpha$, with $\boldsymbol{A}$ stored as a native sparse matrix in MATLAB, it is actually more efficient to zero the nonrelevant components of $\boldsymbol{f}$ than to index the corresponding columns of $\boldsymbol{A}$. That is, letting `diagA = abs(diag(A))`, it is more efficient in MATLAB to execute

```
support = find(diagA.*f > alpha*tol);
g(1:n) = 0;  g(support) = (1/alpha)*f(support);
f = f + A*g;
```

rather than

```
support = find(diagA.*f > alpha*tol);
f = f + A(:,support)*((1/alpha)*f(support));
```

Thus the advantage of the inexact matrix-vector product is lost in this environment. The ideal is of course to completely avoid arithmetic operations involving zeros or the negligible part of the support. The reader wishing to experiment with our approach in MATLAB should bear in mind these issues. Our numerical experiments were conducted using the CCS format in FORTRAN, which offers a finer granularity on controlling the operations without an excessive tradeoff. We anticipate that such a procedural language is the most appropriate for very large problems.

3.3. Detecting the stationary probability distribution. A particular feature of a Markov chain is that its transition matrix $\boldsymbol{A}$ has 0 as a simple eigenvalue, and for any other nonzero eigenvalue λ of $\boldsymbol{A}$, $\Re e(\lambda) < 0$. Hence the differential system (1) is stable and has a steady-state solution. It follows from this property that if $\tilde{\boldsymbol{w}}$ denotes the eigenvector associated with the eigenvalue 0, that is, $\boldsymbol{A}\tilde{\boldsymbol{w}} = 0$, then we have

$$\lim_{t \to \infty} e^{t\boldsymbol{A}} \boldsymbol{v} = \tilde{\boldsymbol{w}}.$$

The vector $\tilde{\boldsymbol{w}}$ is known as the *stationary* probability vector. When integrating therefore over a very long time t, it is possible for the uniformization method to tend to $\tilde{\boldsymbol{w}}$. However, implementations often shy away from attempting to detect this situation not only because it is not obvious how to do so cheaply and reliably, but also because of the unwarranted overhead of testing for an event that is more often than not unlikely to happen. However, we observed that when the stationary distribution

is reached (to within the error tolerance), a vast amount of computation is wasted unnecessarily. We can afford to check this in our approach. Indeed if in the course of our algorithm the stationary probability distribution is reached, then (10) implies that

$$\boldsymbol{P}\boldsymbol{f} \approx \boldsymbol{f}.$$

Detecting this situation enables further computational savings by avoiding the addition of unnecessary terms in the series. We implemented this check only in between the steps of (5). In other words, there are at most $m-1$ checkpoints in Algorithm 2, with the steady solution assumed if $\|\boldsymbol{P}\boldsymbol{f} - \boldsymbol{f}\|_\infty \leq \varepsilon_{\text{tol}}$, where $\boldsymbol{P}\boldsymbol{f}$ is itself approximated with the inexact scheme.

4. Numerical experiments. Experiments were undertaken at the National Facility of the Australian Partnership for Advanced Computing (APAC). We used its SGI Altix supercomputer infrastructure, but in a single node execution (Intel Itanium2 with 1.6 GHz). We implemented the code in FORTRAN using double precision (machine precision $= 0.2\,10^{-15}$), and used the Intel FORTRAN compiler (ifort) with a compilation switch to request its highest level of optimization (`-O3`). For comparison, we include the exact uniformization where we added the provision of detecting the steady solution as described in section 3.3. We also include the Krylov code (DGEXPV) from the EXPOKIT package [14], which implements Arnoldi's full orthogonalization method (FOM), where we set 30 as the dimension of the Krylov basis everywhere. This is labeled as Krylov(30) in the tables. The same ε_{tol} is used for the error control criteria. Refer to [14] for more details.

We compute the transient probability vector, $\boldsymbol{w}(t) = \exp(t\boldsymbol{A})\boldsymbol{v}$, for $t = 1,\ 10,\ 100$ over nine different data sets with $\boldsymbol{v} = (1, 0, \ldots, 0)^T$ of appropriate length. The choice of e_1 as the starting vector may seem too special, but it reflects the fact that we order the state space by reachability and that the Markov chain starts in the first state before evolving into other states. We repeated some of the calculations with a random starting vector. In general the computation time for the inexact uniformization increased from the case where e_1 is the starting vector, but it was still notably faster than the exact uniformization, although the differences in performance were less marked the more stringent the error tolerance became.

Recall from (7) that, in order to achieve a desired accuracy on the final result, we should set a comparable (or higher) accuracy for the inexact matrix-vector product, leading evidently to the exact uniformization if the matrix-vector product is computed to full accuracy. In the experiments, we set the same ε_{tol} value to control the inexact matrix-vector product (10). We run the experiments with the accuracy tolerance parameter ranging from $\varepsilon_{\text{tol}} = 10^{-5}$ to $\varepsilon_{\text{tol}} = 10^{-10}$. Note that the graphical charts use a logarithmic scale.

4.1. Mutual exclusion (MUTEX) problem. Mutual exclusion is a situation commonly encountered in computer systems. In this model, N distinguishable processes compete for a shared resource. Each of these processes alternates between a *sleeping* state and a resource *using* state. However, the number of processes that may concurrently use the resource is limited to P, where $1 \leq P \leq N$, so that when a process wishing to move from the sleeping state to the resource using state finds P processes already using the resource, that process fails to access the resource and returns to the sleeping state. Notice that when $P = 1$ this model reduces to the usual mutual exclusion problem. When $P = N$ all of the processes are independent. Let

$\lambda^{(i)}$ be the rate at which process i awakes from the sleeping state wishing to access the resource, and let $\mu^{(i)}$ be the rate at which this same process releases the resource when it has possession of it. In the experiments, we set the pair (N, P) to be $(16, 4)$, $(20, 8)$, and $(20, 16)$, with the rates chosen as $\lambda^{(i)} = 1/i$ and $\mu^{(i)} = i$, for $i = 1, \ldots, N$. The largest matrix in this series is of order $n = 1,047,225$ with $nz = 21,972,345$ nonzero elements, and $\|A\|_\infty = 232.65$. A modified version of the MARCA Markov chain modeling package [17] was used for the generation.

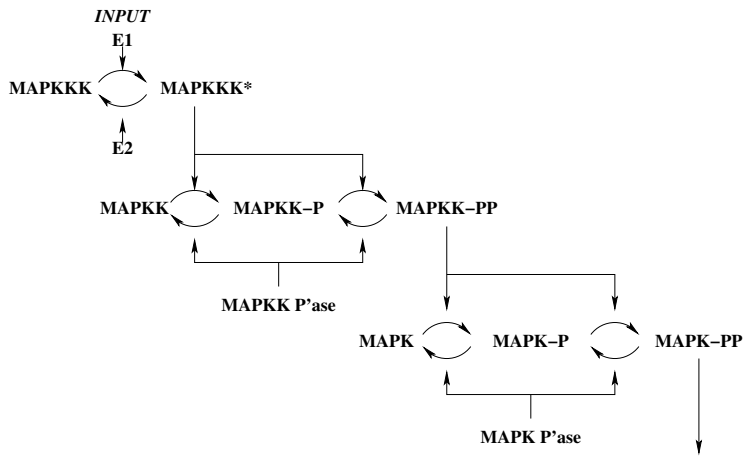
4.2. Nearly completely decomposable (NCD) queueing network. This model represents a multiuser interactive computer environment in which the system architecture is a time-shared, multiprogrammed, paged, virtual memory computer. The system consists of a set of N terminals, from which N users generate commands; a central processing unit (CPU); a secondary memory device (SM); and a filing device (FD). A queue of requests is associated with each device, and the scheduling is assumed to be FCFS (first come first served). When a command is generated, the user at the terminal remains inactive until the system responds. Symbolically, a user having generated a command enters the CPU queue. The behavior of the process in the system is characterized by a compute time followed either by a page fault, after which the process enters the SM queue, or an input/output (file request), in which case the process enters the FD queue. Processes that terminate their service at the SM or FD queue return to the CPU queue. Symbolically, completion of a command is represented by a departure of the process from the CPU to the terminals. This model has been often used in previous studies (see [12]). The matrices that are obtained are NCD, a factor that makes computation of stationary distribution by certain methods rather difficult. Our interest is in seeing how it affects methods for obtaining transient solutions. Parameter values were chosen to generate three matrices, the largest of which is of order $n = 91,881$ with $nz = 623,241$ nonzero elements (see Table 1).

4.3. Mitogen-activated protein kinase (MAPK) cascade. The mitogen-activated protein kinase (MAPK) cascade is implicated in a variety of signaling processes governing transitions in a cell's phenotype [9]. The MAPK cascade consists of 22 chemical species and 30 chemical reaction channels. A schematic of the cascade and the reactions, with their associated rate constants, is given in Figure 1. By simulating the dynamics of the MAPK cascade using the chemical master equation (CME), we obtain a Markov chain of very large dimension [4]. This example shows that it is possible to solve the CME for the MAPK cascade, at least for small numbers of molecules.

TABLE 1

Problem characteristics. The table shows n, the order of the transition rate matrix; its number of nonzero entries, $nz(\boldsymbol{A})$; the minimum and maximum number of nonzero entries on its columns, $\min nz(\boldsymbol{a}_j)$ and $\max nz(\boldsymbol{a}_j)$; and its L-$\infty$ norm (maximum row sum) $\|\boldsymbol{A}\|_\infty = \max_i \sum_{j=1}^{n} |a_{ij}|$. The MAPK and MUTEX problems are described in sections 4.3 *and* 4.1*, respectively, while the NCD problems are described in* 4.2.

	n	$nz(\boldsymbol{A})$	$\min nz(\boldsymbol{a}_j)$	$\max nz(\boldsymbol{a}_j)$	$\|.\|_\infty$
MAPK1	3,505	27,583	2	13	26.80
MAPK2	484,770	5,820,720	2	21	95.10
MAPK3	1,566,390	20,190,096	2	22	129.70
MUTEX1	2,517	20,949	5	17	154.44
MUTEX2	263,950	4,031,310	9	21	232.65
MUTEX3	1,047,225	21,972,345	17	21	232.65
NCD1	5,456	35,216	2	7	23.26
NCD2	12,341	81,221	2	7	35.51
NCD3	91,881	623,241	2	7	99.39



1	KKK + E1 $\xrightarrow{1.0}$ KKK.E1	16	KKPP + KKP'ase $\xrightarrow{1.0}$ KKPP.KKP'ase
2	KKK + E1 $\xleftarrow{1.0}$ KKK.E1	17	KKPP + KKP'ase $\xleftarrow{1.0}$ KKPP.KKP'ase
3	KKK.E1 $\xrightarrow{0.1}$ KKK* + E1	18	KKPP.KKP'ase $\xrightarrow{0.1}$ KKP + KKP'ase
4	KKK* + E2 $\xrightarrow{1.0}$ KKK.E2	19	KKPP + K $\xrightarrow{1.0}$ KKPP.K
5	KKK* + E2 $\xleftarrow{1.0}$ KKK.E2	20	KKPP + K $\xleftarrow{1.0}$ KKPP.K
6	KKK.E2 $\xrightarrow{0.1}$ KKK + E2	21	KKPP.K $\xrightarrow{0.1}$ KKPP + KP
7	KK + KKK* $\xrightarrow{1.0}$ KK.KKK*	22	KP + KP'ase $\xrightarrow{1.0}$ KP.KP'ase
8	KK + KKK* $\xleftarrow{1.0}$ KK.KKK*	23	KP + KP'ase $\xleftarrow{1.0}$ KP.KP'ase
9	KK.KKK* $\xrightarrow{0.1}$KKP+KKKx*	24	KP.KP'ase $\xrightarrow{0.1}$ K+KP'ase
10	KKP + KKP'ase $\xrightarrow{1.0}$ KKP.KKP'ase	25	KP + KKPP $\xrightarrow{1.0}$ KP.KKPP
11	KKP + KKP'ase $\xleftarrow{1.0}$ KKP.KKP'ase	26	KP + KKPP $\xleftarrow{1.0}$ KP.KKPP
12	KKP.KKP'ase $\xrightarrow{0.1}$ KK + KKP'ase	27	KP.KKPP $\xrightarrow{0.1}$ KPP + KKPP
13	KKP + KKK* $\xrightarrow{1.0}$ KKP.KKK*	28	KPP + KP'ase $\xrightarrow{1.0}$ KPP.KP'ase
14	KKP + KKK* $\xleftarrow{1.0}$ KKP.KKK*	29	KPP + KP'ase $\xleftarrow{1.0}$ KPP.KP'ase
15	KKP.KKK* $\xrightarrow{0.1}$ KKPP + KKK*	30	KPP.KP'ase $\xrightarrow{0.1}$ KP+KP'ase

Reactions and their rate constants

FIG. 1. *Schematic and detailed reactions in a mitogen-activated protein kinase (MAPK) cascade consisting of* 10 *coupled Michaelis–Menten enzyme kinetic models (adapted from* [9]*).*

In the largest experiment here, there can be anywhere from 0 to 5 molecules of each of the 22 species, so that an upper bound on the dimension of the full state space is 6^{22}. This reflects the so-called *curse of dimensionality*, which motivates the need for efficient solution techniques in systems biology. Since not all of these states are reachable in one step, the matrix is sparse and is generated based on the reachability from seven key species: the MAPK, MAPK kinase, MAPK kinase-kinase (KKK), and enzymes denoted E1, E2, MAPK phosphatase, and MAPKK phosphatase.

4.4. Discussion. The results of our extensive simulations are presented in the appendix. These detailed tables show how the results are affected by varying the choice of ε_{tol} (which determines the ε-support) from $\varepsilon_{\text{tol}} = 10^{-5}$ to $\varepsilon_{\text{tol}} = 10^{-10}$. The reported computation times (in seconds) over such a range give an indication of the extra work required per decimal place of accuracy. For each test problem, we present $w_1(t)$ and $w_n(t)$, the first and last components of $\boldsymbol{w}(t)$, as computed by the three different

methods. The results show that the inexact method produces results accurate to the specified tolerance.

In order to encapsulate the simulation results given in the appendix we present a number of figures. Figure 2 shows how the probability vector and the support evolve over time for each of the three classes of problems. Clearly the three problems have very different characteristics.

In Figure 3, we present timing results for the three classes of problems with $\varepsilon_{\text{tol}} = 10^{-7}$ and $\varepsilon_{\text{tol}} = 10^{-10}$. We see that the inexact uniformization (inexactUnif) outperforms the exact uniformization (exactUnif) in nearly all cases (except one where the exactUnif detects equilibrium early). Furthermore, for integration intervals that are not too large, the inexactUnif is very competitive with the Krylov approach. However, as the integration interval increases, the inexactUnif becomes less competitive. One of the reasons for this is that the Krylov method is very efficient in detecting equilibrium.

In Figure 4, we compare the numbers of uMATVEC between the three methods.

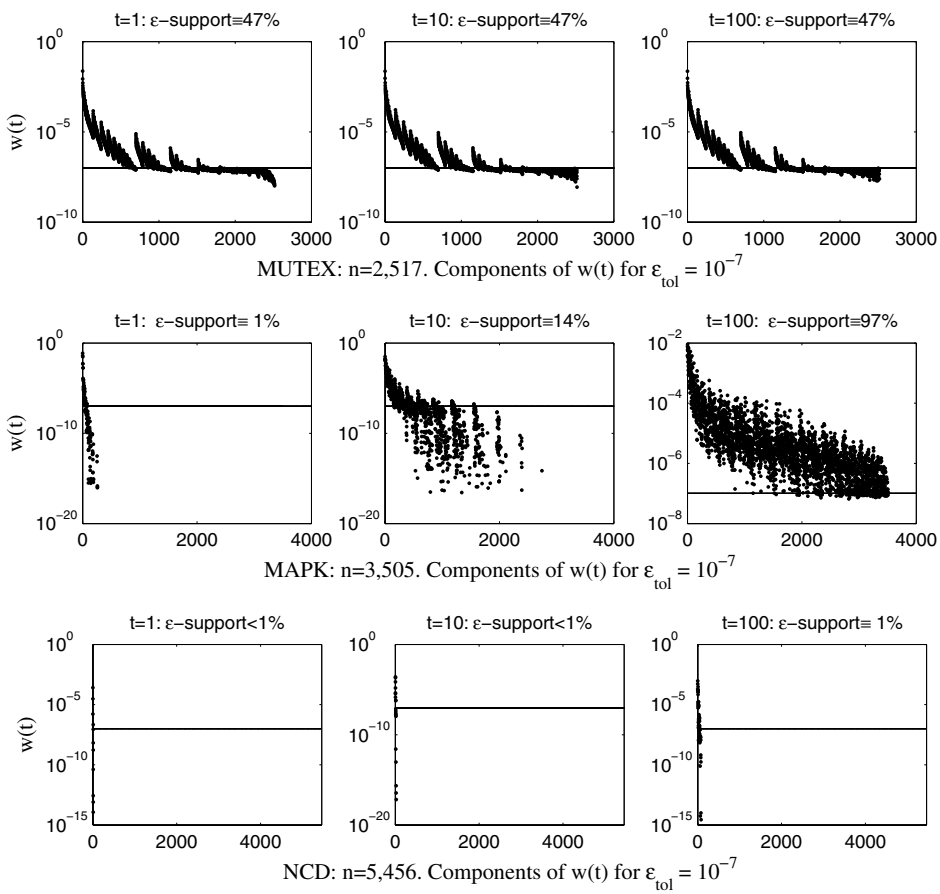


FIG. 2. *Evolution of the probability vector over time. The figure shows the* scaled *components as explained in Remark* 3.4. *Components above the straight line are those in the scaled ε-support, and thus they are those involved in the inexact matrix-vector product. The reported percentage is their proportion with respect to the number of components. It is seen that the ε-support changes over time, and various situations can happen depending on the matrix.*

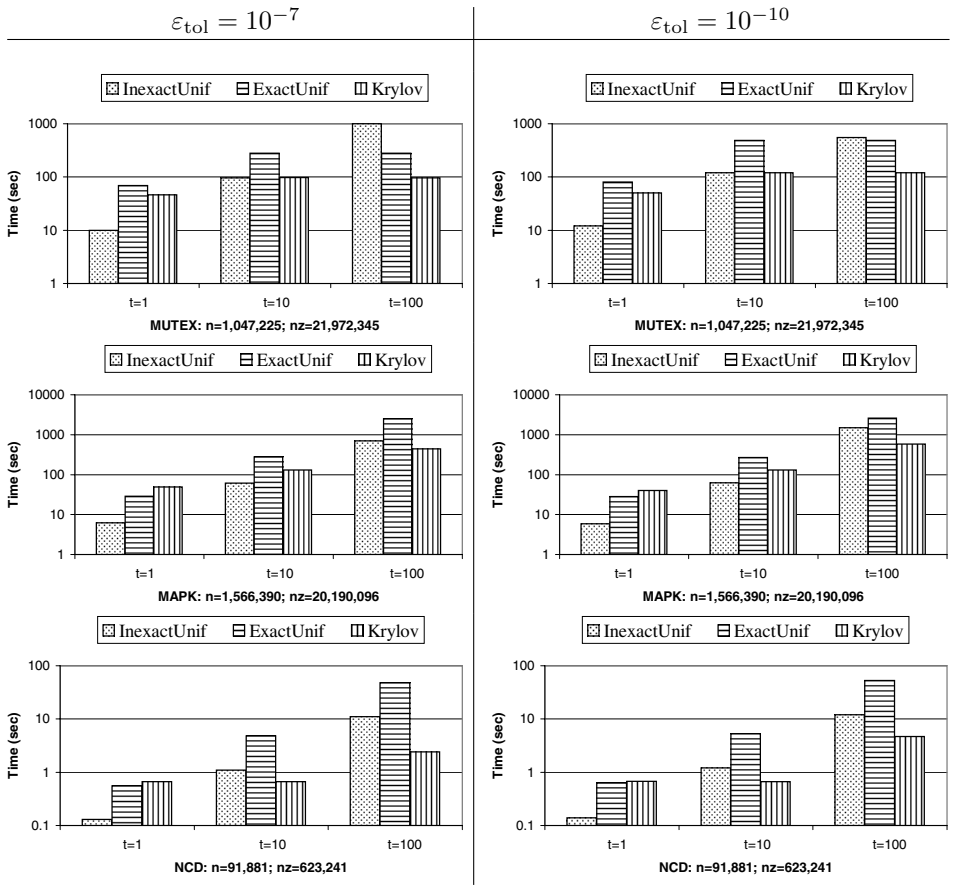


FIG. 3. *Timings for the largest matrices in the series when* $\varepsilon_{\text{tol}} = 10^{-7}$ *and* $\varepsilon_{\text{tol}} = 10^{-10}$. *At short integration intervals,* $t = 1$ *and* $t = 10$, *the times are all small in the NCD plots, but on the other problems it is seen that the inexactUnif is effective. Note in the MUTEX plots that the steady state occurs early at* $t = 10$. *The Krylov method is able to detect this almost immediately (due to an invariant subspace in its Arnoldi loop) and keeps a constant execution time as soon as* $t \geq 10$. *The exactUnif detects equilibrium later than the Krylov method but sooner than the inexact method.*

uMATVEC measures one unit of a matrix-vector product. This amounts to the standard matrix-vector product for the normal matrix-free methods, exactUnif and Krylov. However, in the case of inexactUnif, since not all columns participate in its matrix-vector operation, the number of uMATVEC is obtained by counting the number of participating columns over the entire run and dividing this by the order n.

In all cases, we see that there are substantially fewer uMATVEC operations with the inexactUnif technique. However, in this case, a single uMATVEC becomes increasingly more expensive as more and more conditional checks need to be processed. Indeed, this is one of the reasons why the inexactUnif becomes less competitive with the Krylov technique as the integration domain increases. It should be recalled that inexactUnif still uses almost the same number of iterations as exactUnif. Thus uMATVEC is a simple metric that gives a unified way of contrasting the methods in terms of matrix-vector products while giving a sense of the number of columns discarded by inexactUnif. Since it does not capture the fact that we still have to loop over the entire matrix using conditional checks to decide whether to skip over columns

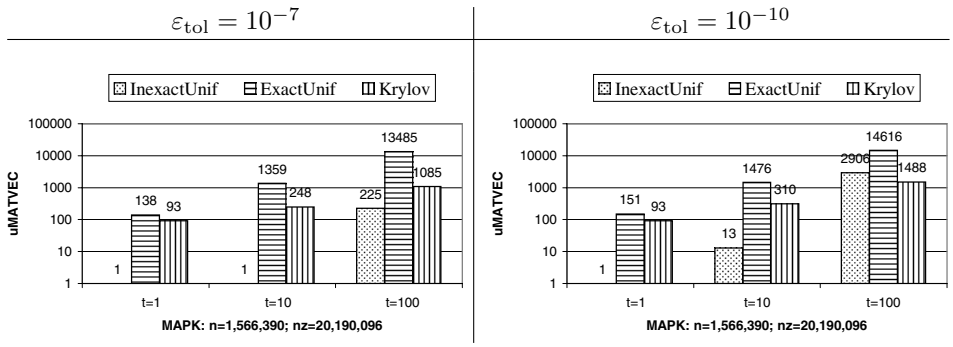


FIG. 4. *Numbers of uMATVEC operations for the largest MAPK matrix when* $\varepsilon_{\text{tol}} = 10^{-7}$ *and* $\varepsilon_{\text{tol}} = 10^{-10}$. *uMATVEC measures one unit of a matrix-vector product. This amounts to the standard matrix-vector product for the normal matrix-free methods, exactUnif and Krylov. However, in the case of inexactUnif, since not all columns participate in the matrix-vector operation, the reported number of uMATVEC operations is obtained by summing the number of participating columns over the entire run and dividing this by the order* n.

TABLE 2

*Distribution of the numbers of nonzeros in the columns. For example, the MAPK*1 *matrix has one column with two nonzero elements,* 11 *columns with three nonzero elements, and so forth. The location of the nonzero elements in these columns is not relevant.*

$nz(\boldsymbol{a}_j)$	Numbers of columns that have the indicated numbers of nonzeros, $nz(\boldsymbol{a}_j)$								
	MAPK1	MAPK2	MAPK3	MUTEX1	MUTEX2	MUTEX3	NCD1	NCD2	NCD3
2	1	1	1	0	0	0	3	3	3
3	11	17	17	0	0	0	87	117	237
4	61	140	151	0	0	0	407	742	3,082
5	179	668	880	1,820	0	0	87	117	237
6	426	2,617	3,708	0	0	0	1,218	2,223	9,243
7	712	7,664	12,162	0	0	0	3,654	9,139	79,079
8	866	18,460	32,120	0	0	0			
9	717	35,511	69,410	0	125,970	0			
10	386	57,179	125,404	0	0	0			
11	125	75,710	189,459	0	0	0			
12	20	84,437	242,549	0	0	0			
13	1	77,521	261,368	0	0	0			
14		59,439	238,129	0	0	0			
15		36,983	181,494	0	0	0			
16		18,630	115,291	0	0	0			
17		7,234	59,765	697	0	4,845			
18		2,100	24,713		0	0			
19		410	7,800		0	0			
20		47	1,724		0	0			
21		2	235		137,980	1,042,380			
22			10						

(together with scaling the components as indicated before), the reader should look at the value of inexactUnif in conjunction with that of exactUnif, keeping in mind the increasing number of conditional checks for matrices of large dimension over thousands of steps. This explains why the execution time of inexactUnif, seen in Figure 3, is not directly proportionate to its number of uMATVEC operations. Rather, the pay-off of the inexact matrix-vector routine is worthwhile if the discarded columns have sufficient nonzero elements so that the savings in floating point operations exceeds the conditional checks. Refer to Table 2 for the distribution of the numbers of nonzeros in the columns.

We can draw some other conclusions based on our extensive numerical tests.

- The inexactUnif is a worthy challenger not only to the exactUnif but also to the Krylov method in a number of cases. In general the inexactUnif method brings about a 10-fold improvement over the uniformization method on problems of significant dimension and a wide range of the accuracy tolerance parameter ε_{tol}, to the point of even becoming faster than the Krylov method at short integration domains, $t = 1$ and $t = 10$.
- Only when $t = 100$ does the Krylov method become the fastest due to its inherent higher convergence rate [8]. This comes at the price of the extra storage for the Krylov basis (which may not be possible on all computer systems for problems this large).
- For the MUTEX problems the steady state occurs early at $t = 10$. Since the Krylov method is able to detect this almost immediately (due to an invariant subspace in its Arnoldi loop), this explains why its uMATVEC and execution time are constant as soon as $t \geq 10$. The exactUnif detects equilibrium later than the Krylov method, but sooner than the inexact method. Hence the inexactness tends to delay the detection of the steady state. This can be clearly seen on the MUTEX plots in Figure 3. The inexact method took many more iterations, while the other methods finished early, thus keeping their constant execution time.
- The inexact method is able to track the ε-support and exclude impressive numbers of columns, making the method worthwhile and very efficient on problems with a large number of nonzero elements per column. Understandably, it is disadvantaged on problems with very few nonzero elements per column (such as a bidiagonal or tridiagonal matrix, for example). Determining whether to avoid a column may not be worth the effort on a nearly empty column (and testing this for all columns of a large matrix over thousands of steps would incur a substantial overhead too).

5. Conclusion. We have presented a new approach to substantially reducing the cost of the uniformization method. Our new algorithm consists of relaxing the matrix-vector products so as to perform them only to sufficient accuracy. Our approach is of practical value as it tells precisely how this relaxation can be performed, with the added benefit of retaining the minimalism and malleability that make the method so popular with Markovian analysts. We characterized the ensuing global error and showed that it grows at most linearly with the local errors introduced by the inexact matrix-vector products. We discussed strategies for refining the accuracy while reducing the execution cost. It has been observed that using our technique in an environment such as MATLAB does not reap the full benefits because of the overhead of operating on its native sparse data structure from the scripting level. However, using the appropriate sparse storage format in FORTRAN, numerical experiments showed that significant computational savings can be achieved in practical applications without a marked deterioration of accuracy.

Appendix A. MAPK results. Detailed results with the accuracy tolerance parameter ranging from $\varepsilon_{\text{tol}} = 10^{-5}$ to $\varepsilon_{\text{tol}} = 10^{-10}$, and the first and last components of $\boldsymbol{w}(t)$ as computed by the different methods at various times. See section 4.4 for how to interpret the results.

A.1. MAPK1: $n = 3,505$; $nz = 27,583$; $||.||_\infty = 26.80$.

	$\varepsilon_{\text{tol}} = 10^{-5}$		$\varepsilon_{\text{tol}} = 10^{-6}$		$\varepsilon_{\text{tol}} = 10^{-7}$		$\varepsilon_{\text{tol}} = 10^{-8}$		$\varepsilon_{\text{tol}} = 10^{-9}$		$\varepsilon_{\text{tol}} = 10^{-10}$	
	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time
$t = 1$												
inexactUnif	1	2.0E−03	1	2.0E−03	1	2.0E−03	1	2.9E−03	2	2.9E−03	3	2.9E−03
exactUnif	36	8.8E−03	38	9.8E−03	41	8.8E−03	43	9.8E−03	45	9.8E−03	47	1.1E−02
Krylov(30)	62	2.1E−02	62	2.0E−02	62	2.0E−02	62	2.0E−02	62	2.0E−02	62	1.9E−02
$t = 10$												
inexactUnif	6	1.4E−02	12	1.5E−02	23	1.9E−02	46	2.3E−02	82	3.2E−02	126	4.1E−02
exactUnif	242	5.7E−02	252	6.0E−02	262	6.2E−02	270	6.3E−02	278	6.5E−02	286	6.6E−02
Krylov(30)	124	3.9E−02	124	3.8E−02	124	3.8E−02	124	3.9E−02	155	4.8E−02	155	4.7E−02
$t = 100$												
inexactUnif	473	2.3E−01	1195	3.9E−01	1823	5.2E−01	2186	6.0E−01	2417	6.5E−01	2572	6.8E−01
exactUnif	1795	4.2E−01	2448	5.7E−01	2533	5.9E−01	2601	6.1E−01	2686	6.3E−01	2754	6.4E−01
Krylov(30)	310	9.8E−02	341	1.1E−01	372	1.2E−01	403	1.3E−01	434	1.4E−01	496	1.5E−01

	$w_1(t)$			$w_n(t)$		
	$t = 1$	$t = 10$	$t = 100$	$t = 1$	$t = 10$	$t = 100$
$\varepsilon_{\text{tol}} = 10^{-7}$						
inexactUnif	1.552574100E−01	6.088158784E−02	1.532050930E−02	0.000000000E+00	0.000000000E+00	2.690948138E−07
exactUnif	1.552574114E−01	6.088159971E−02	1.532115373E−02	1.996756526E−28	1.968854041E−13	1.495909999E−07
Krylov(30)	1.552574169E−01	6.088160594E−02	1.532117162E−02	2.154610103E−28	1.968857655E−13	1.495910210E−07
$\varepsilon_{\text{tol}} = 10^{-10}$						
inexactUnif	1.552574168E−01	6.088160591E−02	1.532117147E−02	0.000000000E+00	0.000000000E+00	1.495911835E−07
exactUnif	1.552574168E−01	6.088160593E−02	1.532117150E−02	2.150201066E−28	1.968857862E−13	1.495911800E−07
Krylov(30)	1.552574169E−01	6.088160594E−02	1.532117152E−02	2.154612807E−28	1.968857878E−13	1.495911803E−07

A.2. MAPK2: $n = 484,770$; $nz = 5,820,720$; $||.||_\infty = 95.10$.

	$\varepsilon_{\text{tol}} = 10^{-5}$		$\varepsilon_{\text{tol}} = 10^{-6}$		$\varepsilon_{\text{tol}} = 10^{-7}$		$\varepsilon_{\text{tol}} = 10^{-8}$		$\varepsilon_{\text{tol}} = 10^{-9}$		$\varepsilon_{\text{tol}} = 10^{-10}$	
	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time
$t = 1$												
inexactUnif	1	1.2E+00	1	1.3E+00	1	1.4E+00	1	1.4E+00	1	1.4E+00	1	1.6E+00
exactUnif	101	5.5E+00	105	5.7E+00	110	6.2E+00	114	6.2E+00	117	6.4E+00	121	7.0E+00
Krylov(30)	62	8.2E+00	93	1.2E+01	93	1.3E+01	93	1.2E+01	93	1.2E+01	93	1.3E+01
$t = 10$												
inexactUnif	1	1.1E+01	1	1.2E+01	2	1.3E+01	5	1.3E+01	11	1.5E+01	21	1.6E+01
exactUnif	945	5.2E+01	980	5.3E+01	1015	5.7E+01	1050	6.2E+01	1078	6.5E+01	1106	6.4E+01
Krylov(30)	186	2.4E+01	217	2.9E+01	217	2.8E+01	248	3.5E+01	248	3.4E+01	279	4.0E+01
$t = 100$												
inexactUnif	5	4.0E+01	119	1.4E+02	400	1.7E+02	1079	2.7E+02	2340	3.7E+02	4127	4.5E+02
exactUnif	3169	1.7E+02	7153	3.9E+02	10010	5.5E+02	10335	6.1E+02	10660	6.6E+02	10920	6.3E+02
Krylov(30)	682	8.8E+01	775	1.0E+02	899	1.1E+02	992	1.3E+02	1085	1.5E+02	1178	1.7E+02

	$w_1(t)$			$w_n(t)$		
	$t = 1$	$t = 10$	$t = 100$	$t = 1$	$t = 10$	$t = 100$
$\varepsilon_{\text{tol}} = 10^{-7}$						
inexactUnif	4.488055377E−03	6.724395696E−04	5.309184144E−05	0.000000000E+00	0.000000000E+00	0.000000000E+00
exactUnif	4.488055750E−03	6.724409029E−04	5.365710324E−05	5.578122275E−49	5.323320342E−23	1.485858527E−13
Krylov(30)	4.488055970E−03	6.724412733E−04	5.365741171E−05	1.195373611E−51	5.293177404E−23	1.485866141E−13
$\varepsilon_{\text{tol}} = 10^{-10}$						
inexactUnif	4.488055970E−03	6.724412698E−04	5.365621479E−05	0.000000000E+00	0.000000000E+00	0.000000000E+00
exactUnif	4.488055970E−03	6.724412730E−04	5.365740680E−05	6.017255413E−49	5.323334687E−23	1.485867037E−13
Krylov(30)	4.488055970E−03	6.724412734E−04	5.365740705E−05	2.005569704E−49	5.322076447E−23	1.485867036E−13

A.3. MAPK3: $n = 1,566,390$; $nz = 20,190,096$; $||.||_\infty = 129.70$.

	$\varepsilon_{\text{tol}} = 10^{-5}$		$\varepsilon_{\text{tol}} = 10^{-6}$		$\varepsilon_{\text{tol}} = 10^{-7}$		$\varepsilon_{\text{tol}} = 10^{-8}$		$\varepsilon_{\text{tol}} = 10^{-9}$		$\varepsilon_{\text{tol}} = 10^{-10}$	
	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time
$t = 1$												
inexactUnif	1	5.8E+00	1	5.6E+00	1	6.2E+00	1	5.5E+00	1	5.3E+00	1	5.9E+00
exactUnif	128	2.6E+01	134	2.6E+01	138	2.9E+01	143	2.6E+01	147	2.5E+01	151	2.8E+01
Krylov(30)	93	4.9E+01	93	4.8E+01	93	4.9E+01	93	3.9E+01	93	3.6E+01	93	4.0E+01
$t = 10$												
inexactUnif	1	5.7E+01	1	5.9E+01	1	6.1E+01	3	5.4E+01	6	5.7E+01	13	6.2E+01
exactUnif	1260	2.7E+02	1305	2.6E+02	1359	2.8E+02	1395	2.5E+02	1440	2.6E+02	1476	2.7E+02
Krylov(30)	217	1.2E+02	248	1.3E+02	248	1.3E+02	279	1.2E+02	310	1.3E+02	310	1.3E+02
$t = 100$												
inexactUnif	1	1.5E+02	40	4.9E+02	225	7.0E+02	642	8.1E+02	1486	1.1E+03	2906	1.5E+03
exactUnif	3169	6.2E+02	7951	1.5E+03	13485	2.5E+03	13920	2.5E+03	14268	2.6E+03	14616	2.6E+03
Krylov(30)	837	3.9E+02	930	4.8E+02	1085	4.4E+02	1240	5.0E+02	1395	5.9E+02	1488	5.8E+02

	$w_1(t)$			$w_n(t)$		
	$t = 1$	$t = 10$	$t = 100$	$t = 1$	$t = 10$	$t = 100$
$\varepsilon_{\text{tol}} = 10^{-7}$						
inexactUnif	5.603577847E−04	5.356397548E−05	3.056917426E−06	0.000000000E+00	0.000000000E+00	0.000000000E+00
exactUnif	5.603579078E−04	5.356416637E−05	3.115903889E−06	2.870698209E−55	3.641162019E−26	1.912229657E−15
Krylov(30)	5.603579480E−04	5.356419530E−05	3.115922178E−06	3.717505224E−56	3.582836703E−26	1.912244107E−15
$\varepsilon_{\text{tol}} = 10^{-10}$						
inexactUnif	5.603579480E−04	5.356419257E−05	3.115704160E−06	0.000000000E+00	0.000000000E+00	0.000000000E+00
exactUnif	5.603579480E−04	5.356419305E−05	3.115922477E−06	3.096836990E−55	3.641169451E−26	1.912241195E−15
Krylov(30)	5.603579480E−04	5.356419311E−05	3.115922502E−06	1.672589706E−55	3.640413645E−26	1.912241197E−15

Appendix B. MUTEX results. Parameters as for Appendix A.

B.1. MUTEX1: $n = 2,517$; $nz = 20,949$; $||.||_\infty = 154.44$.

	$\varepsilon_{\text{tol}} = 10^{-5}$		$\varepsilon_{\text{tol}} = 10^{-6}$		$\varepsilon_{\text{tol}} = 10^{-7}$		$\varepsilon_{\text{tol}} = 10^{-8}$		$\varepsilon_{\text{tol}} = 10^{-9}$		$\varepsilon_{\text{tol}} = 10^{-10}$	
	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time
$t = 1$												
inexactUnif	9	5.9E−03	23	1.1E−02	50	1.4E−02	83	1.9E−02	107	2.1E−02	114	2.2E−02
exactUnif	98	1.7E−02	103	1.8E−02	107	1.9E−02	111	1.9E−02	115	2.0E−02	118	2.1E−02
Krylov(30)	93	2.2E−02	93	2.1E−02	93	2.2E−02	93	2.2E−02	93	2.1E−02	93	2.1E−02
$t = 10$												
inexactUnif	99	5.8E−02	245	9.1E−02	508	1.3E−01	463	9.9E−02	733	1.4E−01	927	1.8E−01
exactUnif	263	4.6E−02	412	7.0E−02	569	9.8E−02	585	1.0E−01	756	1.3E−01	931	1.6E−01
Krylov(30)	156	3.7E−02	156	3.6E−02	156	3.8E−02	186	4.3E−02	186	4.4E−02	187	4.4E−02
$t = 100$												
inexactUnif	994	5.8E−01	2438	8.9E−01	5004	1.3E+00	504	1.1E−01	632	1.2E−01	837	1.6E−01
exactUnif	287	5.1E−02	448	7.7E−02	463	7.9E−02	637	1.1E−01	653	1.1E−01	841	1.4E−01
Krylov(30)	156	3.7E−02	156	3.6E−02	156	3.6E−02	187	4.4E−02	187	4.4E−02	187	4.4E−02

	$w_1(t)$			$w_n(t)$		
	$t = 1$	$t = 10$	$t = 100$	$t = 1$	$t = 10$	$t = 100$
$\varepsilon_{\text{tol}} = 10^{-7}$						
inexactUnif	5.908162476E−01	5.759677948E−01	5.759646421E−01	1.025981203E−08	8.623696382E−09	2.452064837E−08
exactUnif	5.908914429E−01	5.760429646E−01	5.760438606E−01	1.813281322E−10	1.765543744E−10	1.765546611E−10
Krylov(30)	5.908914877E−01	5.760430075E−01	5.760423408E−01	1.813281459E−10	1.765543864E−10	1.765541820E−10
$\varepsilon_{\text{tol}} = 10^{-10}$						
inexactUnif	5.908914876E−01	5.760430259E−01	5.760430262E−01	1.813281459E−10	1.765543919E−10	1.765543920E−10
exactUnif	5.908914876E−01	5.760430259E−01	5.760430262E−01	1.813281459E−10	1.765543919E−10	1.765543920E−10
Krylov(30)	5.908914877E−01	5.760430251E−01	5.760429693E−01	1.813281459E−10	1.765543917E−10	1.765543746E−10

B.2. MUTEX2: $n = 263,950$; $nz = 4,031,310$; $||.||_\infty = 232.65$.

	$\varepsilon_{\text{tol}} = 10^{-5}$		$\varepsilon_{\text{tol}} = 10^{-6}$		$\varepsilon_{\text{tol}} = 10^{-7}$		$\varepsilon_{\text{tol}} = 10^{-8}$		$\varepsilon_{\text{tol}} = 10^{-9}$		$\varepsilon_{\text{tol}} = 10^{-10}$	
	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time
$t = 1$												
inexactUnif	1	1.1E+00	1	1.3E+00	1	1.5E+00	3	1.5E+00	7	1.8E+00	13	2.4E+00
exactUnif	216	7.1E+00	226	7.9E+00	236	9.1E+00	244	8.5E+00	252	8.8E+00	258	9.5E+00
Krylov(30)	93	4.9E+00	93	4.8E+00	124	7.0E+00	124	6.4E+00	124	6.5E+00	124	6.7E+00
$t = 10$												
inexactUnif	2	1.2E+01	6	1.2E+01	14	1.3E+01	33	1.5E+01	71	1.8E+01	138	2.4E+01
exactUnif	549	2.0E+01	716	2.5E+01	1037	3.6E+01	1217	4.3E+01	1414	5.0E+01	1772	6.4E+01
Krylov(30)	187	9.7E+00	218	1.1E+01	218	1.2E+01	218	1.1E+01	249	1.4E+01	249	1.3E+01
$t = 100$												
inexactUnif	19	1.2E+02	57	1.2E+02	146	1.4E+02	337	1.5E+02	711	1.9E+02	1386	2.3E+02
exactUnif	581	2.0E+01	751	2.9E+01	931	3.4E+01	1281	4.5E+01	1486	5.7E+01	1691	6.0E+01
Krylov(30)	187	9.7E+00	218	1.2E+01	218	1.1E+01	218	1.1E+01	249	1.4E+01	249	1.3E+01

	$w_1(t)$			$w_n(t)$		
	$t = 1$	$t = 10$	$t = 100$	$t = 1$	$t = 10$	$t = 100$
$\varepsilon_{\text{tol}} = 10^{-7}$						
inexactUnif	5.841439919E−01	5.683451353E−01	5.669922628E−01	0.000000000E+00	0.000000000E+00	0.000000000E+00
exactUnif	5.846449101E−01	5.699475028E−01	5.699496995E−01	8.689328648E−21	8.466573484E−21	8.466606732E−21
Krylov(30)	5.846449817E−01	5.699465427E−01	5.699465011E−01	1.057141134E−19	3.344706750E−18	3.344706506E−18
$\varepsilon_{\text{tol}} = 10^{-10}$						
inexactUnif	5.846430475E−01	5.699412545E−01	5.699381226E−01	0.000000000E+00	0.000000000E+00	0.000000000E+00
exactUnif	5.846449816E−01	5.699465444E−01	5.699465457E−01	8.689329709E−21	8.466558908E−21	8.466558928E−21
Krylov(30)	5.846449817E−01	5.699465222E−01	5.699457741E−01	8.705266675E−21	-2.082401649E−20	-2.082398916E−20

B.3. MUTEX3: $n = 1,047,225$; $nz = 21,972,345$; $||.||_\infty = 232.65$.

	$\varepsilon_{\text{tol}} = 10^{-5}$		$\varepsilon_{\text{tol}} = 10^{-6}$		$\varepsilon_{\text{tol}} = 10^{-7}$		$\varepsilon_{\text{tol}} = 10^{-8}$		$\varepsilon_{\text{tol}} = 10^{-9}$		$\varepsilon_{\text{tol}} = 10^{-10}$	
	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time
$t = 1$												
inexactUnif	1	9.3E+00	1	1.1E+01	1	1.0E+01	1	1.1E+01	2	1.1E+01	4	1.2E+01
exactUnif	333	6.4E+01	348	7.4E+01	360	6.9E+01	372	7.3E+01	384	7.5E+01	396	8.0E+01
Krylov(30)	93	3.5E+01	93	3.6E+01	124	4.6E+01	124	4.8E+01	124	4.8E+01	124	5.0E+01
$t = 10$												
inexactUnif	1	8.7E+01	2	9.1E+01	5	9.6E+01	11	1.0E+02	24	1.1E+02	48	1.2E+02
exactUnif	716	1.5E+02	1044	2.1E+02	1387	2.8E+02	1750	3.8E+02	2120	4.2E+02	2506	4.9E+02
Krylov(30)	218	8.5E+01	249	1.0E+02	249	9.8E+01	249	1.1E+02	280	1.1E+02	311	1.2E+02
$t = 100$												
inexactUnif	6	8.9E+02	18	9.4E+02	49	1.0E+03	114	1.1E+03	244	1.2E+03	224	5.5E+02
exactUnif	726	1.6E+02	1051	2.0E+02	1405	2.8E+02	1761	3.5E+02	2146	4.6E+02	2536	4.9E+02
Krylov(30)	218	8.6E+01	249	9.2E+01	249	9.6E+01	249	9.7E+01	280	1.2E+02	311	1.2E+02

	$w_1(t)$			$w_n(t)$		
	$t = 1$	$t = 10$	$t = 100$	$t = 1$	$t = 10$	$t = 100$
$\varepsilon_{\text{tol}} = 10^{-7}$						
inexactUnif	5.839384439E−01	5.676057995E−01	5.655566443E−01	0.000000000E+00	0.000000000E+00	0.000000000E+00
exactUnif	5.846448305E−01	5.699513949E−01	5.699507492E−01	3.215221919E−36	3.144217184E−36	3.144213576E−36
Krylov(30)	5.846449817E−01	5.699465374E−01	5.699462861E−01	8.454329836E−24	-1.173634626E−17	-1.173634108E−17
$\varepsilon_{\text{tol}} = 10^{-10}$						
inexactUnif	5.846420671E−01	5.699383602E−01	5.699350419E−01	0.000000000E+00	0.000000000E+00	0.000000000E+00
exactUnif	5.846449815E−01	5.699465481E−01	5.699465470E−01	3.215222751E−36	3.144190105E−36	3.144190099E−36
Krylov(30)	5.846449817E−01	5.699465393E−01	5.699463596E−01	1.026507866E−20	2.463669522E−21	2.463668745E−21

Appendix C. NCD results. Parameters as for Appendix A.

C.1. NCD1: $n = 5,456$; $nz = 35,216$; $||.||_\infty = 23.26$.

	$\varepsilon_{\text{tol}} = 10^{-5}$		$\varepsilon_{\text{tol}} = 10^{-6}$		$\varepsilon_{\text{tol}} = 10^{-7}$		$\varepsilon_{\text{tol}} = 10^{-8}$		$\varepsilon_{\text{tol}} = 10^{-9}$		$\varepsilon_{\text{tol}} = 10^{-10}$	
	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time
$t = 1$												
inexactUnif	1	2.0E−03	1	2.9E−03	1	2.9E−03	1	2.9E−03	1	2.9E−03	1	2.9E−03
exactUnif	29	8.8E−03	31	9.8E−03	33	1.1E−02	35	1.2E−02	37	1.3E−02	39	1.3E−02
Krylov(30)	62	2.9E−02	62	2.7E−02	62	2.7E−02	62	2.6E−02	62	2.5E−02	62	2.6E−02
$t = 10$												
inexactUnif	1	1.4E−02	1	1.5E−02	1	1.5E−02	1	1.5E−02	1	1.5E−02	1	1.8E−02
exactUnif	186	6.2E−02	196	6.4E−02	204	6.7E−02	212	7.0E−02	218	7.2E−02	226	7.4E−02
Krylov(30)	62	2.6E−02	62	2.6E−02	62	2.7E−02	62	2.6E−02	62	2.7E−02	62	2.6E−02
$t = 100$												
inexactUnif	3	1.3E−01	5	1.3E−01	8	1.4E−01	11	1.4E−01	15	1.5E−01	19	1.5E−01
exactUnif	1704	5.6E−01	1764	5.9E−01	1824	6.0E−01	1884	6.2E−01	1944	6.4E−01	1992	6.5E−01
Krylov(30)	124	5.4E−02	124	5.3E−02	124	5.4E−02	155	6.6E−02	155	6.7E−02	186	8.0E−02

	$w_1(t)$			$w_n(t)$		
	$t = 1$	$t = 10$	$t = 100$	$t = 1$	$t = 10$	$t = 100$
$\varepsilon_{\text{tol}} = 10^{-7}$						
inexactUnif	9.970072970E−01	9.706652267E−01	7.506102592E−01	0.000000000E+00	0.000000000E+00	0.000000000E+00
exactUnif	9.970072973E−01	9.706652300E−01	7.506102657E−01	0.000000000E+00	1.48178429E−146	9.62797897E−105
Krylov(30)	9.970073730E−01	9.706653663E−01	7.506110649E−01	1.27652477E−197	7.52708063E−167	4.377266630E−83
$\varepsilon_{\text{tol}} = 10^{-10}$						
inexactUnif	9.970073730E−01	9.706653662E−01	7.506110643E−01	0.000000000E+00	0.000000000E+00	0.000000000E+00
exactUnif	9.970073730E−01	9.706653662E−01	7.506110643E−01	0.000000000E+00	1.49205224E−146	9.62805426E−105
Krylov(30)	9.970073730E−01	9.706653663E−01	7.506110649E−01	1.38760040E−199	8.47997046E−170	3.164209892E−64

C.2. NCD2: $n = 12,341$; $nz = 81,221$; $||.||_\infty = 35.51$.

	$\varepsilon_{\text{tol}} = 10^{-5}$		$\varepsilon_{\text{tol}} = 10^{-6}$		$\varepsilon_{\text{tol}} = 10^{-7}$		$\varepsilon_{\text{tol}} = 10^{-8}$		$\varepsilon_{\text{tol}} = 10^{-9}$		$\varepsilon_{\text{tol}} = 10^{-10}$	
	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time
$t = 1$												
inexactUnif	1	6.8E−03	1	7.8E−03	1	7.8E−03	1	8.8E−03	1	8.8E−03	1	9.8E−03
exactUnif	38	2.8E−02	41	3.1E−02	44	3.4E−02	46	3.5E−02	48	3.7E−02	50	3.8E−02
Krylov(30)	62	6.7E−02	62	6.3E−02	62	6.3E−02	62	6.4E−02	62	6.3E−02	62	6.3E−02
$t = 10$												
inexactUnif	1	4.8E−02	1	4.9E−02	1	5.2E−02	1	5.3E−02	1	5.5E−02	1	5.7E−02
exactUnif	264	2.0E−01	274	2.1E−01	284	2.2E−01	292	2.2E−01	302	2.3E−01	310	2.4E−01
Krylov(30)	62	6.3E−02	62	6.3E−02	62	6.4E−02	62	6.4E−02	62	6.4E−02	62	6.4E−02
$t = 100$												
inexactUnif	2	4.7E−01	4	4.9E−01	6	5.0E−01	8	5.2E−01	11	5.4E−01	15	5.6E−01
exactUnif	2592	2.0E+00	2682	2.0E+00	2772	2.1E+00	2862	2.2E+00	2952	2.2E+00	3024	2.3E+00
Krylov(30)	124	1.3E−01	124	1.3E−01	124	1.3E−01	155	1.6E−01	186	1.9E−01	248	2.6E−01

	$w_1(t)$			$w_n(t)$		
	$t = 1$	$t = 10$	$t = 100$	$t = 1$	$t = 10$	$t = 100$
$\varepsilon_{\text{tol}} = 10^{-7}$						
inexactUnif	9.960117786E−01	9.610794944E−01	6.821540892E−01	0.000000000E+00	0.000000000E+00	0.000000000E+00
exactUnif	9.960117791E−01	9.610794990E−01	6.821540968E−01	0.000000000E+00	1.73272207E−201	5.34734533E−147
Krylov(30)	9.960118222E−01	9.610796409E−01	6.821552696E−01	0.000000000E+00	0.000000000E+00	1.78287456E−142
$\varepsilon_{\text{tol}} = 10^{-10}$						
inexactUnif	9.960118221E−01	9.610796408E−01	6.821552686E−01	0.000000000E+00	0.000000000E+00	0.000000000E+00
exactUnif	9.960118221E−01	9.610796408E−01	6.821552686E−01	0.000000000E+00	1.75235449E−201	5.34739831E−147
Krylov(30)	9.960118222E−01	9.610796409E−01	6.821552696E−01	0.000000000E+00	0.000000000E+00	3.603659264E−94

C.3. NCD3: $n = 91,881$; $nz = 623,241$; $||.||_\infty = 99.39$.

	$\varepsilon_{\text{tol}} = 10^{-5}$		$\varepsilon_{\text{tol}} = 10^{-6}$		$\varepsilon_{\text{tol}} = 10^{-7}$		$\varepsilon_{\text{tol}} = 10^{-8}$		$\varepsilon_{\text{tol}} = 10^{-9}$		$\varepsilon_{\text{tol}} = 10^{-10}$	
	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time	uMATVEC	Time
$t = 1$												
inexactUnif	1	1.2E−01	1	1.2E−01	1	1.3E−01	1	1.3E−01	1	1.4E−01	1	1.4E−01
exactUnif	82	5.1E−01	87	5.4E−01	90	5.6E−01	94	5.8E−01	97	6.0E−01	101	6.3E−01
Krylov(30)	62	6.8E−01	62	6.7E−01	62	6.7E−01	62	6.7E−01	62	6.8E−01	62	6.8E−01
$t = 10$												
inexactUnif	1	1.0E+00	1	1.1E+00	1	1.1E+00	1	1.1E+00	1	1.2E+00	1	1.2E+00
exactUnif	725	4.5E+00	750	4.7E+00	775	4.8E+00	800	5.0E+00	825	5.1E+00	845	5.3E+00
Krylov(30)	62	6.7E−01	62	6.8E−01	62	6.7E−01	62	6.7E−01	62	6.7E−01	62	6.7E−01
$t = 100$												
inexactUnif	1	1.0E+01	2	1.1E+01	3	1.1E+01	4	1.1E+01	6	1.2E+01	8	1.2E+01
exactUnif	7250	4.5E+01	7500	4.7E+01	7750	4.8E+01	8000	5.0E+01	8250	5.1E+01	8450	5.3E+01
Krylov(30)	155	1.7E+00	155	1.7E+00	217	2.4E+00	279	3.0E+00	341	3.7E+00	434	4.7E+00

	$w_1(t)$			$w_n(t)$		
	$t = 1$	$t = 10$	$t = 100$	$t = 1$	$t = 10$	$t = 100$
$\varepsilon_{\text{tol}} = 10^{-7}$						
inexactUnif	9.920394538E−01	9.236736066E−01	4.653047977E−01	0.000000000E+00	0.000000000E+00	0.000000000E+00
exactUnif	9.920394553E−01	9.236736193E−01	4.653048069E−01	0.000000000E+00	0.000000000E+00	0.000000000E+00
Krylov(30)	9.920395500E−01	9.236740522E−01	4.653069901E−01	0.000000000E+00	0.000000000E+00	0.000000000E+00
$\varepsilon_{\text{tol}} = 10^{-10}$						
inexactUnif	9.920395499E−01	9.236740518E−01	4.653069870E−01	0.000000000E+00	0.000000000E+00	0.000000000E+00
exactUnif	9.920395499E−01	9.236740518E−01	4.653069870E−01	0.000000000E+00	0.000000000E+00	0.000000000E+00
Krylov(30)	9.920395500E−01	9.236740522E−01	4.653069888E−01	0.000000000E+00	0.000000000E+00	3.17331408E−220

Acknowledgments. The comments of the referees contributed to improving the paper and are gratefully acknowledged.

REFERENCES

[1] R. Barrett, M. W. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, 1993.

[2] A. Berman and R. J. Plemmons, *Nonnegative Matrices in the Mathematical Sciences*, Academic Press, New York, 1979.

[3] A. Bouras and V. Frayssé, *Inexact matrix-vector products in Krylov methods for solving linear systems: A relaxation strategy*, SIAM J. Matrix Anal. Appl., 26 (2005), pp. 660–678.

[4] K. Burrage, M. Hegland, S. MacNamara, and R. B. Sidje, *A Krylov-based finite state projection algorithm for solving the chemical master equation arising in the discrete modelling of biological systems*, in Proceedings of the 150$^{\text{th}}$ Markov Anniversary Meeting, A. N. Langville and W. J. Stewart, eds., Boson Books, Raleigh, NC, 2006, pp. 21–38.

[5] J. R. Gilbert, C. Moler, and R. Schreiber, *Sparse matrices in MATLAB: Design and implementation*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 333–356.

[6] W. Grassmann, *Transient solutions in Markovian queueing systems*, Comput. Oper. Res., 4 (1977), pp. 47–56.

[7] D. Gross and D. R. Miller, *The randomization technique as modelling tool and solution procedure for transient Markov processes*, Oper. Res., 32 (1984), pp. 343–361.

[8] M. Hochbruck and Ch. Lubich, *On Krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal., 34 (1997), pp. 1911–1925.

[9] C.-Y. F. Huang and J. E. Ferrell, Jr., *Ultrasensitivity in the mitogen-activated protein kinase cascade*, Proc. Natl. Acad. Sci. USA, 93 (1996), pp. 10078–10083.

[10] S. MacNamara, R. B. Sidje, and K. Burrage, *An improved dynamic finite state projection algorithm for the numerical solution of the chemical master equation with applications*, in Proceedings of the 13th Biennial Computational Techniques and Applications Conference (CTAC-2006), W. Read and A. J. Roberts, eds., Anziam J., 48 (2007), pp. C397–C419.

[11] C. Moler and C. Van Loan, *Nineteen dubious ways to compute the exponential of a matrix, Twenty-five years later*, SIAM Rev., 45 (2003), pp. 3–49.

[12] B. Philippe, Y. Saad, and W. J. Stewart, *Numerical methods in Markov chains modeling*, Oper. Res., 40 (1992), pp. 1156–1179.

[13] B. Philippe and R. B. Sidje, *Transient solutions of Markov processes by Krylov subspaces*, in Computations with Markov Chains: 2nd International Workshop on the Numerical Solution of Markov Chains, W. J. Stewart, ed., Kluwer International Publishers, Boston, MA, 1995, pp. 95–119.

[14] R. B. Sidje, *Expokit: A software package for computing matrix exponentials*, ACM Trans. Math. Software, 24 (1998), pp. 130–156; available online at http://www.expokit.org.

[15] R. B. Sidje and W. J. Stewart, *A numerical study of large sparse matrix exponentials arising in Markov chains*, Comput. Statist. Data Anal., 29 (1999), pp. 345–368.

[16] V. Simoncini and D. B. Szyld, *Theory of inexact Krylov subspaces methods and applications to scientific computing*, SIAM J. Sci. Comput., 25 (2003), pp. 454–477.

[17] W. J. Stewart, *Marca: Markov Chain Analyser. A Software Package for Markov Modelling*, Technical report, Department of Computer Science, North Carolina State University, Raleigh, NC, 1988; also IEEE Computer Repository report R76 232, 1976, and IRISA Publication Interne 45, Université de Rennes, France.

[18] W. J. Stewart, *Introduction to the Numerical Solution of Markov Chains*, Princeton University Press, Princeton, NJ, 1994.

[19] J. van den Eshof and G. L. G. Sleijpen, *Inexact Krylov subspace methods for linear systems*, SIAM J. Matrix Anal. Appl., 26 (2004), pp. 125–153.

SIAM J. SCI. COMPUT.
Vol. 29, No. 6, pp. 2581–2606
© 2007 Society for Industrial and Applied Mathematics

A HIGH-ORDER SOLVER FOR THE HEAT EQUATION IN 1D DOMAINS WITH MOVING BOUNDARIES*

SHRAVAN K. VEERAPANENI† AND GEORGE BIROS‡

Abstract. We describe a fast high-order accurate method for the solution of the heat equation in domains with moving Dirichlet or Neumann boundaries and distributed forces. We assume that the motion of the boundary is prescribed. Our method extends the work of Greengard and Strain [*Comm. Pure Appl. Math.*, XLIII (1990), pp. 949–963]. Our scheme is based on a time-space Chebyshev pseudo-spectral collocation discretization, which is combined with a recursive product quadrature rule to accurately and efficiently approximate convolutions with Green's function for the heat equation. We present numerical results that exhibit up to eighth-order convergence rates. Assuming N time steps and M spatial discretization points, the evaluation of the solution of the heat equation at the same number of points in space-time requires $\mathcal{O}(NM \log M)$ work. Thus, our scheme can be characterized as "fast"; that is, it is work-optimal up to a logarithmic factor.

Key words. integral equations, spectral methods, Chebyshev polynomials, moving boundaries, heat equation, quadratures, Nyström's method, collocation methods, potential theory

AMS subject classifications. 35K05, 31A10, 65D30, 65N35, 65N38

DOI. 10.1137/060677896

1. Introduction.

We present a fast and high-order method for the solution of the one-dimensional (1D) heat equation in domains with moving boundaries. We assume that the boundary motion is prescribed. Given smooth functions f, g, and w we seek to compute $u(x,t)$ such that

$$\begin{aligned} \frac{\partial u}{\partial t} &= \triangle u(x,t) + f(x,t) \quad \text{in} \quad \omega(t), \qquad t > 0, \\ u(x,0) = w(x) \quad \text{in} \quad \omega(0), &\quad u(x,t) = g(x,t) \quad \text{on} \quad \gamma(t). \end{aligned} \tag{1.1}$$

All of our algorithmic choices can be extended to 2D and 3D. The details, however, become quite involved, and additional algorithmic components are necessary. For clarity and due to space limitations, we present the core ideas of our method for the one-dimensional case. We will report the extensions to higher dimensions in a future article.

Formulation. The proposed algorithm is based on potential theory [13]. Without loss of generality, let $\omega(t)$ be contained in the unit box $\Omega = [0,1]$. Let Γ denote the boundary of the unit box. By linearity, we decompose the problem (1.1) into the initial condition component u_i, the distributed force component u_f, and the boundary contribution u_b:

$$\frac{\partial u_i}{\partial t} = \triangle u_i \quad \text{in} \quad \Omega, \quad u_i(x,0) = w(x), \quad u_i(\Gamma,t) = 0, \quad t > 0, \tag{1.2}$$

*Received by the editors December 17, 2006; accepted for publication (in revised form) June 20, 2007; published electronically October 24, 2007. This work is partially supported by the U.S. Department of Energy under grant DE-FG02-04ER25646 and by the U.S. National Science Foundation grants CCF-0427985 and DMS-0612578.

http://www.siam.org/journals/sisc/29-6/67789.html

†Department of Mechanical Engineering and Applied Mechanics, University of Pennsylvania, Philadelphia, PA 19104 (shravan@seas.upenn.edu).

‡Department of Mechanical Engineering and Applied Mechanics and Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104 (biros@seas.upenn.edu.)

$$\frac{\partial u_f}{\partial t} = \triangle u_f + f \quad \text{in} \quad \Omega, \quad u_f(x,0) = 0, \quad u_f(\Gamma,t) = 0, \quad t > 0, \tag{1.3}$$

$$\frac{\partial u_b}{\partial t} = \triangle u_b \quad \text{in} \quad \omega(t), \quad u_b(x,0) = 0, \quad u_b(\gamma(t),t) = g - u_i - u_f, \quad t > 0. \tag{1.4}$$

We solve (1.2), (1.3), and (1.4) using an integral equation formulation; we compute u_i and u_f by

$$\begin{aligned} u_i(x,t) &= \mathcal{V}_0[w](x,t) = \int_\Omega G(x,t;y)w(y)dy, \\ u_f(x,t) &= \mathcal{V}[f](x,t) = \int_0^t \int_\Omega G(x,t;y,\tau)f(y,\tau)dy\,d\tau, \end{aligned} \tag{1.5}$$

where $G(x,t;y,\tau)$ is Green's function for the unit box Ω with periodic boundary conditions.[1] The solution of (1.4) is obtained by the solution of a double layer indirect integral equation formulation, which is given by

$$u_b(x,t) = \mathcal{D}[\phi](x,t) = \int_0^t \int_{\gamma(\tau)} \frac{\partial G(x,t;y,\tau)}{\partial n(y,\tau)} \phi(y,\tau) ds(y) d\tau, \ x \in \omega \times (0,T]. \tag{1.6}$$

Here n is the outward normal to γ. The double layer potential denoted by $\mathcal{D}[\phi]$ is the solution for problem (1.4) provided the boundary density ϕ solves the following Fredholm second-kind integral equation for $t > 0$:

$$-\frac{1}{2}\phi(x,t) + \mathcal{D}[\phi](x,t) = u_b(x,t) \quad \forall x \in \gamma(t). \tag{1.7}$$

Computational complexity. The solution of (1.1) through potential theory requires evaluation of double layer and volume potentials. A direct evaluation of $\mathcal{V}[f]$ using $M \times N$ quadrature points at $M \times N$ locations in space and time requires $\mathcal{O}(M^2N^2)$ work. The history dependence of this convolution can be overcome by using the fast algorithm of Greengard and Strain [6]. The key idea is to use two equivalent expansions for the kernel $G(x,t)$: one that converges fast at distant times and the other that converges fast locally. The volume potential $\mathcal{V}$ is split into a local or near part $\mathcal{V}_L$ and a far part $\mathcal{V}_F$. For the evaluation of the local part, a method-of-images expansion of G is used; for the far part, a Fourier expansion is used. A recurrence relation to update the Fourier coefficients in the far part eliminates the need to integrate over the entire history. As pointed out in [22], discrete sums of the form $\sum_{n=1}^{p} C_n \sin(n\pi x)$ and $\sum_{k=1}^{M} \sin(n\pi y_k) f_k$ that arise in the computation of $\mathcal{V}_F$ can be computed in optimal time by using the nonuniform FFT (NUFFT) [4]. The local part can be computed optimally using the fast Gauss transform [7]. Using these methods the overall complexity of computing the volume potential can be reduced to $\mathcal{O}(MN \log M)$; the overall scheme is second-order accurate.

Synopsis of the new method. Here we extend the Greengard and Strain algorithm to a high-order accurate scheme. We develop special product integration rules to compute the local and far parts of the heat potentials. The basic idea is to approximate the boundary density using Chebyshev polynomials and then compute the resulting moments exactly. To achieve optimal complexity, we propose a scheme

[1] Here we describe the decomposition into three subproblems for the case of a Dirichlet boundary condition. The extension to the Neumann case is analogous.

that makes use of fast summation methods [4, 6, 7]. We then build a solver for (1.1) based on a fast and high-order accurate evaluation of volume and layer potentials.

Following Greengard and Strain, this is the outline of our method:

- We use a double layer integral equation formulation for (1.1), which results in a well-conditioned linear system;
- we discretize using a pseudo-spectral collocation method using Chebyshev polynomials;
- we develop high-order quadratures for the heat (and other sharply peaked) kernels using recurrences;
- we derive a high-order scheme for the heat equation in domains with moving boundaries (with prescribed motion) and distributed forces.

The main contributions of this work are the high-order time-marching scheme and the extension to problems with moving boundaries and distributed sources.

Related work. Much of the research on solving the heat equation on moving boundaries has been concentrated on Stefan's problem for which the motion of the boundary is unknown. In this paper, we consider only problems in which the motion of the boundary is *prescribed.*[2] Fast algorithms introduced recently dramatically improved the computational complexity of solving integral equation formulations for parabolic PDEs. The fast Gauss transform [7] can be used to accelerate the solution of the free-space initial value problem for the heat equation. The Greengard and Strain algorithm [6] can be used for the efficient evaluation of single and double layer heat potentials in bounded domains. Fast algorithms for unbounded domains are discussed in [5, 14]. The work of [2, 20] is one of the first attempts to design fast methods for the heat equation. A direct formulation was used in [10, 28] for the 1D Stefan problem. In the case of prescribed Dirichlet data, a direct integral equation formulation leads to a Volterra system of equations that is ill-conditioned.[3] Ill-conditioning can be avoided by using an indirect double layer formulation, which we describe in detail in section 5. Recently, epitaxial step flow growth in 1D was simulated in [9] using layer potentials. Despite these remarkable advances, however, none of the existing methods achieves both high accuracy and optimal complexity.

Indeed, fast summation algorithms and high-order schemes are necessary to build work-efficient solvers for (1.1). The time integrals in all variants of the heat potentials have kernels that are sharply peaked. A generic quadrature rule can be used, but it will not achieve its order of convergence for reasonable discretization sizes. Uniform second-order convergence in computing the single layer was achieved in [6, 20] using asymptotic expansions. To obtain high-order convergence through asymptotic expansions, one needs high-order derivatives of the boundary density ϕ, something that can be computationally expensive. In [22, 23] a uniform second-order convergence was achieved by using a piecewise linear approximation for the boundary density and computing the convolution with the heat kernel exactly.

An alternative is the design of special quadrature rules: Given $I(\alpha) = \int_{-1}^{1} K(\alpha, \theta)\, \phi(\theta) d\theta$, with K being a singular, sharply peaked, or oscillatory kernel, the goal is to design high-order accurate integration schemes. There exists a substantial body

[2]There is a large (and significant) body of work regarding local stencil-based discretizations of (1.1); we do not attempt to review this literature. We believe that, for certain problems, there are significant advantages in using integral equations. Since this work is in 1D, we postpone the discussion to a future paper.

[3]In 1D ill-conditioning has little impact in practical computations. The argument is important for problems in higher dimensions.

of work on quadrature rules for such problems. The main challenge is to ensure optimal algorithmic complexity and compatibility with fast summation methods. In Kapur and Rokhlin [12] and Rokhlin [19], corrections to the trapezoidal rule have been suggested for integrating singular functions. In Ma and Wandzura [11] and in Yarvin and Rokhlin [26], numerical tools to obtain generalized Gaussian quadratures have been devised. In Alpert [1] hybrid Gauss-trapezoidal rules for regular functions and singular kernels were developed. These approaches, however, are not directly applicable to the heat kernel due to the spatial and temporal components of the kernel. Although it may be the case that they can be extended to the heat kernel, we have opted for an alternative approach based on recurrence relationships.

Such recurrences have been established by Piessens and Branders [15] for a wide variety of kernels. In Hasegawa and Torii [8], Cauchy principal value integrals were evaluated using quadrature rules combined with Chebyshev-polynomial approximations. Here we concentrate on heat potentials, but this strategy can be applied to solving other PDEs via integral equations. For example, in Piessens [16] integral equations like the Abel integral equation were solved. The main difference between this method and quadrature rules is that we discretize the density but not the integral operator. Given a density $\phi(\theta)$ and a kernel $K(\alpha,\theta)$ we write:

(1.8)
$$\phi(\theta)=\sum_{k=0}^{q-1}\phi(k)T_k(\theta)\ \Rightarrow I(\alpha)=\sum_{k=0}^{q}\phi(k)I_k(\alpha),\quad \text{where } I_k(\alpha)=\int_{-1}^{1}K(\alpha,\theta)T_k(\theta)d\theta,$$

based on the Chebyshev-polynomial expansion of ϕ. The integrals $I_k(\alpha)$ are computed to machine accuracy using recurrence relations (section 2). Using the FFT, the Chebyshev coefficients $\{\phi(n)\}_{n=0}^{q}$ can be computed in $\mathcal{O}(q\log q)$ work, and all $I_k(\alpha)$ can be computed using recurrence relations in $\mathcal{O}(q)$ work. Hence the overall complexity of computing the integral $I(\alpha)$ is $\mathcal{O}(q\log q)$. Using this method we compute heat potentials by approximating the potential density function using piecewise $(q-1)$th-order Chebyshev polynomials and thus obtain a qth-order accurate method.

Contents. In section 2, we discuss algorithms for the fast evaluation of heat potentials. In section 3, we derive recurrence relations for computing the moments $I_k(\alpha)$ defined in (1.8) for heat kernels. The details of implementing the boundary integral solver for (1.1) with static boundaries are given in section 4. The case of boundaries with prescribed motion is discussed in section 5. We report numerical results that verify the efficiency and accuracy of our scheme in section 6.

2. Fast summation. The direct evaluation of u_b by (1.6) at M spatial locations and N time levels requires $\mathcal{O}(N^2M)$ work in 1D. The Greengard–Strain algorithm reduces this work to $\mathcal{O}(NM)$. The key idea is to use two different representations of Green's function (for a square box with periodic boundary conditions) in different time intervals: one for the history part that represents the influence of the sources located temporally away from the current evaluation time and one for the local part that represents the influence of the sources located at times closer to the current evaluation time.

Green's function for the unit box is the solution to the adjoint problem with homogeneous boundary conditions:

$$\frac{\partial G}{\partial \tau}+\triangle_y G=\delta(x-y,t-\tau)\quad \text{in } \Omega\quad \text{such that (s.t.) } G(\Gamma,t)=0. \tag{2.1}$$

The solution of this problem by Fourier series and by Kelvin's method of images give

us two equivalent expansions:

$$G(x,y;t,\tau) = \sum_{n=1}^{\infty} e^{-n^2\pi^2(t-\tau)} 2\sin(n\pi x)\sin(n\pi y) \tag{2.2}$$

$$= \sum_{n=-\infty}^{\infty} \left(\frac{e^{-(x-y-2n)^2/4(t-\tau)}}{\sqrt{4\pi(t-\tau)}} - \frac{e^{-(x+y-2n)^2/4(t-\tau)}}{\sqrt{4\pi(t-\tau)}} \right). \tag{2.3}$$

These expansions converge with different rates in different time intervals. The Fourier representation converges faster at distant times (from the current evaluation time), whereas the method of images representation converges faster at closer times. This motivates splitting of heat potentials into two parts:

$$\mathcal{V}[f] = \mathcal{V}_F[f] + \mathcal{V}_L[f] = \int_0^{t-\delta} \int_\Omega Gf + \int_{t-\delta}^{t} \int_\Omega Gf. \tag{2.4}$$

Here δ is a parameter that determines the error in truncating both of the series in (2.3). The operator $\mathcal{V}_F$ will be termed as the far part, and $\mathcal{V}_L$ will be termed as the local part. The exact same decomposition is also valid for single and double layer potentials, with f being replaced by the boundary density ϕ.

2.1. Truncating the series. In 1D the boundary γ of the domain ω corresponds to two points b_1 and b_2. Then the far part of the double layer potential simplifies to

$$\mathcal{D}_F[\phi] = \sum_{n=1}^{\infty} \sum_{k=1}^{2} (-1)^k 2n\pi \sin(n\pi x) \int_0^{t-\delta} \cos(n\pi b_k) e^{-n^2\pi^2(t-\tau)} \phi(b_k,\tau) d\tau. \tag{2.5}$$

The error in truncating the series (2.5) after p terms can be bounded from above as follows:

$$\begin{aligned}
\mathcal{E}_F(p) &= \left| \mathcal{D}_F[\phi] - \sum_{n=1}^{p} \sum_{k=1}^{2} (-1)^k 2n\pi \sin(n\pi x) \int_0^{t-\delta} \cos(n\pi b_k) e^{-n^2\pi^2(t-\tau)} \phi(b_k,\tau) d\tau \right| \\
&= \left| \sum_{n=p+1}^{\infty} \sum_{k=1}^{2} (-1)^k 2n\pi \sin(n\pi x) \int_0^{t-\delta} \cos(n\pi b_k) e^{-n^2\pi^2(t-\tau)} \phi(b_k,\tau) d\tau \right| \\
&\le 4|\phi|_\infty \sum_{n=p+1}^{\infty} \left| \int_0^{t-\delta} n\pi e^{-n^2\pi^2(t-\tau)} d\tau \right| = 4|\phi|_\infty \sum_{n=p+1}^{\infty} \left| \frac{e^{-n^2\pi^2\delta} - e^{-n^2\pi^2 t}}{n\pi} \right| \\
&\le \frac{4|\phi|_\infty}{\pi(p+1)} \sum_{n=p+1}^{\infty} e^{-n^2\pi^2\delta} \le \frac{4|\phi|_\infty}{\pi(p+1)} e^{-(p+1)^2\pi^2\delta} \int_0^{\infty} e^{-\pi^2 x^2 \delta} dx \\
&= \frac{2|\phi|_\infty}{\pi\sqrt{\pi\delta}(p+1)} e^{-(p+1)^2\pi^2\delta}.
\end{aligned}$$

The local part expansion represents the influence of Gaussian pulses initiated at $t = 0$ and located at $\{x - 2n, 2n - x\}_{n=-\infty}^{\infty}$. The value of a Gaussian pulse evaluated at a distance $2r\sqrt{t}$ from its center would be of the order $\frac{e^{-r^2}}{\sqrt{t}}$. Hence, if the boundary γ is a distance d away from the boundary of the unit box Γ, then the error in approximating Green's function by just the first term ($n = 0$) is of the order $\frac{e^{-d^2/\delta}}{\sqrt{\delta}}$.

Then the local part of the double layer is

$$\mathcal{D}_L[\phi] = \int_{t-\delta}^{t} \int_{\gamma(\tau)} \frac{x-y}{4[\pi(t-\tau)]^{3/2}} e^{-\frac{(x-y)^2}{4(t-\tau)}} \phi(y,\tau) d\tau + \mathcal{O}\left(\frac{e^{-d^2/\delta}}{\delta^{3/2}}\right). \tag{2.6}$$

2.2. Fast evaluation of the far part. A direct evaluation of $\mathcal{D}_F[\phi]$ at $M \times N$ points in space and time requires $\mathcal{O}(N^2 M p)$ computations. Optimal complexity of $\mathcal{O}((M+p)N)$ can be achieved by using the fast algorithms in [4, 6]. We briefly summarize these algorithms here. Let

$$\hat{\phi}_n(\tau) = \int_{\gamma(\tau)} \frac{\partial \sin(n\pi y)}{\partial n(y)} \phi(y,\tau) ds(y) = n\pi \left(\cos(n\pi b_2)\phi(b_2,\tau) - \cos(n\pi b_1)\phi(b_1,\tau)\right), \tag{2.7}$$

$$C_n(t,\delta) = 2\int_0^{t-\delta} e^{-n^2\pi^2(t-\tau)} \hat{\phi}_n(\tau)\, d\tau. \tag{2.8}$$

Then

$$\mathcal{D}_F[\phi](x,t) = \sum_{n=1}^{p} C_n(t,\delta)\sin(n\pi x). \tag{2.9}$$

The Fourier coefficients can be updated with constant work per time step using the following recurrence:

$$C_n(t+\triangle t,\delta) = e^{-n^2\pi^2\triangle t} C_n(t,\delta) + 2\int_{t-\delta}^{t+\triangle t-\delta} e^{-n^2\pi^2(t+\triangle t-\tau)} \hat{\phi}_n(\tau) d\tau. \tag{2.10}$$

This recurrence is the key step that eliminates the history dependence of the far part. Once we have computed the C_n coefficients, we can evaluate (2.9) using the inverse NUFFT.

2.3. Optimal splitting of the heat potentials. In this section we briefly comment on the selection[4] of the parameters associated with the splitting of the heat potentials to a far and local part. If we retain p terms in the far part expansion and if $\delta = l\triangle t$, the overall complexity of computing $\mathcal{D}[\phi]$ is $W = \mathcal{O}(N(M+p)\log p + MNl)$. For each target point, the local part computation involves integrating over l time steps, hence the local part evaluation contributes $\mathcal{O}(MNl)$ to W. For fixed constants (c_1, c_2), we set $p = c_1\sqrt{M}$ and $l = \left\lceil \frac{c_2 \log M}{N\triangle t} \right\rceil$ to obtain $W = \mathcal{O}(MN\log M)$, which is off from being optimal only by a logarithmic factor. For this choice of parameters (assuming exact evaluation of the far and near integrals) the error estimates due to the splitting are given by

$$\begin{aligned} \mathcal{E}_F(M,N) &= \mathcal{O}\left(\left(\frac{N}{M\log M}\right)^{1/2} e^{-\frac{\pi^2 c_1^2 c_2 M \log M}{N}}\right), \\ \mathcal{E}_L(M,N) &= \mathcal{O}\left(\left(\frac{N}{\log M}\right)^{3/2} e^{-\frac{d^2 N}{c_2 \log M}}\right). \end{aligned} \tag{2.11}$$

[4]The discussion here is important only in the asymptotic limit, i.e., when the size of the time step is small. But for significantly large time-step discretizations, we can just fix the parameters p, δ such that $\delta < \triangle t$ and such that the truncation errors are within the required accuracy. This would suffice to obtain optimal complexity.

Therefore, both errors are exponentially converging as M and N are increased. For instance, if $N = M = 256$, by choosing $c_1 = 2.5, c_2 = 0.1$, both $\mathcal{E}_F, \mathcal{E}_L$ are less than 10^{-15}, and for this choice we get $p = 40, \delta \approx 0.0022$.

Next we describe our construction of recursive product integration rules for the local and far parts.

3. Quadratures. In this section, we compute the moments $I_k(\alpha)$ as defined by (1.8) for the kernels: $e^{-\alpha\theta}$, $\frac{e^{-\frac{\alpha}{\theta}}}{\sqrt{\theta}}$, $\frac{e^{-\frac{\alpha}{\theta}}}{\sqrt{\theta^3}}$ and then show how they can be used to compute the time integrals in $\mathcal{D}_L$ and $\mathcal{D}_F$. First, we review some basic properties of Chebyshev polynomials. For $x \in [-1, 1]$, the closed-form expression for the nth-order Chebyshev polynomial $T_n(x)$ is given by:

$$T_n(x) = \cos(n \cos^{-1} x). \tag{3.1}$$

The Chebyshev polynomials $T_n(x)$ are orthogonal w.r.t. the inner product defined by:

$$\langle f(x), g(x) \rangle = \int_{-1}^{1} \frac{f(x)g(x)}{\sqrt{1-x^2}} dx. \tag{3.2}$$

The nth-order Chebyshev coefficient of $f(x)$ denoted by $f(n)$ can be computed by taking the inner product with $T_n(x)$:

(3.3)
$$f(n) = \int_{-1}^{1} \frac{f(x)T_n(x)}{\sqrt{1-x^2}}\, dx = \frac{c_n}{\pi} \int_0^{\pi} f(\cos\theta) \cos n\theta \, d\theta = \frac{c_n}{2\pi} \int_0^{2\pi} f(\cos\theta) \cos n\theta \, d\theta,$$

$$\text{where} \quad c_0 = 1 \quad \text{and} \quad c_k = 2 \quad \text{for} \quad k \geq 1. \tag{3.4}$$

Now as the function $f(\cos\theta)$ is 2π-periodic, we can use FFT or the fast cosine transform to compute $\{f(n)\}_{n=0}^{q-1}$ in $\mathcal{O}(q \log q)$ time. Also, we will use the following properties of Chebyshev polynomials:

$$T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x), \quad \text{and} \quad \int T_n = \frac{1}{2}\left(\frac{T_{n+1}}{n+1} - \frac{T_{n-1}}{n-1}\right) + C.$$

For more details on the properties of Chebyshev polynomials, we refer to [18]; a review on spectral methods for solving PDEs using the Chebyshev polynomials can be found in [24].

3.1. Recurrence relations. The idea of computing the integrals through the approximation of the integrand by Chebyshev polynomials and then computing the moments of smooth functions by recurrences was first used by Clenshaw and Curtis [3]. Consider integrating a smooth function $f(x)$ in the interval $(0, 1)$:

$$I = \int_0^1 f(x)dx = \sum_{n=0}^{q-1} f(n) \int_0^1 T_n(2x-1)dx = \sum_{n=0}^{q-1} f(n) I_n,$$

$$\text{where} \quad I_n = \int_0^1 T_n(2x-1)dx = \left[\frac{T_{n+1}(2x-1)}{n+1} - \frac{T_{n-1}(2x-1)}{n-1}\right]_0^1.$$

One can easily show that evaluation of this will yield:

$$I_n = \begin{cases} 0 & \text{if } n \text{ is odd,} \\ -\frac{1}{n^2-1} & \text{otherwise.} \end{cases} \tag{3.5}$$

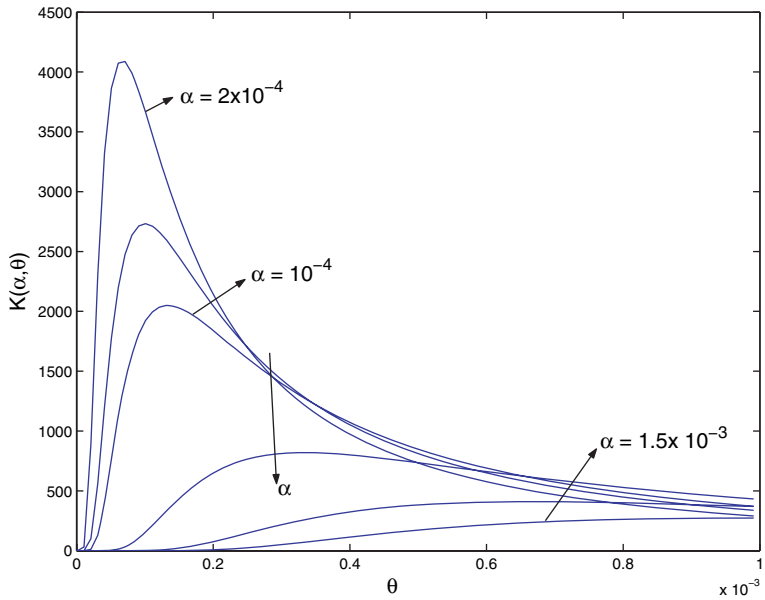


FIG. 3.1. *Kernel* $K(\alpha,\theta) = \frac{\sqrt{\alpha}}{\sqrt{\theta^3}} e^{-\frac{\alpha}{\theta}}$ *for different values of* α. *The kernel becomes sharply peaked as* α *decreases, so an accurate evaluation of* (3.6) *by generic quadratures requires an excessive number of points.*

Similarly we can derive the recurrences for computing the moments $I_k(\alpha) = \int_0^1 K(\alpha,\theta)$ $T_k(\theta)$ for a variety of kernels K. In Appendix A we give recurrences for the singular kernels $\theta^{-\gamma}$ for $\gamma \in (-\infty, 1)$ and $\log\theta$. As the approximation of $f(\theta)$ by Chebyshev polynomials is superalgebraically convergent, the error in the computation of the integral $I(\alpha)$ is approaching zero exponentially (numerical examples can be found in section 6). Now we discuss the computation of integral operators that will arise while evaluating the heat potentials. Let us consider

$$I(\alpha) = \int_0^1 \frac{\sqrt{\alpha}}{\sqrt{\theta^3}} e^{-\frac{\alpha}{\theta}} f(\theta)\, d\theta, \quad \text{and} \quad J(\alpha) = \int_0^1 \frac{1}{\sqrt{\theta}} e^{-\frac{\alpha}{\theta}} f(\theta)\, d\theta. \tag{3.6}$$

In the limit any generic quadrature rule will converge to the correct value of I and J with its optimal convergence rate. The quality of the approximation, however, will be nonuniform in the value of α. In fact, the constants deteriorate dramatically with small α. As a result, I and J become very difficult to compute accurately; a generic rule will require thousands of points—even for a few digits of accuracy. This is because the kernel becomes sharply peaked for smaller values of α; see Figure 3.1.

This poses a challenge for Nyström-type methods as one would need to develop specialized quadratures for different values of α. Instead, we are using a Chebyshev approximation of f and derive recurrence relations to compute the moments I_n. In this way, given α, we can compute all I_n with $\mathcal{O}(q)$ work.

First we compute the base condition for the recurrence:

$$I_0(\alpha) = \int_0^1 \frac{\sqrt{\alpha}}{\sqrt{\theta^3}} e^{-\frac{\alpha}{\theta}}\, d\theta = 2\int_{\sqrt{\alpha}}^{\infty} e^{-z^2} dz = \sqrt{\pi}\mathrm{erfc}(\sqrt{\alpha}).$$

Then we compute higher moments by

$$I_n(\alpha) = \int_0^1 \frac{\sqrt{\alpha}}{\sqrt{\theta^3}} e^{-\frac{\alpha}{\theta}} T_n \, d\theta = \int_0^1 \frac{\sqrt{\alpha}}{\sqrt{\theta^3}} e^{-\frac{\alpha}{\theta}} \left[2(\lambda\theta+\eta)T_{n-1} - T_{n-2}\right] d\theta,$$

$$\Rightarrow I_n(\alpha) = 2\lambda\sqrt{\alpha} J_{n-1} + \eta I_{n-1} - I_{n-2}, \tag{3.7}$$

$$\text{where} \quad J_{n-1}(\alpha) = \int_0^1 \frac{1}{\sqrt{\theta}} e^{-\frac{\alpha}{\theta}} T_{n-1} \, d\theta = 2\lambda \int_0^1 \sqrt{\theta} e^{-\frac{\alpha}{\theta}} T_{n-2} \, d\theta + 2\eta J_{n-2} - J_{n-3}.$$

Integrating by parts, we get:

$$2\lambda \int_0^1 \sqrt{\theta} e^{-\frac{\alpha}{\theta}} T_{n-2} \, d\theta = C_n - \int_0^1 \left(\frac{1}{2\sqrt{\theta}} + \frac{\alpha}{\sqrt{\theta^3}}\right) e^{-\frac{\alpha}{\theta}} \left(\frac{T_{n-1}}{n-1} - \frac{T_{n-3}}{n-3}\right) d\theta.$$

Substituting this back in the expression for J_{n-1}, we obtain the following recurrence:

$$J_{n-1}(\alpha) = \frac{2n-2}{2n-1}\left(C_n - \frac{2n-7}{2n-6} J_{n-3} + 2\eta J_{n-2} - \sqrt{\alpha}\left(\frac{I_{n-1}}{n-1} - \frac{I_{n-3}}{n-3}\right)\right), \tag{3.8}$$

$$\text{where} \quad C_n = e^{-\alpha}\left(\frac{T_{n-1}(\lambda+\eta)}{n-1} - \frac{T_{n-3}(\lambda+\eta)}{n-3}\right). \tag{3.9}$$

Using (3.8) we can compute $J_{n-1}(\alpha)$, from which $I_n(\alpha)$ can be computed using (3.7). Now we establish the recurrences for the kernel $e^{-\alpha\theta}$:

$$I_n = \int_0^1 e^{-\alpha x} T_n(\lambda x + \eta) \, dx,$$

$$= \left[\frac{e^{-\alpha x}}{2\lambda}\left(\frac{T_{n+1}}{n+1} - \frac{T_{n-1}}{n-1}\right)\right]_0^1 + \int_0^1 \left[\frac{e^{-\alpha x}}{2\lambda}\left(\frac{T_{n+1}}{n+1} - \frac{T_{n-1}}{n-1}\right)\right],$$

$$\Rightarrow I_{n+1} = c_{n+1} + (n+1)\left(\frac{2\lambda}{\alpha} I_n + \frac{I_{n-1}}{n-1}\right),$$

$$\text{where } c_{n+1} = \frac{n+1}{\alpha}\left[e^{-\alpha x}\left(\frac{T_{n+1}}{n+1} - \frac{T_{n-1}}{n-1}\right)\right]_0^1. \tag{3.10}$$

The recurrences (3.8) and (3.10) are numerically unstable[5] in the forward direction. More specifically, we get a dominant solution rather than a minimal[6] solution for these recurrences, which prohibits us from computing higher-order moments (see [17]

[5]There are algorithms that can be used for stabilizing these recurrences; see, for example, [25]. The length of the domain ω (which is less than 1) sets an upper bound on the parameter α in the recurrences (3.7), (3.8). Based on this upper bound, we have obtained a global eighth-order accurate method without resorting to any such recurrence-stabilizing algorithms. In general, it is possible to get *arbitrary* order of accuracy by stabilizing the recurrences.

[6]If we take two linearly independent solutions f_n and g_n of a three term recurrence $y_{n+1} + a_n y_n + b_n y_{n-1} = 0$, then f_n is called a minimal solution if $f_n/g_n \to 0$ as $n \to \infty$; g_n is then termed as the dominant solution. Adding any multiple of f_n to g_n would give us another dominant solution, but the minimal solution is always unique.

for more details). Since we use a qth-order piecewise polynomial approximation of $\mu(\theta)$ we need to ensure that the recurrences are stable up to $n = q$.

Remark. The more sharply peaked the integration kernel is, the more stable the recurrences are and thus the higher the order of moments that we can compute accurately. For example, the kernel $e^{-\alpha\theta}$ is sharply peaked for higher values of α.

For $\alpha \ll 1$, instead of using the recurrence (3.10), we use $e^{-\alpha x} = \sum_{k=0}^{p}(-1)^k \frac{(\alpha x)^k}{k!}$ and the recurrences given in Appendix A to compute

$$I_n = \int_0^1 e^{-\alpha x} T_n \, dx = \sum_{k=0}^{p} \frac{(-\alpha)^k}{k!} M_{nk}, \quad \text{where } M_{nk} = \int_0^1 x^k T_n \, dx; \tag{3.11}$$

here p is chosen such that $\frac{\alpha^p}{p!} < \epsilon$, the required accuracy.

The numerical instability is a property of the recurrence relation; different recurrences such as (3.5) and the ones given in Appendix A for the kernels $\frac{1}{\theta^\gamma}$ for $\gamma \in (-\infty, 1)$ and $\log\theta$ are unconditionally stable.

3.2. Heat kernels. Using the recurrences derived here, we now develop product integration rules for the time integrals of far and near parts of the double layer.

(a) *Far part.* To compute $C_n(t + \triangle t, \delta)$ using the recurrence (2.10), we have to compute

$$U_n[\hat{\phi}] = \int_{t-\delta}^{t+\triangle t-\delta} e^{-n^2\pi^2(t+\triangle t-\tau)} \hat{\phi}_n(\tau) \, d\tau. \tag{3.12}$$

As n increases, the kernel becomes sharply peaked, prohibiting the use of smooth quadrature rules to compute (3.12). Instead, we use the recurrences. We compute the Chebyshev coefficients of $\hat{\phi}_n(\tau)$ in the interval[7] $(t - \delta, t + \triangle t - \delta)$, then we have $\hat{\phi}_n(\tau) = \sum_{k=0}^{q-1} \hat{\phi}_n(k) T_k(\frac{2}{\triangle t}(\tau - t + \delta) - 1)$. By setting $\tau = t + \triangle t - \theta\triangle t - \delta$ we obtain

$$U_n[\hat{\phi}] = \triangle t e^{-n^2\pi^2\delta} \sum_{k=0}^{q-1} \hat{\phi}_n(k) \int_0^1 e^{-(n^2\pi^2\triangle t)\theta} T_k(-2\theta + 1) \, d\theta = \sum_{k=0}^{q-1} \hat{\phi}_n(k) E_{nk}. \tag{3.13}$$

For a fixed step size $\triangle t$ we can precompute

$$E_{nk} = \triangle t e^{-n^2\pi^2\delta} \int_0^1 e^{-(n^2\pi^2\triangle t)\theta} T_k(-2\theta + 1), \quad n = 1, \ldots, p, \; k = 1, \ldots, q. \tag{3.14}$$

This precomputation is done in an optimal $\mathcal{O}(pq)$ time using the recurrence (3.10) with $\alpha = n^2\pi^2\triangle t$ and the scaling parameters $\lambda = -2, \eta = 1$.

(b) *Local part.* The local part approximation of the double layer potential is given by

$$\mathcal{D}_L[\phi] = \sum_{k=1}^{2}(-1)^k \int_{t-\delta}^{t} \frac{x - b_k}{4[\pi(t-\tau)]^{3/2}} e^{-\frac{(x-b_k)^2}{4(t-\tau)}} \phi(b_k, \tau) d\tau. \tag{3.15}$$

[7]Here we assume $\delta = l\triangle t$. We have the Chebyshev coefficients of $\hat{\phi}_n(\tau)$ at each time interval Δt (see section 4). If $\delta < \triangle t$, we use the recurrence (3.10) with modified scaling factors (λ, η) instead of computing the Chebyshev coefficients in the interval $(t - \delta, t + \triangle t - \delta)$.

Let

$$\mathcal{D}_{L_{b_k}}[\phi] = (-1)^k \int_{t-\delta}^{t} \frac{x-b_k}{4[\pi(t-\tau)]^{3/2}} e^{-\frac{(x-b_k)^2}{4(t-\tau)}} \phi(b_k,\tau) d\tau. \tag{3.16}$$

Then

$$\mathcal{D}_L[\phi] = \mathcal{D}_{L_{b_1}}[\phi] + \mathcal{D}_{L_{b_2}}[\phi]. \tag{3.17}$$

Setting $\theta = \frac{t-\tau}{\delta}$ and substituting $\alpha = \frac{(x-b_k)^2}{4\delta}$ we get

$$\mathcal{D}_{L_{b_k}}[\phi] = \frac{1}{2\sqrt{\pi}} \int_0^1 \frac{\sqrt{\alpha}}{\sqrt{\theta^3}} e^{-\frac{\alpha}{\theta}} \phi(b_k, t-\delta\theta)\, d\theta. \tag{3.18}$$

Similarly to the far part case, we compute the Chebyshev coefficients[8] of $\phi(b_k,\tau)$ for $\tau \in (t-\delta, t)$. Then we use the recurrence relations (3.7) together with (3.8) to compute (3.18).

4. The overall algorithm for a stationary boundary. So far we have presented (1) an integral equation formulation for (1.1); (2) a near-far decomposition in time of the volume and layer potentials; and (3) quadratures for convolutions of functions (defined in terms of Chebyshev polynomials) for singular or nearly singular kernels and in particular heat kernels. Next we describe how we can combine these techniques to efficiently solve (1.1) on stationary boundaries. In a nutshell, we compute the volume potentials $u_i(x,t)$ and $u_f(x,t)$ using the formulas (1.5), solve the integral equation (1.7) for the double layer density $\phi(\gamma,t)$, and use (1.6) to compute $u_b(x,t)$.

4.1. Evaluation of the volume potentials. We will need two additional algorithms.

Computing Fourier coefficients for the far part. We need a fast algorithm for computing the sine transforms $\hat{w}$ and $\hat{f}(\tau)$ used in $\mathcal{V}_{0,F}[w]$ and $\mathcal{V}_F[f]$:

$$\hat{w}_n = \int_\Omega w(y)\sin(n\pi y)\, dy; \quad \hat{f}_n(\tau) = \int_\Omega f(y,\tau)\sin(n\pi y)\, dy, \quad n = 1,\dots,p. \tag{4.1}$$

We assume that both f and w are given in regular grid points in space time. If f and w are periodic, then (4.1) can be computed using FFTs. Otherwise, we use the high-order hybrid Gauss trapezoidal rules developed in [1]. The advantage of using those rules is that, except for a few Gaussian nodes, all of the other quadrature nodes are on a regular grid. Then if n_g is the number of Gaussian nodes and M is the number of trapezoidal nodes, the complexity of computing the discretization of (4.1) can be reduced from $\mathcal{O}((M+n_g)p)$ to $\mathcal{O}((M+p)\log M + pn_g)$ using the FFT.

Fast Gauss transform. We need an accurate and fast evaluation of the Gauss transform defined by

$$G_\sigma[\mu](x) = \frac{1}{\sqrt{\pi\sigma}} \int_\Omega e^{-\frac{(x-y)^2}{\sigma}} \mu(y)\, dy. \tag{4.2}$$

[8] If $\delta = l\triangle t$, we need to loop through the l time intervals, each of which has a Chebyshev polynomial representation of $\phi(b_k,\tau)$. On the other hand, if $\delta < \triangle t$, we compute (3.18) from the polynomial representation of $\phi(b_k,\tau)$ in the interval $(t-\triangle t, t)$.

Using quadrature rules to discretize (4.2) and applying the fast Gauss transform (FGT) [7] on the resulting discrete sum, the complexity of computing G_σ at M points in space using M quadrature points is reduced from $\mathcal{O}(M^2)$ to $\mathcal{O}(M)$. But through this approach, uniform high-order convergence would not be possible for smaller values of σ, since the Gaussian becomes sharply peaked. So, to get the expected convergence with optimal complexity, we proceed as follows:

- We divide Ω into M/q_s uniform cells, each of size l_C; q_s is defined below.
- In each cell we use a fixed number of quadrature nodes q_s. One way to choose q_s is to ensure that a Gaussian of support equal to the size of cell can be resolved using q_s points.
- An approximate support of a Gaussian is given by $2r\sqrt{\sigma}$, where r is such that $e^{-r^2} < \epsilon$ and ϵ is the desired accuracy. If $2r\sqrt{\sigma} > l_C$, we use the FGT to compute G_σ at the M target points. Since there are M sources the complexity is $\mathcal{O}(M)$.
- If $2r\sqrt{\sigma} < l_C$ we compute G_σ directly: We truncate the domain of integration from Ω to the support $(x - r\sqrt{\sigma}, x + r\sqrt{\sigma})$ and use q_s quadrature points to discretize the integral. The forcing term at these points is computed using an appropriate high-order interpolation. The complexity is $\mathcal{O}(M)$.

4.1.1. Initial condition volume potential. The kernel $G(x, y, t)$ is approximated by the truncated Fourier series expansion if $t > \delta$ and by the Kelvin expansion otherwise. Hence,

$$u_i(x,t) = \begin{cases} G_{2t}[w](x), & \text{if } t \le \delta, \\ \sum_{n=1}^{p} (2e^{-n^2\pi^2 t}\hat{w}_n)\sin(n\pi x), & \text{otherwise.} \end{cases} \tag{4.3}$$

The transform $G_{2t}[w]$ is evaluated using the aforementioned strategy with a quadrature rule of order q. The inverse NUFFT is used to compute the discrete sum in (4.3) for $t > \delta$.

4.1.2. Distributed forcing volume potential. The evaluation of (1.5) is split into a far part $(0, t-\delta)$ and a local part $(t-\delta, t)$. At time interval $(t, t+\triangle t)$ the evaluation of the far part $\mathcal{V}_F[f]$ involves the following steps:

- Computation of the Chebyshev coefficients for each of $\{\hat{f}_n(\tau)\}_{n=1}^p$ by evaluating (4.1) (using quadrature rules in space) at the Chebyshev nodes belonging to $(t-\delta, t+\triangle t-\delta)$.
- For $n = 1, \ldots, p$, computation of the update $U_n[\hat{f}]$ as defined in (3.12) using Chebyshev recurrences developed in section 3.
- Update of $\{C_n\}_{n=1}^p$ using $C_n(t+\triangle t, \delta) = C_n(t, \delta) + 2U_n[\hat{f}]$.
- Evaluation of the discrete sum $\mathcal{V}_F[f](x, t+\triangle t) = \sum_{n=1}^{p} C_n(t+\triangle t, \delta)\sin(n\pi x)$ at the target points using the inverse NUFFT.

The local part of $u_f(x,t)$ denoted by $\mathcal{V}_L[f]$ is given by

$$\mathcal{V}_L[f](x,t) = \int_{t-\delta}^{t}\int_{\Omega} \frac{e^{-\frac{(x-y)^2}{4(t-\tau)}}}{\sqrt{4\pi(t-\tau)}} f(y,\tau)dy\,d\tau. \tag{4.4}$$

Setting $\sigma^2 = t-\tau$ and noting that, outside the interval $(x-2r\sigma, x+2r\sigma)$, the Gaussian decays exponentially fast to zero, we obtain

$$\mathcal{V}_L[f](x,t) \approx \int_0^{\sqrt{\delta}} g(\sigma)\,d\sigma, \quad \text{where } g(\sigma) = \frac{1}{\sqrt{\pi}}\int_{x-2r\sigma}^{x+2r\sigma} e^{-\frac{(x-y)^2}{4\sigma^2}} f(y, t-\sigma^2)dy. \tag{4.5}$$

By substituting $y = x + 2\xi\sigma$, we get $g(\sigma) = \frac{2\sigma}{\sqrt{\pi}} \int_{-r}^{r} e^{-\xi^2} f(x + 2\xi\sigma, t - \sigma^2) d\xi$; the integrand is smooth, and hence, $g(\sigma)$ is a smooth function in σ. Then, we use a high-order quadrature rule of order q to integrate $g(\sigma)$, i.e., $\mathcal{V}_L[f](x,t)] = \sum_{k=1}^{q} w_k g(\sigma_k)$, where w_k, σ_k are the quadrature weights and nodes, respectively. Notice that $g(\sigma_k) = 2\sigma_k G_{2\sigma_k}[f](t - \sigma_k^2)$ at each σ_k. Hence, the optimal strategy for computing the Gauss transform discussed previously can be used for the accurate computation of $g(\sigma_k)$.

4.2. Computation and evaluation of the boundary double layer potential. To evaluate $u_b(x,t)$, we first need to compute the density $\phi(\gamma, t)$ by solving (1.7). At each time step, we incrementally solve (1.7) for the Chebyshev coefficients of $\phi(b_1, t)$ and $\phi(b_2, t)$. Let $\phi(b_1, n), \phi(b_2, n)$ denote the corresponding nth-order Chebyshev coefficients. Then, for any $\tau \in [t, t + \triangle t]$, we have

$$\phi(b_k, \tau) = \sum_{n=0}^{q-1} \phi(b_k, n) T_n \left(\frac{2}{\triangle t}(\tau - t) - 1 \right), \quad k = 1, 2.$$

Since we require q Chebyshev coefficients at each point on the boundary, we need to solve (1.7) at q collocation nodes $\{t + t_i\}_{i=1}^{q}$. The natural choice of the collocation nodes are the zeros of qth-order Chebyshev polynomial defined in the interval $[t, t + \triangle t]$. Then the transformation between physical and spectral space can be performed using the FFT. In addition, interpolation at the Chebyshev nodes gives rise to a stable approximation of the interpolated function.

The Chebyshev nodes are given by

$$t_i = \frac{\triangle t}{2} \left(1 + \cos \frac{\pi(i-1)}{q} \right), \qquad i = 1, \ldots, q. \tag{4.6}$$

The integral equation (1.7) can be rewritten as

$$\begin{aligned} &-\frac{1}{2}\phi(x, t + t_i) + \int_t^{t+t_i} \int_\gamma \frac{\partial G(x, t + t_i; y, \tau)}{\partial n(y)} \phi(y, \tau) ds(y) d\tau \\ &= u_b(x, t + t_i) - \int_0^t \int_\gamma \frac{\partial G(x, t + t_i; y, \tau)}{\partial n(y)} \phi(y, \tau) ds(y)\, d\tau \quad \forall x \in \gamma. \end{aligned} \tag{4.7}$$

The resulting finite-dimensional linear equation is $K\phi_h = r$, where

$$\phi_h = [\phi(b_1, 0), \ldots, \phi(b_1, q-1),\ \phi(b_2, 0), \ldots, \phi(b_2, q-1)]^T. \tag{4.8}$$

Assuming $\delta \geq \triangle t$, the entries of K are given by $K = \begin{bmatrix} A & B \\ B & A \end{bmatrix}$,

$$\text{where} \quad A_{ij} = -\frac{1}{2} T_{j-1} \left(\frac{2}{\triangle t} t_i - 1 \right), \quad i, j = 1, \ldots, q, \tag{4.9}$$

$$B_{ij} = \int_t^{t+t_i} \frac{b_2 - b_1}{4[\pi(t-\tau)]^{3/2}} e^{-\frac{(b_2 - b_1)^2}{4(t-\tau)}} T_{j-1} \left(\frac{2}{\triangle t}(\tau - t) - 1 \right) d\tau. \tag{4.10}$$

The entries of B can be computed using recurrences on $I_n(\alpha) = \int_0^1 \frac{\sqrt{\alpha}}{\sqrt{\theta^3}} e^{-\frac{\alpha}{\theta}} T_n(\lambda\theta + \eta) d\theta$. The operator K is well-conditioned and can by "inverted" using a matrix-free GMRES.[9] The integration of the history part of double layer on the right-hand side

[9] Given a fixed time step $\triangle t$, we can precompute the inverse K^{-1} once and subsequently solve for ϕ_h by just one matrix vector multiplication $K^{-1}r$. This is can be done efficiently only in 1D and for static boundaries.

of (4.7), denoted by $\mathcal{D}_h$, is split into $(0, t+t_i-\delta)$ and $(t+t_i-\delta, t)$. The first interval integral is

$$\sum_{n=1}^{p} \left(e^{-n^2\pi^2 t_i} C_n(t,\delta) + 2\int_{t-\delta}^{t+t_i-\delta} e^{-n^2\pi^2(t+t_i-\tau)} \hat{\phi}(\tau)\, d\tau \right) \sin(n\pi x). \tag{4.11}$$

We use the local part expansion of the kernel for the second interval. The time integration in both intervals is performed using the product integration rules discussed in section 3. Since the coefficients C_n are updated at each time step, r can be computed with constant work per time step. Then $u_b(x,t)$ can be evaluated at the target M spatial locations using (1.6) and the fast summation method for the double layer described in section 2.

4.3. Summary of the overall scheme for stationary boundaries. The overall algorithm is summarized in Algorithm 1. The output of the algorithm is $u(x,t)$ at the target points. The inputs are

- **parameters:** A fixed time step Δt and an error tolerance ϵ.
- **boundary conditions:** In each time interval $(t, t+\triangle t)$, the boundary data at the collocation nodes $g(b_k, t+t_i), i = 1, \ldots, q$.
- **distributed force:** The forcing f at points belonging to a regular grid of size $M \times N$ imposed on $\Omega \times [0,T]$.
- **initial condition:** w at M regular points on Ω.

Since we use special quadrature points to compute the volume integrals (4.1), f and w are evaluated using high-order interpolation schemes; for example, FFT combined with B-splines can be used [27].

The overall complexity of the algorithm is $\mathcal{O}(MN \log M)$. Below, we discuss the different sources for errors in the final solution.

- Truncating the series expansion for Green's function in the near and far parts. These have been shown to be exponentially convergent with increasing M, N.
- The order of the Chebyshev polynomials used for approximating the density ϕ in each time interval.
- The quadrature rule used for the computing the spatial integrals (4.1) and time integral (4.5).

By approximating ϕ with $(q-1)$th-order Chebyshev polynomials and picking a quadrature rule of order q to perform the space and time integration for the volume potentials, we get a qth-order method. In section 6, we present results that validate our scheme.

5. Moving boundary. Now we will describe the methodology for solving (1.1) for boundaries with a prescribed motion. By linearity, we decompose the solution into three parts: $u = u_i + u_f + u_b$. The three parts correspond to the solutions of an initial-value problem (1.2), an inhomogeneous problem with homogeneous boundary conditions (1.3), and a homogeneous problem with inhomogeneous boundary conditions (1.4). The motion of the boundary $\omega(t)$ is assumed to be contained within the unit box Ω; hence, the solutions $u_i(x,t)$ and $u_f(x,t)$ are given by (1.5). At any point x inside the domain $\omega(t)$ we assume that $u_b(x,t) = \mathcal{D}[\phi](x,t)$. As $G(x,t;y,\tau)$ satisfies the adjoint equation (2.1) and the potential u_b satisfies the heat equation, the initial condition $u_b(x,0) = 0$ is satisfied by construction. To satisfy the prescribed boundary conditions on $\gamma(t)$, we need to compute the jumps in the double layer as $x \to \gamma(t)$. The conditions have been derived before; for example, see [21, 9]. We have included their derivation for completeness. We first discuss the computation of the double layer potential at a point away from the boundary, and then we give the jump conditions.

Algorithm 1. Overall algorithm for stationary boundaries.

INITIALIZATION
Choose p and l using (2.11)
Compute E (3.14) using (3.10)
Compute $\hat{w}_n$ and set $C_n = 2e^{-n^2\pi^2(l-1)\triangle t}\hat{w}_n$, $n = 1 \dots p$ $\mathcal{O}((M+p)\log M)$
Compute t_i using (4.6)
Compute K^{-1} $\mathcal{O}(q^3)$

TIME MARCHING
for $j = 1 : N$ do
  If $j < l$, $l = j$ *first l time steps: $\delta = j\triangle t$*

  SOLVE FOR THE BOUNDARY DENSITY
  loop over boundary nodes
  for $k = 1 : 2$ do
    Loop over time-collocation nodes
    for $i = 1 : q$ do
      $t_c = (j-1)\triangle t + t_i$
      $r_k(i) = g(b_k, t_c) - \mathcal{V}[f](b_k, t_c) - \mathcal{V}_0[w](b_k, t_c)$ $\mathcal{O}(M)$, *using FGT*
      $r_k(i) = r_k(i) - \mathcal{D}_h[\phi](b_k, t_c)$
    end for
  end for
  $\phi_h = K^{-1} r$

  EVALUATION AND UPDATE
  Local part
  $u(x_k, j\triangle t) = \mathcal{V}_L[f](x_k, j\triangle t)$, $k = 1 \dots M$ $\mathcal{O}(M)$
  $u(x_k, j\triangle t) = u(x_k, j\triangle t) + \mathcal{D}_L[\phi](x_k, j\triangle t)$, $k = 1 \dots M$ $\mathcal{O}(Mq)$
  if $j < l$ then
    Initial Condition
    $u(x_k, j\triangle t) = u(x_k, j\triangle t) + G_{j\triangle t}[w](x_k)$, $k = 1 \dots M$ $\mathcal{O}(M)$
  else
    Far part
    for n = 1:p do
      Compute $U_n[\hat{\phi} + \hat{f}]((j-1)\triangle t)$ using (3.12)
      $C_n = e^{-n^2\pi^2\triangle t} C_n + 2U_n[\hat{f} + \hat{\phi}]$
    end for $\mathcal{O}(pq + p)$
    $u(x_k, j\triangle t) += \sum_{n=1}^{p} C_n \sin(n\pi x_k)$, $k = 1 \dots M$ $\mathcal{O}((M+p)\log M)$
  end if
end for

To evaluate the double layer, we split it into a local and a far part: $\mathcal{D}[\phi] = \mathcal{D}_L[\phi] + \mathcal{D}_F[\phi]$. We first discuss the local part computation. We rewrite the local part as $\mathcal{D}_L[\phi] = \mathcal{D}_{L_{b_1}}[\phi] + \mathcal{D}_{L_{b_2}}[\phi]$, with each term being defined as in (3.16) but using the time-dependent boundaries. Let's look at computing $D_{L_{b_2}}[\phi](x,t)$ defined by

$$\mathcal{D}_{L_{b_2}}[\phi] = \int_{t-\delta}^{t} \frac{x - b_2(\tau)}{4[\pi(t-\tau)]^{3/2}} e^{-\frac{(x-b_2(\tau))^2}{4(t-\tau)}} \phi(b_2(\tau), \tau) d\tau. \tag{5.1}$$

Setting $\tau = t - \delta\theta$, $\frac{x - b_2(\tau)}{2\sqrt{\delta}} = \beta(\theta)$, and $\phi(b_2, t - \delta\theta) = \psi(\theta)$ we get

$$\mathcal{D}_{L_{b_2}}[\phi] = \frac{1}{2\sqrt{\pi}} \int_0^1 \frac{\beta(\theta)}{\sqrt{\theta^3}} e^{-\frac{\beta^2(\theta)}{\theta}} \psi(\theta)\, d\theta. \tag{5.2}$$

Unlike the static boundary case, we cannot directly use the recurrences (3.7), (3.8) to compute (5.2) as β is not a constant. Instead, we rewrite (5.2) as

$$\mathcal{D}_{L_{b2}} = \frac{1}{2\sqrt{\pi}} \int_0^1 \frac{e^{-\frac{\beta_0^2}{\theta}}}{\sqrt{\theta^3}} \left\{\beta(\theta) E_\beta(\theta) \psi(\theta)\right\} d\theta, \quad \text{where } E_\beta(\theta) = e^{-\frac{\beta^2 - \beta_0^2}{\theta}}, \tag{5.3}$$

and $\beta_0 = \beta(0)$. The kernel in this expression is similar to the static boundaries case (3.18). Therefore, we first compute the Chebyshev coefficients of the function $\beta E_\beta \psi$ and then use the recurrences (3.7), (3.8) with $\alpha = \beta_0^2$ to compute (5.3).

Remark. Assuming $\beta(\theta)$ is sufficiently smooth, it is easy to see that E_β is also smooth: By expanding $\beta(\theta)$ about $\theta = 0$ using Taylor's series as $\beta(\theta) = \sum_{k=0}^{\infty} \beta_k \theta^k$, we can see that $E_\beta(\theta)$ is of the form $e^{-\sum_{k=0}^{\infty} c_k \theta^k}$.

Similarly, the single layer potential is calculated by rewriting it as

$$\mathcal{S}_{L_{b_2}}[\phi] = \frac{1}{2\sqrt{\pi}} \int_0^1 \frac{e^{-\frac{\beta^2(\theta)}{\theta}}}{\sqrt{\theta}} \psi(\theta)\, d\theta = \frac{1}{2\sqrt{\pi}} \int_0^1 \frac{e^{-\frac{\beta_0^2}{\theta}}}{\sqrt{\theta}} \left\{E_\beta \psi(\theta)\right\} d\theta \tag{5.4}$$

and using the recurrence (3.8). Now let us examine the limit as the evaluation point x approaches the boundary. As $x \to b_2(t)$, the constant term in Taylor's expansion of $\beta(\theta)$ given by $\beta_0 = \frac{x - b_2(t)}{2\sqrt{\delta}}$ approaches zero.

PROPOSITION 5.1. *We define the single layer potential as* $\mathcal{S}_L[f](\beta_0) = \int_0^1 \frac{e^{-\frac{\beta_0^2}{\theta}}}{\sqrt{\theta}} f(\theta)\, d\theta$ *and the double layer potential as* $\mathcal{D}_L[f](\beta_0) = \int_0^1 \beta_0 \frac{e^{-\frac{\beta_0^2}{\theta}}}{\sqrt{\theta^3}} f(\theta)\, d\theta$. *Then* $\mathcal{S}_L[f]$ *is continuous as* $\beta_0 \to 0$, *whereas* $\mathcal{D}_L[f]$ *sustains a jump equal to* $f(0)$.

$$\lim_{\beta_0 \to 0^-} \mathcal{S}_L[f] = \lim_{\beta_0 \to 0^+} \mathcal{S}_L[f] = \int_0^1 \frac{f(\theta)}{\sqrt{\theta}}\, d\theta, \tag{5.5}$$

$$\lim_{\beta_0 \to 0^-} \mathcal{D}_L[f] = -\frac{1}{2} f(0), \quad \lim_{\beta_0 \to 0^+} \mathcal{D}_L[f] = \frac{1}{2} f(0). \tag{5.6}$$

We give a proof of this proposition in Appendix B. In the case of static boundaries, using this proposition, we arrived at the Volterra system (1.7). For moving boundaries, clearly the proposition cannot be applied directly, since β is not constant. Here we write the double layer potential defined in (5.2) as follows:

$$\mathcal{D}_{L_{b_2}} = \frac{1}{2\sqrt{\pi}} \int_0^1 \frac{e^{-\frac{\beta_0^2}{\theta}}}{\sqrt{\theta}} \left\{ \left(\frac{\beta(\theta) - \beta_0}{\theta} \right) E_\beta \psi(\theta) \right\} d\theta + \frac{1}{2\sqrt{\pi}} \int_0^1 \frac{\beta_0 e^{-\frac{\beta_0^2}{\theta}}}{\sqrt{\theta^3}} \left\{E_\beta \psi(\theta)\right\} d\theta. \tag{5.7}$$

Let us define $F_\beta(\theta) = \frac{\beta(\theta) - \beta_0}{\theta}$. Using Taylor's series (and assuming sufficient smoothness), we have $F_\beta(\theta) = \sum_{k=1}^{\infty} \beta_k \theta^{k-1}$. Therefore, $F_\beta(\theta)$ is smooth and independent of β_0 and $\lim_{\theta \to 0} F_\beta(\theta) = \beta_1$, where $\beta_1 = -\frac{\dot{b}(t)}{2\sqrt{\delta}}$. We denote the first integral in (5.7) by $\tilde{\mathcal{S}}_{L_{b_1}}$; it is a single layer potential with constant β_0, and hence it is continuous as $\beta_0 \to 0$. By Proposition 5.1, the second integral approaches $-\frac{1}{2} E_\beta(0) \psi(0)$

as $\beta \to 0^-$. We can show that $\lim_{\theta \to 0_{\beta_0 \to 0}} E_\beta(\theta) = 1$. The Volterra system that we obtain for moving boundaries is given by

$$-\frac{1}{2}\phi(x,t) + \int_0^t \int_{\gamma(\tau)} \frac{\partial G(x,t;y,\tau)}{\partial n(y)} \phi(y,\tau)\, ds(y)\, d\tau = u_b(x,t)\, \forall x \in \gamma(\tau). \tag{5.8}$$

In the case of static boundaries, the Volterra equation (1.7) at the boundary point b_1 at time t can be written as

$$\left(-\frac{1}{2}I + \mathcal{D}_{F_{b_1}} + \mathcal{D}_{b_2}\right)[\phi](b_1,t) = u_b(b_1,t), \tag{5.9}$$

since the local part of the double layer $D_{L_{b_1}}[\phi](b_1,t) \to -\frac{1}{2}\phi(b_1,t)$. The far part does not contribute to the jump in the double layer. From (5.7), we can show that the Volterra equation in the case of moving boundaries at $(b_1(t),t)$ is given by

$$\left(-\frac{1}{2}I + \mathcal{D}_{F_{b_1}} + \tilde{\mathcal{S}}_{L_{b_1}} + \mathcal{D}_{b_2}\right)[\phi](b_1(t),t) = u_b(b_1(t),t). \tag{5.10}$$

We can verify that, as the boundary motion becomes negligible, (5.10) leads to (5.9): $\beta(\theta) \to \beta_0$ and $\lim_{\beta(\theta)\to\beta_0} \tilde{\mathcal{S}}_{L_b}[\phi](b(t),t) = 0$ as the function $F_\beta(\theta) \to 0$ since $\beta_k \to 0$ for all $\{\beta_k\}_{k=1}^\infty$.

We now summarize the changes in the algorithm described in section 4, which are necessary for moving boundary problems. At each time step, we solve (5.8) for the Chebyshev coefficients of $\phi(b_1(t),t)$ and $\phi(b_2(t),t)$ using GMRES. Assuming $\phi(b_2,t)$ is given, our aim is to solve the resulting algebraic system after discretizing the operator equation (5.10) by collocation at nodes t_i. Starting with an initial guess $\{\phi_n(b_1)\}_{n=0}^{q-1}$, we first compute $\phi_0(b_1,t_i)$ at the collocation nodes in time t_i by inverse FFT. Computing the left-hand side of (5.10) involves computing three operators at (b_1,t_i) : the far part $\mathcal{D}_{F_{b_1}}[\phi]$, the local part of the single layer $\tilde{\mathcal{S}}_{L_{b_1}}[\phi]$, and the double layer $\mathcal{D}_{b_2}[\phi]$.

- As with the static case, the far part $\mathcal{D}_{F_{b_1}}[\phi](b_1,t)$ is computed by storing and updating $\{C_n(t,\delta)\}_{n=1}^p$ and evaluating (2.9). The only difference is that the boundaries are time-dependent in the definition of $\hat{\phi}_n(\tau)$.
- To compute $\tilde{\mathcal{S}}_{L_{b_1}}[\phi](b_1,t_i)$, we need to evaluate the function $F_\beta(\theta)E_\beta(\theta)\psi(\theta)$ in (5.7). Let

$$\phi_{\tilde{\mathcal{S}}}(t,\tau) = \frac{b_1(t) - b_1(\tau)}{2(t-\tau)} e^{-\frac{(b_1(t)-b_1(\tau))^2}{4(t-\tau)}} \phi(b_1(\tau),\tau), \tag{5.11}$$

and then

$$\begin{aligned}\tilde{\mathcal{S}}_{L_{b_1}}[\phi](b_1(t_i),t_i) &= \int_{t_i-\delta}^{t_i} G\left(b_1(t_i), b_1(t_i); t_i, \tau\right) \phi_{\tilde{\mathcal{S}}}(t_i,\tau)\, d\tau \\ &= \int_{t_i-\delta}^{t_i} \frac{\phi_{\tilde{\mathcal{S}}}(t_i,\tau)}{\sqrt{4\pi(t_i-\tau)}}\, d\tau.\end{aligned}$$

To compute the Chebyshev coefficients $\{\phi_{\tilde{\mathcal{S}}}(t_i,n)\}_{n=0}^{q-1}$, we evaluate $\phi_{\tilde{\mathcal{S}}}(t_i,\tau)$ at the collocation nodes $\{t_j\}_{j=1}^q$ (which also are the Chebyshev nodes). Note that, when $\tau \to t$, we have $\lim_{\tau\to t} \phi_{\tilde{\mathcal{S}}}(t,\tau) = -\dot{b}_1(t)$. Then we evaluate $\tilde{\mathcal{S}}_{L_{b_1}}[\phi](b_1(t_i),t_i)$ using the recurrence (3.8) with appropriate scaling factors and $\alpha = 0$.

- To compute $\mathcal{D}_{b_2}[\phi]$, we split it into a far and a local part. The local part is computed by rewriting it as in (5.3) with $\beta_0 = \frac{b_1(t_i)-b_2(t_i)}{2\sqrt{\delta}}$.

Once we solve for the Chebyshev coefficients of $\phi(b_1, t)$ and $\phi(b_2, t)$ the potential u_b is evaluated at all target locations using (1.6). The asymptotic complexity and accuracy of the algorithm are the same with the stationary case.

6. Numerical results. In this section we illustrate the accuracy of the quadrature scheme described in section 3 for different kernels. Then we present numerical results for solving the heat equation (1.1).

6.1. Quadratures. We compare the hybrid quadratures of Alpert [1] with the Chebyshev spectral method. First we compute the following integral:

$$\int_0^1 \cos(200x)s(x) + \cos(200x + 0.5)\ dx \tag{6.1}$$

for the functions $s(x) = 0, s(x) = \frac{1}{\sqrt{x}}, s(x) = \log x$. The recurrence relations for computing the moments $I_k = \int_0^1 s(x)T_k(2x-1)dx$ are given in Appendix A. We report relative errors for both the Chebyshev method and the quadrature methods in [1] for convergence orders 4, 8, and 16. See Figure 6.1.

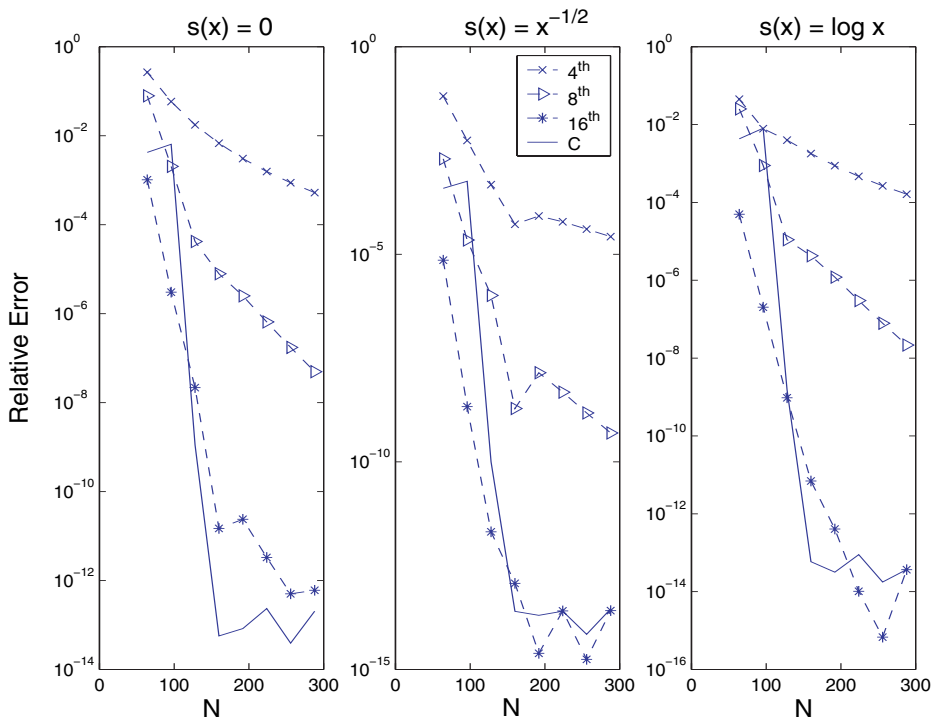


FIG. 6.1. *Plot of the relative errors given in Tables* 6.1, 6.2, *and* 6.3 *w.r.t. p, the number of Chebyshev modes used. As the number of quadrature points is increased, we observe that the quadrature error gets reduced by a constant multiple, whereas in the case of the Chebyshev approximation, the error decays exponentially.*

TABLE 6.1

In this table we report a numerical comparison between our method and the quadrature rules proposed in [1] *for convergence orders* 4, 8, *and* 16. *Relative errors in computing the integral* (6.1) *for the case* $s(x) = 0$, p *indicates the number of Chebyshev coefficients or the total number of points used in the quadrature rule.*

p	4	8	16	Chebyshev
64	2.66e-01	7.85e-02	1.01e-03	4.21e-03
96	5.83e-02	2.03e-03	3.04e-06	6.39e-03
128	1.75e-02	4.18e-05	2.16e-08	1.11e-09
160	6.72e-03	7.87e-06	1.48e-11	5.66e-14
192	3.05e-03	2.50e-06	2.37e-11	8.32e-14
224	1.56e-03	6.50e-07	3.28e-12	2.33e-13
256	8.75e-04	1.74e-07	5.00e-13	3.88e-14
288	5.25e-04	4.95e-08	6.00e-13	2.04e-13

TABLE 6.2

Relative errors in computing the integral (6.1) *for the case* $s(x) = \frac{1}{\sqrt{x}}$.

p	4	8	16	Chebyshev
64	6.42e-02	1.98e-03	7.17e-06	3.87e-04
96	5.53e-03	2.19e-05	2.14e-09	5.73e-04
128	4.74e-04	1.02e-06	2.06e-12	1.00e-10
160	5.27e-05	1.92e-09	1.18e-13	2.50e-14
192	8.35e-05	1.40e-08	2.44e-15	2.01e-14
224	6.09e-05	4.78e-09	2.57e-14	2.53e-14
256	4.04e-05	1.50e-09	1.74e-15	6.99e-15
288	2.66e-05	4.94e-10	2.62e-14	2.88e-14

TABLE 6.3

Relative errors in computing the integral (6.1) *with* $s(x) = \log x$.

p	4	8	16	Chebyshev
64	4.46e-02	2.52e-02	4.96e-05	4.31e-03
96	7.84e-03	8.83e-04	2.03e-07	8.07e-03
128	3.97e-03	1.09e-05	9.53e-10	1.53e-09
160	1.79e-03	4.25e-06	6.93e-12	5.88e-14
192	8.76e-04	1.20e-06	4.04e-13	3.16e-14
224	4.66e-04	2.99e-07	1.01e-14	8.93e-14
256	2.67e-04	7.82e-08	6.79e-16	1.76e-14
288	1.62e-04	2.16e-08	3.65e-14	3.68e-14

We now present numerical results for evaluating the following integral for two different functions $f(x)$:

$$J(\alpha) = \int_0^1 \frac{e^{-\frac{\alpha}{x}}}{\sqrt{x}} f(x)\, dx. \tag{6.2}$$

The relative errors in computing (6.2) using two different methods are presented in Tables 6.4 and 6.5. One is to use *Nyström's method* with the quadrature nodes and weights of the singular kernel $x^{-1/2}$, and the other is to approximate the function $f(x)$ using Chebyshev polynomials and then using the recurrence relations (3.8). In Table 6.4, we can see that the relative error is high in the case of Nyström's method even if we take as many as 512 points. This can be expected as the behavior of the heat kernels is different from $x^{-1/2}$. So we need to develop separate quadrature rules. Developing such quadratures for each value α is not practical. Instead, we can compute $\{I_n(\alpha)\}_{n=1}^{p}$ for each α in $\mathcal{O}(p)$ work using the recurrence relations (3.8). For

TABLE 6.4

Relative errors in computing the double layer heat potential with density $f_1(x) = 1 + x^{10}$ and $\alpha = 2 \times 10^{-4}$. Here N indicates the number of quadrature nodes or the order of Chebyshev polynomials used to approximate the function $f(x)$; q denotes the order of singular quadrature rule used. Observe that the error in the case of quadrature doesn't reduce as $\mathcal{O}(N^{-q})$, whereas in the second case the error reaches machine precision for $N > 10$. Recall that the smaller the α, the more difficult the computation of the heat kernel becomes.

N	4	8	12	16	20	512
$q = 4$	1.87e-01	1.09e-02	8.31e-03	6.36e-03	4.78e-03	3.64e-04
Chebyshev	1.38e-03	8.59e-06	2.17e-16	2.17e-16	2.17e-16	2.17e-16

TABLE 6.5

Relative errors in computing (6.2) *with the function $f_2(x) = \cos(20x)$ and $\alpha = 2 \times 10^{-4}$. The 16th-order singular quadrature rule from* [1] *is used.*

N	16	20	24	28	32	36
$q = 16$	2.94e-01	7.75e-02	7.64e-02	7.53e-02	7.46e-02	7.39e-02
Chebyshev	3.34e-04	2.16e-06	5.75e-09	7.51e-12	4.00e-15	1.20e-15

TABLE 6.6

Relative errors in computing $I(\alpha)$ defined by (6.3) *using recurrences for the functions $f_1(x) = 1 + x^{10}$ and $f_2(x) = \cos(20x)$, for $\alpha = 2 \times 10^{-8}$.*

N	4	8	16	32	64
$f_1(x)$	7.38e-02	3.61e-04	3.75e-16	1.25e-16	1.25e-16
$f_2(x)$	2.62e-01	5.32e-01	2.61e-03	6.84e-14	2.50e-16

TABLE 6.7

We report l_∞–norm errors for the solution of the heat equation with an exact solution given by $u(x,t) = e^{-10^2 t}\sin(10x)$. In this example we solve a problem with Neumann boundary conditions and zero distributed forces. We compute the numerical solution using a direct integral equation formulation that results in a well-conditioned Volterra equation.

$N = M$	2	4	8	16	32	64
$q = 4$	4.62e-02	1.01e-03	5.34e-05	3.03e-06	2.66e-07	2.11e-08
$q = 6$	1.72e-03	7.09e-05	4.15e-06	1.62e-07	1.39e-09	4.94e-11
$q = 8$	4.36e-04	7.86e-06	3.33e-07	1.05e-08	1.71e-11	5.36e-13

the same functions $f_1(x)$ and $f_2(x)$, we evaluate the integrals

$$I(\alpha) = \int_0^1 \frac{\sqrt{\alpha} e^{-\frac{\alpha}{x}}}{\sqrt{x^3}} f(x)\, dx. \tag{6.3}$$

See Table 6.6. In Figure 6.2, we plot the relative errors in computing aforementioned integral operators by the Chebsyshev method.

6.2. The heat equation: Stationary boundaries. In this section we report results for the diffusion problems in domains with static boundaries. In all of the examples, the domain ω is taken to be $[0.4, 0.6]$, $\delta = 10^{-3}$, and the total time $T = 0.1$; N indicates the number of time steps and q the number of Chebyshev polynomials used to approximate the functions in the boundary integrals at each time step. We report the l_∞ error evaluated at M spatial points and N time levels in Tables 6.7 and 6.8 for two different analytic solutions.

6.3. The heat equation on domains with moving boundaries. Here we present numerical results for the solution of the heat equation on domains with prescribed boundary motion. The boundary position, as a function of time, is given by

TABLE 6.8

In this table we report l_∞-norm errors for the heat equation (1.1) problem with an analytic solution given by $u(x,t) = \cos(100x)\cos(200t)$ and a corresponding nonzero distributed force. We consider Dirichlet boundary conditions and an indirect double layer formulation. We use eight quadrature points in computing the volume integrals in (4.1). See Table 6.9 for running times.

N	2	4	8	16	32	64
$q = 4$	7.16e-01	1.28e-04	3.54e-05	2.41e-06	1.51e-07	1.17e-08
$q = 6$	8.27e-01	8.10e-06	7.80e-07	6.72e-08	1.08e-09	2.92e-11
$q = 8$	7.75e-01	3.74e-07	8.38e-08	3.40e-09	2.46e-11	3.34e-13

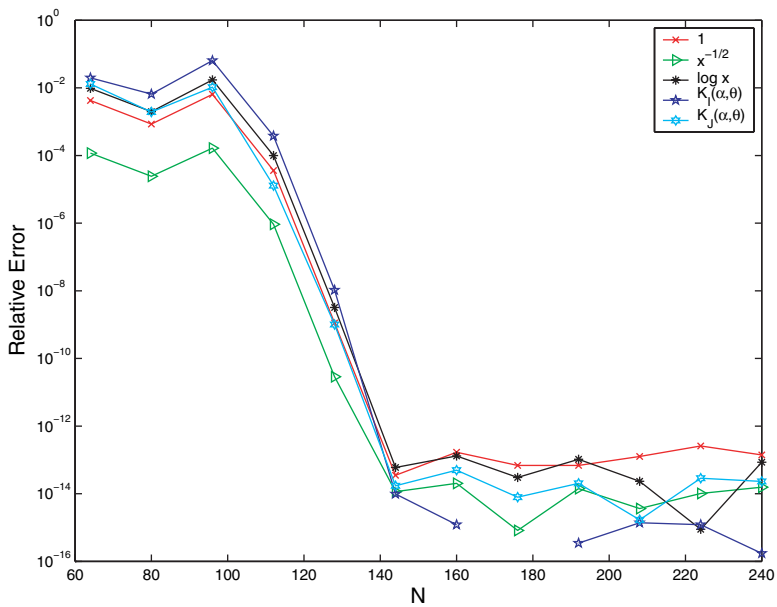


FIG. 6.2. *Relative error plot (logarithmic scale) versus the number of Chebyshev modes N used for approximating the function $f(x) = \cos(200x)$ in computing the integral $I(\alpha) = \int_0^1 K(\alpha, x) f(x)\, dx$ for the kernels $1, x^{-1/2}, \log x$ and for the heat kernels $\frac{\sqrt{\alpha}}{\sqrt{x^3}} e^{-\frac{\alpha}{x}}, \frac{e^{-\frac{\alpha}{x}}}{\sqrt{x}}$, with $\alpha = 2 \times 10^{-4}$. The integration errors follow the approximation error of the function $f(x)$. We can observe that, once $f(x)$ is resolved, the integration is accurate to machine precision—independently of the kernel in which we used in the integration.*

TABLE 6.9

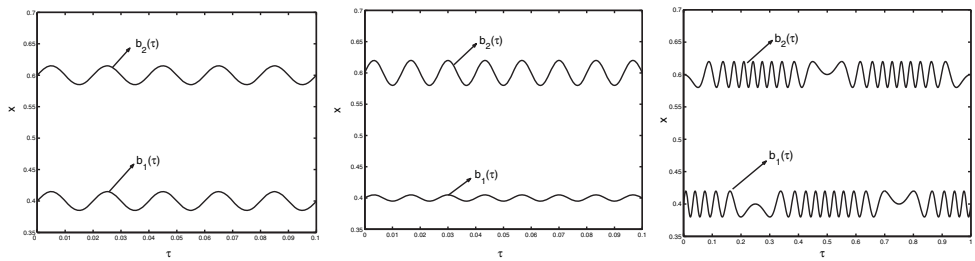
Representative running times for the example in Table 6.8. Notice that only the relative increase is important here, as the code is implemented in MATLAB *and has not been optimized. The following observations can be made: First, the computation time grows linearly with N; second, for a fixed N, the computation time increases by a constant factor as the order q is increased.*

N	2	4	8	16	32	64
$q = 4$	0.8	1.3	2.6	5.3	10.3	21.1
$q = 6$	1.0	1.7	3.3	6.0	12.9	26.3
$q = 8$	1.3	2.3	4.0	8.0	16.0	32.6

$b_1(t) = 0.4 + s_1(t)$ and $b_2(t) = 0.6 + s_2(t)$. We present results for three different motions of the boundary for which the functions $s_1(t)$ and $s_2(t)$ are defined in Table 6.10. The motion of the boundary is depicted in Figure 6.3.

TABLE 6.10
Test cases for the boundary motion.

Case	$s_1(t)$	$s_2(t)$
(i)	$0.01\sin(100\pi t)$	$0.01\sin(100\pi t)$
(ii)	$0.005\sin(150\pi t)$	$0.02\sin(150\pi t)$
(iii)	$0.02\sin[10\pi\sin(2\pi t)]$	$0.02\sin[10\pi\cos(2\pi t)]$



(a) case (i) (b) case (ii) (c) case (iii)

FIG. 6.3. *Plot of test cases for the boundary motion.*

TABLE 6.11
l_∞–norm errors for the solution of Example 1 *with the $\omega(t)$ varying in time as given in case* (i). *See Table* 6.12 *for CPU times.*

$N = M$	2	4	8	16	32	64
$q = 4$	1.85e-01	1.00e-01	8.59e-03	3.28e-04	1.72e-05	9.42e-07
$q = 6$	1.37e-01	9.85e-03	2.08e-04	4.90e-06	3.80e-08	1.47e-09
$q = 8$	6.36e-02	6.13e-04	9.96e-06	3.39e-08	2.62e-10	9.01e-13

TABLE 6.12
CPU times for the example in Table 6.11.

$N = M$	2	4	8	16	32	64
$q = 4$	0.80	1.27	2.60	4.94	10.34	21.47
$q = 6$	1.19	2.32	4.49	8.83	17.84	36.86
$q = 8$	1.92	3.58	7.14	13.71	27.81	56.25

Example 1 (prescribed Neumann conditions). We solve the homogeneous heat equation with Neumann boundary conditions given by $\frac{\partial u}{\partial x}(b_k, t) = 2e^{-4t}\cos(2b_k(t))$ and an initial condition given by $u(x, 0) = \sin(2x)$. For this problem the exact solution is given by $u(x, t) = e^{-4t}\sin(2x)$. We report l_∞-norm errors of our numerical approximation scheme in Tables 6.11, 6.13, and 6.14. We have used a direct formulation that results in a well-conditioned integral equation. We solve this problem for the three cases of boundary motion. At each time step, we sample the numerical and exact solution at a uniformly spaced spatial grid. We report the maximum of the absolute error values over all time steps and spatial points. We observe optimal convergence rates for all three cases.

Example 2 (prescribed Dirichlet data). In this second example we solve a heat equation problem with distributed forces and prescribed Dirichlet boundary condi-

TABLE 6.13
l_∞–norm errors for the solution of Example 1 *with the $\omega(t)$ varying in time as given in case* (ii).

$N = M$	2	4	8	16	32	64
$q = 4$	1.31e+00	1.00e+00	9.83e-02	4.81e-03	2.50e-04	1.35e-05
$q = 6$	3.00e+00	1.42e-01	7.39e-03	2.13e-04	2.37e-06	9.67e-08
$q = 8$	6.23e-01	3.11e-02	7.22e-04	7.20e-06	7.37e-08	2.06e-10

TABLE 6.14
l_∞–norm errors for the solution of Example 1 *with the $\omega(t)$ varying in time as given in case* (iii)*, with $T = 1$.*

$N = M$	16	32	64	128	256	512
$q = 4$	1.49e+000	4.09e-02	7.65e-03	2.44e-04	1.42e-05	5.98e-07
$q = 6$	6.93e-02	3.71e-03	2.02e-04	2.21e-06	1.64e-08	6.48e-10
$q = 8$	8.73e-03	8.07e-04	2.70e-06	5.76e-08	2.79e-10	5.51e-13

TABLE 6.15
l_∞–norm errors for the solution of Example 2 *with the $\omega(t)$ varying in time as given in case* (i).

N	2	4	8	16	32	64
$q = 4$	2.19e-02	5.24e-03	2.20e-03	5.55e-05	4.54e-06	4.15e-07
$q = 6$	2.29e-03	1.54e-03	1.05e-04	2.50e-06	5.39e-08	4.33e-10
$q = 8$	3.27e-03	5.04e-04	8.82e-06	9.07e-08	1.28e-10	7.83e-13

TABLE 6.16
l_∞–norm errors for the solution of Example 2 *with the $\omega(t)$ varying in time as given in case* (ii).

N	4	8	16	32	64	128
$q = 4$	2.04e-02	1.58e-02	4.94e-04	3.47e-05	2.71e-06	8.48e-08
$q = 6$	1.29e-02	3.36e-03	2.56e-05	8.92e-07	1.00e-08	1.64e-10
$q = 8$	2.30e-03	4.54e-04	2.49e-06	6.40e-09	3.85e-11	2.54e-13

TABLE 6.17
l_∞–norm errors for the solution of Example 2 *with the $\omega(t)$ varying in time as given in case* (iii)*; the total time here is $T = 1$.*

N	32	64	128	256	512	1024
$q = 4$	2.54e-01	4.85e-03	5.52e-04	6.69e-05	6.41e-06	4.21e-07
$q = 6$	1.01e-01	7.57e-04	3.54e-05	7.14e-07	1.23e-08	2.10e-10
$q = 8$	2.07e-02	3.67e-05	1.85e-06	2.09e-09	1.24e-11	4.28e-14

tions:

$$\frac{\partial u}{\partial t} = \triangle u + 4\cos(2x)\cos(100t) - 100\cos(2x)\sin(100t) \quad \text{in } \omega(t), \tag{6.4}$$

$$\text{and} \quad u(x,0) = \cos(2x),\ u(b_k(t),t) = \cos(2b_k(t))\cos(100t). \tag{6.5}$$

The exact solution for this problem is given by $u(x,t) = \cos(2x)\cos(100t)$. We report l_∞ errors, computed as in the case of the other examples, in Tables 6.15, 6.16, and 6.17. Again, we observe optimal convergence rates.

7. Conclusions. We have presented an extension of the Greengard and Strain algorithm [6] for the fast solution of the heat equation with moving boundaries. Our main contribution is the introduction of the product integration quadrature and a

scheme for moving boundaries. Our approach results in a high-order scheme in both space and time. We presented results for the Neumann and Dirichlet problems with distributed forces. In all cases the accuracy and efficiency of the proposed scheme is verified—even in the case of highly oscillatory boundary motions. Optimal complexity has been achieved by using the fast Gauss transform and nonuniform FFT. Also, our method can be extended to variable-coefficient and nonlinear problems by introducing a volume density for which we have to solve.

A shortcoming of our method is that it has a complex implementation. A second, more important drawback is that high-order accuracy can be achieved for sufficiently smooth data only (forces, boundary conditions, and boundary motion). For certain applications, in particular industrial ones, this can be a quite restrictive assumption. For general nonsmooth data the high-order properties of our scheme will be lost.

We are currently working on extending this work to higher dimensions. As we mentioned in the introduction, all algorithmic choices extend to higher dimensions: the product quadrature rules, the Chebyshev approximation (using adaptive, tensor products), the kernel splitting for the local and far parts, and the approximations in the case of prescribed boundaries. The most significant complication is the fact that the boundary of the target domain becomes an infinite-dimensional manifold. Boundary convolution with the heat kernel requires additional algorithmic machinery. Although one can outline the extensions of the method in higher dimensions, such a task is difficult when one wants to preserve work optimality and high accuracy. Finally, the assumption that f and w are defined everywhere is restrictive; in most practical applications the data will be defined only on a regular grid or at random points. Smooth extensions and design of special quadrature rules for such data are not trivial. We are currently working on addressing some of these issues, and we will report our work in a future paper.

Appendix A. Here we give recurrence relations to compute $I_k = \int_0^1 \sigma(\theta) T_k(2\theta - 1) d\theta$ for the singular kernels $\sigma_1(\theta) = \theta^{-\gamma}$ for any $\gamma \in (-\infty, 1)$ and $\sigma_2(\theta) = \log(\theta)$. These recurrences are unconditionally stable.

$$I_n = -\frac{n}{n+1-\gamma}\left(\frac{2}{n(n-2)} + 2I_{n-1} + \frac{n-3+\gamma}{n-2} I_{n-2}\right) \quad n > 2 \quad \text{for } \sigma_1(\theta). \tag{A.1}$$

$$I_n = \frac{n}{n+1}\left(C_n - 2I_{n-1} - \frac{n-3}{n-2} I_{n-2}\right) \quad n > 2 \quad \text{for } \sigma_2(\theta), \tag{A.2}$$

$$\text{where} \quad C_n = \begin{cases} 0 & n \text{ is odd,} \\ -\frac{6}{n^4 - 4n^3 + n^2 + 6n} & \text{otherwise.} \end{cases}$$

Appendix B.

Proof of the direct formulation. The Leibniz rule for differentiating the volume integral, also called Reynold's transport theorem, states that, for any function $q(x,t)$ defined in $\Omega(t)$,

$$\frac{d}{dt}\int_{\Omega(t)} q d\Omega = \int_{\Omega(t)} \frac{\partial q}{\partial t} d\Omega + \int_{\gamma(t)} q(v.n) d\gamma,$$

where v is the velocity of the boundary. Substituting $q = Gu$, we get

$$\begin{aligned}\frac{d}{dt}\int_{\Omega(t)} Gu d\Omega &= \int_{\Omega(t)} \frac{\partial G}{\partial t}u + G\frac{\partial u}{\partial t} + \int_{\gamma(t)} Gu(v\cdot n)d\gamma \\ &= \int_{\Omega(t)} \frac{\partial G}{\partial t}u + G(\triangle u + b) + \int_{\gamma(t)} Gu(v\cdot n) \\ &= \int_{\Omega(t)} \left(\frac{\partial G}{\partial t} + \triangle G\right)u + Gb + \int_{\gamma(t)} \left(G\frac{\partial u}{\partial n} - u\frac{\partial G}{\partial n}\right) + Gu(v\cdot n).\end{aligned}$$

Integrating both sides from 0 to t and using $G(x,y,t,t) = 0$ for any t, we get

$$-\int_{\Omega_0} Gw = \int_0^t \int_{\Omega(\tau)} [-\delta(x,t)u + Gb] + \int_0^t \int_{\gamma(\tau)} \left(G\frac{\partial u}{\partial n} - u\frac{\partial G}{\partial n}\right) + Gu(v\cdot n),$$

$$\Rightarrow u(x,t) = \int_{\Omega_0} Gw + \int_0^t \int_{\Omega(\tau)} Gb \; + \int_0^t \int_{\gamma(\tau)} \left(G\frac{\partial u}{\partial n} - u\frac{\partial G}{\partial n}\right) + Gu(v\cdot n). \tag{B.1}$$

Jump conditions: (a) Defining $I(\beta_0) = \int_0^1 \frac{e^{-\frac{\beta_0^2}{\theta}}}{\sqrt{\theta}} f(\theta)\, d\theta$, we have to prove:

$$\lim_{\beta_0\to 0^-} I(\beta_0) = \lim_{\beta_0\to 0^+} I(\beta_0) = \int_0^1 \frac{1}{\sqrt{\theta}} f(\theta)\, d\theta. \tag{B.2}$$

Proof. Consider the sequence of functions g_n defined as $g_n = e^{-\frac{\beta_n^2}{\theta}}$ with the sequence β_n chosen such that $\lim_{n\to\infty} \beta_n = 0$. Clearly $|g_n| \le 1$ and $g_n \to 1$ pointwise almost everywhere. The result follows by Lebesgue's dominated convergence theorem.

(b) Assuming that $f(\theta)$ is uniformly continuous in $\theta \in (0,1)$, we need to prove:

$$\begin{gathered}\lim_{\beta_0\to 0^+} I(\beta_0) = \frac{1}{2}f(0); \\ \lim_{\beta_0\to 0^-} I(\beta_0) = -\frac{1}{2}f(0), \quad \text{where } I(\beta_0) = \frac{\beta_0}{2\sqrt{\pi}}\int_0^1 \frac{e^{-\frac{\beta_0^2}{\theta}}}{\sqrt{\theta^3}} f(\theta)\, d\theta. \qquad \square\end{gathered} \tag{B.3}$$

Proof. We split the integral into two parts: $I(\beta_0) = I_1(\beta_0) + I_2(\beta_0)$. The integrals I_1 and I_2 are defined as follows:

$$I_1 = \frac{\beta_0 f(0)}{2\sqrt{\pi}} \int_0^1 \frac{e^{-\frac{\beta_0^2}{\theta}}}{\sqrt{\theta^3}}\, d\theta = \frac{\beta_0}{2\,|\beta_0|}\mathrm{Erfc}(\beta_0) f(0). \tag{B.4}$$

$$\therefore \lim_{\beta_0\to 0^-} I_1(\beta_0) = -\frac{1}{2}f(0) \quad \text{and} \quad \lim_{\beta_0\to 0^+} I(\beta_0) = \frac{1}{2}f(0). \tag{B.5}$$

$$I_2 = \frac{1}{2\sqrt{\pi}} \int_0^1 \frac{e^{-\frac{\beta_0^2}{\theta}}}{\sqrt{\theta^3}} \left(f(\theta) - f(0)\right) d\theta = \frac{1}{\sqrt{\pi}} \int_{\beta_0}^\infty e^{-\sigma^2} \left(f\left(\frac{\beta_0^2}{\sigma^2}\right) - f(0)\right), \tag{B.6}$$

by the change of variables $\sigma = \frac{\beta_0}{\sqrt{\theta}}$. Now breaking the interval of integration into two parts $\left(\beta_0, \sqrt{\beta_0}\right)$ and $\left(\sqrt{\beta_0}, \infty\right)$, we see that as $\beta_0 \to 0$ the integral of the first interval vanishes. The integral of the second interval $\sigma \ge \sqrt{\beta_0} \Rightarrow |\frac{\beta_0^2}{\sigma^2}| \le |\beta_0|$. Since $f(\theta)$ is uniformly continuous, we can choose δ such that $|f(\frac{\beta_0^2}{\sigma^2}) - f(0)| < \epsilon$ whenever $|\beta_0| < \delta$. Thus,

$$\lim_{\beta_0\to 0} |I_2(\beta_0)| < \lim_{\epsilon\to 0} \frac{\epsilon}{2\sqrt{\pi}} \int_{\sqrt{\beta_0}}^\infty e^{-\sigma^2}\, d\sigma = 0. \qquad \square \tag{B.7}$$

REFERENCES

[1] B. K. Alpert, *Hybrid Gauss-trapezoidal quadrature rules*, SIAM J. Sci. Comput., 20 (1999), pp. 1551–1584.

[2] K. Brattkus and D. I. Meiron, *Numerical simulations of unsteady crystal growth*, SIAM J. Appl. Math., 52 (1992), pp. 1303–1320.

[3] C. W. Clenshaw and A. R. Curtis, *A method for numerical integration on an automatic computer*, Numer. Math., 2 (1960), pp. 197–205.

[4] A. Dutt and V. Rokhlin, *Fast Fourier transforms for nonequispaced data*, SIAM J. Sci. Comput., 14 (1993), pp. 1368–1393.

[5] L. Greengard and P. Lin, *Spectral approximation of the free–space heat kernel*, Appl. Comput. Harmon. Anal., 9 (2000), pp. 83–97.

[6] L. Greengard and J. Strain, *A fast algorithm for the evaluation of heat potentials*, Comm. Pure Appl. Math., XLIII (1990), pp. 949–963.

[7] L. Greengard and J. Strain, *The fast Gauss transform*, SIAM J. Sci. Comput., 12 (1991), pp. 79–94.

[8] T. Hasegawa and T. Torii, *An automatic quadrature for cauchy principal value integrals*, Math. Comp., 194 (1991), pp. 741–754.

[9] J. Huang, M.-C. Lai, and Y. Xiang, *An integral equation method for epitaxial step-flow growth simulations*, J. Comput. Phys., 216 (2006), pp. 724–743.

[10] M. T. Ibanez and H. Power, *An efficient direct BEM numerical scheme for phase change problems using Fourier series*, Comput. Methods Appl. Mech. Engrg., 191 (2002), pp. 2371–2402.

[11] J. Ma, V. Rokhlin, and S. Wandzura, *Generalized Gaussian quadrature rules for systems of arbitrary functions*, SIAM J. Numer. Anal., 33 (1996), pp. 971–996.

[12] S. Kapur and V. Rokhlin, *High-order corrected trapezoidal quadrature rules for singular functions*, SIAM J. Numer. Anal., 34 (1997), pp. 1331–1356.

[13] R. Kress, *Linear Integral Equations*, Appl. Math. Sci., Springer, New York, 1999.

[14] P. P.-Y. Lin, *On the Numerical Solution of the Heat Equation in Unbounded Domains*, Ph.D. thesis, New York, NY, 1993.

[15] R. Piessens and M. Branders, *Numerical-solution of integral-equations of mathematical physics, using Tschebyscheff polynomials*, J. Comput. Phys., 21 (1976), pp. 178–196.

[16] R. Piessens, *Computing integral transforms and solving integral equations using Chebyshev polynomial approximations*, J. Comput. Appl. Math., 121 (2000), pp. 113–124.

[17] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, New York, NY, 1992.

[18] T. J. Rivlin, *Chebyshev Polynomials*, Wiley–Interscience, New York, 1990.

[19] V. Rokhlin, *End-point corrected trapezoidal quadrature rules for singular functions*, Comput. Math. Appl., 20 (1990), pp. 51–62.

[20] J. A. Sethian and J. Strain, *Crystal-growth and dendritic solidification*, J. Comput. Phys., 98 (1992), pp. 231–253.

[21] J. Strain, *Linear-stability of planar solidification fronts*, Phys. D, 30 (1988), pp. 297–320.

[22] J. Strain, *Fast potential theory.* II. *Layer potentials and discrete sums*, J. Comput. Phys., 99 (1992), pp. 251–270.

[23] J. Strain, *Fast adaptive methods for the free–space heat equation*, SIAM J. Sci. Comput., 15 (1994), pp. 185–206.

[24] L. N. Trefethen, *Spectral Methods in Matlab*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2000.

[25] J. Wimp, *Computation with Recurrence Relations*, Pitman, Boston, 1984.

[26] N. Yarvin and V. Rokhlin, *Generalized Gaussian quadratures and singular value decompositions of integral operators*, SIAM J. Sci. Comput., 20 (1998), pp. 699–718.

[27] L. Ying, G. Biros, and D. Zorin, *A high-order* 3*D boundary integral equation solver for elliptic PDEs in smooth domains*, J. Comput. Phys., 219 (2006), pp. 247–275.

[28] M. Zerroukat and L. C. Wrobel, *A boundary element method for multiple moving boundary problems*, J. Comput. Phys., 138 (1997), pp. 501–519.

SIAM J. SCI. COMPUT.
Vol. 29, No. 6, pp. 2607–2620
© 2007 Society for Industrial and Applied Mathematics

IMPLICIT SCHEME FOR HYPERBOLIC CONSERVATION LAWS USING NONOSCILLATORY RECONSTRUCTION IN SPACE AND TIME*

KARTHIKEYAN DURAISAMY† AND JAMES D. BAEDER‡

Abstract. The efficiency of high order accurate schemes for the solution of unsteady hyperbolic conservation laws is adversely affected by time-step restrictions that arise from monotonicity requirements. When applied to the solution of problems involving discontinuities, these restrictions render conventional high order implicit time integration schemes impractical. In the present study, a new single step second order implicit scheme that uses nonoscillatory reconstruction in space and time is presented. Both the spatial and temporal limiters are dependent on the evolving solution, and this nonlinearity allows for a circumvention of total variation diminishing bounds. Numerical results on model scalar and vector hyperbolic equations suggest that the scheme holds promise in achieving accurate and unconditionally nonoscillatory solutions.

Key words. high resolution schemes, implicit schemes, time limiting

AMS subject classifications. 65M06, 35L65

DOI. 10.1137/070683271

1. Introduction.

The present study is concerned with the solution of hyperbolic conservation equations using high order accurate implicit schemes. Hyperbolic conservation laws allow for discontinuities, and hence a good numerical solution scheme should be able to accurately represent such features without generating spurious oscillations. Over the past few decades, much research has focused on designing high order accurate nonoscillatory numerical schemes. Spatial discretizations like the uniformly accurate (UNO) schemes [1], the essentially nonoscillatory (ENO) and weighted ENO (WENO) [2] schemes, and the monotonicity preserving (MP) [3] schemes have been successfully used with high order explicit Runge–Kutta methods (for time integration) in the solution of a variety of hyperbolic conservation laws.

The stability of explicit schemes is governed by the CFL condition, placing a restriction on the allowable time-step size. Hence these methods could become inefficient when the hyperbolic system is highly stiff or when the spatial mesh is largely nonuniform. In such cases, it is desirable to use implicit schemes for which the time step is usually limited by accuracy and not stability. Implicit schemes will also be beneficial for unsteady flows in which the dominant time scales are much larger than the characteristic-based time scales (an example is the flow around a pitching airfoil). Also, implicit time integration may be preferred in practical computations when the time-step size required to achieve the desired accuracy may be several times larger than the explicit limit. However, ensuring nonoscillatory behavior of linear implicit high order time integration schemes imposes severe time-step restrictions (see Gottlieb, Shu, and Tadmor [4]). These restrictions are a result of nonlinear stability constraints, a well-known measure of which is the total variation diminishing con-

*Received by the editors February 21, 2007; accepted for publication (in revised form) March 6, 2007; published electronically October 31, 2007.
http://www.siam.org/journals/sisc/29-6/68327.html
†Department of Aerospace Engineering, University of Maryland, College Park, MD 20742. Current address: Department of Aerospace Engineering, University of Glasgow, Scotland (dkarthik@eng.gla.ac.uk).
‡Department of Aerospace Engineering, University of Maryland, College Park, MD 20742.

TABLE 1
TVD limits for some existing implicit schemes.

Method[1]	State update formula	$(\Delta t)_{imp}/(\Delta t)_{exp}$
Imp. Euler	$u^{n+1} = u^n + \Delta t L(u^{n+1})$	∞
Imp. Trap	$u^{n+1} = u^n + \frac{\Delta t}{2}[L(u^n) + L(u^{n+1})]$	2.0
Imp. BDF2	$u^{n+1} = \frac{3\Delta t L(u^{n+1}) - 4u^n + u^{n-1}}{2}$	0.5
SDIRK-2 ($\gamma = \frac{2-\sqrt{2}}{2}$)	$u^{(1)} = u^n + \Delta t \gamma L(u^{(1)})$ $u^{n+1} = u^n + \Delta t[(1-\gamma)L(u^{(1)}) + \gamma L(u^{n+1})]$	2.41

[1]Imp. BDF2: Implicit second order backwards difference; SDIRK-2: Singly diagonally implicit Runge–Kutta method (two stages) [6].

dition (TVD) [5]. For a given spatial discretization, Table 1 shows the ratio of the maximum allowable time step (from a TVD viewpoint) of a few implicit schemes to that of explicit schemes (specifically, the explicit Euler method). It is evident that linear high order implicit schemes become impractical in the solution of flow fields with discontinuities since the allowable time steps are only slightly higher than that of explicit schemes.

In an attempt to overcome this time-step barrier, Duraisamy, Baeder, and Liu [7] proposed a class of nonlinear implicit time integration schemes in a method-of-lines framework. In this method, a convex combination of a first and a second order implicit scheme is used, and the coefficients of this combination are changed such that the time accuracy locally drops to first order near discontinuities. Forth [8] presented a scheme for scalar equations that switches between the implicit Euler and second order backwards difference methods in different parts of the domain. Both of the aforementioned approaches yielded nominally nonoscillatory schemes but turned out to be highly diffusive at large CFL numbers. In this work, an implicit scheme that is accurate and nonoscillatory at large time steps is presented.

2. Formulation. Consider a scalar conservation law of the form

$$\frac{\partial u(x,t)}{\partial t} + \frac{\partial f(u(x,t))}{\partial x} = 0 \tag{1}$$

with suitable initial and boundary conditions. To discretize this equation, space is divided into cells $[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}]$, and time is divided into intervals $[t^n, t^{n+1}]$ as depicted in Figure 1. The integral form of (1) in the control volume $[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}] \times [t^n, t^{n+1}]$ is given by

$$\int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} [u(x,t^{n+1}) - u(x,t^n)]dx = -\int_{t^n}^{t^{n+1}} [f(u(x_{j+\frac{1}{2}},t)) - f(u(x_{j-\frac{1}{2}},t))]dt.$$

This can then be written in the *conservation form*

$$\bar{u}_j^{n+1} = \bar{u}_j^n - \tau(\hat{f}_{j+\frac{1}{2}} - \hat{f}_{j-\frac{1}{2}}), \tag{2}$$

where

$$\bar{u}_j^n = \frac{1}{\Delta x}\int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} u(x,t^n)dx \quad \text{and} \quad \hat{f}_{j+\frac{1}{2}} = \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} f(u(x_{j+\frac{1}{2}},t))dt \tag{3}$$

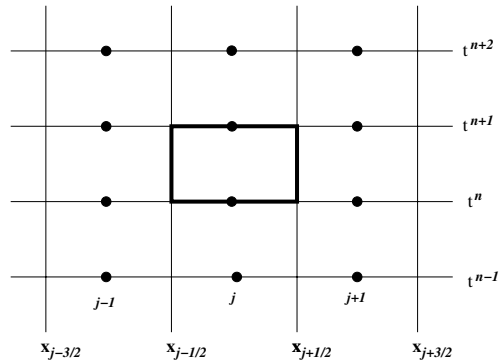


FIG. 1. *Schematic of discretization in time and space.*

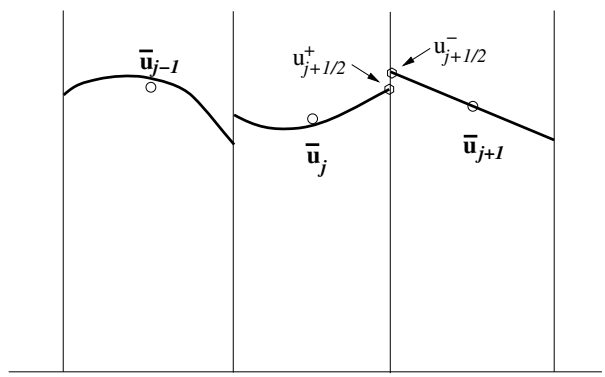


FIG. 2. *Spatial reconstruction.*

with $\Delta x = x_{j+\frac{1}{2}} - x_{j-\frac{1}{2}}$, $\Delta t = t^{n+1} - t^n$, and $\tau = \frac{\Delta t}{\Delta x}$. By defining a suitable approximation to the flux integral (3), numerical schemes can be constructed to an arbitrary order of accuracy. Higher order accurate schemes can be constructed using multiple stages or multiple steps or by increasing the stencil size. In this work, we are concerned with Godunov-type MUSCL methods [9]. In this method, the flux $f(u(x_{j+\frac{1}{2}}))$ at a cell interface is evaluated by the following:

(1) Defining *reconstructed* values of the solution to the left (+) and right (−) of the interface (Figure 2): These values are usually computed using monotone interpolation of the cell-average values.

(2) Using an appropriate upwind solver treating the left and right states as defining a Riemann problem: For instance, the Roe upwind solver would yield $f(u(x_{j+\frac{1}{2}})) = \frac{1}{2}(f(u^-_{j+\frac{1}{2}}) + f(u^+_{j+\frac{1}{2}})) - \frac{1}{2}|a|(u^-_{j+\frac{1}{2}} - u^+_{j+\frac{1}{2}})$, where a is an approximation to the interface flux Jacobian.

For the new scheme, $\hat{f}_{j+\frac{1}{2}}$ is approximated by

$$\begin{aligned}\hat{f}_{j+\frac{1}{2}} &= \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} f(u(x_{j+\frac{1}{2}}, t))dt \\ &= f(u(x_{j+\frac{1}{2}}, t^{n+\frac{1}{2}})) + O(\Delta t^2).\end{aligned}$$

Second order accuracy is achieved by considering the midpoint in time. To define the

interface values, a Taylor series approximation is used:

$$u(x,t) = \bar{u}(x_j, t^{n+1}) + \frac{\partial u}{\partial x}(x_j, t^{n+1})(x - x_j) + \frac{\partial u}{\partial t}(x_j, t^{n+1})(t - t^{n+1}) + O(\Delta x^2, \Delta t^2).$$

Since the Taylor series is centered about the new time-level t^{n+1}, the resulting numerical scheme would be implicit in time. Similar approaches with series centered at time-level t^n abound in the literature (see, for instance, [10]). The usual practice is to approximate the time derivative using the governing equation. For instance,

$$\frac{\partial u}{\partial t}(x_j, t^n) = -\frac{\partial f}{\partial u}(x_j, t^n)\frac{\partial u}{\partial x}(x_j, t^n).$$

High order accurate schemes can be constructed by using higher order terms in the Taylor series and replacing the time derivative terms by equivalent spatial derivatives. The present scheme differs in the sense that it is implicit and the time derivative is not approximated by spatial derivatives. This allows a definition of the reconstructed value in terms of spatial and temporal differences. The interface value (to the left of the interface) would then be defined as

$$u^{+}{}_{j+\frac{1}{2}}^{n+\frac{1}{2}} = \bar{u}_j^{n+1} + \frac{(\Delta_x)_{j+1/2}^{n+1}}{2} - \frac{(\Delta_t)_j^{n+1/2}}{2}, \tag{4}$$

where $(\Delta_x)_{j+1/2}^{n+1} = \bar{u}_{j+1}^{n+1} - \bar{u}_j^{n+1}$ and $(\Delta_t)_j^{n+1/2} = \bar{u}_j^{n+1} - \bar{u}_j^n$.

Figure 3 shows results for the new method as applied to the linear convection equation $u_t + u_x = 0$ with $u(x,0) = \sin^4(x/2)$ and periodic boundary conditions. A CFL number $\sigma = \Delta t/\Delta x = 3.5$ is used. The shaded profile is the exact evolution of the solution at the half–time step. It is evident that the addition of the time reconstruction greatly improves the final solution when compared to an implicit Euler method which can be viewed as using a purely spatial reconstruction.

It has to be mentioned that the current scheme gives an exact solution to the linear advection equation at $\sigma = 1.0$ *and* $\sigma = 2.0$. This is especially interesting because implicit time integration (in a method-of-lines framework) is typically characterized by large diffusive errors at modest CFL numbers.

3. Monotonicity constraints. The reconstruction defined in (4) is not guaranteed to be monotone in the presence of discontinuities, and hence limiters are introduced as follows:

$$u^{+}{}_{j+\frac{1}{2}}^{n+\frac{1}{2}} = \bar{u}_j^{n+1} + \phi_j \frac{(\Delta_x)_{j+1/2}^{n+1}}{2} - \psi_j \frac{(\Delta_t)_j^{n+1/2}}{2}.$$

The design of the spatial ($\phi_j(r_j)$) and temporal ($\psi_j(s_j)$) limiters will be guided by the TVD condition, and these limiters are functions of $r_j = \frac{(\Delta_x)_{j-1/2}^{n+1}}{(\Delta_x)_{j+1/2}^{n+1}}$ and $s_j = \frac{(\Delta_t)_{j+1}^{n+1/2}}{(\Delta_t)_j^{n+1/2}}$, respectively.

Consider the linear advection equation $u_t + au_x = 0$, with $a > 0$. From this point forward, the overbar for cell-averaged quantities and superscripts for the spatial $(n+1)$ and temporal $(n+1/2)$ differences are dropped for clarity. For the new scheme,

$$\hat{f}_{j+\frac{1}{2}} = a\left[u_j^{n+1} + \phi_j \frac{(\Delta_x)_{j+1/2}}{2} - \psi_j \frac{(\Delta_t)_j}{2}\right]. \tag{5}$$

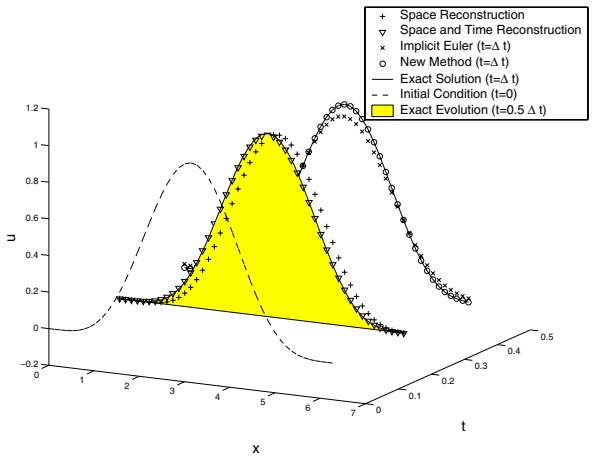


FIG. 3. *Linear advection equation with* $\sigma = 3.5$.

Therefore,

$$\begin{aligned} u_j^{n+1} &= u_j^n - \tau(\hat{f}_{j+\frac{1}{2}} - \hat{f}_{j-\frac{1}{2}}) \\ &= u_j^n - \sigma\Bigg[(u_j^{n+1} - u_{j-1}^{n+1}) \\ &\quad + \frac{1}{2}\left(\frac{\phi_j}{r_j} - \phi_{j-1}\right)(\Delta_x)_{j-1/2} - \frac{1}{2}\left(\psi_j - \frac{\psi_{j-1}}{s_{j-1}}\right)(\Delta_t)_j\Bigg], \end{aligned}$$

where $\sigma = a\frac{\Delta t}{\Delta x}$. Rearranging terms,

$$\left[1 + \frac{\sigma}{2}\left(\frac{\psi_{j-1}}{s_{j-1}} - \psi_j\right)\right](u_j^{n+1} - u_j^n) = -\sigma\left[1 + \frac{1}{2}\left(\frac{\phi_j}{r_j} - \phi_{j-1}\right)\right](\Delta_x)_{j-1/2}.$$

Finally, this gives

$$u_j^{n+1} + \sigma\left[\frac{1 + \frac{1}{2}(\frac{\phi_j}{r_j} - \phi_{j-1})}{1 + \frac{\sigma}{2}(\frac{\psi_{j-1}}{s_{j-1}} - \psi_j)}\right](\Delta_x)_{j-1/2} = u_j^n. \tag{6}$$

Casting the scheme in this form, the above scheme is TVD (see appendix) if

$$\infty > \frac{1 + \frac{1}{2}(\frac{\phi_j}{r_j} - \phi_{j-1})}{1 + \frac{\sigma}{2}(\frac{\psi_{j-1}}{s_{j-1}} - \psi_j)} \geq 0. \tag{7}$$

For $a < 0$, a similar expression involving $j+1$ is obtained. For the nonlinear case $u_t + f(u)_x = 0$, a similar condition involving a local CFL-like number is obtained. Both these expressions and the corresponding TVD conditions are provided in the appendix.

Notice that the numerator and denominator represent spatial and temporal reconstruction, respectively, and hence both should be individually positive. For the spatial part, any standard limiter can be used, but in this work, the following is used:

$$\phi(r) = \begin{cases} r & \text{if } |r| \leq 1, \\ 1 & \text{if } |r| > 1. \end{cases} \tag{8}$$

This is equivalent to second order accurate ENO reconstruction. The ENO method is spatially second order accurate *and* strictly TVD when used with the implicit Euler time integration. In other words, the numerator is nonnegative for any value of r. It is apparent that the temporal terms (denominator) are similar to the spatial terms except for the scaling by the CFL number; hence, in order to satisfy the TVD conditions (nonnegativity of the denominator), the time limiter can be defined as

$$\psi(s) = \begin{cases} s/\sigma & \text{if} \quad |s| \leq 1 \quad \text{and} \quad \sigma > 1, \\ 1/\sigma & \text{if} \quad |s| > 1 \quad \text{and} \quad \sigma > 1, \\ s & \text{if} \quad |s| \leq 1 \quad \text{and} \quad \sigma \leq 1, \\ 1 & \text{if} \quad |s| > 1 \quad \text{and} \quad \sigma \leq 1. \end{cases} \tag{9}$$

Equations (8) and (9) are sufficient to guarantee an *unconditionally TVD scheme*. However, from (9) it is seen that for (local) $\sigma > 1$, the TVD region has shrunk because of the scaling, and this results in a local drop in temporal accuracy even when the solution is smooth in space and time ($s \approx 1$). This can be overcome by taking advantage of the implicitness of the scheme. The denominator of (7) is positive (and the numerator divided by the denominator is bounded) if

$$\psi_j < \frac{2}{\sigma} + \frac{\psi_{j-1}}{s_{j-1}}.$$

In implicit schemes, the system of equations is solved in an approximate linearized form at each time step, but to restore time accuracy, Newton *subiterations* have to be performed. Refer to [12] for a review of the approach. In the current method, the first subiteration is performed assuming $\psi_j = 0$ throughout the domain. After this, for any subiteration $p+1$, ψ can be set using information from p:

$$\psi_j^{p+1} = \min\left[1, \max\left[\frac{2}{\sigma} + \frac{\psi_{j-1}^p}{s_{j-1}^p} - \epsilon, 0\right]\right], \tag{10}$$

where $\epsilon > 0$ is a small number (to ensure that the denominator does not equal zero). This restores accuracy (see Figure 4) when the solution is smooth. To an extent, this also ensures smooth variation of the time limiter between grid points. In practice, this marginally degrades the convergence of subiterations.

Equations (8) and (10) define the limiters that will be used in this work for space and time reconstruction.

4. Numerical results. Numerical results are presented for the one-dimensional inviscid Burgers equation and the Euler equations. For each of the above cases, Newton-type subiterations [12] are used to solve the implicit set of equations at each time step. Typically, about five subiterations are required for convergence within one time step. The new scheme is termed the *implicit second order* space-time reconstruction (STR). For comparative purposes, all linear time integration schemes are used with the second order ENO spatial discretization. For this spatial scheme, explicit time integration schemes will be nonoscillatory up to a maximum CFL number $\sigma = \max_j\{\sigma_j\} = 0.5$.

4.1. The Burgers equation. The first test case is the inviscid Burgers equation $u_t + (\frac{u^2}{2})_x = 0$, with periodic boundary conditions and a domain $[0, 2\pi]$ of 100 equally spaced points. The initial condition is composed of an expansion wave and a

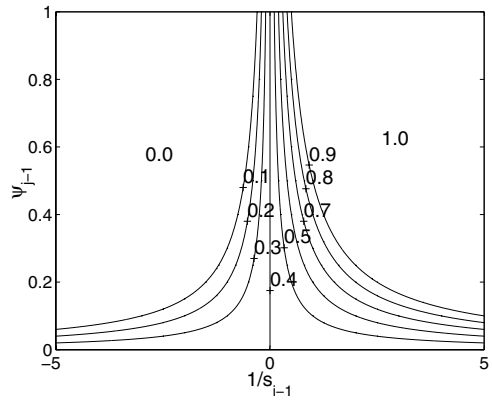


FIG. 4. *ψ_j as a function of ψ_{j-1} and s_{j-1} for $\sigma = 5$. For higher σ, there is more limiting in the left half-plane.*

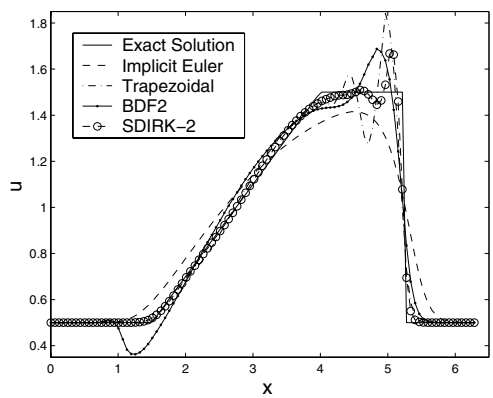


FIG. 5. *The Burgers equation with $\sigma = 4.0$, $N = 100$, and $T_{final} = 2.0$.*

compression wave. This profile is convected to a time $t = 2.0$, before which the compression wave becomes a shock. The Roe–Riemann solver is used for the interfacial flux.

For a time step corresponding to a CFL number $\sigma = \max_j\{u_j \frac{\Delta t}{\Delta x}\} = 4.0$, Figure 5 shows that the implicit Euler method shows high dissipation and the linear second order time integration schemes develop oscillations in the vicinity of the shock. The trapezoidal (Crank–Nicolson) method performs particularly poorly since it provides very little damping for high frequency dispersive errors. It is seen that the new scheme (Figure 6) is nonoscillatory. As expected, limiter-1 (9) is very dissipative and is only marginally better than the implicit Euler method. Limiter-2 (10) is seen to resolve the shock and the expansion wave well and will be used in the rest of the paper. Figures 7 and 8 assess the performance of the scheme up to $\sigma = 10.0$. It is evident that the new method remains nonoscillatory and reasonably accurate for the entire range of time-step sizes.

4.2. One-dimensional Euler equations. The one-dimensional Euler equations of gas dynamics are given by

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} = 0,$$

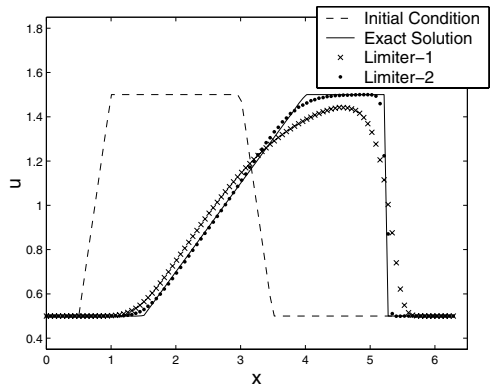


FIG. 6. *The Burgers equation with* $\sigma = 4.0$, $N = 100$, *and* $T_{final} = 2.0$.

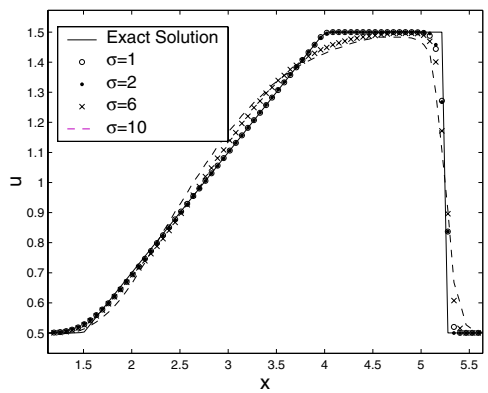


FIG. 7. *Performance of second order STR for various CFL numbers for the Burgers equation.* $N = 100$ *and* $T_{final} = 2.0$.

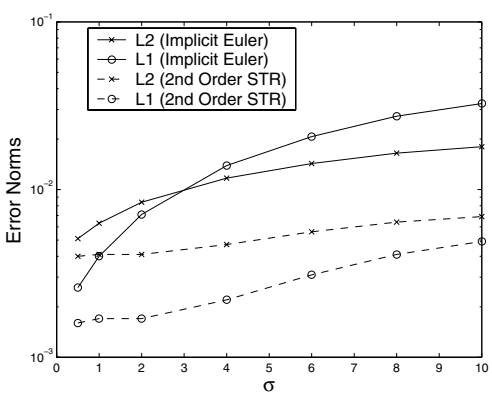


FIG. 8. L_1 *and* L_2 *norms for the Burgers equation.* $N = 100$ *and* $T_{final} = 2.0$.

where $\mathbf{U}$, the vector of conserved variables, and $\mathbf{F}$, the flux vector, are defined by

$$\mathbf{U} = \left\{ \begin{array}{l} \rho \\ \rho u \\ e \end{array} \right\}, \quad \mathbf{F} = \left\{ \begin{array}{l} \rho u \\ p + \rho u^2 \\ (e + p)u \end{array} \right\},$$

where ρ, u, and p are density, velocity, and pressure, respectively. e is the total energy per unit volume given by

$$e = \frac{p}{\gamma - 1} + \frac{\rho u^2}{2}.$$

Space and time limiters can be applied to each scalar equation $i = 1, \ldots, 3$ at each point $j = 1, \ldots, j\max$ in the domain. The reconstruction may be based on primitive, conservative, or characteristic variables. However, if primitive or conservative variables are used, the time limiter is somewhat arbitrary, and σ_j will have to be based on the maximum eigenvalue $\max\{|\lambda_i|)\}_j$. Based on numerical computations, this approach still seems to work if the time limiter (ψ_i) for a given cell is restricted to the minimum of the three computed limiters, resulting in increased dissipation. If characteristic variables are used for reconstruction, $(\sigma_i)_j$ can be based on $(|\lambda_i|)_j$, and hence different ψ_i can be used. Computations prove that this is indeed much more accurate, especially in treating contact discontinuities, which being linear, are more susceptible to smearing. It is also known [10], [3] that in many cases (including Lax's problem), even high order explicit schemes operating at very small CFL numbers require characteristic variables for nonoscillatory reconstruction. Hence, this approach will be used to obtain the following numerical results. Once the reconstruction is accomplished, Roe's approximate Riemann solver [13] is used to calculate the interfacial fluxes.

Sod's problem. Sod's problem for an infinite length constant area duct is defined by the initial left and right states given by $\{p_L, \rho_L, u_L\} = \{1.0,\ 1.0,\ 0.0\}$ and $\{p_R, \rho_R, u_R\} = \{0.1,\ 0.125,\ 0.0\}$. Figures 9–11 confirm that the scheme is less diffusive than the first order method and stable compared to the second order methods.

Lax's problem. Lax's problem is defined by $\{p_L, \rho_L, u_L\} = \{3.528,\ 0.445,0.698\}$ and $\{p_R, \rho_R, u_R\} = \{0.571,\ 0.5,\ 0.0\}$. This problem is slightly more difficult than Sod's problem in that the density is not monotone decreasing, and hence the contact discontinuity and the shock are more difficult to capture without spurious oscillations. As mentioned earlier, reconstructions based on primitive and conserved variables yield a fair amount of noise at high CFL numbers. Figures 12 and 13 outline some results for this problem.

5. Perspectives. For second order accurate schemes with symmetric spatial differencing, the determination of the interface value $u_{j+\frac{1}{2}}^{n+\frac{1}{2}}$ essentially comes from the set of four values $\{u_j^n, u_{j+1}^n, u_j^{n+1}, u_{j+1}^{n+1}\}$. Therefore, a family of schemes can be constructed, and many well-known methods can be defined based on particular combinations of these values. For instance, the calculation of the interface value in the unlimited STR-2 reduces to just a diagonal average as shown in Figure 14, and a few other schemes are shown in Table 2 (for a linear advection equation with a positive wave speed). Definition of better limiters can take advantage of the solution geometry as suggested in [14], [3]. In general, adopting less stringent monotonicity criteria than the TVD condition can improve solution accuracy, especially near extrema.

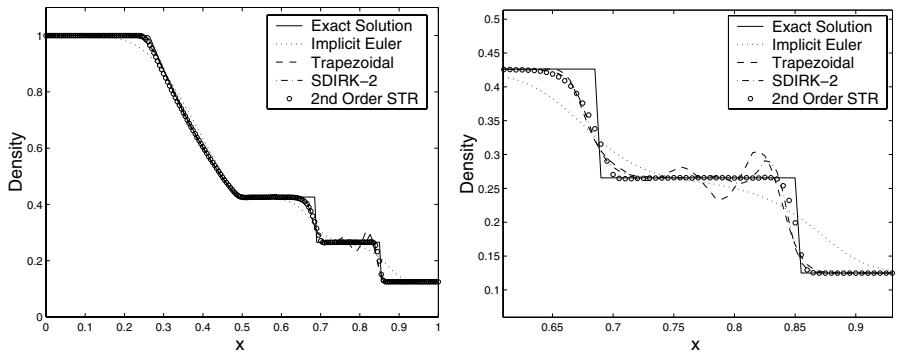


FIG. 9. *Sod's problem with* $\sigma = 4.0$, $N = 200$, *and* $T_{final} = 0.2$.

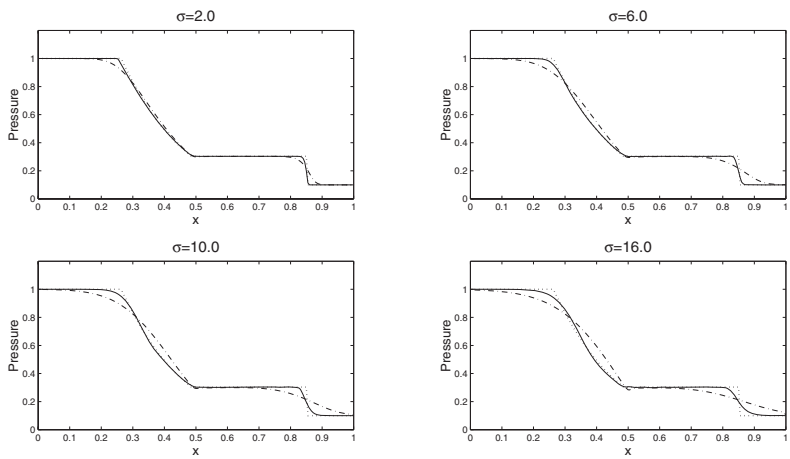


FIG. 10. *Sod's problem* ($N = 200$)*: Performance of second order STR (solid lines) and implicit Euler (broken lines) for various CFL numbers.*

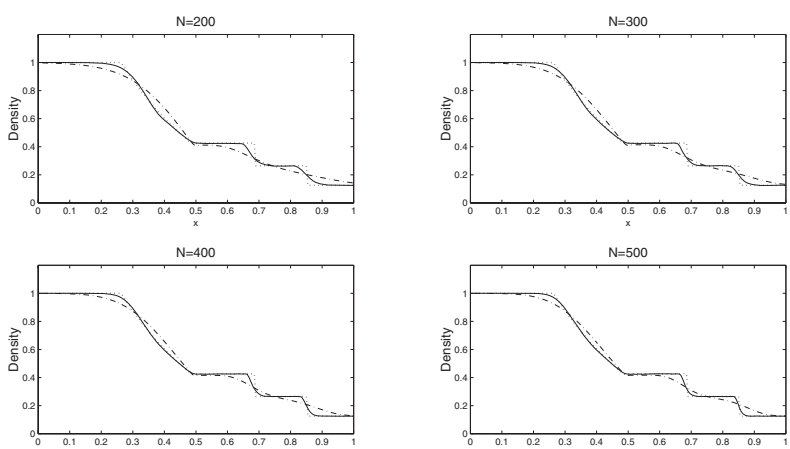


FIG. 11. *Sod's problem* ($\sigma = 16.0$)*: Performance of second order STR (solid lines) and implicit Euler (broken lines) with mesh refinement.*

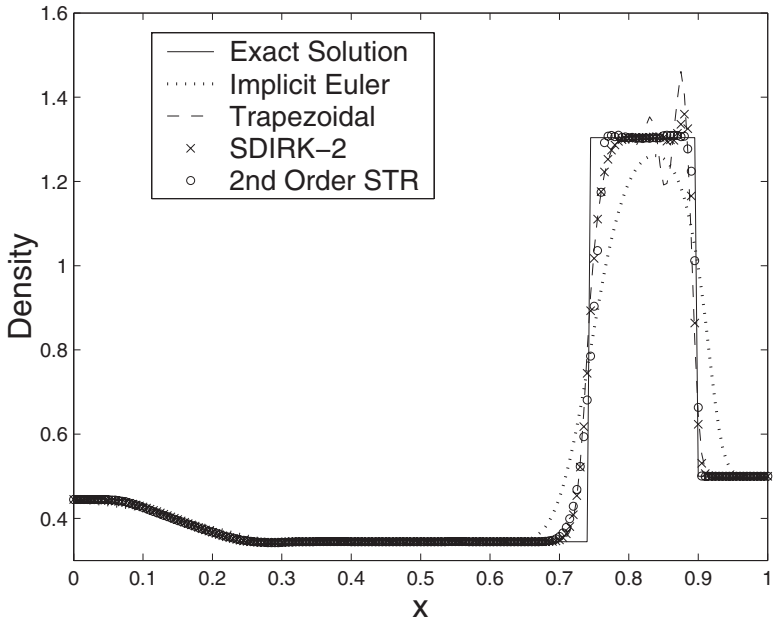


FIG. 12. *Lax's problem with* $\sigma = 4.0$, $N = 200$, *and* $T_{final} = 0.16$.

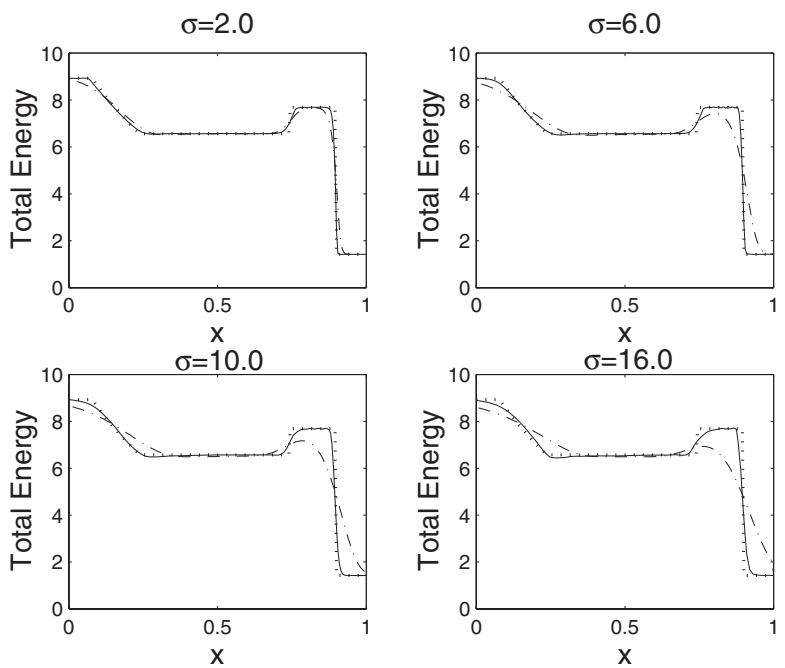


FIG. 13. *Lax's problem* ($N = 200$)*: Performance of second order STR (solid lines) and implicit Euler (broken lines) for various CFL numbers.*

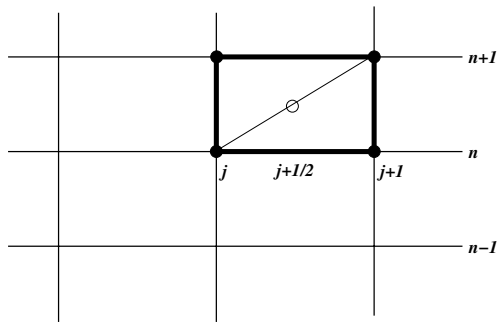


FIG. 14. *Schematic of support for the determination of* $u_{j+\frac{1}{2}}^{n+\frac{1}{2}}$.

TABLE 2
Interpolation formula to compute $u_{j+\frac{1}{2}}^{n+\frac{1}{2}}$ *using various second order accurate schemes.*

Method	Interpolation formula
STR2 (unlimited)	$\frac{u_j^n+u_{j+1}^{n+1}}{2}$
Exp. Lax–Wendroff	$\frac{u_j^{n(1+\sigma)}+u_{j+1}^{n}(1-\sigma)}{2}$
Imp. Lax–Wendroff	$\frac{u_j^{n+1}(1-\sigma)+u_{j+1}^{n+1}(1+\sigma)}{2}$
Crank–Nicolson	$\frac{(u_{j+1}^n-u_j^n)+(u_{j+1}^{n+1}-u_j^{n+1})}{2}$

6. Conclusions. A new single step second order accurate implicit scheme that uses *reconstruction in space and time* has been developed. Conventionally, to achieve high order accuracy in single step methods, temporal derivatives are replaced by equivalent spatial derivatives. In the present approach, the temporal derivative is retained and approximated by a difference in time. The resulting STR is made nonoscillatory by the introduction of limiters that satisfy TVD conditions. Numerical experiments on model problems show accurate nonoscillatory solutions over a large range of time steps. Future work can involve increasing the order of accuracy by possibly having more quadrature points in the time interval.

7. Appendix.

7.1. Proof of TVD property. Lemma 3.1 in [11] states that if a numerical scheme for a conservation law can be written as $L.v^{n+1} = v^n$, then the scheme is TVD if the finite difference operator L is total variation increasing (TVI). Further, Lemma 3.2 in [11] states that if L can be represented in the form $(L.v)_j = v_j - C^-_{j-\frac{1}{2}}(\Delta_x)_{j-\frac{1}{2}}$, then L is TVI if $-\infty < C^-_{j-\frac{1}{2}} \leq 0$. Therefore, the TVD condition for the numerical scheme (6) is

$$-\infty < -\frac{1+\frac{1}{2}(\frac{\phi_j}{r_j}-\phi_{j-1})}{1+\frac{\sigma}{2}(\frac{\psi_{j-1}}{s_{j-1}}-\psi_j)} \leq 0 \tag{11}$$

or

$$\infty > \frac{1+\frac{1}{2}(\frac{\phi_j}{r_j}-\phi_{j-1})}{1+\frac{\sigma}{2}(\frac{\psi_{j-1}}{s_{j-1}}-\psi_j)} \geq 0. \tag{12}$$

7.2. Limiter in a general framework. For the linear convection case $u_t + au_x = 0$ with $a < 0$, the TVD condition is

$$\frac{1+\frac{1}{2}(-\frac{\phi_{j+1}}{r_{j+1}}+\phi_j)}{1+\frac{|\sigma|}{2}(\frac{\psi_{j+1}}{s^-_{j+1}}-\psi_j)} \geq 0,$$

where $s^-_{j+1} = \frac{(\Delta_t)_j}{(\Delta_t)_{j+1}}$.

For a nonlinear case $u_t + f(u)_x = 0$, with $\frac{\partial f}{\partial u}$ being positive or negative, we get similar conditions except for the fact that σ is replaced by $\sigma_j = \frac{\Delta t}{\Delta x}\frac{(f_j^{n+1}-f_j^n)}{(u_j^{n+1}-u_j^n)}$. Hence the time limiter (10) for a general case would be

$$\psi_j^{p+1} = \max\left[0, \min\left[\frac{2}{|\sigma_j|}+\frac{\psi^p_{j-1}}{s^p_{j-1}}, \frac{2}{|\sigma_j|}+\frac{\psi^p_{j+1}}{(t^-_{j+1})^p}, 1\right]\right].$$

7.3. Linear stability of the unlimited scheme. A conventional Fourier stability analysis of the unlimited scheme as applied to the linear convection equation ($a > 0$) results in the amplification function G as a function of the spatial wave number β and the CFL number σ given by

$$G(\beta,\sigma) = \frac{1-\frac{\sigma}{2}(1-e^{-i\beta})}{1-\frac{\sigma}{2}(1-e^{i\beta})}.$$

It is seen that $|G| = 1$ for all possible β and σ. This implies unconditional linear stability, and the scheme performs well for smooth solutions. However, for discontinuous solutions, accumulation of dispersion errors will result in spurious oscillations.

REFERENCES

[1] A. Harten and S. Osher, *Uniformly high-order accurate nonoscillatory schemes.* I, SIAM J. Numer. Anal., 24 (1987), pp. 279–309.

[2] X. Liu, S. Osher, and T. Chan, *Weighted essentially non-oscillatory schemes*, J. Comput. Phys., 115 (1994), pp. 200–212.

[3] A. Suresh and H. T. Huynh, *Accurate monotonicity-preserving schemes with Runge–Kutta time stepping*, J. Comput. Phys., 136 (1997), pp. 83–99.

[4] S. Gottlieb, C.-W. Shu, and E. Tadmor, *Strong stability-preserving high-order time discretization methods*, SIAM Rev., 43 (2001), pp. 89–112.

[5] A. Harten, *High resolution schemes for hyperbolic conservation laws*, J. Comput. Phys., 49 (1983), pp. 357–393.

[6] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations*, Vol. 2, Springer-Verlag, Berlin, 1991.

[7] K. Duraisamy, J. D. Baeder, and J.-G. Liu, *Concepts and application of time-limiters to high resolution schemes*, J. Sci. Comput., 19 (2003), pp. 139–162.

[8] S. A. Forth, *A second order accurate, space-time limited, BDF scheme for the linear advection equation*, in Godunov Methods: Theory and Applications, Kluwer Academic/Plenum Press, New York, 2001.

[9] B. Van Leer, *Towards the ultimate conservative difference scheme* V, *A second order sequel to Godunov's method*, J. Comput. Phys., 32 (1979), pp. 101–136.

[10] A. Harten, B. Engquist, S. Osher, and S. Chakravarthy, *Uniformly high-order accurate essentially nonoscillatory schemes.* III, J. Comput. Phys., 71 (1987), pp. 231–303.
[11] A. Harten, *On a class of high resolution total-variation-stable finite-difference schemes*, SIAM J. Numer. Anal., 21 (1984), pp. 1–23.
[12] T. Pulliam, *Time Accuracy and the Use of Implicit Methods*, AIAA Paper 93-3360, 1993.
[13] P. Roe, *Approximate Riemann solvers, parameter vectors, and difference schemes*, J. Comput. Phys., 43 (1981), pp. 357–372.
[14] A. Suresh, *The Reconstruction Problem Revisited*, NASA TM 1999-209082, NASA, Washington, D.C., 1999.

SIAM J. SCI. COMPUT.
Vol. 29, No. 6, pp. 2621–2643
© 2007 Society for Industrial and Applied Mathematics

THE EFFECT OF PROBLEM PERTURBATIONS ON NONLINEAR DYNAMICAL SYSTEMS AND THEIR REDUCED-ORDER MODELS*

RADU SERBAN†, CHRIS HOMESCU‡, AND LINDA R. PETZOLD‡

Abstract. Reduced-order models are used extensively in many areas of science and engineering for simulation, design, and control. Reduction techniques for nonlinear dynamical systems produce models that depend strongly on the nominal set of parameters for which the reduction is carried out. In this paper we address the following two questions: "What is the effect of perturbations in the problem parameters on the output functional of a nonlinear dynamical system?" and "to what extent does the reduced-order model capture this effect?"

Key words. model reduction, small-sample statistical condition estimation, adjoint method, valid parameter range, singular value analysis

AMS subject classifications. 65L10, 65L99

DOI. 10.1137/050625278

1. Introduction.

In this paper we consider nonlinear dynamical systems of the generic form

$$\mathcal{M}(t, y', y, p) = 0, \quad y(t_0) = y^0(p), \tag{1.1}$$

where y are the model states, y^0 are the initial conditions, and p are model parameters. The system may be an ordinary differential equation (ODE), a semidiscretized partial differential equation (PDE), or a differential-algebraic equation (DAE). An important problem which arises in a wide variety of engineering and scientific applications is to find the amount of change which can be expected in a model response functional $h(t, y(t), p)$ at a later time t, as a function of changes in the parameters p. The direction in the parameter space of the perturbation that produces the maximum change in the output may also be needed. For example, in weather prediction one wishes to find those perturbations in the initial conditions that can produce the greatest change in the accuracy of the forecast, because they identify the locations where the acquisition of additional data can be most effective [7, 16].

For large-scale dynamical systems, such decisions making use of the sensitivity of the system to perturbations in the problem parameters may necessarily be based on the corresponding sensitivities of a reduced-order model (ROM). A ROM corresponding to (1.1) is a system of the form

$$\widetilde{\mathcal{M}}(t, \tilde{y}', \tilde{y}, p) = 0, \quad \tilde{y}(t_0) = \tilde{y}^0(p), \tag{1.2}$$

such that its solution $\tilde{y}$ is an approximation to the solution y of (1.1). The ROM must be computationally less expensive than the full model. Depending on the particular model reduction technique employed, this can be achieved either through a

*Received by the editors February 24, 2005; accepted for publication (in revised form) June 22, 2007; published electronically October 31, 2007. This work was supported by DOE DE-FG03-00ER25430, NSF/NCSA ACI-9619019, NSF CCF 0428912, and NSF/ITR ACI-0086061.
http://www.siam.org/journals/sisc/29-6/62527.html

†Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA 94551 (radu@llnl.gov). The work of the first author was performed under the auspices of the U.S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under contract W-7405-Eng-48.

‡Department of Computer Science, University of California, Santa Barbara, CA 93106 (homescu@cs.ucsb.edu, petzold@engineering.ucsb.edu).

reduction in the dimension of the state space, through a reduction in the complexity of the forcing term, or both. Nonlinear ROMs have been obtained through proper orthogonal decomposition (POD) [14, 20], reduction of complexity [19, 22], balanced truncation [9], principal component analysis [12], and many other techniques [1]. An important property of nonlinear ROMs is that they are (necessarily) based on a nominal set of parameters or a limited range of operating conditions. Thus the question naturally arises: To what extent does a given nonlinear ROM capture the changes in the output functional of the full system that result from perturbations in the problem parameters?

We present in section 2 an efficient method for the estimation of perturbation-induced errors in the solution of a dynamical system. The method is based on a combination of adjoint sensitivity analysis [4] and the small-sample statistical condition estimate (SCE) method [13, 8]. We show how to apply these ideas to systems described by ODEs, semiexplicit index-1 DAEs, and Hessenberg index-2 DAEs [10, 3]. This algorithm is closely related to our earlier work on estimating errors for POD-based ROM applied to ODE systems [11] but extends those techniques to DAEs and to more general nonlinear reduced-order models.

Using this methodology, in section 3 we show how the direction in parameter space of maximum error growth in the output functional can be computed with very few solutions of the adjoint model. This is related to singular vector (SV) analysis in the weather prediction literature, e.g., [5]. By interpreting the initial perturbation as the error (or uncertainty) at the initial time, it was found in [5] that the maximum possible error growth is the largest singular value of the operator obtained from the linearization of the differential system. In section 4 we tackle the issue of estimating the extent to which a given ROM preserves the value of the maximum perturbation-induced error and/or the direction of the perturbations that induce it. We introduce two types of similarity index which measure this property. The problem of estimating the error which combines the effects of both model reduction and perturbations in parameters is addressed in section 5.

Numerical results are given in section 6. Models described by both ODEs and DAEs and corresponding ROMs generated by several different methods are presented. An example is given for which the ROM is quite accurate for the nominal values of the parameters but fails to capture the behavior of the full model when the parameters are slightly perturbed. We demonstrate that the similarity index can be very effective in detecting this type of situation.

Before proceeding, we would like to state that studying the errors induced by change in parameter values via a linearized equation is not new, and sensitivity analysis has been used extensively as a model analysis tool. However, the two usual sensitivity analysis methods based on solving continuous sensitivity equations, the forward and adjoint sensitivity analysis methods, respectively, are limited (from computational effort considerations) in the number of parameters that are allowed to vary or in the number of model responses that are monitored. As a consequence, methods based on a forward sensitivity approach are feasible only for problems with relatively few parameters, while methods based directly on an adjoint sensitivity approach allow the inclusion in the analysis of only a few model responses.

The new aspect of this work is in using aproximate schemes (namely, the SCE approach) which, combined with adjoint methods, allow the effective estimation of the "many parameters-many responses" case, e.g., estimating the influence of perturbations in all initial conditions on the entire state vector at some final time. Using the methodology described in the following, this capability enables a more robust

assessment of the quality of ROMs under perturbations by capturing in the analysis the influence of perturbations in any of the (potentially) many problem parameters on (potentially) the entire model state.

Finally, we note that rigurous error bounds for the various approximations used in the proposed approach are difficult, if not impossible, to get within the scope of the current scheme. Indeed approximations due to the SCE (see section 2.1) would defeat any attempt at deriving such error bounds. However, as demonstrated by the examples presented in section 6, the methodology proposed proves to be a very effective indicator of potential shortcomings of any given ROM under perturbations of problem parameters away from their nominal values (the parameter values used in constructing the ROM).

2. Estimation of perturbation-induced errors. In this section we present a method for the estimation of errors induced by perturbations in the initial conditions of a dynamical system. This approach, based on adjoint sensitivity analysis and small-sample SCE, will be used throughout this paper.

For dynamical systems of the form (1.1), we wish to find an estimate of the error $e(t) = Y(t) - y(t)$ at some time $t > t_0$, where $Y(t)$ is the solution of the perturbed system

$$\mathcal{M}(t, Y', Y, p + \delta p) = 0, \quad Y(t_0) = y^0(p + \delta p). \tag{2.1}$$

Subtracting (1.1) from (2.1) yields the following equation for the error $e(t)$:

$$\mathcal{M}_{y'}(t)e' + \mathcal{M}_y(t)e + \mathcal{M}_p(t)\delta p + \mathcal{O}(\|e\|^2) = 0,$$

where $\mathcal{M}_{y'}$, $\mathcal{M}_y$, and $\mathcal{M}_p$ are the Jacobians of $\mathcal{M}$ with respect to y', y, and p, respectively, evaluated at $(t, y'(t), y(t), p)$. Thus, to a first-order approximation, the error satisfies

$$\mathcal{M}_{y'}(t)e' + \mathcal{M}_y(t)e + \mathcal{M}_p(t)\delta p = 0, \quad e(t_0) = y_p^0(p)\,\delta p. \tag{2.2}$$

Note that the problem of estimating the error for perturbations in the initial conditions is a particular case of the problem of estimating the error for perturbations in the parameters, for which $p = y^0$. For any given perturbation δp, one can solve the forward sensitivity equation (2.2) to obtain the time evolution of the error $e(t)$ at any time $t > t_0$. However, this forward approach to error estimation can become expensive if one wishes to estimate the error over a range of perturbations, because (2.2) must be solved for *each* perturbation of interest.

We can, with the SCE technique described below, obtain an estimate of the norm of the error at some arbitrary but fixed time $t > t_0$, i.e., $\|e(t)\|$, as long as scalar products of the form $z^T e(t)$ are available, for any given unit vector z. This estimate takes the form

$$\|e(t)\| \approx \frac{W_q}{W_n} \left(\sum_{j=1}^{q} \left| z_j^T e(t) \right|^2 \right)^{1/2},$$

where $\{z_j\}_{j=1\ldots q}$ are q orthogonal vectors, uniformly and randomly selected from the unit sphere $\mathbb{S}_{n-1}$, where n is the dimension of y.

In the remainder of this section, after a short introduction to the SCE technique, we present an adjoint-based method for efficiently computing the scalar products

$z^T e(t)$. First, we consider dynamical systems that can be written as ODEs, and then we present the derivation for DAE systems. We base our SCE estimates on the solution of adjoint systems that are *independent* of δp and can thus be used for the estimation of errors induced by *any* perturbation.

2.1. Small-sample statistical method for condition estimation. The small-sample SCE method, originally proposed in [13], offers an efficient means for condition estimation for general nonlinear functions, at the cost of allowing moderate relative errors in the estimate. The basic idea is described below (for complete details, see [13, 8]).

For any vector $v \in \mathbb{R}^n$, if u is selected uniformly and randomly from the unit sphere $\mathbb{S}_{n-1}$, the expected value of $u^T v$ is proportional to the norm of v:

$$E(|u^T v|) = W_n \|v\|.$$

The proportionality factor, called the *Wallis factor*, depends only on n. It is defined as $W_1 = 1$ and

$$W_n = \begin{cases} \dfrac{1 \cdot 3 \ldots (n-2)}{2 \cdot 4 \ldots (n-1)} & \text{for } n \text{ odd}, \\ \dfrac{2}{\pi} \dfrac{2 \cdot 4 \ldots (n-2)}{1 \cdot 3 \ldots (n-1)} & \text{for } n \text{ even} \end{cases}$$

and can be approximated by $W_n \approx \sqrt{\frac{2}{\pi(n-\frac{1}{2})}}$.

Thus the expression $\xi = \frac{|u^T v|}{W_n}$ is used to estimate the norm $\|v\|$. This estimate is first order in the sense that the probability of a relative error in the estimate is inversely proportional to the size of the error. That is, for $\gamma > 1$, we have

$$\Pr\left(\frac{\|v\|}{\gamma} \le \xi \le \gamma \|v\| \right) \ge 1 - \frac{2}{\pi\gamma} + O\left(\frac{1}{\gamma^2} \right).$$

Additional function evaluations can improve the accuracy of the estimate.

Suppose that we obtain estimates $\xi_1, \xi_2, \ldots, \xi_q$ corresponding to orthogonal vectors $u_1, u_2, \ldots, u_q$ selected uniformly and randomly from the unit sphere $\mathbb{S}_{n-1}$. The expected value of the norm of the projection of v onto the span $\mathcal{U}$ generated by $u_1, u_2, \ldots, u_q$ is

$$E\left(\sqrt{|u_1^T v|^2 + |u_2^T v|^2 + \cdots + |u_q^T v|^2} \right) = \frac{W_n}{W_q} \|v\|.$$

The estimate $\nu(q) = \frac{W_q}{W_n} \sqrt{|u_1^T v|^2 + |u_2^T v|^2 + \cdots + |u_q^T v|^2}$ is qth order accuracy; i.e., a relative error of size γ in the estimate occurs with probability proportional to γ^{-q} [13]:

$$\begin{aligned} \Pr\left(\frac{\|v\|}{\gamma} \le \nu(2) \le \gamma \|v\| \right) &\approx 1 - \frac{\pi}{4\gamma^2}, \\ \Pr\left(\frac{\|v\|}{\gamma} \le \nu(3) \le \gamma \|v\| \right) &\approx 1 - \frac{32}{3\pi^2\gamma^3}, \\ \Pr\left(\frac{\|v\|}{\gamma} \le \nu(4) \le \gamma \|v\| \right) &\approx 1 - \frac{81\pi^2}{512\gamma^4}. \end{aligned}$$

Usually at most four random vectors are required in practice.

2.2. Ordinary differential equation systems. Consider the following system:

$$f(t, y', y, p) = 0, \quad y(t_0) = y^0(p), \tag{2.3}$$

for $t \in [t_0, t_f]$, y, $y^0 \in \mathbb{R}^n$, and $f : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$. The partial derivative matrix $f_{y'}$ is nonsingular. The linearized error equation in this case is

$$M(t)e' + A(t)e = -f_p(t)\delta p, \quad e(t_0) = y_p^0(p)\delta p, \tag{2.4}$$

where the matrices of partial derivatives $M(t) = f_{y'}$, $A(t) = f_y$, and f_p are evaluated at $(t, y'(t), y(t), p)$. For the remainder of this section, for the sake of readability, we will drop the time argument for these matrices, except where absolutely necessary.

We begin by noting that the solution of (2.4) can be written as

$$e(t) = \Phi(t) \left(y_p^0 - \int_{t_0}^{t} \Phi^{-1}(\tau) M^{-1} f_p \, d\tau \right) \delta p,$$

where $\Phi(t) \in \mathbb{R}^{n \times n}$ is the fundamental matrix corresponding to (2.4), i.e.,

$$M\Phi' + A\Phi = 0, \quad \Phi(t_0) = I. \tag{2.5}$$

It is straightforward to verify that the solution λ of the adjoint system

$$(M^T\lambda)' - A^T\lambda = 0, \tag{2.6}$$

with final condition $\lambda(t) = z$, satisfies

$$\begin{aligned} &\lambda^T(\tau) = z^T\Phi(t)\Phi^{-1}(\tau)M^{-1}(\tau) \quad \forall \tau \in [t_0, t], \\ &\lambda^T(t_0)M(t_0) = z^T\Phi(t), \end{aligned} \tag{2.7}$$

and therefore,

$$z^T e(t) = \left(\lambda^T(t_0)M(t_0)y_p^0 - \int_{t_0}^{t} \lambda^T(\tau) f_p \, d\tau \right) \delta p. \tag{2.8}$$

The case of perturbations in initial conditions is obtained by setting $f_p(t) = 0$ for all t and $y_p^0 = I$, while the case of perturbations in the right-hand side corresponds only to $y_p^0 = 0$.

If we consider q orthogonal vectors $z_j, 1 \le j \le q$, uniformly and randomly selected from the unit sphere $\mathbb{S}_{n-1}$, we obtain the SCE estimate

$$\begin{aligned} \|e(t)\| &\approx \frac{W_q}{W_n} \left(\sum_{j=1}^{q} \left| z_j^T e(t) \right|^2 \right)^{1/2} \\ &= \frac{W_q}{W_n} \left(\sum_{j=1}^{q} \left| \left(\lambda_j^T(t_0)M(t_0)y_p^0 - \int_{t_0}^{t} \lambda_j^T(\tau) f_p(\tau) \, d\tau \right) \delta p \right|^2 \right)^{1/2}, \end{aligned} \tag{2.9}$$

where λ_j are solutions of the adjoint systems (2.6) with final conditions based on z_j.

An important property of this error estimate is that the adjoint system (2.6) on which it is based is *independent* of the perturbation δp. Thus the same adjoint solutions λ_j can be used to estimate the error induced by any perturbation. Finally,

we note that very few (less than four) unit vectors z_j (and hence adjoint system solutions) are usually sufficient for acceptable SCE estimates.

In preparation for using the above estimate in comparing two different models, let us consider the perturbation-induced error in a model response functional $h(t, y(t), p)$. We seek the norm of this error, namely, $\epsilon(t) := h(t, Y(t), p + \delta p) - h(t, y(t), p)$ (see Figure 5.1). A linearization around $\delta p = 0$ yields

$$\epsilon(t) = h_y e(t) + h_p \delta p.$$

Thus

$$\begin{aligned}\|\epsilon(t)\| &\approx \frac{W_q}{W_{n_h}} \left(\sum_{j=1}^{q} \left| z_j^T h_y e(t) + z_j^T h_p \delta p \right|^2 \right)^{1/2} \\ &= \frac{W_q}{W_{n_h}} \left(\sum_{j=1}^{q} \left| \left(\lambda_j^T(t_0) M(t_0) y_p^0 - \int_{t_0}^{t} \lambda_j^T(\tau) f_p(\tau)\, d\tau + z_j^T h_p \right) \delta p \right|^2 \right)^{1/2},\end{aligned}$$

where λ_j is the solution of the adjoint system (2.6) with the final condition $\lambda_j(t) = h_y^T(t) z_j$. We select the q vectors z_j uniformly and randomly from the unit sphere $\mathbb{S}_{n_h - 1}$, where n_h is the dimension of h.

2.3. Semiexplicit index-1 differential-algebraic equation systems. A semiexplicit index-1 DAE takes the form

$$\begin{aligned} & f(t, y', y, x, p) = 0, \\ & g(t, y, x, p) = 0, \\ & y(t_0) = y^0(p),\ x(t_0) = x^0(p), \end{aligned} \tag{2.10}$$

where y are the differential variables, x are the algebraic variables, and p are model parameters. The Jacobian matrices $f_{y'}$ and g_x are nonsingular. Note that the initial conditions y^0, x^0 are assumed to be consistent with the algebraic constraints; that is, $g(t_0, y^0, x^0, p) = 0$.

We are interested in estimating the norm of the error $e(t)$ between the solution $(Y(t), X(t))$ of the system with perturbed parameters and the solution $(y(t), x(t))$ of the unperturbed system. We decompose $e(t)$ into the differential and algebraic components: $e_d(t) = Y(t) - y(t)$ and $e_a(t) = X(t) - x(t)$. As in the ODE case, a linearization along the solution of the unperturbed DAE leads to

$$\begin{aligned} & M(t) e_y' + A(t) e_d + B(t) e_a = -f_p(t) \delta p, \\ & C(t) e_d + D(t) e_a = -g_p(t) \delta p, \\ & e_d(0) = y_p^0 \delta p\,,\ e_a(t_0) = x_p^0 \delta p, \end{aligned} \tag{2.11}$$

where the matrices of partial derivatives $M(t) = f_{y'}$, $A(t) = f_y$, $B(t) = f_x$, $C(t) = g_y$, $D(t) = g_x$, f_p, and g_p are all evaluated at $(t, y(t), x(t), p)$. Consistency of the initial conditions for the above DAE is equivalent to the requirement that the perturbation δp satisfies

$$(C(t_0) y_p^0 + D(t_0) x_p^0 + g_p(t_0)) \delta p = 0. \tag{2.12}$$

We reduce the analysis to the ODE case by using the essential underlying ODE (EUODE) [2]:

$$Me_y' + (A - BD^{-1}C)e_d = -(f_p - BD^{-1}g_p)\delta p, \quad e_d(t_0) = y_p^0 \delta p. \tag{2.13}$$

The corresponding fundamental matrix Φ_1 satisfies

$$M\Phi_1' + (A - BD^{-1}C)\Phi_1 = 0, \quad \Phi_1(t_0) = I, \tag{2.14}$$

and therefore

$$\begin{aligned} e_d(t) &= \Phi_1(t)\left(y_p^0 - \int_{t_0}^{t} \Phi_1^{-1}(\tau)M^{-1}(f_p - BD^{-1}g_p)\,d\tau\right)\delta p, \\ e_a(t) &= -D^{-1}\left(Ce_d(t) + g_p\delta p\right). \end{aligned}$$

The SCE norm estimate of the error vector $[e_d^T(t);\, e_a^T(t)]^T$ is based on scalar products of the form

$$z_d^T e_d(t) + z_a^T e_a(t) = \left(z_d^T - z_a^T D^{-1}C\right)e_d(t) - z_a^T D^{-1}g_p\delta p,$$

for some vector $z = [z_d^T;\, z_a^T]^T \in \mathbb{S}_{n-1}$. It can be easily verified that the differential component λ of the solution of the adjoint system

$$\begin{aligned} &(M^T\lambda)' - A^T\lambda - C^T\mu = 0, \\ &-B^T\lambda - D^T\mu = 0, \\ &\lambda(t) = z_d - C^T D^{-T} z_a, \end{aligned} \tag{2.15}$$

satisfies $\lambda(\tau) = M^{-T}(\tau)\Phi_1^{-T}(\tau)\Phi_1^T(t)\lambda(t)$. As a consequence,

$$\begin{aligned} \left(\lambda^T(t_0)M(t_0)y_p^0 - \int_{t_0}^{t}\lambda^T(f_p - BD^{-1}g_p)\,d\tau\right)\delta p &= \left(z_d - C^T D^{-T} z_a\right)^T \\ &\times \left(\Phi_1(t)\Phi_1^{-1}(t_0)y_p^0 - \int_{t_0}^{t}\Phi_1(t)\Phi_1^{-1}(\tau)M^{-1}(f_p - BD^{-1}g_p)\,d\tau\right)\delta p. \end{aligned}$$

Thus

$$z_d^T e_d(t) + z_a^T e_a(t) = \left(\lambda^T(t_0)M(t_0)y_p^0 - \int_{t_0}^{t}\lambda^T(f_p - BD^{-1}g_p)\,d\tau - z_a^T D^{-1}g_p\right)\delta p,$$

or, using $\mu^T = -\lambda^T BD^{-1}$,

$$z_d^T e_d(t) + z_a^T e_a(t) = \left(\lambda^T(t_0)M(t_0)y_p^0 - \int_{t_0}^{t}\left(\lambda^T f_p + \mu^T g_p\right)\,d\tau - z_a^T D^{-1}g_p\right)\delta p. \tag{2.16}$$

Following the same argument as in section 2.2, an SCE norm estimate for the perturbation-induced error $\epsilon(t)$ in a model response $h(t, y(t), x(t), p)$ can be obtained by replacing the final condition on λ with

$$\lambda(t) = (h_y - h_x D^{-1}C)^T z, \quad z \in \mathbb{S}_{n_h - 1},$$

and using

$$z^T\epsilon(t) = \left(\lambda^T(t_0)M(t_0)y_p^0 - \int_{t_0}^{t}\left(\lambda^T f_p + \mu^T g_p\right)\,d\tau - z^T h_x D^{-1}g_p + z^T h_p\right)\delta p.$$

If we consider only perturbations in the initial conditions ($f_p = g_p = y_p^0 = 0$, implying $x_p^0 = I$) and if we are interested in the norm of the entire error vector $e(t)$, the following refinement of the above procedure, which takes advantage of the geometry of the problem, results in more accurate SCE estimates. We begin by observing that, since $e_d(t)$ and $e_a(t)$ must be consistent with the algebraic constraints, we need only consider unit vectors z that satisfy the linearized constraint $C(t)z_d + D(t)z_a = 0$. Let $n_y = \dim(y)$ be the number of degrees of freedom of the system (2.10). We start by selecting q vectors z_d uniformly and randomly from $\mathbb{S}_{n_y-1}$ and, for each one of them, construct the vector $z_a = -D^{-1}(t)C(t)z_d$. We then orthonormalize the set of q vectors $z = [z_d^T; z_a^T]^T$ and thus obtain q orthogonal unit vectors in $\mathbb{R}^n$ which are uniformly and randomly distributed over the sphere $\mathbb{S}_{n_y-1}$ embedded on the linear subspace of $\mathbb{R}^n$ of codimension $n - n_y$.

2.4. Hessenberg index-2 differential-algebraic equation systems. Consider next a dynamical system expressed in Hessenberg index-2 DAE form:

$$\begin{aligned} &f(t, y', y, x, p) = 0, \\ &g(t, y, p) = 0, \\ &y(t_0) = y^0(p),\ x(t_0) = x^0(p), \end{aligned} \tag{2.17}$$

where y are the differential variables, x are the algebraic variables, and p are model parameters. The matrices $f_{y'}$ and $g_y f_{y'}^{-1} f_x$ are nonsingular, and the initial conditions are assumed to be consistent, i.e., $g(t_0, y^0, p) = 0$ and $\frac{d}{dt} g(t_0, y^0, p) = 0$.

Let $A(t) = f_y$, $B(t) = f_x$, $C(t) = g_y$, and $M(t) = f_{y'}$. To keep the derivation as simple as possible, we start by assuming that $M(t) = I$. Results for a general mass matrix will be given at the end of this section. With $M(t) = I$, the linearized error equation becomes

$$\begin{aligned} &e_y' + A(t)e_d + B(t)e_a = -f_p(t)\delta p, \\ &C(t)e_d = -g_p(t)\delta p, \\ &e_d(t_0) = y_p^0 \delta p,\ e_a(t_0) = x_p^0 \delta p \end{aligned} \tag{2.18}$$

for the errors $e_d(t) = Y(t) - y(t)$ and $e_a(t) = X(t) - x(t)$, where $Y(t)$, $X(t)$ is the solution of (2.17) with perturbed parameters. Consistency of the initial conditions in (2.18) requires

$$\begin{aligned} &C y_p^0 \delta p = -g_p \delta p, \\ &-(C' - CA) y_p^0 \delta p + CB x_p^0 \delta p = (g_p' - C f_p)\delta p \end{aligned} \tag{2.19}$$

at $t = t_0$.

For the Hessenberg index-2 DAE (2.18), the EUODE is derived as follows [2]. If B is sufficiently smooth, there exists a smooth bounded matrix function $R(t) \in \mathbb{R}^{(n_y - n_x) \times n_y}$ whose linearly independent, normalized rows form a basis for the nullspace of B^T. Thus, $RB = 0$, and $\begin{bmatrix} R \\ C \end{bmatrix}$ is invertible.

With the change of variables $u(t) = R(t)e_d(t)$, e_d is obtained as

$$e_d = \begin{bmatrix} R \\ C \end{bmatrix}^{-1} \begin{bmatrix} u \\ -g_p \delta p \end{bmatrix} = Su - F g_p \delta p,$$

where $S(t) \in \mathbb{R}^{n_y \times (n_y - n_x)}$ satisfies $R(t)S(t) = I$, and $C(t)S(t) = 0$ and $F = B(CB)^{-1}$. The EUODE is obtained as $u' = (R' - RA)Su - (R' - RA)F g_p \delta p - R f_p \delta p$.

Before proceeding further, we note that

$$F = B(CB)^{-1},$$
$$SR = I - B(CB)^{-1}C,$$

and thus the EUODE can be written as

$$u' = (R' - RA)Su + R\left((B' + AB)(CB)^{-1}g_p - f_p\right)\delta p, \quad u(t_0) = u_0, \tag{2.20}$$

where we have used $R'F = R'B(CB)^{-1} = -RB'(CB)^{-1}$. Note that, to satisfy the first condition in (2.19), a perturbation δp must be such that

$$y_p^0 \delta p = S(t_0)u_0 - B(t_0)(C(t_0)B(t_0))^{-1}g_p(t_0)\delta p, \tag{2.21}$$

for an arbitrary u_0.

If Φ_2 is the fundamental matrix corresponding to (2.20), that is,

$$\Phi_2' = (R' - RA)S\Phi_2, \quad \Phi_2(t_0) = I,$$

then

$$u(t) = \Phi_2(t)\left(u_0 + \int_{t_0}^{t} \Phi_2^{-1}(\tau)R\left((B' + AB)(CB)^{-1}g_p - f_p\right)\delta p\, d\tau\right),$$
$$e_d(t) = Su(t) - Fg_p\delta p = Su(t) - B(CB)^{-1}g_p\delta p,$$
$$e_a(t) = (CB)^{-1}\Big((C' - CA)e_d(t) + (g_p' - Cf_p)\delta p\Big).$$

Consider next the adjoint system

$$\begin{aligned} \lambda' - A^T\lambda - C^T\mu &= 0, \\ -B^T\lambda &= 0, \end{aligned} \tag{2.22}$$

whose EUODE can be obtained with the change of variables $v = S^T\lambda$ (implying $\lambda = R^T v$) as

$$v' = \left(S'^T + S^T A^T\right)R^T v = -S^T(R' - RA)^T v, \quad v(t) = \nu.$$

Since the EUODE of the adjoint is the adjoint of the EUODE (2.20), we get

$$\lambda(\tau) = R^T(\tau)v(\tau) = R^T(\tau)\Phi_2^{-T}(\tau)\Phi_2^T(t)\nu. \tag{2.23}$$

The scalar products required by the SCE procedure in this case are of the form

$$z_d^T e_d(t) + z_a^T e_a(t) = z_d^T e_d(t) + z_a^T(CB)^{-1}(C' - CA)e_d(t) + z_a^T(CB)^{-1}(g_p' - Cf_p)\delta p$$
$$= \left(z_d^T + z_a^T(CB)^{-1}(C' - CA)\right)\left(Su(t) - B(CB)^{-1}g_p\right)\delta p + z_a^T(CB)^{-1}(g_p' - Cf_p)\delta p.$$

Using (2.23) and (2.21) we get

$$\left(\lambda^T(t_0)y_p^0 + \int_{t_0}^{t} \lambda^T\left((B' + AB)(CB)^{-1}g_p - f_p\right)\, d\tau\right)\delta p$$
$$= \nu\Phi_2(t)\left(u_0 + \int_{t_0}^{t} \Phi_2^{-1}(\tau)R\left((B' + AB)(CB)^{-1}g_p - f_p\right)\delta p\, d\tau\right),$$

where we have used $R(t_0)S(t_0) = I$ and $R(t_0)B(t_0) = 0$. Taking into account $\mu^T = -\lambda^T(B' + AB)(CB)^{-1}$ and selecting the final condition for (2.22) to be

$$\lambda(t) = \left(I - B(CB)^{-1}C\right)^T \left(z_d + (C' - CA)^T(CB)^{-T}z_a\right),$$

we get

$$z_d^T e_d(t) + z_a^T e_a(t) = \Bigg(\lambda^T(t_0)y_p^0 - \int_{t_0}^{t} \left(\lambda^T f_p + \mu^T g_p\right)\, d\tau \\ - z_d^T B(CB)^{-1}g_p + z_a^T(CB)^{-1}\left(g_p' - (C' - CA)B(CB)^{-1}g_p - Cf_p\right)\Bigg)\delta p.$$

If the final condition for λ is replaced with

$$\lambda(t) = \left(I - B(CB)^{-1}C\right)^T \left(h_y^T + (C' - CA)^T(CB)^{-T}h_x^T\right) z,$$

for some $z \in \mathbb{R}^{n_h}$, the resulting adjoint variable can be used in evaluating the scalar product $z^T\epsilon(t)$, which is necessary in an SCE estimate of the norm of the perturbation-induced error in the model response $h(t, y(t), x(t), p)$, as

$$z^T\epsilon(t) = \Bigg(\lambda^T(t_0)y_p^0 - \int_{t_0}^{t} \left(\lambda^T f_p + \mu^T g_p\right)\, d\tau - z^T h_y B(CB)^{-1}g_p \\ + z^T h_x(CB)^{-1}\left(g_p' - (C' - CA)B(CB)^{-1}g_p - Cf_p\right) + z^T h_p\Bigg)\delta p.$$

Finally, for a general mass matrix $M(t)$, we note that the matrix $CM^{-1}B$ must always be nonsingular. Then the scalar product $z_d^T e_d(t) + z_a^T e_a(t)$ can be obtained, using the solution $\{\lambda, \nu\}$ of the following adjoint system:

$$\begin{aligned} (M^T\lambda)' - A^T\lambda - C^T\mu &= 0, \\ -B^T\lambda &= 0, \\ M^T(t)\lambda(t) = \left(I - M^{-1}B(CM^{-1}B)^{-1}C\right)^T &\left(z_d + (C' - CM^{-1}A)^T(CM^{-1}B)^{-T}z_a\right), \end{aligned} \tag{2.24}$$

as

$$\begin{aligned} z_d^T e_d(t) + z_a^T e_a(t) = \Bigg(&\lambda^T(t_0)M(t_0)y_p^0 - \int_{t_0}^{t} \left(\lambda^T f_p + \mu^T g_p\right)\, d\tau \\ &-z_d^T M^{-1}B(CM^{-1}B)^{-1}g_p \\ &+z_a^T(CM^{-1}B)^{-1}\left(g_p' - (C' - CM^{-1}A)M^{-1}B(CM^{-1}B)^{-1}g_p - CM^{-1}f_p\right)\Bigg)\delta p. \end{aligned} \tag{2.25}$$

3. Singular vector analysis. SV analysis, introduced by Lorenz in the 1960s [18], is employed to compute the largest error growth and to help estimate model predictability. Finding the direction of maximal error growth implies solving a generalized eigenvalue problem for a matrix involving the product $\Phi^T\Phi$, where Φ is the fundamental matrix of the differential system.[1] Solving the resulting eigenvalue problem is quite computationally expensive. It is typically accomplished using Lanczos-type

[1] In the meteorological and weather prediction literature this matrix is often called the *propagator*.

algorithms which require only the evaluation of matrix-vector products. However, every such product involves a forward integration of a linearized model followed by a backward in time solution of the corresponding adjoint model. Based on the combination of SCE estimate and adjoint models, we describe below a method for finding an approximation to the leading SV (representing the direction of maximal error growth) that requires only very few solutions of the adjoint model (equal to the number of unit vectors considered in the SCE estimate).

Consider again the error $\epsilon(t) := h(t, Y(t), p + \delta p) - h(t, y(t), p)$ in some model response functional $h(t, y, p)$. We wish to find the direction of perturbations in the parameters p that gives rise to the maximum error $\|\epsilon(t)\|$ at some time $t \geq t_0$. In other words, we seek

$$\mathcal{E} := \sup_{\substack{\|u\| = 1 \\ u \in \mathbb{R}^{n_u}}} \|\epsilon(t, u)\|, \tag{3.1}$$

where ϵ is the response error induced by a perturbation $\delta p = \delta p(u)$ in the parameters p. The directions u over which we search for the maximum error norm growth are constrained to belong to a search space $\mathbb{R}^{N_u}$ such that the resulting perturbations $\delta p(u)$ are admissible. The dimension N_u is model- and problem-dependent and will be defined below for each of the dynamical systems discussed in section 2.

Using an SCE estimate for the error norm in (3.1) we get

$$\mathcal{E}^2 = \left(\frac{W_q}{W_{n_h}}\right)^2 \sup_{\substack{\|u\| = 1 \\ u \in \mathbb{R}^{n_u}}} \sum_{j=1}^{q} |z_j^T \epsilon(t, u)|^2,$$

where, for the dynamical systems considered in this paper, the scalar products $z_j^T \epsilon(t, u)$ were derived in sections 2.2–2.4. The essence of these formulas is that, in all cases,

$$z_j^T \epsilon(t, u) = \alpha_j^T \delta p,$$

with α_j defined in terms of the solution λ_j of an appropriate adjoint system. Therefore

$$\mathcal{E}^2 = \left(\frac{W_q}{W_{n_h}}\right)^2 \sup_{\substack{\|u\| = 1 \\ u \in \mathbb{R}^{n_u}}} \sum_{j=1}^{q} |\alpha_j^T \delta p(u)|^2 = \left(\frac{W_q}{W_{n_h}}\right)^2 \sup_{\substack{\|u\| = 1 \\ u \in \mathbb{R}^{n_u}}} \delta p^T(u) \left(\sum_{j=1}^{q} \alpha_j \alpha_j^T\right) \delta p(u)$$

$$= \left(\frac{W_q}{W_{n_h}}\right)^2 \sup_{\substack{\|u\| = 1 \\ u \in \mathbb{R}^{n_u}}} u^T P^T \left(\sum_{j=1}^{q} \alpha_j \alpha_j^T\right) P u,$$

where we have assumed that the admissible perturbations can be written as $\delta p(u) = Pu$, for some matrix P. Let $\Psi := P^T(\sum_{j=1}^{q} \alpha_j \alpha_j^T)P$. The matrix Ψ is symmetric and has $\operatorname{rank}(\Psi) \leq q$. Thus its singular value decomposition (SVD) is $\Psi = U^T \Sigma U = \sum_{j=1}^{q} \sigma_j u_j u_j^T$, and $u^T \Psi u = \sum_{j=1}^{q} \left(\sqrt{\sigma_j} u^T u_j\right)^2$. If the singular values of Ψ are numbered in decreasing order, it is easy to see that

$$\sup_{\substack{\|u\| = 1 \\ u \in \mathbb{R}^{n_u}}} u^T \Psi u = \sigma_1,$$

which is attained for $u^* = u_1$. Thus

$$\mathcal{E} = \frac{W_q}{W_{n_h}}\sqrt{\sigma_1}, \tag{3.2}$$

and the direction of perturbations along which the error is maximized is $\delta p^* = Pu_1$.[2]

Next we establish the dimension n_u in (3.1) and the corresponding matrix P (defining the allowable parameter perturbations $\delta p(u) = Pu$) for the dynamical systems discussed in the previous section.

1. For an ODE system, any perturbation δp is admissible. Thus $n_u = n_p \equiv \dim(p)$ and $P = I_{n_p}$.
2. For a semiexplicit index-1 DAE, requiring the initial condition for (2.12) to be consistent is equivalent to requiring that δp be in the nullspace of the matrix $Q_1 = C(t_0)y_p^0 + D(t_0)x_p^0 + g_p(t_0)$. Then $P = \mathcal{N}(Q_1)$ and $n_u = \dim(\mathcal{N}(Q_1))$, where $\mathcal{N}(Q_1)$ is a basis for the nullspace of Q_1. In the particular case of perturbations in initial conditions $e_d(t_0) = \delta p$ and $e_a(t_0) = -D^{-1}(t_0)C(t_0)\delta p$. Therefore $\mathcal{U} \equiv \mathbb{R}^{n_y}$ and $P = I_{n_y}$.
3. For a Hessenberg index-2 DAE, similarly to the index-1 DAE case, the conditions (2.19), modified here to account for a general mass matrix, are equivalent to the requirement that δp be in the nullspace of

$$Q_2 = \begin{bmatrix} Cy_p^0 + g_p \\ -(C' - CM^{-1}A)y_p^0 + CM^{-1}Bx_p^0 - g_p' + CM^{-1}f_p \end{bmatrix}$$

at $t = t_0$. Thus $P = \mathcal{N}(Q_2)$ and $n_u = \dim(\mathcal{N}(Q_2))$. In the case of perturbations in the initial conditions, $e_d(t_0) = \delta p$ and $e_a(t_0) = (C(t_0)B(t_0))^{-1}(C'(t_0) - C(t_0)A(t_0))\delta p$. Consistent perturbations must then satisfy $\delta p = S(t_0)u$, with $u \in \mathbb{R}^{n_y - n_x}$. Note that $n_y - n_x$ represents the number of degrees of freedom of the system (2.18).

Finally, if the parameter perturbations δp are to be further restricted to the range of some matrix Π, then $P = \Pi$ in the ODE case, and $P = \Pi\mathcal{N}(Q\Pi)$, where $Q = Q_1$ for index-1 DAEs or $Q = Q_2$ for index-2 DAEs.

4. Quality measures for reduced-order models. The quality of a given ROM can be judged by the extent to which it preserves the value of the maximum possible error and/or the direction of perturbations that induce it. We begin by constructing a matrix whose entries

$$m_{ij} = \langle u_i, \tilde{u}_j \rangle^2$$

are the squared scalar products of the ith SV of the full model and the jth SV of the reduced model. These entries represent the amount of energy[3] of the ith full model SV that can be explained by the jth reduced model SV. The SV analysis must be based on a model response that is common to both the full and reduced-order models.

[2] As indicated by one of the reviewers of this paper, if considering only perturbations in initial conditions, such an error estimate is equivalent to an approximation to the dominant local (i.e., finite-time) Lyapunov exponent of the system. For a discussion on the connection between singular vectors and Lyapunov vectors, see, for example, [21]. See also [6] for methods for the computation of Lyapunov exponents based on orthogonal decompositions of the system Jacobian.

[3] The energy depends on the particular inner product used. Here we use the usual Euclidean inner product.

Based on the matrix m_{ij}, a similarity index can be defined [17] as

$$\mathcal{S}_1 = \frac{1}{q} \sum_{i,j=1}^{q} m_{ij}. \tag{4.1}$$

Although the proposed method (based on SCE norm estimation) provides only an approximation to $\mathcal{S}_1$, since the evaluation of $\mathcal{S}_1$ requires several singular vectors, it is more computationally attractive than the alternative offered by Lanczos-type algorithms, not to mention the option of computing the entire fundamental matrix Φ.

A second measure of similarity can be defined in terms of the ratio between the norms of the errors induced in the full model and in the reduced-order model by a perturbation of the unit norm along the dominant SV of the full model

$$\mathcal{S}_2 = \min\left(\frac{\|\tilde{\epsilon}(t,u_1)\|}{\|\epsilon(t,u_1)\|}, \frac{\|\epsilon(t,u_1)\|}{\|\tilde{\epsilon}(t,u_1)\|}\right). \tag{4.2}$$

Since both similarity indexes are related to the assessment of ROM quality, one may ask when one index is more relevant than the other. Each similarity index evaluates different aspects of the ROM quality. The first index $\mathcal{S}_1$ quantifies the similarity between subspaces spanned by the first q SVs for the full model and, respectively, for the reduced model. Thus it provides an overall picture of ROM approximation of the phenomena described by the full model. The second index $\mathcal{S}_2$ gives a more localized result, by estimating the extent to which the ROM preserves the error due to perturbations along the direction of maximal error growth, given by the dominant SV of the full model.

Finally, we note that both indexes take values between 0 and 1 and that, for identical models, they will equal unity. The index $\mathcal{S}_2$ is a more stringent ROM quality test, in the sense that, typically, $\mathcal{S}_2 < \mathcal{S}_1$.

5. ROM error under perturbations. Consider next the problem of estimating the error $\|\Theta(t)\|$, which combines the effects of both model reduction and perturbations in parameters (see Figure 5.1). For an arbitrary, but fixed, time $t > t_0$, let[4]

$$\begin{aligned}\Theta(t) &= h(Y) - \tilde{h}(\tilde{Y}) = \big(h(y) - \tilde{h}(\tilde{y})\big) + \big(h(Y) - h(y)\big) - \big(\tilde{h}(\tilde{Y}) - \tilde{h}(\tilde{y})\big)\\ &= \theta(t) + \epsilon(t) - \tilde{\epsilon}(t)\\ &\approx \theta(t) + h_y e(t) - \tilde{h}_{\tilde{y}}\tilde{e}(t) + \big(h_p - \tilde{h}_p\big)\delta p,\end{aligned} \tag{5.1}$$

where $\theta(t) = h(t, y(t), p) - \tilde{h}(t, \tilde{y}(t), p)$ is the error in the model response due solely to model reduction, $e(t) = Y(t) - y(t)$ is the error introduced by the perturbation in the original system, and $\tilde{e}(t) = \tilde{Y}(t) - \tilde{y}(t)$ is the error introduced by the perturbation in the reduced system. We distinguish two cases. In both, the projections $z_j^T h_y e(t)$ and $z_j^T \tilde{h}_{\tilde{y}} \tilde{e}(t)$ are based on the solution of adjoint problems corresponding to the full and reduced models, respectively, using the procedure of section 2.

[4]For the sake of clarity, only the relevant function arguments were kept.

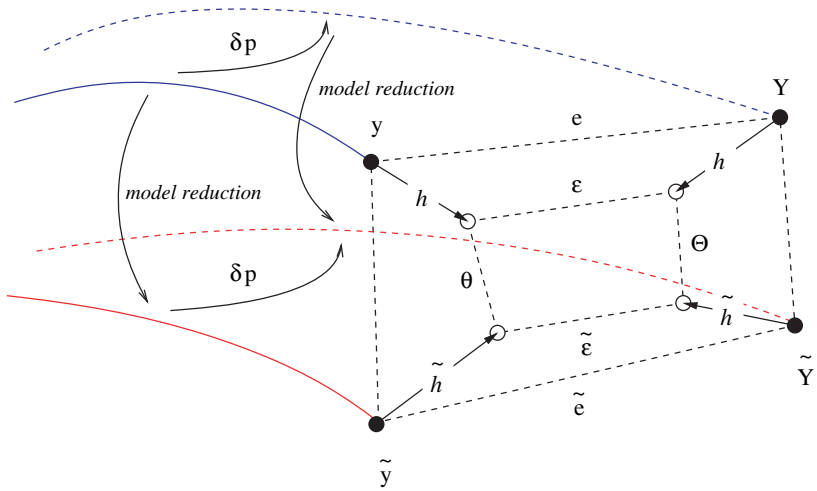


FIG. 5.1. *Graphical representation of model reduction and perturbation-induced errors. Solid-line trajectories and quantities in lowercase $(y, \tilde{y}, \theta)$ correspond to unperturbed systems, while dashed-line trajectories and states in uppercase $(Y, \tilde{Y}, \Theta)$ correspond to the perturbed systems. Quantities with a tilde correspond to reduced-order models.*

1. If the components of the error $\theta(t)$ are known, we can directly use an SCE estimate for the norm $\|\Theta(t)\|$:

$$\|\Theta(t)\| = \|\theta(t) + h_y e(t) - \tilde{h}_{\tilde{y}}\tilde{e}(t)\|$$
$$\approx \frac{W_q}{W_{n_h}} \left[\sum_{j=1}^{q} \left| z_j^T \theta(t) + z_j^T \left(h_p - \tilde{h}_p\right)\delta p + z_j^T h_y e(t) - z_j^T \tilde{h}_{\tilde{y}} \tilde{e}(t) \right|^2 \right]^{1/2}.$$

2. If only the norm $\|\theta(t)\|$ is available, we use the bounds

$$\left| \|\theta(t) + \left(h_p - \tilde{h}_p\right)\delta p\| - \|h_y e(t) - \tilde{h}_{\tilde{y}}\tilde{e}(t)\| \right| \le \|E(t)\|$$
$$\le \|e(t) + \left(h_p - \tilde{h}_p\right)\delta p\| + \|h_y e(t) - \tilde{h}_{\tilde{y}}\tilde{e}(t)\|$$

and the following SCE estimate:

$$\|h_y e(t) - \tilde{h}_{\tilde{y}}\tilde{e}(t)\| \approx \frac{W_q}{W_{n_h}} \left[\sum_{j=1}^{q} \left| z_j^T h_y e(t) - z_j^T \tilde{h}_{\tilde{y}} \tilde{e}(t) \right|^2 \right]^{1/2}.$$

6. Numerical results. We have implemented our methodology to problems representative of the dynamical systems considered in this paper. We consider models described by systems of ODEs—the two-dimensional Brusselator (modeling autocatalytic, oscillating chemical reactions) and the Gas Research Institute (GRI) combustion model (from chemical kinetics)—and by DAEs (the heat shock model in biology).

6.1. 2D Brusselator. The following 2D time-dependent PDE models a chemically reacting system:

$$u_t = \epsilon_1 \Delta u + A + u^2 v - (B+1)u,$$
$$v_t = \epsilon_2 \Delta v + Bu - u^2 v,$$

where the diffusion coefficients are $\epsilon_1 = \epsilon_2 = 2 \cdot 10^{-3}$ and the chemical parameters are given by $A = 1$ and $B = 3.4$. The PDE is defined over the square $\Omega = [0,1] \times [0,1]$, for $t \in [0,1]$. Homogeneous Neumann conditions are imposed at the boundary of the domain, while the initial conditions are

$$u(x,y,0) = 1 - 0.5 \cdot \cos(\pi y), \quad v(x,y,0) = 3.5 - 2.5 \cdot \cos(\pi x). \tag{6.1}$$

Using the method of lines, we convert this PDE to an ODE initial value problem using central finite differences to approximate the spatial derivatives on a grid of $N_x \times N_y = 16 \times 16$ interior points, with the boundary conditions explicitly used to define the necessary ghost boundary points. The resulting system, of the form (2.3), has dimension $n = 512$.

Proper orthogonal decomposition (POD) provides a method for finding the best approximating affine subspace to a given set of data. When using POD for model reduction of dynamical systems, the data are time snapshots of the solution obtained via numerical simulations or from experiments. Consider next the solutions of (1.1) at m time points, collected in the $n \times m$ matrix $\mathcal{Y} = [y(t_1) - \overline{\mathbf{y}}, y(t_2) - \overline{\mathbf{y}}, \ldots y(t_m) - \overline{\mathbf{y}}]$, where $\overline{\mathbf{y}}$ is the mean of these observations. POD seeks a subspace $S \in \mathbb{R}^n$ and the corresponding projection matrix $P = \rho\rho^T$ so that the total square distance

$$\|\mathcal{Y} - P\mathcal{Y}\|^2 = \sum_{i=1}^{m} \| \left(y(t_i) - \overline{\mathbf{y}}\right) - P\left(y(t_i) - \overline{\mathbf{y}}\right) \|^2$$

is minimized. The solution to the above minimization problem can be easily obtained by considering the SVD of the correlation matrix $R = \mathcal{Y}\mathcal{Y}^T$ (see [20, 11] for details). A POD-based reduced model that approximates the original problem can then be constructed by projecting onto S the model (1.1):

$$\rho^T \mathcal{M}(t, \rho\tilde{y} + \overline{\mathbf{y}}, \rho\tilde{y}') = 0, \quad \tilde{y}(t_0) = \rho^T(y_0 - \overline{\mathbf{y}}).$$

Returning to the Brusselator model, a POD-based ROM is constructed using 100 equally spaced snapshots from the trajectory corresponding to the initial conditions (6.1) and retaining 12 modes. We note that the resulting ROM retains 99.52% of the "energy" of the full system. Moreover, comparing the solutions of the full and reduced models at the final time, we find that

$$\frac{\|y(t) - \tilde{y}(t)\|}{\|y(t)\|} \cdot 100.0 = 0.3661\%. \tag{6.2}$$

We are therefore led to believe that this ROM (of dimension only 12) can very accurately represent the solution of full model. However, using the analysis of the preceding sections, it turns out that this ROM does not perform well under perturbations in the initial conditions.

We have applied the approach presented in section 3 to estimate the leading singular vectors for both the full model and the ROM, using $q = 12$ and $q = 4$ unit vectors for the SCE, respectively. The results are summarized in Table 6.1. Next, applying the approach presented in section 4 and using the first four leading singular vectors, (4.1) gives a similarity index of only $\mathcal{S}_1 = 0.03287$, while (4.2) gives $\mathcal{S}_2 = 0.02328$. The reason for this poor agreement is that the direction of maximum error growth for the full system is almost orthogonal to the POD subspace in which the ROM is bound to evolve. Indeed, if u_1 is the leading singular vector for the full system, we have that $\|(I - \rho * \rho^T)u_1\| = 0.95266$, while

TABLE 6.1

Brusselator example: SV analysis results. For both the full and reduced models the following quantities are given: the estimated maximum perturbation-induced error norm given by (3.2) *at the final time* $t = 1.0$*, the actual norm of the error in the estimated direction of maximum error growth, and the norms of errors induced by several random perturbations.*

Full model		Reduced-order model	
$\mathcal{E}$	6.70690e+01	$\tilde{\mathcal{E}}$	6.50318e+00
$\|e(t,u_1)\|$	5.18372e+01	$\|\tilde{e}(t,\tilde{u}_1)\|$	5.85795e+00
$\|e(t,u_{rnd})\|$	2.82639e+00	$\|\tilde{e}(t,\tilde{u}_{rnd})\|$	2.54461e+00
	5.18880e+00		2.58347e+00
	3.79540e+00		3.32946e+00
	5.81685e+00		1.59148e+00
	4.14449e+00		1.65752e+00

$\|\rho^T u_1\| = 0.30404$, where ρ is the POD projection matrix. Moreover, the leading singular vector $\tilde{u}_1$ of the ROM fails to approximate well even the projection of u_1 on the POD subspace: the angle $\measuredangle(\tilde{u}_1, \rho^T u_1) = 150.87°$ (an angle of 0° or 180° indicates perfect agreement). The discrepancies between the leading singular vector of the full model and that of the ROM can also be seen in Figure 6.1 in which the singular vectors are color coded and superimposed over the unperturbed initial conditions.

As a final verification, introducing a perturbation along u_1 of magnitude of only $1/100 \cdot \|y_0\|$, the quantity (6.2) becomes

$$\frac{\|Y(t) - \tilde{Y}(t)\|}{\|Y(t)\|} \cdot 100.0 = 16.268\%,$$

which is in perfect agreement with the similarity index $\mathcal{S}_2$ (indeed, $0.02328 = \mathcal{S}_2 \approx 0.3661/16.268 = 0.0225$).

6.2. GRI combustion model. Given n_s chemical species with mass fractions y_i and m reactions, the constant-volume batch reactor is modeled by the ODE system

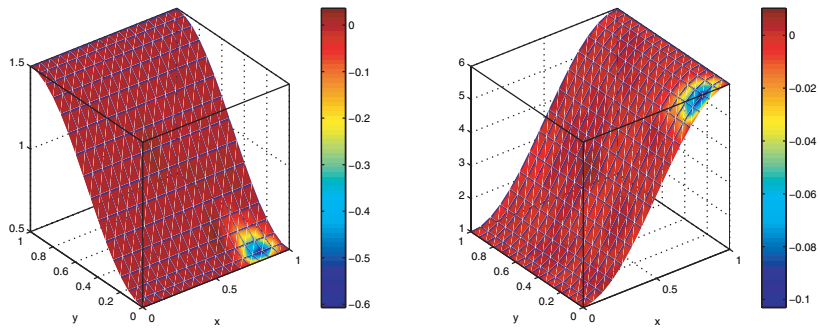
$$y' = S(y)F(y), \quad y(t_0) = y_0, \tag{6.3}$$

where $S \in \mathbb{R}^{n_s \times m}$ includes the stoichiometric matrix and the thermodynamic equations. $F \in \mathbb{R}^m$ is the vector of nonlinear responses corresponding to elementary reactions, with $F_i(y)$ denoting the reaction rate of the ith physical elementary reaction.

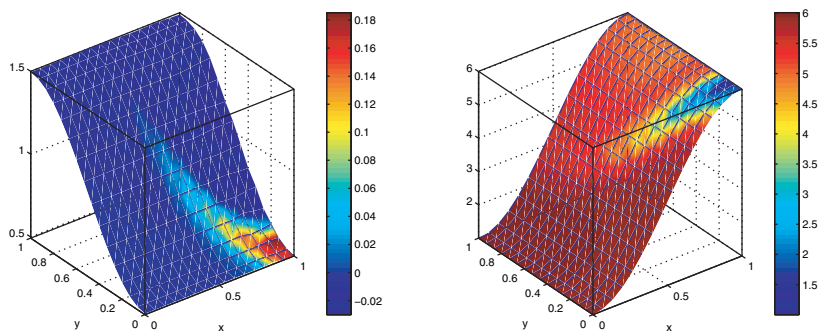
A ROM can be constructed by choosing some (small) subset of the reactions to use in the mechanism so that the behavior of the reduced system is as close as possible to that of the original system, given a range of operating conditions. Often this is done by a chemist based on physical knowledge and intuition. A numerical method for the construction of such reduced models is given by [19]. In that approach, one seeks a ROM in the form

$$\tilde{y}' = S(\tilde{y})DF(\tilde{y}), \quad \tilde{y}(t_0) = y_0, \tag{6.4}$$

such that the norm of the error $e = y - \tilde{y}$ is minimized over a given interval $[t_0, t]$. In (6.4), $D \in \mathbb{R}^{m \times m}$ is a diagonal matrix whose diagonal elements d_i are either 1 or 0 (depending on whether or not the reaction i is selected for the reduced mechanism).



(a) Leading singular vector for the full model.



(b) Leading singular vector for the reduced-order model.

FIG. 6.1. *Brusselator example: Singular vectors color coded and superimposed over the initial conditions (u component on the left, v component on the right).*

The problem of finding the reduced mechanism is written as a discrete constrained optimization where the minimum is over $d_1, d_2, \ldots, d_m$, with $d_i \in \{0, 1\}$, and $\sum d_i = k \ll m$ is prescribed.

The GRI combustion model is a set of $m = 177$ reactions describing the combustion of natural gas and involves $n_s = 32$ chemical species. For more details on this model, see [19, 22]. The resulting ODE system has dimension $n = n_s + 1 = 33$, the last differential equation describing the evolution of the temperature; i.e., $y = [c, T/T_0]$, where c are the species concentrations and T is the temperature, scaled by its initial value T_0.

We consider two different ROMs, corresponding to $k = 32$ and $k = 78$ reactions selected in the reduced mechanism and denoted by ROM_{32} and ROM_{78}, respectively. The time evolution of O_2, CH_4, and CO_2, as well as T/T_0 are shown in Figure 6.2. The full system solution is depicted with a thin solid line, while the ROM solutions are depicted with thick lines (solid for ROM_{32} and dashed for ROM_{78}).

For each ROM the similarity indexes $\mathcal{S}_1$ and $\mathcal{S}_2$ were calculated using $q = 5$ SCE unit vectors. The model parameters were defined to be scaling factors, with nominal values of 1.0, for the $m = 177$ reaction rates in the full model. The model response was taken to be the set of all species concentrations; i.e., $h(t, y, p) = c(t)$. Table 6.2 provides the similarity indexes at two different times $t_1 = 5$ and $t_2 = 10$, before and after ignition, respectively.

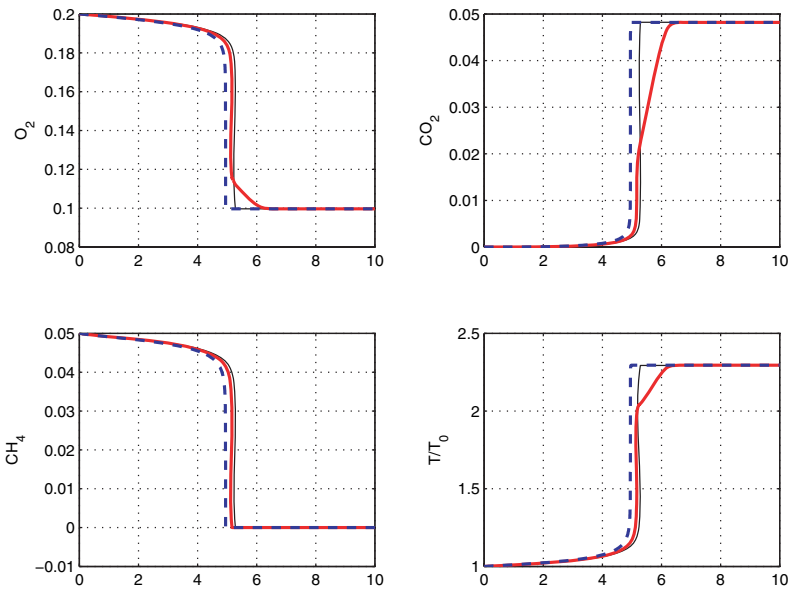


FIG. 6.2. *GRI combustion example: Full system (thin line) and ROM solutions (*ROM_{32}*: thick solid line,* ROM_{78}*: thick dashed line).*

TABLE 6.2
GRI combustion example: Similarity indexes.

	ROM_{32}			ROM_{78}	
	$t_1 = 5$	$t_2 = 10$		$t_1 = 5$	$t_2 = 10$
$\mathcal{S}_1$	0.8991	0.6076	$\mathcal{S}_1$	0.3656	0.6306
$\mathcal{S}_2$	0.4992	$9.366 \cdot 10^{-3}$	$\mathcal{S}_2$	$1.870 \cdot 10^{-2}$	$7.206 \cdot 10^{-2}$

As these results show, ROM_{78} is more robust than ROM_{32} based on the similarity indexes at $t_2 = 10$ but much less robust at the time before ignition $t_1 = 5$. To explain this (apparently) surprising result, note first that the selection of reactions kept in the reduced-order models was based on minimizing the norm $\|e\|_{L_2} = \|y - \tilde{y}\|_{L_2}$ over the time interval $[t_0, t_2]$. For ROM_{78} we have $\|e^*_{78}\| = 1.3736 \cdot 10^{-2}$, while for ROM_{32}, $\|e^*_{32}\| = 3.0349 \cdot 10^{-1}$. In this global measure, ROM_{78} is a better reduced-order model, but, as can be seen in Figure 6.2, at the price of predicting an earlier ignition. Second, note that a parameter perturbation along the direction of largest error growth at $t_1 = 5$ for the full model also leads to an earlier ignition time. As a consequence, ROM_{32} is better at predicting a perturbation-induced hastening of the ignition. This can also be seen in Figure 6.3, which depicts the time evolution up to $t_1 = 5$ of the temperature for the perturbed and unperturbed, full and reduced-order models.

6.3. Heat shock model. The previous reduced models were obtained via automatic procedures that are not specifically designed with the objective of robustness of the reduced model to perturbations in the problem parameters. Those reduced models did not perform well in that situation. The following reduced model was obtained using physics-based methods. It turns out to yield a robust reduced model. It is a DAE system, but this fact is not so important in the interpretation of the results.

This problem models the production of heat shock proteins in *Escherichia coli* cells as a response to extreme heat. These special proteins refold other cell proteins,

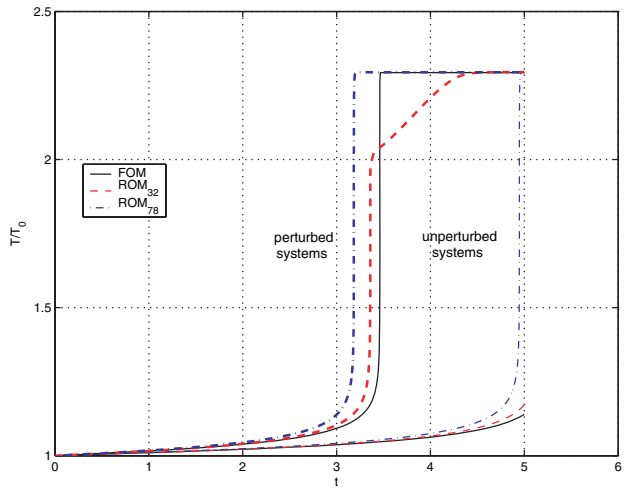


FIG. 6.3. *GRI combustion example: Evolution of the temperature over the time interval* $[t_0, t_1]$ *for the full model (solid line),* ROM_{32} *(dashed line), and* ROM_{78} *(dotted-dashed line). Both unperturbed solutions (thin lines) and perturbed solutions (thick lines) are represented. The perturbation is in the direction of maximum error growth in the norm of the concentrations at time* $t_1 = 5$ *for the full model.*

essential to cell survival, that have become unfolded due to excessive heat. The optimization problem is to find a balance between the effect of the protective proteins and the cost of their production. The mathematical model of the heat shock response is reported in [15]. The model employs first-order kinetics (law of mass action) to describe the various components of the heat shock system, namely, the synthesis of heat shock proteins and their regulator σ_{32}, in addition to protein folding and the association/dissociation activity of molecules. Reactions involving association/dissociation of molecules are often faster than synthesis and degradation of new molecules. Therefore the model exhibits a wide range of time scales and is subject to numerical stiffness. The differential equations that describe the fast states were transformed into algebraic constraints by means of a partial equilibrium approximation, yielding an index-1 DAE system. This system describes the evolution of 11 differential variables, coupled to the evolution of 20 algebraic variables, and includes 31 parameters:

$$\begin{aligned} &y' = F(t, y, x, p^{[1]}, p^{[2]}) = 0, \\ &G(t, y, x, p^{[2]}) = 0, \\ &y(0) = y_0, \;\; x(0) = x_0, \end{aligned}$$

where we have split the model parameters $p \in \mathbb{R}^{31}$ into two subsets, with the parameters $p^{[1]} \in \mathbb{R}^{11}$ appearing only in the differential equations. The integration time interval was taken such that the heat shock response occurs exactly at the middle of the time interval.

The reduced model is obtained in two steps. During the first step we reduce the number of differential variables from 11 to 5 by using an approximation which relates 6 variables (through a proportionality constant) to 2 other variables. The second step reduces further the number of variables by 15, taking advantage of the fact that the reduced variables can be well approximated by constants before and after the heat shock response. In the heat shock response region their behavior is highly nonlinear,

TABLE 6.3

Heat shock example: SV analysis results. Errors induced by parameter perturbations along the direction of maximum error growth are compared against errors induced by perturbations in random directions.

Full model			Reduced-order model		
δ in:	$p^{[1]}$	$p^{[1]}$ and $p^{[2]}$	δ in:	$p^{[1]}$	$p^{[1]}$ and $p^{[2]}$
$\mathcal{E}$	$8.2541 \cdot 10^4$	$5.6598 \cdot 10^4$	$\tilde{\mathcal{E}}$	$5.2887 \cdot 10^3$	$7.4958 \cdot 10^3$
$\|e(t,u_1)\|$	$6.0817 \cdot 10^4$	$6.7626 \cdot 10^4$	$\|\tilde{e}(t,\tilde{u}_1)\|$	$6.6292 \cdot 10^3$	$6.6292 \cdot 10^3$
$\|e(t,u_{rnd})\|$	$1.7648 \cdot 10^4$	$4.2042 \cdot 10^3$	$\|\tilde{e}(t,\tilde{u}_{rnd})\|$	$1.8461 \cdot 10^3$	$4.7902 \cdot 10^2$
	$1.9156 \cdot 10^4$	$2.2488 \cdot 10^4$		$1.3622 \cdot 10^3$	$1.3374 \cdot 10^3$
	$1.8371 \cdot 10^4$	$7.3998 \cdot 10^3$		$2.2759 \cdot 10^3$	$1.1942 \cdot 10^4$
	$1.9269 \cdot 10^3$	$9.5864 \cdot 10^3$		$1.4381 \cdot 10^3$	$1.2135 \cdot 10^3$

but the length of this region is much smaller than the length of any neighboring region, allowing us to make the approximation.

We analyze the full and reduced-order models under perturbations in the model parameters. The following two cases are considered:

(a) Perturbation only in the subset $p^{[1]}$ of model parameters representing those parameters that do *not* appear in the algebraic equations.

(b) Perturbation in all model parameters $p^{[1]}$ and $p^{[2]}$.

For both the full and reduced-order models, the direction of maximum error growth was estimated with the method described in section 3 using 4 adjoint systems, i.e., 4 unit vectors in the SCE estimates. We considered perturbations either in only the subset $p^{[1]}$ of model parameters or in all of the model parameters. In both cases, all of the states of the corresponding model were included in the perturbation-induced errors. Table 6.3 provides the estimated value $\mathcal{E}$ compared against the norm of the error along the estimated direction of maximum error growth computed by integrating the forward error equation $\|e(t,u_1)\|$. We also provide the norms of errors due to perturbations along a few random directions $\|e(t,u_{rnd})\|$ to emphasize the fact that it is very unlikely to get close to the "worst-case" scenario by only using random perturbations. All perturbations were $0.01 \cdot u$ (with u either $u^* = u_1$ or some random unit vector).

Next, we computed the similarity indexes of section 4 for this example. If we base the errors on all states of the ROM, the two indexes are as follows:

	Perturbation in $p^{[1]}$	Perturbation in $p^{[1]}$ and $p^{[2]}$
$\mathcal{S}_1$	0.9788	0.4844
$\mathcal{S}_2$	0.1089	0.0882

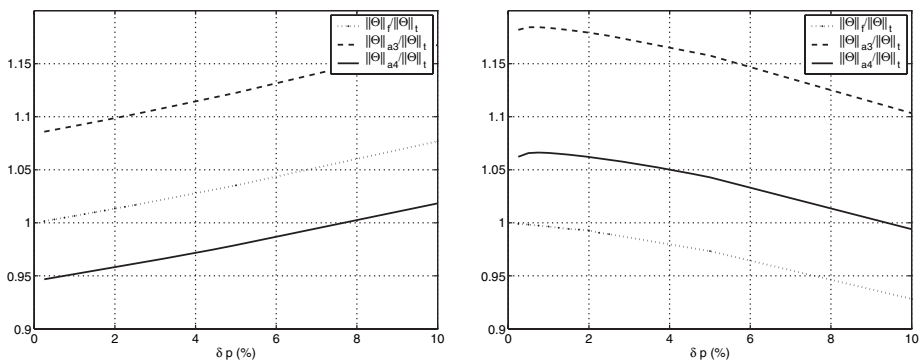
Considering only the differential variables in the ROM in the error estimates, the reduced-order model is found to be closer (in the sense of these similarity indexes) to the full model:

	Perturbation in $p^{[1]}$	Perturbation in $p^{[1]}$ and $p^{[2]}$
$\mathcal{S}_1$	0.9998	0.7340
$\mathcal{S}_2$	0.1499	0.1213

Table 6.4 summarizes the analysis of the effect of parameter perturbations on the ROM through the estimation of the norm of the cumulative error due to model reduction and perturbations in parameters. The same information (in a slightly dif-

TABLE 6.4
Heat shock example: Cumulative error due to model reduction and perturbations in parameters.

δp	Perturbation in $p^{[1]}$			Perturbation in $p^{[1]}$ and $p^{[2]}$		
(%)	$\|\Theta\|_{true}$	$\|\Theta\|_{fwd}$	$\|\Theta\|_{SCE}$	$\|\Theta\|_{true}$	$\|\Theta\|_{fwd}$	$\|\Theta\|_{SCE}$
0.25	$7.498 \cdot 10^2$	$7.510 \cdot 10^2$	$7.098 \cdot 10^2$	$8.437 \cdot 10^1$	$8.428 \cdot 10^1$	$8.962 \cdot 10^1$
0.50	$1.497 \cdot 10^3$	$1.502 \cdot 10^3$	$1.420 \cdot 10^3$	$1.722 \cdot 10^2$	$1.719 \cdot 10^2$	$1.835 \cdot 10^2$
0.75	$2.243 \cdot 10^3$	$2.254 \cdot 10^3$	$2.131 \cdot 10^3$	$2.601 \cdot 10^2$	$2.595 \cdot 10^2$	$2.774 \cdot 10^2$
1.00	$2.985 \cdot 10^3$	$3.005 \cdot 10^3$	$2.841 \cdot 10^3$	$3.493 \cdot 10^2$	$3.481 \cdot 10^2$	$3.723 \cdot 10^2$
1.25	$3.726 \cdot 10^3$	$3.756 \cdot 10^3$	$3.552 \cdot 10^3$	$4.367 \cdot 10^2$	$4.347 \cdot 10^2$	$4.652 \cdot 10^2$
1.50	$4.464 \cdot 10^3$	$4.508 \cdot 10^3$	$4.262 \cdot 10^3$	$5.253 \cdot 10^2$	$5.223 \cdot 10^2$	$5.590 \cdot 10^2$
1.75	$5.199 \cdot 10^3$	$5.259 \cdot 10^3$	$4.973 \cdot 10^3$	$6.141 \cdot 10^2$	$6.102 \cdot 10^2$	$6.529 \cdot 10^2$
2.00	$5.931 \cdot 10^3$	$6.011 \cdot 10^3$	$5.684 \cdot 10^3$	$7.032 \cdot 10^2$	$6.982 \cdot 10^2$	$7.468 \cdot 10^2$
2.25	$6.662 \cdot 10^3$	$6.762 \cdot 10^3$	$6.394 \cdot 10^3$	$7.920 \cdot 10^2$	$7.846 \cdot 10^2$	$8.402 \cdot 10^2$
2.50	$7.389 \cdot 10^3$	$7.514 \cdot 10^3$	$7.105 \cdot 10^3$	$8.822 \cdot 10^2$	$8.727 \cdot 10^2$	$9.346 \cdot 10^2$
5.00	$1.451 \cdot 10^4$	$1.502 \cdot 10^4$	$1.421 \cdot 10^4$	$1.796 \cdot 10^3$	$1.748 \cdot 10^3$	$1.873 \cdot 10^3$
10.00	$2.791 \cdot 10^4$	$3.005 \cdot 10^4$	$2.842 \cdot 10^4$	$3.774 \cdot 10^3$	$3.503 \cdot 10^3$	$3.751 \cdot 10^3$



(a) Only those parameters *not* affecting the algebraic equations were perturbed.

(b) All model parameters were perturbed.

FIG. 6.4. *Heat shock example: Cumulative error due to model reduction and perturbations in parameters. The plots show the ratio of the estimated norm of the cumulative error and the norm of the true (nonlinear) cumulative error for different magnitudes of parameter perturbations along some random direction.*

ferent form) is shown in Figure 6.4. The norms of the following quantities were computed: the true (nonlinear) error ($\|\Theta\|_{true}$), the estimation using integration of the forward error equation ($\|\Theta\|_{fwd}$), and the SCE estimate based on $q = 4$ unit vectors ($\|\Theta\|_{SCE}$). All of the ROM variables (which are a subset of the full model variables) were considered in these errors. The calculations were done for different perturbation magnitudes, given as a percentage of the norm of the perturbed parameters. The perturbation itself was along some random direction.

7. Conclusions. Our research extends the work presented in [11], focusing on estimating validity concepts for general nonlinear ROMs of dynamical models, with application to systems described by ODEs and DAEs. Our estimates, based on a combination of adjoint sensitivity analysis and the small-sample SCE, quantify how well a given ROM captures the changes in the output functional of the full system that result from perturbations in the problem parameters. We measure the extent to

which the ROM preserves the value of maximum perturbation-induced error (and/or the direction of the perturbations that induce it) by considering two types of similarity indexes. The direction in parameter space of maximum error growth in the output functional is efficiently computed within the framework of singular vector analysis, using the SCE estimate for the error norm.

Numerical examples (for both ODE and DAE systems) validate our approach. They also show that the similarity index concept is very effective for detecting situations where the ROM fails to capture the behavior of the full model when the parameters are slightly perturbed. Our numerical experiments indicate that it is quite possible, particularly when the reduced model is constructed via automatic procedures, to generate reduced models that agree well with the original data but are not robust to even small perturbations in the system parameters. This is perhaps not surprising, given that such automatic procedures are in general not specifically constructed with this kind of robustness in mind. We plan to address the construction of reduced models that are robust to perturbations in the problem data in future work.

REFERENCES

[1] A. Antoulas and D. Sorensen, *Approximation of large-scale dynamical systems: An overview*, Internat. J. Appl. Math. Comput. Sci., 11 (2001), pp. 1093–1121.

[2] U. M. Ascher and L. R. Petzold, *Stability of computational methods for constrained dynamics systems*, SIAM J. Sci. Comput., 14 (1993), pp. 95–120.

[3] K. E. Brenan, S. L. Campbell, and L. R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, SIAM, Philadelphia, PA, 1996.

[4] Y. Cao, S. Li, L. Petzold, and R. Serban, *Adjoint sensitivity analysis for differential-algebraic equations: The adjoint DAE system and its numerical solution*, SIAM J. Sci. Comput., 24 (2003), pp. 1076–1089.

[5] Y. Chen, D. Battisti, T. Palmer, J. Barsugli, and E. Sarachik, *Study of the predictability of tropical Pacific SST in a coupled atmosphere-ocean model using singular vector analysis: The role of the annual cycle and the ENSO cycle*, Mon. Weather Rev., 125 (1997), pp. 831–845.

[6] L. Dieci, R. D. Russell, and E. S. Van Vleck, *On the computation of Lyapunov exponents for continuous dynamical systems*, SIAM J. Numer. Anal., 34 (1997), pp. 402–423.

[7] P. Faucher, M. Gavart, and P. D. Mey, *Isopycnal empirical orthogonal functions (EOFs) in the north and tropical Atlantic and their use in estimation problems*, J. Geophys. Res-Oceans, 107 (2002), p. 3107.

[8] T. Gudmundsson, C. S. Kenney, and A. J. Laub, *Small-sample statistical estimates for matrix norms*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 776–792.

[9] S. Gugercin and A. C. Antoulas, *A survey of model reduction by balanced truncation and some new results*, Internat. J. Control, 77 (2004), pp. 748–766.

[10] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations* II, *Stiff and Differential-Algebraic Problems*, Springer-Verlag, Berlin, 1991.

[11] C. Homescu, L. R. Petzold, and R. Serban, *Error estimation for reduced-order models of dynamical systems*, SIAM J. Numer. Anal., 43 (2005), pp. 1693–1714.

[12] I. T. Jolliffe, *Principal Component Analysis*, Springer-Verlag, Berlin, 2002.

[13] C. S. Kenney and A. J. Laub, *Small-sample statistical condition estimates for general matrix functions*, SIAM J. Sci. Comput., 15 (1994), pp. 36–61.

[14] K. Kunisch and S. Volkwein, *Control of the Burgers equation by a reduced-order approach using proper orthogonal decomposition*, J. Optim. Theory Appl., 102 (1999), pp. 345–371.

[15] H. Kurata, H. El-Samad, T. Yi, M. Khammash, and J. Doyle, *Feedback regulation of the heat shock response in E. Coli*, in Proceedings of the 40th IEEE Conference on Decision and Control, 2001, pp. 837–842.

[16] H. Kuroda and M. Kishi, *A data assimilation technique applied to estimate parameters for the NEMURO marine ecosystem model*, Ecological Modelling, 172 (2004), pp. 69–85.

[17] Z. Li, I. Navon, and M. Hussaini, *Analysis of the singular vectors of the full-physics fsu global spectral model*, Tellus A, 57 (2005), pp. 560–574.

[18] E. Lorenz, *A study of the predictability of a* 28 *variable atmospheric model*, Tellus, 17 (1965), pp. 321–333.
[19] L. Petzold and W. Zhu, *Model reduction for chemical kinetics: An optimization approach*, AIChe J., 45 (1999), pp. 869–886.
[20] M. Rathinam and L. R. Petzold, *A new look at proper orthogonal decomposition*, SIAM J. Numer. Anal., 41 (2003), pp. 1893–1925.
[21] A. Trevisan and F. Pancotti, *Periodic orbits, Lyapunov vectors, and singular vectors in the lorenz system*, J. Atmospheric Sci., 55 (1998), pp. 390–398.
[22] P. Tupper, *Adaptive model reduction for chemical kinetics*, BIT, 42 (2002), pp. 447–465.

SIAM J. SCI. COMPUT.
Vol. 29, No. 6, pp. 2644–2667
© 2007 Society for Industrial and Applied Mathematics

NUMERICAL METHODS FOR TWO-PARAMETER LOCAL BIFURCATION ANALYSIS OF MAPS*

W. GOVAERTS†, R. KHOSHSIAR GHAZIANI‡, YU. A. KUZNETSOV§, AND H. G. E. MEIJER¶

Abstract. We discuss new and improved algorithms for the bifurcation analysis of fixed points and periodic orbits (cycles) of maps and their implementation in MATCONT, a MATLAB toolbox for continuation and bifurcation analysis of dynamical systems. This includes the numerical continuation of fixed points of iterates of the map with one control parameter, detecting and locating their bifurcation points (i.e., limit point, period-doubling, and Neimark–Sacker) and their continuation in two control parameters, as well as detection and location of all codimension 2 bifurcation points on the corresponding curves. For all bifurcations of codim 1 and 2, the critical normal form coefficients are computed, both numerically with finite directional differences and using symbolic derivatives of the original map. Using a parameter-dependent center manifold reduction, explicit asymptotics are derived for bifurcation curves of double and quadruple period cycles rooted at codim 2 points of cycles with arbitrary period. These asymptotics are implemented into the software and allow one to switch at codim 2 points to the continuation of the double and quadruple period bifurcations. We provide two examples illustrating the developed techniques: a generalized Hénon map and a juvenile/adult competition model from mathematical biology.

Key words. bifurcations of fixed points, cycles, normal forms, branch switching

AMS subject classifications. 34C20, 37G05, 65P30

DOI. 10.1137/060653858

1. Introduction.

Discrete-time dynamical systems generated by iterated maps appear in many scientific areas, such as economics, engineering, and ecology. The usual method for getting an insight into their behavior is to compute many orbits, starting from various initial points (i.e., *simulation*). Another tool for gaining a better understanding of these systems is *bifurcation analysis*. It provides theoretic results on the classification of possible modes of behavior, which may explain results of the simulations at different values of control parameters. The application of such theoretical results requires robust numerical computational tools, as a purely symbolic approach is often futile. In the present paper we describe how CL_MATCONT, which supports only ODEs [7, 8], has been complemented by the addition of the new package CL_MATCONTM, which supports local bifurcation analysis of generic finite-dimensional smooth maps. The software and a manual are freely available at http://www.matcont.ugent.be/. Before discussing and comparing existing computer tools for maps, we fix some notation and discuss some events which any standard

*Received by the editors March 7, 2006; accepted for publication (in revised form) April 4, 2007; published electronically November 2, 2007.

http://www.siam.org/journals/sisc/29-6/65385.html

†Department of Applied Mathematics and Computer Science, Ghent University, Krijgslaan 281-S9, B-9000 Gent, Belgium (Willy.Govaerts@UGent.be).

‡Department of Applied Mathematics and Computer Science, Ghent University, Krijgslaan 281-S9, B-9000 Gent, Belgium, and Department of Mathematics, Faculty of Science, Shahrekord University, P.O. Box 115, Shahrekord, Iran (Reza.KhoshsiarGhaziani@UGent.be).

§Department of Mathematics, Utrecht University, Budapestlaan 6, 3584 CD Utrecht, The Netherlands, and Institute of Mathematical Problems of Biology, Russian Academy of Sciences, Pushchino, Moscow Region, 142290 Russia (kuznet@math.uu.nl).

¶Department of Electrical Engineering, Mathematics and Computer Science, Universiteit Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands (meijerhge@math.utwente.nl).

software should be able to handle.

Consider a map

$$x \mapsto f(x, \alpha), \tag{1.1}$$

where $x \in \mathbb{R}^n$ is a state variable vector and $\alpha \in \mathbb{R}^p$ is a parameter vector. We assume that f is sufficiently smooth so that all partial derivatives below are well defined. Write the Kth iterate of (1.1) at some parameter value as

$$x \mapsto f^{(K)}(x, \alpha), \quad f^{(K)} : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^n, \tag{1.2}$$

where

$$g(x, \alpha) := f^{(K)}(x, \alpha) = \underbrace{f(f(f(\cdots f}_{K \text{ times}}(x, \alpha), \alpha), \alpha), \alpha). \tag{1.3}$$

The bifurcation analysis of (1.1) usually starts with fixed points. Numerically we continue fixed points of this map, i.e., solutions to the equation

$$F(x, \alpha) \equiv f(x, \alpha) - x = 0, \tag{1.4}$$

with one control parameter.

While varying one parameter, one may encounter codim 1 bifurcations of fixed points, i.e., critical parameter values where the stability of the fixed point changes. The eigenvalues of the Jacobian matrix of f are called multipliers. The fixed point is asymptotically stable if $|\lambda| < 1$ for every multiplier λ. If there exists a multiplier λ with $|\lambda| > 1$, then the fixed point is unstable. While following a curve of fixed points, three codim 1 singularities related to stability changes can generically occur, namely a *limit point* (*fold*, LP), a *period-doubling* (*flip*, PD), and a *Neimark–Sacker* (NS) point; see section 2 for details. Encountering such a bifurcation, one may use the formulas for the normal form coefficients derived via the center manifold reduction (see, e.g., [18, section 5.4]) to analyze the bifurcation. A nongeneric situation occurs at a branch point (BP) where the Jacobian matrix $[F_x(x, \alpha), F_\alpha(x, \alpha)]$ of (1.4) is rank deficient. Here the implicit function theorem cannot be applied to ensure the existence of a unique smooth branch of solutions. However, it is encountered often in practical problems that exhibit some form of symmetry (equivariance).

When two control parameters are allowed to vary, eleven codim 2 bifurcations can be met in generic families of maps (1.1), where curves of codim 1 bifurcations intersect or meet tangentially. The critical normal form coefficients for all generic codim 2 bifurcation points have been derived earlier in [19, 20] using a combined reduction-normalization technique. Note that the full verification of the genericity of a codim 2 bifurcation requires not only establishing its nondegeneracy at the critical parameter values but also a careful analysis of how the system depends on parameters. This "transversality" issue, which includes the expression of the canonical unfolding parameters of the normal form in terms of the original map parameters, has received little attention in the literature to date. When available, this information could be used to approximate codim 1 bifurcation curves that emanate from the codim 2 points.

There are several standard software packages supporting bifurcation analysis of iterated maps. Orbits of maps and one-dimensional invariant manifolds of saddle fixed points can be computed and visualized using DYNAMICS [24] and DsTool [3]. Location and continuation of fixed-point bifurcations is implemented in AUTO [9]

and the LBFP-version of LOCBIF [15]. The latter program computes the critical normal form coefficient at LP points and locates some codim 2 bifurcations along branches of codim 1 fixed points and cycles. CONTENT [17] was the first software that computed the critical normal form coefficients for all three codim 1 bifurcations of fixed points and cycles and allowed one to continue these bifurcations in two parameters and to detect all eleven codim 2 singularities along them. Branch switching at PD and BP points is also implemented in AUTO, LOCBIF, and CONTENT. However, only trivial branch switching is possible at codim 2 points and only for two (cusp and 1:1 resonance) of eleven codim 2 bifurcations are the critical normal form coefficients computed by the current version of CONTENT. No software supports switching at codim 2 points to the continuation of the double- and quadruple-period bifurcation curves. In the present paper we describe how CL_MATCONTM continues fixed points of an iterate of a map and handles the above-mentioned bifurcations.

The paper is organized as follows. In section 2, we first discuss continuation of fixed points of the iterated map (1.2) and detection of fold, flip, and NS bifurcations and branch points. After a short discussion of all generic codim 1 bifurcations of period-K orbits, we present the continuation of fold, flip, and NS curves of a cycle and detection of codim 2 points along these curves. The algorithms used here are similar to those implemented in CONTENT [12]. In section 3, we provide a (parameter-dependent) normal form for each codim 2 bifurcation point of period-K orbits in the minimal possible phase dimension. Then we consider the branch switching at codim 2 points, where we use parameter-dependent center-manifold reduction. In section 4 we present some algorithmic and numerical details, including the computation of partial derivatives of (1.2) up to and including order 5, using both symbolic partial derivatives of (1.1) and finite differences. In section 5 we demonstrate the use of the algorithms and implementation. Finally, in section 6 we draw some conclusions and mention a few problems for future work.

2. Continuation and detection for codim 1 bifurcations of fixed points and cycles.

2.1. Defining the system and singularities for cycles. Assume that for some $\alpha = \alpha_0$ and $K \geq 1$ the map g has a fixed point $x = x_0$, which is not a fixed point of $f^{(J)}(\cdot, \alpha_0)$ for $1 \leq J < K$ when $K > 1$. In other words, x_0 is a fixed point or a cycle with minimal period K of $f(\cdot, \alpha_0)$.

Each point x of the cycle of period K then satisfies the fixed point equation for the Kth iterate

$$g(x, \alpha) - x = 0. \tag{2.1}$$

As in CONTENT, branches of period-K cycles are computed in CL_MATCONTM by a variant of the Gauss–Newton continuation algorithm [1] applied to (2.1); see [7, 8].

If the Jacobian matrix $A = g_x(x_0, \alpha_0)$ has no eigenvalue λ with $|\lambda| = 1$, i.e., for a hyperbolic fixed point, the dynamics near the origin is topologically equivalent to that of the linear map $x \mapsto Ax$ by the Grobman–Hartman theorem. If eigenvalues with $|\lambda| = 1$ are present, the center manifold theorem guarantees the existence of local stable, unstable, and center invariant manifolds near the fixed point for parameter values close to α_0. Exact formulations of these theorems, together with references to the original papers, can be found in any standard book on dynamical systems, e.g., [2]. On the stable and unstable manifolds, the local dynamics is still determined by the linear part of the map. In contrast, the dynamics in the center manifolds depends on

both linear and nonlinear terms. Not all nonlinear terms are equally important, since some of them can be eliminated by an appropriate smooth coordinate transformation that depends smoothly on parameters and that puts the map restricted to the center manifold into a normal form, at least up to some order. Nonhyperbolic fixed points bifurcate; i.e., the dynamics near such points changes topologically under parameter variations. The birth of extra invariant objects, such as cycles or tori, is described by a parameter-dependent normal form of the restriction of g to a center manifold. Even though neither the center manifold nor the normal form on it is unique, the qualitative conclusions do not depend on the choices that are made [2, p. 266].

For the computations of the center manifold we write the Taylor expansion of g about (x_0, α_0):

$$\begin{aligned} g(x_0+x,\alpha_0+\alpha) = x_0 &+ Ax + \frac{1}{2}B(x,x) + \frac{1}{6}C(x,x,x) \\ &+ \frac{1}{24}D(x,x,x,x) + \frac{1}{120}E(x,x,x,x,x) \\ &+ J_1\alpha + \frac{1}{2}J_2(\alpha,\alpha) \\ &+ A_1(x,\alpha) + \frac{1}{2}B_1(x,x,\alpha) + \frac{1}{6}C_1(x,x,x,\alpha) + \frac{1}{24}D_1(x,x,x,x,\alpha) \\ &+ \frac{1}{2}A_2(x,\alpha,\alpha) + \frac{1}{4}B_2(x,x,\alpha,\alpha) + \frac{1}{12}C_2(x,x,x,\alpha,\alpha) \\ &+ \cdots, \end{aligned} \tag{2.2}$$

where all functions are multilinear forms of their arguments and the dots denote higher order terms in x and α. In particular, $A = g_x(x_0, \alpha)$, and the components of the multilinear functions B and C are given by

$$B_i(u,v) = \sum_{j,k=1}^{n} \frac{\partial^2 g_i(x_0,\alpha_0)}{\partial x_j \partial x_k} u_j v_k, \quad C_i(u,v,w) = \sum_{j,k,l=1}^{n} \frac{\partial^3 g_i(x_0,\alpha_0)}{\partial x_j \partial x_k \partial x_l} u_j v_k w_l, \tag{2.3}$$

for $i = 1, 2, \ldots, n$. From now on, I_n is the unit $n \times n$ matrix and $\|x\| = \sqrt{\langle x, x\rangle}$, where $\langle u, v\rangle = \bar{u}^T v$ is the standard scalar product in $\mathbb{C}^n$ (or $\mathbb{R}^n$).

Generically, the following events happen in the state space near a bifurcation (see, for example, [18]). When the control parameter crosses the critical value corresponding to a fold (LP) bifurcation, two fixed points of g collide and disappear, provided that $a_1 \neq 0$. This implies the collision of two period-K cycles of the original map f. When the control parameter crosses the critical value corresponding to a flip (PD) bifurcation and $b_1 \neq 0$, a cycle of period 2 for g bifurcates from the fixed point, which is a cycle of period $2K$ for map f. Finally, provided that $c_1 = \Re(d_1) \neq 0$, a unique closed invariant curve for g around the fixed point appears on the center manifold, when the control parameter crosses the critical value corresponding to the Neimark–Sacker (NS) bifurcation. For the original map, this means the appearance of K disjoint curves, cyclically shifted by f.

To detect the aforementioned bifurcations, as well as branch points of (2.1), we use in CL_MATCONTM the standard test functions listed in Table 2.1, where t is the tangent vector to the curve (2.1) in the X-space for $X = (x, \alpha)^T$, $F(X) = g(x, \alpha) - x$, $A = g_x$ is the Jacobian matrix of $g = f^{(K)}$, and $\odot$ is the *bialternate matrix product*; see, e.g., [13, section 4.4.4]. We notice that $\det(A \odot A - I_m)$, where $m = \frac{1}{2}\, n(n-1)$,

TABLE 2.1
Detection of codim 1 *bifurcations of cycles.*

Bifurcation	Test function(s)
LP	$t_{n+1} = 0, \begin{pmatrix} F_X \\ t^T \end{pmatrix} \neq 0$
PD	$\det(A + I_n) = 0$
NS	$\det(A \odot A - I_m) = 0$
BP	$\det \begin{pmatrix} F_X \\ t^T \end{pmatrix} = 0$

also vanishes at neutral saddles. We distinguish true NS bifurcations from neutral saddles when processing the NS points.

2.2. Defining systems and singularities for codim 1 bifurcations of cycles. In CL_MATCONTM, LP, PD, and NS curves for period-K cycles are computed by the mentioned Gauss–Newton continuation algorithm applied to *minimally extended defining systems*; cf. [13]. These systems were first implemented, together with the *standard extended defining systems*, in CONTENT [12]. We have adopted in CL_MATCONTM the most robust and efficient methods tested there. For completeness we include a brief description of the defining systems, as we will refer to them in section 4 and use some of the auxiliary variables for the detection of singularities.

The *limit point curve* and *period-doubling curve* are both defined by the following system:

$$\begin{cases} g(x,\alpha) - x &= 0, \\ s(x,\alpha) &= 0, \end{cases} \tag{2.4}$$

where $(x,\alpha) \in \mathbb{R}^{n+2}$, g is given by (1.3), while s is obtained by solving one of the algebraic systems

$$\begin{pmatrix} g_x(x,\alpha) \mp I_n & w_{bor} \\ v_{bor}^T & 0 \end{pmatrix} \begin{pmatrix} v \\ s \end{pmatrix} = \begin{pmatrix} 0_n \\ 1 \end{pmatrix}, \tag{2.5}$$

where $w_{bor}, v_{bor} \in \mathbb{R}^n$ are chosen such that the matrix in (2.5) is nonsingular. One should take the "$-$" sign in (2.5) for the LP-curve and the "$+$" sign for the PD-curve. The derivatives of s can be obtained easily from the derivatives of $g_x(x,\alpha)$:

$$s_z = -w^T (g_x)_z v, \tag{2.6}$$

where z is a state variable or an active parameter and w is obtained by solving

$$\begin{pmatrix} g_x^T(x,\alpha) \mp I_n & v_{bor} \\ w_{bor}^T & 0 \end{pmatrix} \begin{pmatrix} w \\ s \end{pmatrix} = \begin{pmatrix} 0_n \\ 1 \end{pmatrix}. \tag{2.7}$$

We note that the quantities called s in (2.5) and (2.7) are the same since they are both equal to the bottom-right element of the inverse of the square matrix in (2.5).

The *NS* and *neutral-saddle curves* are defined by the following system:

$$\begin{cases} g(x,\alpha) - x = 0, \\ s_{i_1 j_1}(x,\alpha,\kappa) = 0, \\ s_{i_2 j_2}(x,\alpha,\kappa) = 0, \end{cases} \tag{2.8}$$

TABLE 2.2
Detection of codim 2 bifurcations of cycles.

	LP	PD	NS
CP	$a_1 = 0$		
GPD		$b_1 = 0$	
CH			$c_1 = \Re(d_1) = 0$
R1	$\langle w, v\rangle = 0$		$\det(A - I_n) = \kappa - 1 = 0$
R2		$\langle w, v\rangle = 0$	$\det(A + I_n) = \kappa + 1 = 0$
R3			$\kappa + \frac{1}{2} = 0$
R4			$\kappa = 0$
LPPD	$\det(A + I_n) = 0$	$\det(A - I_n) = 0$	
LPNS	$\det(A \odot A - I_m) = 0$		$\det(A - I_n) = 0, \kappa - 1 \neq 0$
PDNS		$\det(A \odot A - I_m) = 0$	$\det(A + I_n) = 0, \kappa + 1 \neq 0$
NSNS			$\det(A_1 \odot A_1 - I_{m_1}) = 0$

TABLE 2.3
Smooth normal forms for generic codim 1 bifurcations of fixed points on center manifolds.

	Eigenvectors	Normal form	Critical coefficients
LP	$Aq = q$ $A^T p = p$	$w \mapsto \beta + w + a_1 w^2$ $+O(w^3), \quad w \in \mathbb{R}$	$a_1 = \frac{1}{2}\langle p, B(q,q)\rangle$
PD	$Aq = -q$ $A^T p = -p$	$w \mapsto -(1+\beta)w + b_1 w^3$ $+O(w^4), \quad w \in \mathbb{R}$	$b_1 = \frac{1}{6}\langle p, C(q,q,q) + 3B(q, h_{200})\rangle$ $h_{200} = (I_n - A)^{-1} B(q,q)$
NS	$Aq = \mathrm{e}^{i\theta_0} q$ $A^T p = \mathrm{e}^{-i\theta_0} p$ $\mathrm{e}^{i\nu\theta_0} \neq 1$ $\nu = 1, 2, 3, 4.$	$w \mapsto w e^{i\theta}\left(1 + \beta + d_1 \lvert w\rvert^2\right)$ $+ O(\lvert w\rvert^4), \quad w \in \mathbb{C}$	$d_1 = \frac{1}{2} e^{-i\theta_0}\langle p, C(q,q,\bar{q})$ $+ 2B(q, h_{1100})$ $+ B(\bar{q}, h_{2000})\rangle$ $h_{1100} = (I_n - A)^{-1} B(q,\bar{q})$ $h_{2000} = (e^{2i\theta_0} I_n - A)^{-1} B(q,q)$

i.e. by $n + 2$ equations for the $(n + 3)$ unknowns $x \in \mathbb{R}^n$, $\alpha \in \mathbb{R}^2$, $\kappa \in \mathbb{R}$. Here $(i_1, j_1, i_2, j_2) \in \{1, 2\}$ and $s_{i,j}$ are the components of S,

$$S = \begin{pmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{pmatrix},$$

which is obtained by solving

$$\begin{pmatrix} (g_x)^2(x,\alpha) - 2\kappa g_x + I_n & W_{bor} \\ V_{bor}^T & O \end{pmatrix} \begin{pmatrix} V \\ S \end{pmatrix} = \begin{pmatrix} 0_{n,2} \\ I_2 \end{pmatrix}, \tag{2.9}$$

where $V_{bor}, W_{bor} \in \mathbb{R}^{n\times 2}$ are chosen (and can be adapted) so that the matrix in (2.9) is nonsingular. Along the NS curve, κ is the real part of the critical multipliers $\mathrm{e}^{\pm i\theta}$. The derivatives of s_{ij} can be obtained easily from the derivatives of $g_x(x,\alpha)$ as before.

Table 2.2 specifies test functions used in CL_MATCONTM to detect and locate relevant codim 2 singularities along the codim 1 bifurcation curves. Here a_1 and b_1 are the critical normal form coefficients given in Table 2.3, where q and p should be replaced by the vectors v and w, respectively, computed in (2.5) and (2.7). The test function c_1 is given by the *Lyapunov coefficient* $c_1 = \Re(d_1)$, where d_1 is also specified in Table 2.3. Matrix A_1 is defined as $A_1 = A|_{N^\perp}$, where $A = g_x$ and $N^\perp$ is the orthogonal complement in $\mathbb{R}^n$ of the two-dimensional eigenspace associated with the pair of multipliers with unit product of A^T; $m_1 = \frac{1}{2}(n-2)(n-3)$.

It is possible and sometimes necessary to adapt the defining system while continuing a bifurcation curve, i.e., to update the auxiliary variables used in the defining

system of the computed branch. The bordering vectors v_{bor} and w_{bor} may require updating since they must ensure that the matrices in (2.5) and (2.7) are nonsingular. Updating is done in CL_MATCONTM by replacing v_{bor} and w_{bor} with the normalized vectors v, w computed in (2.5), (2.7), respectively. Updating of V and W in (2.9) is done similarly, while (i_1, j_1, i_2, j_2) are updated in such a way that the linearized system of (2.8) is as well-conditioned as possible.

3. Numerical analysis of codim 2 bifurcations.

3.1. Codim 2 cases. While, as mentioned in the introduction, methods to study codim 1 bifurcations are implemented in several packages [12], the numerical analysis of codim 2 bifurcations has remained a laborious task. We implemented in CL_MATCONTM the formulas for the critical center manifold reduction from [19, 20] for all eleven codim 2 bifurcations. Below we give normal forms to which the restriction of a generic map $g(x, \alpha) = f^{(K)}(x, \alpha)$ to the parameter-dependent center manifold can be transformed near the corresponding bifurcation by smooth invertible coordinate and parameter transformations. When CL_MATCONTM detects a codim 2 bifurcation, the critical coefficients of these normal forms are automatically computed and reported to the user, so that nondegeneracy is now easily checked. We stress that they can also be used to predict certain behavior of the system. In 6 out of 11 cases, branches of local codim 1 bifurcations are rooted at codim 2 bifurcations, and for these cases we also incorporate the parameter-dependent part of the normal form and provide asymptotic expressions for the new curves. In section 3.2 we specify how we switch the continuation to those branches.

The $\mathcal{O}$-symbol denotes higher order terms in phase-variables, the coefficients of which may also depend on parameters. But the qualitative picture is determined by the lowest order terms listed below. We refer to [18, Chapter 9] and [19, 20] for more details, including explicit expressions for all critical normal form coefficients which are not repeated in the present paper. If a complex critical eigenvalue λ is involved, it is always assumed that $\lambda^\nu \neq 1$ for $\nu = 1, 2, 3, 4$. In some cases, we combine two real unfolding parameters (β_1, β_2) into one complex parameter $\beta = \beta_1 + i\beta_2 \in \mathbb{C}$.

3.1.1. CP. The critical smooth normal form on the center manifold at a *cusp bifurcation* is

$$w \mapsto w + a_2 w^3 + \mathcal{O}(|w|^4), \quad w \in \mathbb{R}, \tag{3.1}$$

where, generically, $a_2 \neq 0$. Under this condition, a generic two-parameter unfolding of this singularity has two fold curves in the parameter plane which form a cuspidal wedge. For nearby parameter values, the map g has up to three fixed points that pairwise collide along the fold curves. In the direct product of the state and the parameter spaces, there is one smooth fold curve, so no branch switching is needed. CL_MATCONTM reports the value of a_2 at the bifurcation.

3.1.2. GPD. Near a *generalized flip* bifurcation the restriction of the map g to the parameter-dependent center manifold is smoothly equivalent to the normal form

$$w \mapsto -(1 + \beta_1)w + \beta_2 w^3 + b_2(\beta) w^5 + \mathcal{O}(|w|^6), \quad w \in \mathbb{R}, \tag{3.2}$$

where, generically, the coefficient $b_2(0) \neq 0$, while the components of $\beta = (\beta_1, \beta_2)$ are smooth functions of α, which can serve as new unfolding parameters. The value of $b_2(0)$ is reported by the software. The fixed point $w = 0$ of the map (3.2) exhibits a flip bifurcation for $\beta_1 = 0$. It is well known that from the point $\beta = 0$, corresponding

to the degenerate flip bifurcation, a fold curve of double-period cycles emanates. The asymptotic expression for this curve in (3.2) is given by

$$(w, \beta_1, \beta_2) = (\varepsilon, -b_2\varepsilon^4 + \mathcal{O}(\varepsilon^5), -2b_2\varepsilon^2 + \mathcal{O}(\varepsilon^3)). \tag{3.3}$$

3.1.3. CH. If $e^{i\nu\theta_0} \neq 1$ for $\nu = 1, 2, \ldots, 6$, the critical smooth normal form on the center manifold at the *Chenciner bifurcation* can be written as

$$z \mapsto ze^{i\theta_0}(1 + d_1|z|^2 + d_2|z|^4 + \mathcal{O}(|z|^6)), \quad z \in \mathbb{C}, \tag{3.4}$$

where $c_1 = \Re(d_1) = 0$ but, generically, $c_2 = \Re(d_2) + \frac{1}{2}\Im(d_1)^2 \neq 0$ and is reported to the user. A generic two-parameter unfolding of this singularity has a complicated bifurcation set due to the "collision" and destruction of two closed invariant curves of different stability born via the sub- and supercritical NS bifurcations, respectively. There are no cycle bifurcation curves rooted at this bifurcation.

3.1.4. R1. The restriction of the map at a 1:1 *resonance* to the corresponding center manifold can be written in the form

$$\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \mapsto \begin{pmatrix} w_1 + w_2 \\ w_2 + a_1 w_1^2 + b_1 w_1 w_2 \end{pmatrix} + \mathcal{O}(\|w\|^3), \quad w \in \mathbb{R}^2. \tag{3.5}$$

Generically, an NS bifurcation curve of the fixed point meets the fold bifurcation curve tangentially. CL_MATCONTM reports the sign of $a_1(b_1 - 2a_1)$, giving the first Lyapunov coefficient along the NS curve near the bifurcation. The local branch switching problem is trivial here, since both curves correspond to fixed points of g. The full bifurcation diagram near the codim 2 point is complicated and involves global bifurcations, e.g., tangencies of stable and unstable invariant manifolds of saddle fixed points of g and destruction of a closed invariant curve born via the NS bifurcation.

3.1.5. R2. Near a 1:2 resonance the restriction of the map g to the parameter-dependent center manifold is smoothly equivalent to the normal form

$$\begin{aligned} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \mapsto & \begin{pmatrix} -w_1 + w_2 \\ \beta_1 w_1 + (-1 + \beta_2) w_2 + C_1(\beta) w_1^3 + D_1(\beta) w_1^2 w_2 \end{pmatrix} \\ & + \mathcal{O}(\|w\|^4), \quad w \in \mathbb{R}^2, \end{aligned} \tag{3.6}$$

which depends on two unfolding parameters (β_1, β_2). If $C_1(0) < 0$, then there is an NS curve of fixed points of g with double period that emanates from the flip bifurcation curve $\beta_2 = 0$ of fixed points. It has the following asymptotic expression:

$$(w_1^2, w_2, \beta_1, \beta_2) = \left(-\frac{1}{C_1}, 0, 1, \left(2 + \frac{D_1}{C_1}\right)\right)\varepsilon + \mathcal{O}(\varepsilon^2). \tag{3.7}$$

There are also global bifurcations associated with the destruction of closed invariant curves. CL_MATCONTM reports the values of $4C_1(0)$ and $-2D_1(0) - 6C_1(0)$ (relevant for the flow approximation) to the user.

3.1.6. R3. At a 1:3 resonance the restriction of the map g to the parameter-dependent center manifold is smoothly equivalent to the normal form

$$z \mapsto (e^{2i\pi/3} + \beta)z + B_1(\beta)\bar{z}^2 + C_1(\beta)z|z|^2 + \mathcal{O}(|z|^4), \quad z \in \mathbb{C}, \tag{3.8}$$

where $\beta = \beta_1 + i\beta_2 \in \mathbb{C}$. A generic unfolding of this singularity has a period-3 saddle cycle that does not bifurcate for nearby parameter values, although it merges with the primary fixed point as the parameters approach R3. Only global bifurcations related to the destruction of a closed invariant curve born via the primary NS bifurcation occur in a neighborhood of this codim 2 point.

Note that the period-3 cycle becomes neutral near this bifurcation. Recall that a saddle cycle is called *neutral* if the corresponding fixed point has a pair of real eigenvalues with product 1. This singularity is important in analyzing global bifurcations of invariant manifolds of cycles. Moreover, the curve of neutral period-3 saddle cycles may turn into a true NS bifurcation at R1 or R2. Therefore, we give here an asymptotic of this curve.

First we need a vector field for which the time-1 flow approximates the third iterate of the map, i.e.,

$$\widetilde{g}(\eta, \widetilde{\beta}) = \widetilde{\beta}\eta + \bar{\eta}^2 + C_0(\beta)\eta^2\bar{\eta} + \mathcal{O}(|\eta|^4), \tag{3.9}$$

where

$$\widetilde{\beta} = 3\mathrm{e}^{-2i\pi/3}\beta, \quad z = \frac{1}{|B_1(\beta)|}\mathrm{e}^{i\arg(B_1(\beta))/3}\eta, \quad C_0(\beta) = \frac{1}{3}\left(\frac{C_1(\beta)}{|B_1(\beta)|^2}\mathrm{e}^{-2i\pi/3} - 1\right).$$

We write $C_0 = a + ib$, where a is reported upon detection, so that for $\eta = \rho\mathrm{e}^{i\phi}$ the neutral saddle curve has the following asymptotic expression:

$$(\rho, \phi, \beta_1, \beta_2) = \left(\varepsilon, s(\pi/6 - a\varepsilon/3), -2a\varepsilon^2, s\varepsilon - b\varepsilon^2\right) + \mathcal{O}(\varepsilon^3), \tag{3.10}$$

where $s = \pm 1$.

3.1.7. R4. Near a 1:4 resonance the restriction of the map g to the parameter-dependent center manifold is smoothly equivalent to the normal form

$$z \mapsto (i + \beta)z + C_1(\beta)z^2\bar{z} + D_1(\beta)\bar{z}^3 + \mathcal{O}(|z|^4), \quad z \in \mathbb{C}, \tag{3.11}$$

where $\beta = \beta_1 + i\beta_2 \in \mathbb{C}$. For this bifurcation we need not only this parameter-dependent normal form, but also an approximation of its fourth iterate by a unit-time shift along orbits of a vector field

$$\widetilde{g}(\eta, \widetilde{\beta}) = \widetilde{\beta}\eta + A_0(\beta)\eta^2\bar{\eta} + \bar{\eta}^3 + \mathcal{O}(|\eta|^4), \tag{3.12}$$

where $\eta \in \mathbb{C}$ and $\widetilde{\beta} = \tilde{\beta}_1 + i\tilde{\beta}_2$, $\tilde{\beta}_i \in \mathbb{R}$. Here we use

$$z = \frac{1}{\sqrt{|D_1(\beta)|}}\mathrm{e}^{i\arg(D_1(\beta))/4}\eta, \qquad A_0(\beta) = -i\frac{C_1(\beta)}{|D_1(\beta)|}.$$

Moreover, we have

$$\begin{pmatrix}\widetilde{\beta}_1\\ \widetilde{\beta}_2\end{pmatrix} = \begin{pmatrix}0 & 4\\ -4 & 0\end{pmatrix}\begin{pmatrix}\beta_1\\ \beta_2\end{pmatrix}. \tag{3.13}$$

There are three possible branch switches for this bifurcation. Denote the reported values by $a = \Re(A_0(0))$ and $b = \Im(A_0(0))$. If $\Delta \equiv a^2 + b^2 - 1 > 0$, then there are two half-lines $l_{1,2}$ of a limit-point curve of cycles with four times the original period. If

$$|b| > \frac{(1 + a^2)}{\sqrt{1 - a^2}},$$

then there is a curve n_1 along which a cycle of four times the primary period exhibits an NS bifurcation. Using $\eta = re^{i\phi}$, we have the following approximations:

$$l_{1,2} : (r^2, \phi, \widetilde{\beta}_1, \widetilde{\beta}_2) = \left(\varepsilon, \frac{1}{4}\arctan\left(\frac{ab \pm \sqrt{\Delta}}{b^2-1}\right) + \mathcal{O}(\varepsilon),\right.$$

$$\left.\frac{-a\Delta \mp b\sqrt{\Delta}}{a^2+b^2}\varepsilon, \frac{-b\Delta \pm a\sqrt{\Delta}}{a^2+b^2}\varepsilon\right) + \mathcal{O}(\varepsilon^2), \tag{3.14}$$

$$n_1 : (r^2, \phi, \widetilde{\beta}_1, \widetilde{\beta}_2) = \Big(\varepsilon + \mathcal{O}(\varepsilon^2), \operatorname{sign}(b)\arccos(a)/4 + \mathcal{O}(\varepsilon),$$

$$-2a\varepsilon + \mathcal{O}(\varepsilon^2), -(b - \operatorname{sign}(b)\sqrt{1-a^2})\varepsilon + \mathcal{O}(\varepsilon^2)\Big).$$

Taking into account (3.13), we obtain expressions for the unfolding parameters β_1 and β_2. If, in the formula for n_1, we replace $\operatorname{sign}(b)$ by $-\operatorname{sign}(b)$, then this gives the asymptotic for a neutral saddle singularity of the period-4 cycle.

Generically, there are also global bifurcations near `R4`.

3.1.8. LPPD. Near a *fold-flip* bifurcation, the restriction of the map g to the parameter-dependent center manifold is smoothly equivalent to the normal form

$$\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \mapsto \begin{pmatrix} \beta_1 + (1+\beta_2)w_1 + a(\beta)w_1^2 + b(\beta)w_2^2 + c_1(\beta)w_1^3 + c_2(\beta)w_1w_2^2 \\ -w_2 + e(\beta)w_1w_2 + c_3(\beta)w_1^2w_2 + c_4(\beta)w_2^3 \end{pmatrix} + \mathcal{O}(\|w\|^4), \quad w \in \mathbb{R}^2. \tag{3.15}$$

A new branch predicted by (3.15) for a generic map g is an NS bifurcation of double period that exists if $be > 0$ and has the asymptotic expression

$$(x, y^2, \beta_1, \beta_2) = \left(-\frac{c_4}{e}, 1, -b, -\frac{2b + ec_2 - 2(a+e)c_4}{e}\right)\varepsilon + \mathcal{O}(\varepsilon^2). \tag{3.16}$$

CL_MATCONTM reports the coefficients $\frac{a}{e}$ and be and the sign of the Lyapunov coefficient if applicable. As for the majority of the considered cases, there are also global bifurcations near this codim 2 point.

3.1.9. LPNS. For a *fold-NS* bifurcation, the critical normal form on the center manifold is given by

$$\begin{pmatrix} w \\ z \end{pmatrix} \mapsto \begin{pmatrix} w + sz\bar{z} + w^2 + cx^3 \\ e^{i\theta_0}z + awz + bzw^2 \end{pmatrix} + \mathcal{O}(\|(w,z)\|^4), \quad (w,z) \in \mathbb{R}\times\mathbb{C}. \tag{3.17}$$

The critical coefficients s, a, b, c are reported. Depending on their values, several bifurcation scenarios are possible in parameter-dependent unfoldings, which all involve global phenomena.

3.1.10. PDNS. Near a *flip-NS* bifurcation, the restriction of the map g to the parameter-dependent center manifold is smoothly equivalent to the parameter-dependent normal form

$$\begin{pmatrix} w \\ z \end{pmatrix} \mapsto \begin{pmatrix} -w(1 + \beta_1 + c_1(\beta)w^2 + c_2(\beta)|z|^2) \\ ze^{i\theta(\beta)}(1 + \beta_2 + c_3(\beta)w^2 + c_4(\beta)|z|^2) \end{pmatrix} + \mathcal{O}(\|(w,z)\|^4), \quad (w,z) \in \mathbb{R}\times\mathbb{C}, \tag{3.18}$$

where $\Re(c_i(0))$ are reported and $\theta(0) = \theta_0$. Besides global bifurcations, an NS bifurcation curve of double period for g is rooted at $\beta = 0$; it is always present. The asymptotic expression of this curve is given by

$$(w^2, z, \beta_1, \beta_2) = (1, 0, -c_1, -\text{sign}(c_1)\Re(c_3))\,\varepsilon + \mathcal{O}(\varepsilon^2). \tag{3.19}$$

3.1.11. NSNS. For a *double NS bifurcation*, provided that $l\theta_0 \neq j\theta_1$ for integer l and j with $l + j \leq 4$, the critical normal form on the center manifold is

$$\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \mapsto \begin{pmatrix} z_1(e^{i\theta_0} + c_1|z_1|^2 + c_2|z_2|^2) \\ z_2(e^{i\theta_1} + c_3|z_1|^2 + c_4|z_2|^2) \end{pmatrix} + \mathcal{O}(\|z\|^4), \quad z \in \mathbb{C}^2. \tag{3.20}$$

Depending on the values of $\Re(c_i)$, which are reported, several bifurcation scenarios are possible in parameter-dependent unfoldings, which all involve global phenomena. To analyze some of them, one has to take into account fourth- and fifth-order terms.

3.2. Branch switching from codim 2 bifurcations. Here we address the problem of branch switching at codim 2 bifurcation points of maps, when the emanating curve corresponds to a local bifurcation. This involves the degenerate flip, 1:2 resonance, 1:4 resonance, fold-flip, and flip-NS bifurcations only. To obtain appropriate initial continuation data for the original map, we combine parameter-dependent center-manifold reduction with asymptotic expressions for the new curves given in section 3.1.

Although we know that in several cases global bifurcations are also involved, we will not try to switch to those branches, as the continuation of these global bifurcations is not supported by the current version of CL_MATCONTM.

3.2.1. Parameter-dependent center-manifold reduction. In all our cases, the map $g(x, \alpha) : \mathbb{R}^n \times \mathbb{R}^2 \to \mathbb{R}^n$, where g is defined by (1.3), satisfies $g(x_0, \alpha_0) = x_0$, and its Jacobian matrix $A = g_x(x_0, \alpha_0)$ has at most three multipliers on the unit circle. Furthermore we know a parameter-dependent smooth normal form $G(w, \beta)$ for the corresponding bifurcation; see section 3.1. Then we assume a relation

$$\alpha - \alpha_0 = V(\beta) = v_{10}\beta_1 + v_{01}\beta_2 + \mathcal{O}(\|\beta\|^2) \tag{3.21}$$

between the original and the unfolding parameters. Note that V incorporates linear scalings. Occasionally, we interpret $\beta_1 = \bar{\beta}_2$ as one parameter $\beta \in \mathbb{C}$; in such cases, $v_{01} = \bar{v}_{10} \in \mathbb{C}^2$.

To find a parameter-dependent center manifold as the graph of $x = x_0 + H(w, \beta)$ we make a Taylor expansion of the *homological equation*

$$g(x_0 + H(w, \beta), \alpha_0 + V(\beta)) = H(G(w, \beta), \beta) \tag{3.22}$$

in w and β at $(w, \beta) = (0, 0)$, where we expand g as in (2.2) and write

$$H(w, \beta) = \sum_{|\mu|+|\nu|\geq 1} h_{\mu,\nu} w^\mu \beta^\nu, \tag{3.23}$$

and μ, ν are multi-indices. All coefficients must vanish, and this leads to a solution for H and V. Below we will focus only on the parameter-dependent computations and assume full knowledge of the critical center manifold and the critical normal form coefficients; see [18, 19, 20]. The solvability conditions imposed coincide with the transversality of the original family to the bifurcation manifold. A similar technique

was introduced in [5, section 11] to switch at codim 2 bifurcations of equilibria in ODEs.

Initial data for the new curve is now provided by substituting for w and β the asymptotic expression in ε of section 3.1 into H and V up to a certain order in ε, usually 2. In CL_MATCONTM the user can specify ε to obtain a starting point near the emanating curve for which the continuation code converges to a point on the new curve.

It will be convenient to introduce some notation. Let p denote an eigenvector of A^T corresponding to the eigenvalue -1 of A. We will then write $\Gamma : \mathbb{R}^{n+2} \to \mathbb{R}^n$ for $\Gamma(q,v) = \langle p, A_1(q,v) + B(q,(I_n - A)^{-1}J_1 v)\rangle$ and $\gamma_i = \Gamma(q,e_i)$ for the evaluation of Γ on the standard basis vectors in $\mathbb{R}^2$. If $\gamma_i \neq 0$ for $i = 1,2$, then $s_1 = \frac{1}{(\gamma_1^2+\gamma_2^2)}(\gamma_1,\gamma_2)^T$ and $s_2 = (-\gamma_2,\gamma_1)^T$ compose a new orthogonal basis in $\mathbb{R}^2$.

3.2.2. Degenerate flip. The homological equation (3.22) provides the following systems to be solved:

$$(A - I_n)[h_{010},\ h_{001}] = -J_1[v_{10},\ v_{01}], \tag{3.24}$$

$$(A + I_n)[h_{110},\ h_{101}] = -[q,\ 0] - A_1(q,[v_{10},\ v_{01}]) - B(q,[h_{010},\ h_{001}]), \tag{3.25}$$

where $[a,\ b]$ is an $n \times 2$-matrix with columns $a, b \in \mathbb{R}^n$. The higher orders give

$$\begin{aligned}(A - I_n)h_{210} = 2h_{200} - [&B_1(q,q,v_{10}) + B(h_{200},h_{010}) + A_1(h_{200},v_{10}) \\ &+ 2B(q,h_{110}) + C(q,q,h_{010})]\,,\end{aligned} \tag{3.26}$$

$$\begin{aligned}(A - I_n)h_{201} = -[&B_1(q,q,v_{01}) + B(h_{200},h_{001}) + A_1(h_{200},v_{01}) \\ &+ 2B(q,h_{101}) + C(q,q,h_{001})]\,,\end{aligned} \tag{3.27}$$

$$\begin{aligned}(A + I_n)h_{310} = -3h_{300} - [&D(q,q,q,h_{010}) + 3C(q,q,h_{110}) + C_1(q,q,q,v_{10}) \\ &+ 3B(h_{110},h_{200}) + B(h_{300},h_{010}) + 3B_1(h_{200},q,v_{10}) \\ &+ A_1(h_{300},v_{10}) + 3C(h_{200},q,h_{010}) + 3B(h_{210},q)]\,,\end{aligned} \tag{3.28}$$

$$\begin{aligned}(A + I_n)h_{301} = 6q - [&D(q,q,q,h_{001}) + 3C(q,q,h_{101}) + C_1(q,q,q,v_{01}) \\ &+ 3B(h_{101},h_{200}) + B(h_{300},h_{001}) + 3B_1(h_{200},q,v_{01}) \\ &+ A_1(h_{300},v_{01}) + 3C(h_{200},q,h_{001}) + 3B(h_{201},q)]\,.\end{aligned} \tag{3.29}$$

The linear part of V involves four unknowns to be found from equations with singular left-hand sides, i.e., (3.25), (3.28), (3.29). For the solution we start with substitution of $[h_{010},\ h_{001}] = (I_n - A)^{-1}J_1[v_{10},\ v_{01}]$ from (3.24) into (3.25). As the left-hand side of (3.25) is singular, the right-hand side must be orthogonal to the adjoint eigenvector p. Application of the Fredholm alternative leads to

$$\begin{aligned}&\langle p, (A + I_n)[h_{110},\ h_{101}]\rangle \\ &\quad = -\langle p, [q,\ 0] + A_1(q,[v_{10},\ v_{01}]) + B(q,(I_n - A)^{-1}J_1[v_{10},v_{01}])\rangle.\end{aligned}$$

We see that the operator $\Gamma(q,v)$ appears naturally and rewrite it as

$$[\gamma_1,\ \gamma_2][v_{10},\ v_{01}] = [-1,\ 0].$$

The general solution is given by

$$v_{10} = -s_1 + \delta_1 s_2, \quad v_{01} = \delta_2 s_2, \quad \delta_1, \delta_2 \in \mathbb{R}.$$

Since v_{10}, v_{01} appear linearly in these equations (via the multilinear functions), application of the Fredholm alternative to (3.28), (3.29) results in a linear system for the constants δ_1, δ_2.

3.2.3. 1:2 resonance. As before, we first list the necessary equations obtained from the homological equation

$$(A - I_n)[h_{0010},\ h_{0001}] = -J_1[v_{10},\ v_{01}], \tag{3.30}$$

$$(A + I_n)[h_{1010},\ h_{1001}] = [q_1,\ 0] - A_1(q_0, [v_{10},\ v_{01}]) - B(q_0, [h_{0010},\ h_{0001}]), \tag{3.31}$$

$$\begin{aligned}(A + I_n)[h_{0110},\ h_{0101}] = {} & [h_{1010},\ q_1 + h_{1001}] - A_1(q_1, [v_{10},\ v_{01}]) \\ & - B(q_1, [h_{0010},\ h_{0001}]).\end{aligned} \tag{3.32}$$

As for the degenerate flip, we substitute $[h_{0010},\ h_{0001}] = (I_n - A)^{-1} J_1[v_{10},\ v_{01}]$ into (3.31). As (3.31) is similar to (3.25), the solution for v_{10} and v_{01} is now

$$v_{10} = s_1 + \delta_1 s_2, \quad v_{01} = \delta_2 s_2, \quad \delta_1, \delta_2 \in \mathbb{R}.$$

We substitute the s_i into (3.32) and write

$$Q_1 = \langle p_1, A_1(q_0, s_1) + B(q_0, (A - I_n)^{-1} J_1 s_1) \rangle, \qquad Q_2 = \Gamma(q_1, s_1),$$

$$Q_3 = \langle p_1, A_1(q_0, s_2) + B(q_0, (A - I_n)^{-1} J_1 s_2) \rangle, \qquad Q_4 = \Gamma(q_1, s_2).$$

A little algebra shows that

$$\delta_1 = -\left(\frac{Q_1 + Q_2}{Q_3 + Q_4}\right), \quad \delta_2 = \frac{1}{Q_3 + Q_4}.$$

3.2.4. 1:3 resonance. We follow a slightly different procedure here. We want to find $V(\beta) = v\beta + \bar{v}\bar{\beta}$, where $\beta = \beta_1 + i\beta_2$. Then we treat β and $\bar{\beta}$ as independent variables, which makes it slightly easier to find the solutions. As the final $V(\beta)$ should be real, it follows that $v = v_{10} = \bar{v}_{01}$.

Let $\lambda = \mathrm{e}^{2i\pi/3}$, and introduce $Aq = \lambda q,\ A^T p = \bar{\lambda} p,\ \langle p, q \rangle = 1$. As before, the first linear systems resulting from (3.22) are given by

$$(A - I_n)[h_{0010},\ h_{0001}] = -J_1[v_{10},\ v_{01}],$$

$$(A - \lambda I_n)[h_{1010},\ h_{1001}] = [q,\ 0] - A_1(q, [v_{10},\ v_{01}]) - B(q, [h_{0010},\ h_{0001}]),$$

and two complex conjugated systems for h_{0101} and h_{0110}. With the same approach we will now find complex γ_i, and rewriting the system for $v = v_{10} = \bar{v}_{01}$, we have $(\gamma_1, \gamma_2)v = 1, (\gamma_1, \gamma_2)\bar{v} = 0$, with $v = (\bar{\gamma}_2, -\bar{\gamma}_1)/(\gamma_1\bar{\gamma}_2 - \gamma_2\bar{\gamma}_1)$ as solution.

3.2.5. 1:4 resonance. Replacing $\lambda = i$, we can repeat the procedure for the case of 1:3 resonance.

3.2.6. Fold-flip. Let $Aq_{1,2} = \pm q_{1,2},\ A^T p_{1,2} = \pm p_{1,2},\ \langle p_1, q_1\rangle = \langle p_2, q_2\rangle = 1$. The necessary systems to solve from the homological equation (3.22) are

$$(A - I_n)[h_{0010},\ h_{0001}] = [q_1,\ 0] - J_1[v_{10},\ v_{01}], \tag{3.33}$$

$$\begin{aligned}(A - I_n)[h_{1010},\ h_{1001}] = {}& [h_{2000},\ q_1] - A_1(q_1, [v_{10},\ v_{01}]) \\ & - B(q_1, [h_{0010},\ h_{0001}]),\end{aligned} \tag{3.34}$$

$$\begin{aligned}(A + I_n)[h_{0110},\ h_{0101}] = {}& [h_{1100},\ 0] - A_1(q_2, [v_{10},\ v_{01}]) \\ & - B(q_2, [h_{0010},\ h_{0001}]).\end{aligned} \tag{3.35}$$

First note that all matrices in the left-hand sides are singular. If we take $(\gamma_1, \gamma_2) = p_1^T J_1$, we can define the orthogonal vectors s_1 and s_2 as before, and then $v_{10} = s_1 + \delta_1 s_2$ and $v_{01} = \delta_2 s_2$ solve system (3.33) for arbitrary δ_1, δ_2. Bordering the singular matrix $(A - I_n)$, one can solve for h_{0010} and h_{0001}. Any multiple of q_1 can be added to h_{0010} and h_{0001}, so we use $h_{0010} = (A - I_n)^{INV}(q_1 - J_1 v_{10}) + \delta_3 q_1$ and $h_{0010} = -(A - I_n)^{INV}(J_1 v_{01}) + \delta_4 q_1$. We will use this freedom to solve (3.34) and (3.35) simultaneously for all δ's. Note that h_{2000} and h_{1100} are also found using bordered systems chosen, but such that $\langle p_1, h_{2000}\rangle = \langle p_2, h_{1100}\rangle = 0$.

Then we obtain the following four-dimensional system:

$$\begin{pmatrix} L & 0_{2\times 2} \\ 0_{2\times 2} & L \end{pmatrix} \begin{pmatrix} \delta_1 \\ \delta_3 \\ \delta_2 \\ \delta_4 \end{pmatrix} = \begin{pmatrix} -\langle p_1, A_1(q_1, s_1) + B(q_1, (A - I_n)^{INV}(q_1 - J_1 s_1))\rangle \\ -\langle p_2, A_1(q_2, s_1) + B(q_2, (A - I_n)^{INV}(q_1 - J_1 s_1))\rangle \\ 1 \\ 0 \end{pmatrix}, \tag{3.36}$$

where L is defined by

$$L = \begin{pmatrix} \langle p_1, A_1(q_1, s_2) + B(q_1, (I_n - A)^{INV} J_1 s_2)\rangle & \langle p_1, B(q_1, q_1)\rangle \\ \langle p_2, A_1(q_2, s_2) + B(q_2, (I_n - A)^{INV} J_1 s_2)\rangle & \langle p_2, B(q_1, q_2)\rangle \end{pmatrix}. \tag{3.37}$$

Notice that $2a(0) = \langle p_1, B(q_1, q_1)\rangle$ and that q_1 can be chosen such that $e(0) = \langle p_2, B(q_1, q_2)\rangle = 1$. The condition $\gamma_1\gamma_2 \det(L) \neq 0$ is equivalent with the transversality to the bifurcation manifold of the family $g(x, \alpha)$.

3.2.7. Flip-NS. Introduce $Aq_1 = q_1,\ A^T p_1 = p_1,\ \langle p_1, q_1\rangle = 1$ and $Aq_2 = e^{i\theta_0} q_2,\ A^T p_2 = e^{-i\theta_0} p_2,\ \langle p_2, q_2\rangle = 1$. The linear systems obtained from the homological equation (3.22) are

$$(A - I_n)[h_{00010},\ h_{00001}] = -J_1[v_{10},\ v_{01}],$$

$$(A + I_n)[h_{10010},\ h_{10001}] = [-q_1,\ 0] - A_1(q_1, [v_{10},\ v_{01}]) - B(q_1, [h_{00010},\ h_{00001}]),$$

$$(A - e^{i\theta_0} I_n)[h_{01010},\ h_{01001}] = [0,\ q_2 e^{i\theta_0}] - A_1(q_2, [v_{10},\ v_{01}]) - B(q_2, [h_{00010},\ h_{00001}]).$$

The same approach as for the degenerate flip and 1:2-resonance cases is to substitute the formal solution of the first equation into the second, and we write

$$v_{10} = -s_1 + \delta_1 s_2, \quad v_{01} = \delta_2 s_2,$$

where the constants δ_i are to be found from the last equation. We compute

$$Q_i = \langle p_2, A_1(q_2, s_i) + B(q_2, (I_n - A)^{-1} J_1 s_i)\rangle$$

for $i = 1, 2$. To obtain the derivative of the modulus and not the argument of the complex multiplier, we proceed as in [26, Appendix] but adapt to the case of maps. Then we find the following real solutions:

$$\delta_1 = \frac{\Re(e^{-i\theta_0}Q_1)}{\Re(e^{-i\theta_0}Q_2)}, \qquad \delta_2 = -\frac{1}{\Re(e^{-i\theta_0}Q_2)}. \tag{3.38}$$

4. Algorithmic and numerical details. In this section we consider the computation of the derivatives and tensor-vector products, which are necessary not only for the continuation, but also for the computation of the critical normal form coefficients at codim 1 and 2 bifurcation points.

4.1. Recursive formulas for derivatives of iterates of maps.

4.1.1. Derivatives with respect to phase variables. The iteration of (1.1) gives rise to a sequence of points

$$\{x^1, x^2, x^3, \dots, x^{K+1}\},$$

where $x^{J+1} = f^{(J)}(x^1, \alpha)$ for $J = 1, 2, \dots, K$. Suppose that symbolic derivatives of f up to order 5 can be computed at each point. We write

$$A(x^J)_{i,j} = \frac{\partial f_i}{\partial x_j}(x^J), \quad B(x^J)_{i,j,k} = \frac{\partial^2 f_i}{\partial x_j \partial x_k}(x^J), \quad C(x^J)_{i,j,k,l} = \frac{\partial^3 f_i}{\partial x_j \partial x_k \partial x_l}(x^J),$$

and similarly for $D(x^J)$ and $E(x^J)$.

We want to find recursive formulas for the derivatives of the composition (1.2), i.e., the coefficients of the multilinear functions in (2.2) that we now denote by $A^{(J)}, B^{(J)}$, and $C^{(J)}$ to indicate the iterate explicitly:

$$(A^{(J)})_{i,j} = \frac{\partial (f^{(J)}(x^1))_i}{\partial x_j}, \quad (B^{(J)})_{i,j,k} = \frac{\partial^2 (f^{(J)}(x^1))_i}{\partial x_j \partial x_k}, \quad (C^{(J)})_{i,j,k,l} = \frac{\partial^3 (f^{(J)}(x^1))_i}{\partial x_j \partial x_k \partial x_l},$$

and $D^{(J)}$ and $E^{(J)}$ are analogously defined. What follows is a straightforward application of the chain rule.

For $J = 1$ we have $A^{(1)} = A(x^1), B^{(1)} = B(x^1)$, and $C^{(1)} = C(x^1)$, and these are known. Now, as

$$A^{(J)}_{i,j} = \sum_k \frac{\partial f_i}{\partial x_k}(f^{(J-1)}(x^1)) \frac{\partial (f^{(J-1)}(x^1))_k}{\partial x_j} = \sum_l A(x^J)_{i,k} A^{(J-1)}_{k,j}, \tag{4.1}$$

we see that

$$(F(x,\alpha))_x = A(x^K)A(x^{K-1}) \cdots A(x^1) - I_n, \tag{4.2}$$

where $F(x, \alpha) = f^{(K)}(x, \alpha) - x$.

For the second-order derivatives we first write $B^{(J)}$ once in coordinates

$$\begin{aligned} B^{(J)}_{i,j,k} &= \frac{\partial}{\partial x_j}\frac{\partial}{\partial x_k} f_i(f^{(J-1)}(x)) \\ &= \sum_{l,m} \frac{\partial^2 f_i}{\partial x_l \partial x_m}(x^J) \frac{\partial (f^{(J-1)})_m}{\partial x_j} \frac{\partial (f^{(J-1)})_l}{\partial x_k} + \sum_l \frac{\partial f_i}{\partial x_l}(x^J) \frac{\partial^2 (f^{(J-1)})_l}{\partial x_j \partial x_k}. \end{aligned}$$

For any two vectors q_1 and q_2, we can multiply the previous expression by $(q_1)_j(q_2)_k$ and sum over (j,k) to obtain

$$B^{(J)}(q_1,q_2) = B(x^J)(A^{(J-1)}q_1, A^{(J-1)}q_2) + A(x^J)B^{(J-1)}(q_1,q_2). \tag{4.3}$$

As $A(x^J)$ and $B(x^J)$ are known, (4.3) allows us to compute the multilinear form $B^{(J)}(q_1,q_2)$ recursively.

Let $q_i,\ i = 1,2,3,4,5$, be given vectors. Multilinear forms with higher order derivatives can be computed with

$$\begin{aligned} C^{(J)}(q_1,q_2,q_3) &= C(x^J)(A^{(J-1)}q_1, A^{(J-1)}q_2, A^{(J-1)}q_3) \\ &\quad + B(x^J)(B^{(J-1)}(q_1,q_2), A^{(J-1)}q_3)^* \\ &\quad + A(x^J)(C^{(J-1)}(q_1,q_2,q_3)), \end{aligned} \tag{4.4}$$

where * means that all combinatorially different terms have to be included, i.e.,

$$\begin{aligned} B(x^J)(B^{(J-1)}(q_1,q_2), A^{(J-1)}q_3)^* &= B(x^J)(B^{(J-1)}(q_1,q_2), A^{(J-1)}q_3) \\ &\quad + B(x^J)(B^{(J-1)}(q_1,q_3), A^{(J-1)}q_2) \\ &\quad + B(x^J)(B^{(J-1)}(q_2,q_3), A^{(J-1)}q_1). \end{aligned}$$

For $D^{(J)}$ we get

$$\begin{aligned} D^{(J)}(q_1,q_2,q_3,q_4) &= D(x^J)(A^{(J-1)}q_1, A^{(J-1)}q_2, A^{(J-1)}q_3, A^{(J-1)}q_4) \\ &\quad + C(x^J)(B^{(J-1)}(q_1,q_2), A^{(J-1)}q_3, A^{(J-1)}q_4)^* \\ &\quad + B(x^J)(B^{(J-1)}(q_1,q_2), B^{(J-1)}(q_3,q_4))^* \\ &\quad + B(x^J)(C^{(J-1)}(q_1,q_2,q_3), A^{(J-1)}q_4)^* \\ &\quad + A(x^J)D^{(J-1)}(q_1,q_2,q_3,q_4). \end{aligned} \tag{4.5}$$

Finally, for $E^{(J)}$ the following holds:

$$\begin{aligned} E^{(J)}(q_1,q_2,q_3,q_4,q_5) &= E(x^J)(A^{(J-1)}q_1, A^{(J-1)}q_2, A^{(J-1)}q_3, A^{(J-1)}q_4, A^{(J-1)}q_5) \\ &\quad + D(x^J)(B^{(J-1)}(q_1,q_2), A^{(J-1)}q_3, A^{(J-1)}q_4, A^{(J-1)}q_5)^* \\ &\quad + C(x^J)(B^{(J-1)}(q_1,q_2), B^{(J-1)}(q_3,q_4), A^{(J-1)}q_5)^* \\ &\quad + C(x^J)(C^{(J-1)}(q_1,q_2,q_3), A^{(J-1)}q_4, A^{(J-1)}q_5)^* \\ &\quad + B(x^J)(C^{(J-1)}(q_1,q_2,q_3), B^{(J-1)}(q_4,q_5))^* \\ &\quad + B(x^J)(D^{(J-1)}(q_1,q_2,q_3,q_4))(A^{(J-1)}q_5)^* \\ &\quad + A(x^J)(E^{(J-1)}(q_1,q_2,q_3,q_4,q_5)). \end{aligned} \tag{4.6}$$

The multilinear forms $A^{(K)}(q_1)$, $B^{(K)}(q_1,q_2)$, $C^{(K)}(q_1,q_2,q_3)$, $D^{(K)}(q_1,q_2,q_3,q_4)$, and $E^{(K)}(q_1,q_2,q_3,q_4,q_5)$ are then used in the computations of the normal form coefficients for codim 1 and codim 2 bifurcations of period-K cycles and also in branch switching.

4.1.2. Derivatives with respect to parameters. If enough symbolic derivatives of f are available, then MATCONT computes the expressions involving J_1 and A_1 in (2.2) symbolically. The idea is as follows. Taking the derivative of (1.3) with respect to α_k gives

$$\frac{\partial(f^{(J)}(x^1,\alpha))}{\partial\alpha_k} = \frac{\partial f}{\partial\alpha_k}(x^J,\alpha) + \frac{\partial f}{\partial x}(x^J,\alpha)\frac{\partial(f^{(J-1)}(x^1,\alpha))}{\partial\alpha_k}, \tag{4.7}$$

which is recursively computable. Also mixed derivatives, which are necessary for continuation and branch switching, can be found recursively:

$$\frac{\partial^2(f^{(J)}(x^1,\alpha))}{\partial\alpha_k\partial x} = \frac{\partial^2 f}{\partial\alpha_k\partial x}(x^J,\alpha) + \frac{\partial^2 f}{\partial x^2}(x^J,\alpha)\frac{\partial(f^{(J-1)}(x^1,\alpha))}{\partial\alpha_k}. \tag{4.8}$$

In fact, the recursion is not applied to (4.8) itself, but to its product with a fixed vector.

This is sufficient for all continuations of fixed points and their codim 1 bifurcations. It is also sufficient for all cases of branch switching from codim 2 points, except for the case of degenerate flip. For this case, we fall back to a finite difference approximation. Since it is used only in the prediction step for which high accuracy is not needed, this seems acceptable.

4.2. Recursive formulas for derivatives of the defining systems for continuation. For the continuation of fixed points and cycles we need the derivatives of (1.3), which can be computed from (4.2) and (4.7).

Now, we consider the derivatives of s (as defined in (2.4)) with respect to z, a state variable or parameter. The flip and NS cases can be handled in a similar way. Let M be the matrix in (2.5). By taking derivatives of (2.5) with respect to z, we obtain

$$M\begin{bmatrix} v_z \\ s_z \end{bmatrix} + \begin{bmatrix} A_z^{(K)} & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} v \\ s \end{bmatrix} = 0. \tag{4.9}$$

Using (2.7), we obtain

$$s_z = -w^T(A^{(K)})_z v. \tag{4.10}$$

If z represents one of the state variables, then $s_{x_i} = -\langle w, B^{(K)}(e_i, v)\rangle$, as computed in section 4.1. When z is a parameter α_k we can write

$$s_{\alpha_k} = \sum_{J=1}^{K} C_J, \tag{4.11}$$

where

$$C_J = -w^T f_x(x^K)\ \cdots\ (f_x(x^J))_{\alpha_k} f_x(x^{J-1})\ \cdots\ f_x(x^1)v, \tag{4.12}$$

where $J = 1,\dots,K$. In this expression

$$(f_x(x^J))_{\alpha_k} = [f_x(f^{(J)}(x^1,\alpha))]_{\alpha_k} = f_{x\alpha}(x^J,\alpha) + B(x^J)T_J, \tag{4.13}$$

where T_J is a vector that can be recursively defined by

$$T_J = f_{\alpha_k}(x^{J-1},\alpha) + A(x^{J-1})T_{J-1}, \quad T_1 = 0. \tag{4.14}$$

Summarizing, for the computation of s_α we need to compute $f_x, f_{\alpha_k}, f_{xx}, f_{x\alpha_k}$ in all iteration points $x^1,\dots,x^K$, and given these, compute T_J and C_J for $J = 1,\dots,K$.

4.3. Numerical computation of the directional derivatives. If symbolic derivatives of the original map are not available, then finite differences have to be used. However, the full tensors are not needed, but rather the multilinear forms evaluated on vectors, which can be computed with directional derivatives and central finite differences. This is an option of last resort and is not reliable for very high iterates. For a general discussion of directional derivatives, we refer to [18]. Similar to the analysis in [14, Appendix D], it can be shown that to compute a derivative of order k the choice $h_k = (\epsilon_m)^{1/(k+2)}$ for the stepsize minimizes the truncation and roundoff error under modest assumptions. Here ϵ_m denotes machine precision. In CL_MATCONTM, the *Increment* ($= h_1$) can be adjusted by the user. The increments of the higher order derivatives are then adapted according to $h_k = (h_1)^{3/(k+2)}$.

5. Examples. Here we give two examples illustrating the developed techniques and their implementation in CL_MATCONTM.

5.1. Generalized Hénon map. Consider the map

$$F : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \begin{pmatrix} x_2 \\ \alpha - \beta x_1 - x_2^2 + r x_1 x_2 \end{pmatrix}, \tag{5.1}$$

where r is in principle not too large. This map appears in numerous theoretical studies of degenerate homoclinic bifurcations. A bifurcation analysis of this map can be found in [23] for $r = 0$ and in [11] for general r. This map exhibits all planar codim 2 bifurcations with branch switching as in section 3.2. Let us start with the 1:2 resonance and the fold-flip. For these cases we can apply the algorithms analytically, i.e., with r as a parameter. We note that $q = (1, -1)^T$ in both cases is an eigenvector of the Jacobian matrix corresponding to eigenvalue -1.

For the 1:2 resonance we have the critical values $x_1 = x_2 = \frac{\beta-1}{r}$ and $(\alpha_0, \beta_0) = (\frac{4(3-r)}{(2-r)^2}, \frac{2+r}{2-r})$. For the coefficients of (3.6) the critical center manifold reduction yields (see [11])

$$C_1 = -\frac{1+r}{2}, \quad D_1 = \frac{1}{4}(6 + 5r + r^2). \tag{5.2}$$

Applying the algorithm from section 3.2.3, we find the prediction for the NS bifurcation curve of the period-2 cycle:

$$(x, y) = \left(\frac{2}{2-r}, \frac{2}{2-r}\right) + \sqrt{\frac{2\varepsilon}{1+r}}\, q,$$

$$(\alpha, \beta) = \left(\frac{4(3-r)}{(2-r)^2}, \frac{2+r}{2-r}\right) + \left(\frac{2(4+3r^2-r^3)}{(1+r)(2-r)^2}, \frac{2r^2}{(1+r)(2-r)}\right)\varepsilon.$$

The fold-flip bifurcation occurs for $(x_1, x_2) = (0, 0)$ at $(\alpha_0, \beta_0) = (0, -1)$. The critical center manifold reduction yields

$$a = \frac{1}{2}(1-r), \quad b = \frac{1}{2}(1+r), \quad c_2 = -\frac{1}{4}(1-r), \quad c_4 = \frac{1}{4}(1+r)^2. \tag{5.3}$$

Then applying the algorithm from section 3.2.6, we find the prediction for the NS bifurcation curve of the period-2 cycle emanating here:

$$(x, y) = (0, 0) + \sqrt{\frac{2\varepsilon}{1+r}}\, q, \qquad (\alpha, \beta) = (0, -1) + \left(2, \frac{-2r^2}{(1+r)(2-r)}\right)\varepsilon.$$

Let us compare the predictions with the exact expressions for these curves. Consider the following set:

$$\alpha = \frac{(1+\beta)(\beta - 1 - r + r^2)}{r^2}, \tag{5.4}$$

$$x_{1,2} = \frac{r(\beta+1) \pm \sqrt{(r-2)(\beta+1)(2+r-\beta(2-r))}}{2r}, \tag{5.5}$$

when

$$\frac{1}{4}(3-r)(\beta+1)^2 \le \alpha \le \frac{5+4r+r^2+2\beta(3+r)+\beta^2(5+2r-r^2)}{2(2+2r+r^2)}.$$

It consists of two different pieces, where an NS bifurcation of a cycle of period 2 occurs. If we take $0 \le \varepsilon \ll 1$ and consider the linear approximations of (5.4) near $\beta = -1 - \frac{r^2}{2-r}\tilde{\varepsilon}$ and $\beta = (\frac{2+r}{2-r}) + \frac{r^2}{2-r}\tilde{\varepsilon}$, we find for the 1:2 resonance

$$(\alpha,\beta) = \left(\frac{4(3-r)}{(2-r)^2}, \frac{2+r}{2-r}\right) + \left(\frac{4+3r^2-r^3}{(2-r)^2}, \frac{r^2}{2-r}\right)\tilde{\varepsilon} + \mathcal{O}(\tilde{\varepsilon}^2),$$

$$(x,y) = \left(\frac{2}{2-r}, \frac{2}{2-r}\right) + \sqrt{\tilde{\varepsilon}}q + \mathcal{O}(\tilde{\varepsilon}),$$

and for the fold-flip bifurcation

$$(\alpha,\beta) = (0,-1) + \left(1+r, \frac{-r^2}{2-r}\right)\tilde{\varepsilon} + \mathcal{O}(\tilde{\varepsilon}^2),$$

$$(x,y) = (0,0) - \sqrt{\tilde{\varepsilon}}q + \mathcal{O}(\tilde{\varepsilon}).$$

So up to a positive factor our results coincide up to first order in ε.

For the other three cases a numerical approach is more illuminating. We now fix $r = -0.1$ and show in Figure 5.1(a) a (partial) bifurcation diagram in the parameters α, β computed with CL_MATCONTM. We pay closer attention to three specific regions, where we can switch branches: a spring area near $(\alpha,\beta) = (3,-1)$ with fold and PD curves with degeneracies, and two domains close to strong resonances near $\beta = 1$. In the parameter space our predictions are linear, and for small initial amplitude the initialization routines worked well.

In Figure 5.1(b), there is a PD curve of period-4 cycles along which there are two degenerate flips near $(\alpha,\beta) = (3,-1)$, where we found a fold curve and a PD curve of cycles of period 4. Then we produced the approximations to the fold curves of the eighth iterates in the degenerate flip points and from these easily continued the fold curves of cycles of period 8. The bifurcation structure of Figure 5.1(b) is rather universal; see [6]. In Figure 5.1(c) we show a neighborhood of the 1:3 resonance. For $r = -0.1$ the 1:4 resonance involves all of the mentioned local branches (see Figure 5.1(d)). In all subfigures we show the continuation results and also the approximate curves.

5.2. Leslie–Gower competition model. The origin of this example is in [21, 22, 10]. It has been found in biological experiments that two species of flour beetles can coexist under strong competition for the same food. This was rather unexpected at the time, and several models were built to explain this phenomenon. One of the

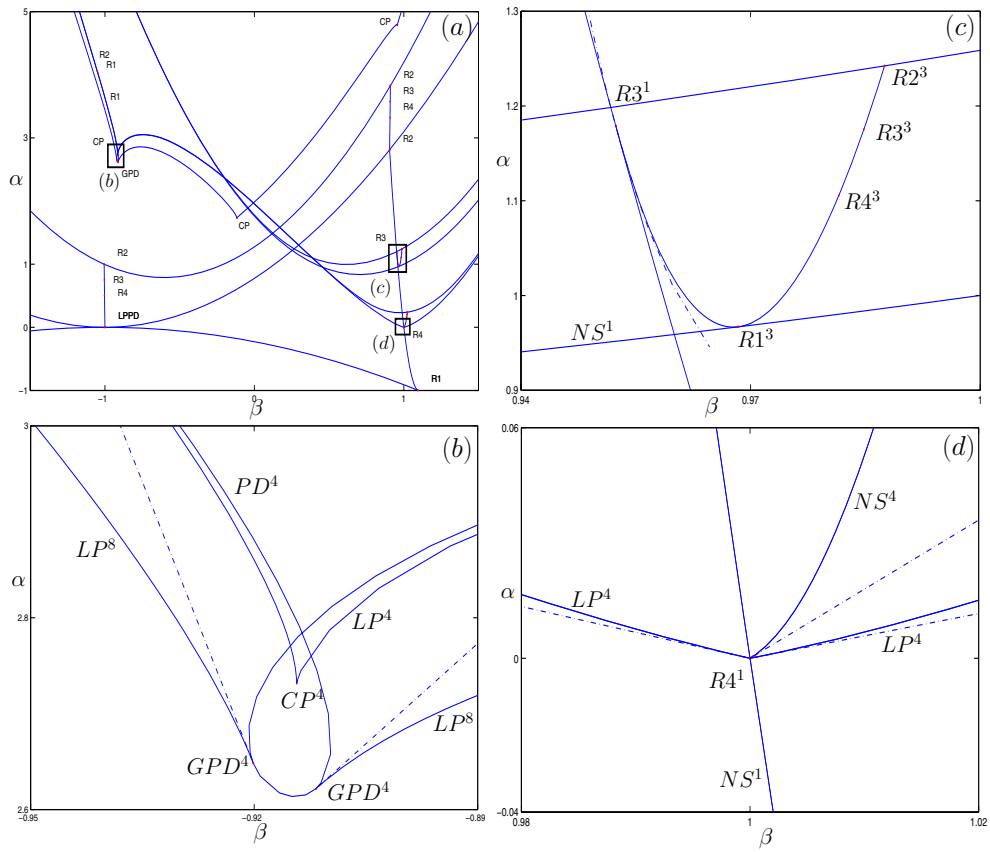


FIG. 5.1. *Bifurcation diagram for the generalized Hénon map* (a). *The close-ups near a spring area of period* 4 (b), 1:3 *resonance of period* 1 (c), 1:4 *resonance of period* 1 (d). *The solid lines are computed with* CL_MATCONTM*; the dashed lines are the predictions from the switching algorithms.*

ideas proposed in [10] and [25] was to use an age-structured population model. We consider the four-dimensional map model with 14 parameters in [25]:

$$M_{LG}: \begin{pmatrix} j \\ a \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} j' \\ a' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \dfrac{b_1 a}{(1 + c_{jj} j + c_{ja} a + c_{jy} y + c_{jz} z)} \\ (1-\mu_j) j + (1-\mu_a) a \\ \dfrac{b_2 z}{(1 + c_{yj} j + c_{ya} a + c_{yy} y + c_{yz} z)} \\ (1-\mu_y) y + (1-\mu_z) z \end{pmatrix}. \tag{5.6}$$

This is the Leslie–Gower competition model for the interaction between the juveniles (j) and adults (a) of one species of the flour beetle *Tribolium* and the juveniles (y) and adults (z) of another species. Each species has its own juvenile recruitment rate $b_1 > 0$, $b_2 > 0$, juvenile death rates μ_j and μ_y, and adult death rates μ_a and μ_z. For biological reasons we have $0 < \mu_j, \mu_a, \mu_y, \mu_z < 1$.

The other coefficients $c_{jj}, c_{ja}, c_{jy}, c_{jz}$ and $c_{yj}, c_{ya}, c_{yy}, c_{yz}$ describe the competition. They are all strictly positive. By assumption, the competition does not affect the adults of either species. As in [25], we will study the influence of the coefficients

TABLE 5.1
Parameter values for the Leslie–Gower model.

$b_1 = 20$	$c_{jj} = 0.36$	$b_2 = 18$	$c_{ja} = 0.55$	$c_{jz} = 0.23$	$\mu_j = 0.23$
$\mu_a = 0.72$	$c_{ya} = 0.08$	$c_{yy} = 0.18$	$c_{yz} = 0.26$	$\mu_y = 0.29$	$\mu_z = 0.98$

c_{yj} and c_{jy} on the behavior of M_{LG} in a case where all other parameters are fixed. In other words, we study the role of the competition between juveniles alone.

The origin $(0, 0, 0, 0)$ is a fixed point of (5.6) but is of little interest. The model also has "horizontal" fixed points of the form $(j^*, a^*, 0, 0)$, where $j^*, a^* > 0$ (i.e., biologically meaningful) when $b_1(1 - \mu_j) > \mu_a$. Similarly, there are unique "vertical" fixed points of the form $(0, 0, y^*, z^*)$. These are biologically meaningful if $y^*, z^* > 0$, i.e., when $b_2(1 - \mu_y) > \mu_z$.

Furthermore there exists a unique coexistence fixed point (j^*, a^*, y^*, z^*), provided

$$H \equiv \delta\gamma - \epsilon\eta \neq 0, \tag{5.7}$$

where

$$\alpha = \frac{1 - \mu_j}{\mu_a}, \qquad \beta = \frac{1 - \mu_y}{\mu_z},$$

$\epsilon = c_{jj} + c_{ja}\alpha$, $\gamma = c_{jy} + c_{jz}\beta$, $\delta = c_{yj} + c_{ya}\alpha$, and $\eta = c_{yy} + c_{yz}\beta$. The equation $H = 0$ defines a hyperbola in (c_{yj}, c_{jy}) space. We will study the overall dynamics of the model for the parameter values specified in Table 5.1. The parameters c_{jy} and c_{yj} will vary. First we consider the horizontal and the vertical fixed points and their stability. For the given model parameters, the horizontal fixed point is stable if $c_{yj} > c_{yj0}$, where $c_{yj0} = 0.474477674$. It is biologically plausible that the horizontal fixed point is stable only if the juveniles of the first species suppress the juveniles of the second species to a sufficient degree. The vertical fixed point is stable if $c_{jy} > c_{jy0}$, where $c_{jy0} = 0.4571312026$.

Now we consider the coexistence fixed point (j^*, a^*, y^*, z^*), starting the continuation from the point where $c_{yj} = c_{jy} = 0$. This fixed point bifurcates into vertical and horizontal fixed points, respectively, when one of c_{jy} and c_{yj} is varied and the other variable is fixed at 0. In the model this means that one species drives another to extinction. Continuation of the coexistence fixed point, where c_{jy} is the free parameter, leads to BP and PD bifurcations at $c_{jy} = c_{yj0}$ and $c_{jy} = 0.170849$, respectively. The coexistence fixed point is stable before the BP and unstable afterwards. If we continue the coexistence fixed points with the free parameter c_{yj}, the points bifurcate into the horizontal fixed point at another BP. The coexistence fixed point is stable before this BP and unstable afterwards.

The solutions to the equation $H = 0$, where H is given by (5.7), are the parameter values for which the existence and uniqueness of the coexistence fixed point is not guaranteed. In the present context, where only c_{yj} and c_{jy} vary, this leads to the condition

$$c_{yj}c_{jy} + ac_{yj} + bc_{jy} - c = 0, \tag{5.8}$$

where $a = 0.1666326531$, $b = 0.0855555552$, and $c = 0.3350275226$, which indeed defines a hyperbola in (c_{yj}, c_{jy}) space. It is not hard to prove that the point (c_{yj0}, c_{jy0}) lies on the hyperbola.

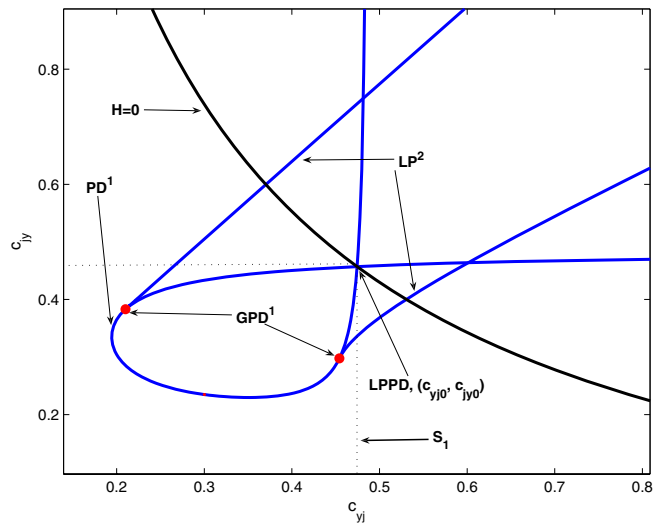


FIG. 5.2. *The flip curve PD^1, the fold curve of the second iterate LP^2, hyperbola $H = 0$, and the rectangle S_1 in the (c_{yj}, c_{jy})-plane.*

Now we return to the stability of the coexistence fixed points. The coexistence fixed point is unstable outside the rectangle

$$S_1 = \{(c_{yj}, c_{jy}) : 0 \le c_{yj} \le c_{yj0}, \quad 0 \le c_{jy} \le c_{jy0}\}.$$

Figure 5.2 shows the hyperbola $H = 0$ and the rectangle S_1.

By numerical continuation we find that there is an interior region inside of S_1 in which the coexistence fixed points are unstable. This region is bounded by the PD curve, where the stability changes. The projection of the PD curve on the (c_{yj}, c_{jy})-plane goes twice through the point (c_{yj0}, c_{jy0}). Indeed, the PD curve has two fold-flip points, where $(c_{yj}, c_{jy}) = (c_{yj0}, c_{jy0})$. Moreover, there are two degenerate PD points `GPD` on the PD curve: $(c_{yj}, c_{jy}) = (0.210138, 0.383143)$ and $(c_{yj}, c_{jy}) = (0.454279, 0.297779)$. The branches of fold curves of the second iterate can be computed by switching at the `GPD` points. These curves emanate tangentially to the PD curve.

The region where there are stable fixed points of the second iterate is bounded by the two fold curves of the second iterate and the lower left part of the PD curve. From the applications point of view, this is the most interesting region because it shows that indeed the two species can coexist even when the competition is strong. If both c_{yj} and c_{jy} are larger than 0.5, then the horizontal fixed points, the vertical fixed points, and the fixed points of the second iterate are all stable. The PD curve and the fold curves of the second iterate are given in Figure 5.2.

6. Conclusions. We have described the implementation of continuation and normal form analysis of fixed points and cycles in MATCONT. Codim 1 bifurcation analysis is standard by now, while the analysis of codim 2 bifurcations, both quantitative and qualitative, on which this paper focuses, is new. Our implementation uses all aspects of the center-manifold reduction. First, the critical normal form coefficients are calculated automatically, which determines the type of the unfolding. Second,

when nondegeneracy and transversality are checked, predictions about new codim 1 branches are made.

We use the iterates of maps to continue cycles. When dealing with strongly expanding/contracting systems and small basins of attraction, a BVP-like approach using extended systems for the fixed point, i.e., $f(x^1) - x^2 = 0, \ldots, f(x^n) - x^1 = 0$, might be more efficient.

Version 2.1 of CL_MATCONTM for maps uses automatic differentiation as an alternative to symbolic derivatives. One reason is that symbolic derivatives may not always be available. The second is that the computation of normal form coefficients for high iterates is faster when automatic differentiation is used.

Finally, we mention that we did not discuss global bifurcations. As a first step, it is well worth the effort to incorporate into our package some of the existing algorithms to compute (un)stable manifolds, e.g., [16]. If a transverse intersection is present, then using the defining systems as in [4], approximations of homoclinic orbits and their tangencies may be computed.

REFERENCES

[1] E. L. Allgower and K. Georg, *Numerical Continuation Methods: An Introduction*, Springer-Verlag, Berlin, 1990.

[2] V. I. Arnol'd, *Geometrical Methods in the Theory of Ordinary Differential Equations*, Springer-Verlag, New York, 1983.

[3] A. Back, J. Guckenheimer, M. Myers, F. Wicklin, and P. Worfolk, *DsTool: Computer assisted exploration of dynamical systems*, Notices Amer. Math. Soc., 39 (1992), pp. 303–309.

[4] W.-J. Beyn and J.-M. Kleinkauf, *The numerical computation of homoclinic orbits for maps*, SIAM J. Numer. Anal., 34 (1997), pp. 1207–1236.

[5] W.-J. Beyn, A. Champneys, E. Doedel, W. Govearts, Yu. A. Kuznetsov, and B. Sandstede, *Numerical continuation, and computation of normal forms*, in Handbook of Dynamical Systems, Vol. 2, B. Fiedler, ed., Elsevier Science, Amsterdam, 2002, pp. 149–219.

[6] J.-P. Carcassés, C. Mira, M. Bosch, C. Simó, and J. C. Tatjer, *"Crossroad area–spring area" transition. I. Parameter plane representation*, Internat. J. Bifur. Chaos Appl. Sci. Engrg., 1 (1991), pp. 183–196.

[7] A. Dhooge, W. Govaerts, Yu. A. Kuznetsov, W. Mestrom, and A. M. Riet, *CL_MATCONT: A continuation toolbox in MATLAB*, in Proceedings of the 2003 ACM Symposium on Applied Computing, Melbourne, FL, 2003, Association for Computing Machinery, New York, pp. 161–166.

[8] A. Dhooge, W. Govaerts, and Yu. A. Kuznetsov, *MATCONT: A MATLAB package for numerical bifurcation analysis of ODEs*, ACM Trans. Math. Software, 29 (2003), pp. 141–164.

[9] E. J. Doedel, A. R. Champneys, T. F. Fairgrieve, Yu. A. Kuznetsov, B. Sandstede, and X.-J. Wang, AUTO97-AUTO2000: *Continuation and Bifurcation Software for Ordinary Differential Equations (with HomCont), User's Guide*, Concordia University, Montreal, Canada (1997–2000), available online from http://indy.cs.concordia.ca.

[10] J. Edmunds, J. M. Cushing, R. F. Costantino, S. M. Henson, B. Dennis, and R. A. Desharnais, *Park's Tribolium competition experiments: A non-equilibrium species coexistence hypothesis*, J. Animal Ecology, 72 (2003), pp. 703–712.

[11] V. S. Gonchenko, Yu. A. Kuznetsov, and H. G. E. Meijer, *Generalized Hénon map and bifurcations of homoclinic tangencies*, SIAM J. Appl. Dyn. Sys., 4 (2005), pp. 407–436.

[12] W. Govaerts,Yu. A. Kuznetsov, and B. Sijnave, *Bifurcations of maps in the software package* CONTENT, in Computer Algebra in Scientific Computing—CASC'99 (Munich), V. G. Ganzha, V. G. E. W. Mayr, and E. V. Vorozhtsov, eds., Springer, Berlin, 1999, pp. 191–206.

[13] W. J. F. Govaerts, *Numerical Methods for Bifurcations of Dynamical Equilibria*, SIAM, Philadelphia, 2000.

[14] B. D. Hassard, N. D. Kazarinoff, and Y.-H. Wan, *Theory and Applications of Hopf Bifurcation*, Cambridge University Press, Cambridge, UK, 1981.

[15] A. I. Khibnik, Yu. A. Kuznetsov, V. V. Levitin, and E. V. Nikolaev, *Continuation techniques and interactive software for bifurcation analysis of ODEs and iterated maps*, Phys. D, 62 (1993), pp. 360–371.
[16] B. Krauskopf and H. M. Osinga, *Growing* 1*D and quasi-*2*D unstable manifolds of maps*, J. Comput. Phys., 146 (1998), pp. 404–419.
[17] Yu. A. Kuznetsov and V. V. Levitin, CONTENT: *A Multiplatform Environment for Analyzing Dynamical Systems*, CWI, Amsterdam, 1997; available online via ftp from ftp.cwi.nl/pub/CONTENT.
[18] Yu. A. Kuznetsov, *Elements of Applied Bifurcation Theory*, 3rd ed., Springer-Verlag, New York, 2004.
[19] Yu. A. Kuznetsov and H. G. E. Meijer, *Numerical normal forms for codim* 2 *bifurcations of fixed points with at most two critical eigenvalues*, SIAM J. Sci. Comput., 26 (2005), pp. 1932–1954.
[20] Yu. A. Kuznetsov and H. G. E. Meijer, *Remarks on interacting Neimark-Sacker bifurcations*, J. Difference Equ. Appl., 12 (2006), pp. 1009–1035.
[21] P. H. Leslie and J. C. Gower, *The properties of a stochastic model for two competing species*, Biometrika, 45 (1958), pp. 316–330.
[22] P. H. Leslie, T. Park, and D. B. Mertz, *The effect of varying the initial numbers on the outcome of competition between two Tribolium species*, J. Animal Ecology, 37 (1968), pp. 9–23.
[23] C. Mira, *Chaotic Dynamics*, World Scientific, Singapore, 1987.
[24] H. Nusse and J. Yorke, *Dynamics: Numerical Explorations*, 2nd ed., Springer-Verlag, New York, 1998.
[25] Y. L. Song, *The Juvenile/Adult Leslie–Gower Competition Model*, preprint, 2004.
[26] D. Roose and V. Hlavaček, *A direct method for the computation of Hopf bifurcation points*, SIAM J. Appl. Math., 45 (1985), pp. 879–894.

SIAM J. SCI. COMPUT.
Vol. 29, No. 6, pp. 2668–2685
© 2007 Society for Industrial and Applied Mathematics

STRUCTURED QUADRATIC INVERSE EIGENVALUE PROBLEM, I. SERIALLY LINKED SYSTEMS*

MOODY T. CHU†, NICOLETTA DEL BUONO‡, AND BO YU§

Abstract. Quadratic pencils arising from applications are often inherently structured. Factors contributing to the structure include the connectivity of elements within the underlying physical system and the mandatory nonnegativity of physical parameters. For physical feasibility, structural constraints must be respected. Consequently, they impose additional challenges on the inverse eigenvalue problems which intend to construct a structured quadratic pencil from prescribed eigeninformation. Knowledge of whether a structured quadratic inverse eigenvalue problem is solvable is interesting in both theory and applications. However, the issue of solvability is problem dependent and has to be addressed structure by structure. This paper considers one particular structure where the elements of the physical system, if modeled as a mass-spring system, are serially linked. The discussion recasts both undamped and damped problems in a framework of inequality systems that can be adapted for numerical computation. Some open questions are described.

Key words. quadratic pencil, inverse eigenvalue problem, linear oscillator, inner-connectivity, nonnegativity, linear system of inequalities

AMS subject classifications. 65F18, 15A22, 93B55

DOI. 10.1137/060672510

1. Introduction.

The time-invariant second order differential system

$$M\ddot{\mathbf{x}} + C\dot{\mathbf{x}} + K\mathbf{x} = f(t), \tag{1.1}$$

where $\mathbf{x} \in \mathbb{R}^n$ and M, C, $K \in \mathbb{R}^{n\times n}$, arises in many important applications, including applied mechanics, electrical oscillation, vibro-acoustics, fluid mechanics, signal processing, and finite element discretization of PDEs. In most cases, parameters characterizing the underlying physical system are embedded in the matrix coefficients M, C, and K. It is well known that if

$$\mathbf{x}(t) = \mathbf{v}e^{\lambda t}$$

represents a fundamental solution to (1.1), then the scalar λ and the vector $\mathbf{v}$ must solve the *quadratic eigenvalue problem* (QEP)

$$(\lambda^2 M + \lambda C + K)\mathbf{v} = 0. \tag{1.2}$$

That is, the dynamical bearing of the differential system (1.1) usually can be interpreted via the eigenvalues and eigenvectors of the algebraic system (1.2). Because of this connection, considerable efforts have been devoted to the QEP in the literature. Readers are referred to the treatise by Tisseur and Meerbergen [28] for a good survey

*Received by the editors October 16, 2006; accepted for publication (in revised form) April 5, 2007; published electronically November 2, 2007.
http://www.siam.org/journals/sisc/29-6/67251.html

†Department of Mathematics, North Carolina State University, Raleigh, NC 27695-8205 (chu@math.ncsu.edu). This research was supported in part by the National Science Foundation under grants CCR-0204157 and DMS-0505880.

‡Dipartimento di Matematica, Università degli Studi di Bari, Via E. Orabona 4, I-70125 Bari, Italy (delbuono@dm.uniba.it).

§Department of Applied Mathematics, Dalian University of Technology, Dalian, Liaoning, 116024, China (yubo@dlut.edu.cn).

of many applications, mathematical properties, and a variety of numerical techniques for the QEP.

The eigenvalue problem associated with the model (1.1) can be considered from two different viewpoints. From a priori known physical parameters such as mass, length, elasticity, inductance, capacitance, and so on, the process of analyzing and deriving the spectral information so as to induce the dynamical behavior of a system is referred to as a *direct* problem. The *inverse* problem, in contrast, is to validate, determine, or estimate the parameters of the system according to its observed or expected behavior. The concern in the direct problem is to express the behavior in terms of the parameters, whereas in the inverse problem the concern is to express the parameters in terms of the behavior. Both problems are important in applications. This paper concerns the inverse problem.

The inverse eigenvalue problem is a diverse area full of research interests and activities. See the newly revised book by Gladwell [20] for mechanics applications, the review article [7] for linear problems, the recently completed monograph by Chu and Golub [8] for general theory, algorithms, and applications, and the many references collected therein from various disciplines. Among current developments, the quadratic inverse eigenvalue problem (QIEP) is particularly more important and challenging, with many unanswered questions [5, 9, 13]. This paper concentrates only on the issue of solvability for one specially structured QIEP.

We have to stress that depending on the applications, the term QIEP has been used in the literature to mean a rather wide variety of formulations. For instance, the QIEP studied by Ram and Elhay in [25] involves only symmetric tridiagonal matrix coefficients where *two* sets of eigenvalues are given. The QIEP studied by Starek and Inman in [27] is associated with nonproportional underdamped systems. Lancaster and Prells [23] considered the QIEP with symmetric and positive semidefinite damping C where *complete* information on eigenvalues and eigenvectors is given and all eigenvalues are simple and nonreal. There are also works which utilize notions of feedback control to reassign the eigenstructure [11, 12, 24]. The list goes on and can hardly be exhaustive. The QIEP considered in this article deals with yet another scenario which is quite common in practice—construct the quadratic pencil with only *a few* eigenvalues and their corresponding eigenvectors.

In vibration industries, including aerospace, automobile, and manufacturing, through vibration tests where the excitation and the response of the structure at selected points are measured experimentally, there are identification techniques to extract a portion of eigenpair information from the measurements. However, the size of the physical system can be so large and complicated that it is not always possible to attain knowledge of the entire spectrum. While there is no reasonable analytical tool available to evaluate all of the spectral information, it is simply unwise to use experimental values of high natural frequencies to reconstruct a model. Additionally, it is often demanded, especially in structural design, that certain eigenvectors should also satisfy some specific conditions [3, 30]. A finite element generated model therefore needs to be updated using only a few measured eigenvalues and eigenvectors [1, 14, 17, 18, 21]. Furthermore, quantities related to high frequency terms in a *finite model* generally are susceptible to measurement errors due to the finite bandwidth of measuring devices. Spectral information, therefore, should not be used at its full extent. For these reasons, it might be more sensible to consider an inverse eigenvalue problem where only a *portion* of eigenvalues and eigenvectors is available. The QIEP that is of interest to us therefore can be formulated as follows:

QIEP. Construct a nontrivial quadratic pencil $Q(\lambda) = \lambda^2 M + \lambda C + K$ so that its matrix coefficients (M, C, K) are of a specified structure and $Q(\lambda)$ has a specified set $\{(\lambda_i, \mathbf{v}_i)\}_{i=1}^k$ as its eigenpairs.

Since we are interested only in real matrices, it is natural to expect that the prescribed eigenpairs are closed under complex-conjugation. Without loss of generality, we shall denote the prescribed eigenpairs in the matrix form (Λ, X), where $\Lambda \in \mathbb{R}^{k\times k}$ is block diagonal with at most 2×2 blocks along the diagonal wherever a complex-conjugate pair of eigenvalues appears in the prescribed spectrum, and $X \in \mathbb{R}^{n\times k}$ represents the "eigenvector matrix" in the sense that each pair of column vectors associated with a 2×2 block in Λ retains the real and the imaginary parts, respectively, of the original complex eigenvector. In this way, we may identify the given eigenpairs (Λ, X) as an element in $\mathbb{R}^k \times \mathbb{R}^{n\times k}$. The QIEP therefore amounts to solving the algebraic equation

$$MX\Lambda^2 + CX\Lambda + KX = 0. \tag{1.3}$$

At first glance, the relationship (1.3) is only a homogeneous linear system of nk algebraic equations. If there are no other constraints, the triplet (M, C, K) constitutes $3n^2$ unknowns. Since k is bounded above by $2n$, the system is well under-determined. It is intuitively true that the system should be solvable in general. The real challenge is to characterize the solution in terms of the given (Λ, X), when the matrix coefficients in the pencil are required to satisfy some structural constraints.

The particular constraints that will be studied in this paper are twofold—the connectivity constraint inherent to the serially linked physical structure and the nonnegative constraint required by the physical parameters. Despite the fact that some general results have been obtained for the QIEPs [22, 23, 25], we have not seen the above-mentioned constraints being taken into account. Our contribution is innovative in that the structured QIEP is cast as a system of inequalities whose solvability can then be checked numerically.

2. Issue of connectivity. An inverse eigenvalue problem without a structure is often trivial and meaningless [8]. This is particularly true for QIEPs arising from applications. The structure imposed on a QIEP depends inherently on the *connectivity* of the underlying physical system. For practical reasons, the physical feasibility must be respected. It is therefore necessary that the reconstructed pencil in the inverse problem satisfies the inherent connectivity structure. In other words, there is more to just solving the algebraic equation (1.3) for a QIEP. The matrices in the solution (M, C, K) must carry a certain specified structure. We give two examples to illustrate how the connectivity affects the structure.

Example 1. Assuming that the springs respond according to Hook's law and that the damping is negatively proportional to the velocity, the coefficient matrices corresponding to the four-degree-of-freedom mass-spring system depicted in Figure 2.1 should be structured as follows:

$$M = \begin{bmatrix} m_1 & 0 & 0 & 0 \\ 0 & m_2 & 0 & 0 \\ 0 & 0 & m_3 & 0 \\ 0 & 0 & 0 & m_4 \end{bmatrix}, \; C = \begin{bmatrix} c_1 + c_2 & 0 & -c_2 & 0 \\ 0 & 0 & 0 & 0 \\ -c_2 & 0 & c_2 + c_3 & -c_3 \\ 0 & 0 & -c_3 & c_3 \end{bmatrix},$$

$$K = \begin{bmatrix} k_1 + k_2 + k_3 & -k_2 & -k_5 & 0 \\ -k_2 & k_2 + k_3 & -k_3 & 0 \\ -k_5 & -k_3 & k_3 + k_4 + k_5 & -k_4 \\ 0 & 0 & -k_4 & k_4 \end{bmatrix}.$$

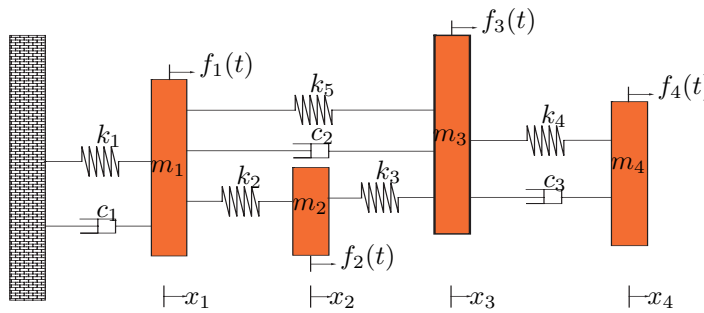


FIG. 2.1. *A four-degree-of-freedom mass-spring system.*

A typical way in the literature to describe the structure of a general mass-spring system is that the mass matrix M is diagonal, both the damping matrix C and the stiffness matrix K are symmetric and banded, M is positive definite, and K is positive semidefinite. But now we see that such a characterization indeed is not fine enough for physical feasibility. One must be specific in that the internal connectivity of the masses and springs defines how the diagonal entries of C and K are related to their off-diagonal entries. A different configuration of the connectivity in the system will lead to a different structure. It is extremely important to point out an additional inherent constraint; that is, all physical parameters are expected to be nonnegative.

Example 2. Assuming Ohm's law and Kirchoff's law in the circuit, the matrix coefficients for the governing equation of the current in the RLC network depicted in Figure 2.2 should have the following structure:

$$M = \begin{bmatrix} -L_2 & L_2 & 0 & 0 \\ L_2 & -L_2 & 0 & 0 \\ 0 & 0 & L_3 & 0 \\ 0 & 0 & 0 & L_4 \end{bmatrix}, \; C = \begin{bmatrix} 0 & R_2 & -R_2 & 0 \\ R_1 + R_4 & 0 & 0 & -R_4 \\ 0 & -R_2 & R_2 + R_3 & 0 \\ -R_4 & 0 & 0 & R_4 \end{bmatrix},$$

$$K = \begin{bmatrix} 0 & \frac{1}{C_2} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{C_3} & -\frac{1}{C_3} \\ 0 & 0 & -\frac{1}{C_3} & \frac{1}{C_3} + \frac{1}{C_4} \end{bmatrix}.$$

In contrast to the structure associated with a mass-spring system, note that the structure associated with an electronic circuit generally may not be definite or even symmetric.

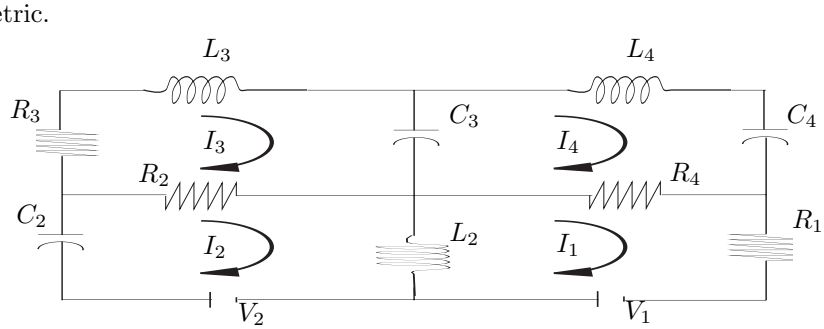


FIG. 2.2. *An RLC electronic network.*

Since structured problems often result in special interrelationships within their eigenstructures, the observed measurement which in many cases is contaminated with random noise may not be consistent with that innate structure. The structural constraint therefore severely limits whether a QIEP is solvable. It would be desirable to have a general theory by which one can obtain an affirmative answer on whether a specifically configured system could be built or not and, if yes, how the parameters should be valued. To our knowledge, such a theory does not exist. Most of the current QIEP techniques consider only the algebraic solvability, that is, solving the QIEP without taking into account the nonnegativity of the solution. The main thrust of this paper is to discuss the (physical) solvability of one particular structure. This is only one step forward with the hope of stimulating further studies toward solving the general structured QIEPs.

3. Symmetric structure. One of the simplest structural constraints in many applications is symmetry. If all matrix coefficients involved in a pencil are symmetric, we say that we have a *self-adjoint* pencil. The theory of solvability for self-adjoint QIEPs has been established recently by Chu et al. in [6]. To make this note more self-contained, it might be informative to summarize some of the results in this section without repeating the details. Define

$$k_{max} = \begin{cases} 3\ell + 1 & \text{if } n = 2\ell, \\ 3\ell + 2 & \text{if } n = 2\ell + 1. \end{cases} \tag{3.1}$$

The following theorem characterizes the general solvability [6].

THEOREM 3.1. *Given any positive integer $k \le k_{max}$, let (Λ, X) represent k arbitrarily prescribed eigenpairs which are closed under complex-conjugation. Then the following hold:*

1. *The self-adjoint QIEP associated with (Λ, X) is always solvable.*
2. *For almost all k prescribed eigenpairs (Λ, X), the solutions to the corresponding self-adjoint QIEP form a subspace of dimensionality $\frac{3n(n+1)}{2} - nk$.*

Elaborating a little further upon the solvability indicated in the above theorem might provide some additional insight. In the event that $k \ge n$, partition the eigenvalues and eigenvectors as

$$\Lambda = \text{diag}\{\Upsilon_1, \Upsilon_2\},$$

$$X = [Z_1, Z_2],$$

with $Z_1 \in \mathbb{R}^{n\times n}$, $Z_2 \in \mathbb{R}^{n\times(k-n)}$, $\Upsilon_1 \in \mathbb{R}^{n\times n}$, and $\Upsilon_2 \in \mathbb{R}^{(k-n)\times(k-n)}$, respectively. Define

$$\Xi := Z_1^{-1} Z_2. \tag{3.2}$$

Then there exists a nontrivial self-adjoint quadratic pencil $Q(\lambda)$ with partial eigenstructure (Λ, X) if and only if the system of matrix equations [8]

$$\begin{cases} \Upsilon_1^\top B - B\Upsilon_1 &= A\Upsilon_1^2 - \Upsilon_1^{2\top} A, \\ \Upsilon_1^\top B\Xi - B\Xi\Upsilon_2 &= A\Xi\Upsilon_2^2 - \Upsilon_1^{2\top} A\Xi, \\ A &= A^\top, \\ B &= B^\top \end{cases} \tag{3.3}$$

has a nontrivial solution (A, B) in $\mathbb{R}^{n\times n} \times \mathbb{R}^{n\times n}$. In that case, the solution to the QIEP is given by

$$\begin{aligned} M &= Z_1^{-\top} A Z_1^{-1}, \\ C &= Z_1^{-\top} B Z_1^{-1}, \\ K &= Z_1^{-\top}(-\Upsilon_1^{\top} B - \Upsilon_1^{2\top} A) Z_1^{-1}. \end{aligned}$$

To better grasp the linear system (3.3), let $\mathbf{vec}(A)$ represent the matrix A by stacking the columns of A into one long vector. Let $\mathcal{K} \in \mathbb{R}^{\frac{n(n-1)}{2}\times n^2}$ denote the matrix representation of the projection of a square matrix onto the subspace of skew-symmetric matrices. Then (3.3) can be expressed as

$$\underbrace{\begin{bmatrix} \Upsilon_1^{2\top} \otimes I_n - I_n \otimes \Upsilon_1^{2\top} & \Upsilon_1^{\top} \otimes I_n - I_n \otimes \Upsilon_1^{\top} \\ (\Xi\Upsilon_2^2)^{\top} \otimes I_n - \Xi^{\top} \otimes \Upsilon_1^{2\top} & (\Xi\Upsilon_2)^{\top} \otimes I_n - \Xi^{\top} \otimes \Upsilon_1^{\top} \\ \mathcal{K} & O_{\frac{n(n-1)}{2}\times n^2} \\ O_{\frac{n(n-1)}{2}\times n^2} & \mathcal{K} \end{bmatrix}}_{\mathcal{T}} \begin{bmatrix} \mathbf{vec}(A) \\ \mathbf{vec}(B) \end{bmatrix} = 0. \tag{3.4}$$

In other words, in order to have nontrivial solution (A, B) for (3.3), the prescribed eigenpairs (Λ, X) must be such that the $n(n + k - 1) \times 2n^2$ coefficient matrix $\mathcal{T}$ has column rank less than $2n^2$.

COROLLARY 3.2. *Given $k \geq n$ arbitrarily prescribed eigenpairs (Λ, X) which are closed under complex-conjugation, define the matrix $\mathcal{T}$ as in* (3.4).

1. *If $n \leq k \leq k_{max}$, then the null space of $\mathcal{T}$ has dimensionality at least $\frac{3n(n+1)}{2} - nk$ and is generically of that dimensionality.*
2. *If $k > k_{max}$, then the matrix $\mathcal{T}$ is generically of full column rank. The collection of (Λ, X) for which* (1.3) *has a nontrivial self-adjoint solution has measure zero in the space $\mathbb{R}^n \times \mathbb{R}^{n\times k}$.*

In the event that $k < n$, we still can solve the QIEP by embedding this eigeninformation in a larger set of n eigenpairs. In particular, we expand $X \in \mathbb{R}^{n\times k}$ to

$$\widehat{X} := [X, \widetilde{X}] \in \mathbb{R}^{n\times n}, \tag{3.5}$$

where $\widetilde{X} \in \mathbb{R}^{n\times(n-k)}$ is arbitrary while making $\hat{X}$ nonsingular and expand $\Lambda \in \mathbb{R}^{k\times k}$ to

$$\widehat{\Lambda} := \operatorname{diag}\{\Lambda, \widetilde{\Lambda}\}, \tag{3.6}$$

where $\widehat{\Lambda}$ is a diagonal matrix with distinct eigenvalues. Taking into account the freedom of this expansion, it is not difficult to prove that the solutions to the QIEP with $k < n$ form a subspace of dimensionality $\frac{n(n+3)}{2} + n(n - k)$.

Example 3. Suppose n distinct eigenvalues Λ and n linearly independent eigenvectors X, both of which are closed under conjugation, are given. Let $A \in \mathbb{R}^{n\times n}$ be an *arbitrary* symmetric matrix. The system (3.3) is reduced to the equation

$$\Lambda^{\top} B - B\Lambda = A\Lambda^2 - \Lambda^{2\top} A \tag{3.7}$$

for the unknown symmetric matrix B. From the block diagonal structure of Λ, it follows that B is determined up to n free parameters. Using A and B as parameters,

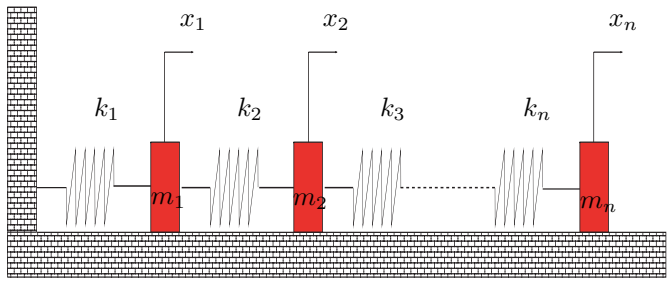


FIG. 4.1. *An undamped mass-spring system.*

the general solution to the self-adjoint QIEP with the prescribed pair (X, Λ) as part of its eigenstructure is given by

$$M = X^{-\top} A X^{-1}, \tag{3.8}$$

$$C = X^{-\top} B X^{-1}, \tag{3.9}$$

$$K = -X^{-\top} \Lambda^{\top} (B + \Lambda^{\top} A) X^{-1}. \tag{3.10}$$

In all cases, so long as the self-adjoint QIEP is solvable, we do have a way to obtain a parametric representation of the solution (M, C, K) of (1.3). Despite the fact that we know how to solve a self-adjoint QIEP algebraically, it must be stressed that algebraic solvability does not necessarily imply physical feasibility. Physical feasibility means, for example, that the special matrix structure resulting from the underlying connectivity must hold and that the physical parameters must be nonnegative. These added conditions make the QIEP much more interesting but harder. We are not aware of many research results in this regard. In what follows, we consider a special structure resulting from serially linked mass-spring systems.

4. Serially linked structure without damping. Our experience seems to indicate that whether a structured QIEP is solvable is probably problem dependent. The issue of solvability needs to be analyzed structure by structure. One special setting, the serially linked, undamped, mass-spring system depicted in Figure 4.1, is considered in this section. Such a setting can be used to model many other physical systems, including a vibrating beam, a composite pendulum, or a string with beads [20].

Matrices corresponding to the quadratic pencil $\lambda^2 M + K$ for the system in Figure 4.1 have the structure

$$M = \begin{bmatrix} m_1 & 0 & \dots & 0 \\ 0 & m_2 & & \\ \vdots & & \ddots & \\ 0 & & & m_n \end{bmatrix}, \quad K = \begin{bmatrix} k_1 + k_2 & -k_2 & 0 & \dots & 0 & 0 \\ -k_2 & k_2 + k_3 & -k_3 & & & 0 \\ 0 & -k_3 & k_3 + k_4 & & & \\ \vdots & & & \ddots & & \vdots \\ 0 & & & & k_{n-1} + k_n & -k_n \\ 0 & & & & -k_n & k_n \end{bmatrix}. \tag{4.1}$$

It is natural to assume that the masses and spring constants are all positive. In this case, eigenvalues associated with this kind of system are necessarily purely imaginary. The inverse eigenvalue problem means to find positive values for the masses $m_1, \dots, m_n$ and spring constants $k_1, \dots, k_n$ from prescribed eigeninformation.

4.1. Sign condition. A typical approach in the literature for this type of inverse problem has been to recast the quadratic pencil as a linear pencil $\mu I + J$ with a Jacobi matrix $J = M^{-1/2}KM^{-1/2}$. A classical theory has been that *two sets of eigenvalues* can uniquely solve the corresponding Jacobi inverse eigenvalue problem [8, section 4.2], but that result does not guarantee the nonnegativity of M and K. Furthermore, what can be said if the system is to be reconstructed from eigenpairs?

A total of $2n$ physical parameters are to be determined in (4.1). Each given eigenpair provides n equations. Imposing two eigenpairs generally will lead to the trivial algebraic solution for such a system, unless the prescribed eigenpairs satisfy some additional internal relationship. So it is reasonable to ask a more fundamental question of constructing the system with *one* prescribed eigenpair $(i\beta, \mathbf{x})$, where $i = \sqrt{-1}$, $\beta \in \mathbb{R}$, and $\mathbf{x} \in \mathbb{R}^n$.

Denote $\mathbf{x} = [x_1, \ldots, x_n]^\top$ and $x_0 = 0$. It is not difficult to see from (1.3) under the structure (4.1) that the recursive relationship

$$k_n = \frac{\beta^2 m_n x_n}{x_n - x_{n-1}}, \tag{4.2}$$

$$k_i = \frac{\beta^2 m_i x_i + k_{i+1}(x_{i+1} - x_i)}{x_i - x_{i-1}}, \quad i = n-1, \ldots, 1, \tag{4.3}$$

must hold. Our goal is to choose a positive value for m_i so as to define a positive value for k_i. In order to achieve this goal, the entries of the eigenvector $\mathbf{x}$ must satisfy some sign properties. More specifically, we observe the following fact directly from (4.2) and (4.3).

THEOREM 4.1. *The necessary condition for a given vector* $\mathbf{x} = [x_1, \ldots, x_n]^\top \in \mathbb{R}^n$ *with distinct entries to be an eigenvector of a quadratic pencil* $\lambda^2 M + K$ *with structure as is specified in* (4.1) *where masses* m_i *and spring constants are positive is that* $x_n(x_n - x_{n-1}) > 0$ *and the signs of the triplets* $(x_{i+1} - x_i, x_i, x_i - x_{i-1})$ *for* $i = 2, \ldots, n-1$ *are neither* $(+,+,-)$ *nor* $(-,-,+)$.

4.2. Numerical algorithm. In the event that the given vector $\mathbf{x}$ does satisfy the necessary condition in Theorem 4.1, there are infinitely many ways to choose positive values for m_i which are then used to define k_i. The following is a *make-or-break* algorithm for the construction where we specify only one particular way of selecting m_i.

ALGORITHM 4.1. *Given an arbitrary eigenpair* $(i\beta, \mathbf{x})$, $\beta \neq 0$ *and* $\mathbf{x} \neq 0$, *assume the normalization* $x_n = 1$ *and* $m_n = 1$. *The following steps either construct the masses* $m_1, \ldots, m_{n-1}$ *and spring constants* $k_1, \ldots, k_n$, *all positive, for the pencil* $\lambda^2 M + K$, *or determine that such a system with the prescribed eigenpair does not exist.*

1. *initialization:*
   $s_n = 0.9$; *(mass decreasing factor)*
   $s_p = 1.1$; *(mass increasing factor)*
   $\eta = \beta^2$;
2. *if* $x_{n-1} < 1$,
   $k_n = \frac{\beta^2}{1 - x_{n-1}}$; *(use formula* (4.2)*)*
   else
   return *(inconsistent eigenvector)*
   end
3. *for* i *from* $n-1$ *to* 2, *do*

(a) $\rho = \frac{\eta}{x_i}$;
(b) *if* $x_{i-1} < x_i$,
  if $\rho > 0$,
    if $x_i < 0$,
      return *(inconsistent eigenvector)*
    else
      $m_i = 1$; *(any $m_i > 0$ will be fine)*
    end
  else
    if $x_i < 0$,
      $m_i = -\frac{s_n \rho}{\beta^2}$; *(need $0 < m_i < -\frac{\rho}{\beta^2}$)*
    else
      $m_i = -\frac{s_p \rho}{\beta^2}$; *(need $m_i > -\frac{\rho}{\beta^2}$)*
    end
  end
else
  if $\rho > 0$,
    if $x_i < 0$,
      $m_i = 1$;
    else
      return *(inconsistent eigenvector)*
    end
  else
    if $x_i < 0$,
      $m_i = -\frac{s_p \rho}{\beta^2}$;
    else
      $m_i = -\frac{s_n \rho}{\beta^2}$;
    end
  end
end
(c) $\eta = \eta + \beta^2 m_i x_i$;
(d) $k_i = \frac{\eta}{x_i - x_{i-1}}$; *(use formula (4.3))*

1. *finale:*
(a) $\rho = \frac{\eta}{x_1}$;
(b) *if* $\rho > 0$,
  $m_1 = 1$;
else
  $m_1 = -\frac{s_p \rho}{\beta^2}$;
end
(c) $\eta = \eta + \beta^2 m_1 x_1$;
(d) $k_1 = \frac{\eta}{x_1}$; *(use formula (4.3))*

The working of the algorithm, based on Theorem 4.1, appears naive enough in just checking a few signs. It is precisely this simplicity of the necessary and sufficient conditions on the given eigenvector for ensuring a *positive* solution that we think is gratifying and elegant.

Note that in the above construction no restriction on the magnitude of β is imposed. This is quite curious in that the physical system can have one arbitrary natural frequency. One might thus ask, after the pencil $\lambda^2 M + K$ is determined, what the remaining eigenstructure should look like. To explore this important question, it is

interesting to recall the classical Courant nodal line theorem [31]. Roughly speaking, it has been known that critical information about a vibration system can be recovered from the places where *nothing happens*. These places are referred to as the *nodal lines*. In our discrete model, the incident of "nothing happens" means that the natural mode crosses the zero equilibrium. Courant's theorem gives an exact count of the numbers of nodal lines. We shall not elaborate on the particulars here, but refer readers to the treatises [10, 31] for more detailed discussion. Its simplest form related to our application can be stated as follows.

THEOREM 4.2. *If the given eigenvector* $\mathbf{x}$ *is feasible and if there are* τ *changes of signs in the sequence when going through* x_1 *to* x_n*, then* $\mathbf{x}$ *is the* $(\tau+1)$*th eigenvector (in the series where the corresponding frequencies are arranged in the ascending order) of the pencil regardless how the masses* m_i *are defined.*

In other words, from the changes of signs in the prescribed eigenvector $\mathbf{x}$, we know precisely how many other natural frequencies are above β and how many are below, although the values of these other frequencies can be quite arbitrary. The normalization $m_n = 1$ is immaterial. It indicates only the unit of mass used relative to m_n. It is the choice of values for other m_i's, which in turn determines values for other k_i's, that affects the total spectrum. How to select values for m_i when there are multiple alternatives so as to satisfy some additional spectral information is a very different but important question. At present, we have no knowledge about how this problem should be tackled. For other types of connectivity including damped systems or RLC configurations, the resulting pencil matrix structure will be different. We are not aware of any similar nodal line theorem. It is very likely that the condition of solvability will also vary. Again, this is a widely open area for further research.

5. Serially linked structure with damping. When damping is present in the mass-spring system in the way depicted in Figure 5.1, the QIEP becomes much more complicated. The associated matrices in the pencil $\lambda^2 M + \lambda C + K$ have the same structure of M and K as in (4.1) and, additionally, that of

$$C = \begin{bmatrix} c_1+c_2 & -c_2 & 0 & \cdots & 0 & 0 \\ -c_2 & c_2+c_3 & -c_3 & & & 0 \\ 0 & -c_3 & c_3+c_4 & & & \\ \vdots & & & \ddots & & \vdots \\ 0 & & & & c_{n-1}+c_n & -c_n \\ 0 & & & & -c_n & c_n \end{bmatrix}. \tag{5.1}$$

There are a total of $3n$ physical parameters to be determined, all of which need to be positive real numbers. It is clear that specifying more than three eigenpairs will make the QIEP over-determined. In the same spirit as before, we shall consider that two pairs of eigenpairs are given, and we want to determine $c_1, \ldots, c_n$ and $k_1, \ldots, k_n$ in terms of $m_1, \ldots, m_n$. Depending upon whether the specified eigenvalues are complex or not, we divide our discussion into two cases.

5.1. Complex eigenpairs. Assume that the given eigenpairs are complex-conjugate to each other. This eigeninformation is given through the matrices

$$\Lambda = \begin{bmatrix} \alpha & \beta \\ -\beta & \alpha \end{bmatrix}, \quad X = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n \end{bmatrix}^\top, \tag{5.2}$$

where the 2×2 matrix Λ represents the complex-conjugate eigenvalues $\alpha \pm i\beta$, and the first and the second columns of X represent the real and the imaginary parts of the corresponding eigenvector, respectively.

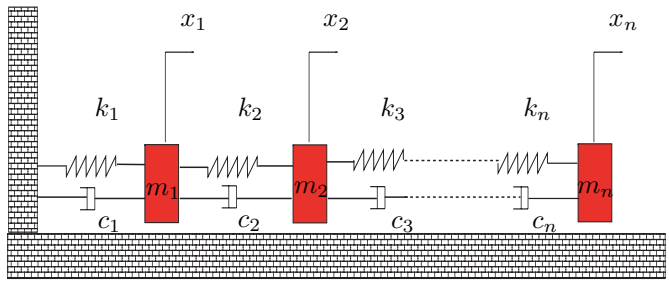


FIG. 5.1. *A damped mass-spring system.*

For $i = 1, \ldots, n$, denote the differences

$$X_i := x_i - x_{i-1}, \tag{5.3}$$

$$Y_i := y_i - y_{i-1} \tag{5.4}$$

with the notation $x_0 = y_0 = 0$. The algebraic system (1.3) characterizing the QIEP can be expressed as a system of n pairs of equations:

$$\begin{cases} \left((\alpha^2 - \beta^2)x_n - 2\alpha\beta y_n\right) m_n + (\alpha X_n - \beta Y_n)c_n + X_n k_n &= 0, \\ \left(2\alpha\beta x_n + (\alpha^2 - \beta^2)y_n\right) m_n + (\alpha Y_n + \beta X_n)c_n + Y_n k_n &= 0 \end{cases} \tag{5.5}$$

and

$$\begin{cases} \left((\alpha^2 - \beta^2)x_i - 2\alpha\beta y_i\right) m_i + (\alpha X_i - \beta Y_i)c_i - (\alpha X_{i+1} - \beta Y_{i+1})c_{i+1} \\ \qquad + X_i k_i - X_{i+1}k_{i+1} = 0, \\ \left(2\alpha\beta x_i + (\alpha^2 - \beta^2)y_i\right) m_i + (\alpha Y_i + \beta X_i)c_i - (\alpha Y_{i+1} + \beta X_{i+1})c_{i+1} \\ \qquad + Y_i k_i - Y_{i+1}k_{i+1} = 0 \end{cases} \tag{5.6}$$

for $i = n-1, \ldots, 1$. Our goal is to seek *positive* solutions $c_1, \ldots, c_n$ and $k_1, \ldots, k_n$ in terms of *positive* values of $m_1, \ldots, m_n$. Toward that end, we find it convenient to rewrite the above equations in matrix form. Define

$$\Gamma := -\begin{bmatrix} \alpha^2 - \beta^2 & -2\alpha\beta \\ 2\alpha\beta & \alpha^2 - \beta^2 \end{bmatrix} = -\Lambda^{2\top} \tag{5.7}$$

and

$$\Theta_i := \begin{bmatrix} \alpha X_i - \beta Y_i & X_i \\ \beta X_i + \alpha Y_i & Y_i \end{bmatrix}, \tag{5.8}$$

$$\mathbf{z}_i := \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{5.9}$$

for $i = 1, \ldots, n$. It is readily seen that (5.5) and (5.6) can be conveniently expressed in the compact form

$$\Theta_n \begin{bmatrix} c_n \\ k_n \end{bmatrix} = m_n \Gamma \mathbf{z}_n, \tag{5.10}$$

$$\Theta_i \begin{bmatrix} c_i \\ k_i \end{bmatrix} = \Theta_{i+1} \begin{bmatrix} c_{i+1} \\ k_{i+1} \end{bmatrix} + m_i \Gamma \mathbf{z}_i, \quad i = n-1, \ldots, 1. \tag{5.11}$$

Clearly, $\det(\Theta_i) = -\beta(X_i^2 + Y_i^2)$. By assuming the generic condition that no two consecutive entries of the given eigenvectors are the same, all matrices Θ_i are invertible. We thus obtain the following recursive relationship:

$$\begin{bmatrix} c_n \\ k_n \end{bmatrix} = m_n \Theta_n^{-1} \Gamma \mathbf{z}_n, \tag{5.12}$$

$$\begin{bmatrix} c_i \\ k_i \end{bmatrix} = \Theta_i^{-1} \Gamma \left[m_n \mathbf{z}_n + m_{n-1} \mathbf{z}_{n-1} + \cdots + m_i \mathbf{z}_i \right], \qquad i = n-1, \ldots, 1, \tag{5.13}$$

where we can express Θ_i^{-1} explicitly as

$$\Theta_i^{-1} = \frac{1}{\beta(X_i^2 + Y_i^2)} \begin{bmatrix} -Y_i & X_i \\ \beta X_i + \alpha Y_i & -\alpha X_i + \beta Y_i \end{bmatrix}.$$

It remains to analyze conditions under which the physical parameters are positive.

Observe that all entries in Γ, Θ_i^{-1}, and $\mathbf{z}_i$, $i = 1, \ldots, n$, are known and fixed from the prescribed eigenpairs. For each $i = 1, \ldots, n$ and $j = i, \ldots, n$, write

$$\Theta_i^{-1} \Gamma \mathbf{z}_j = \begin{bmatrix} a_{ij} \\ b_{ij} \end{bmatrix}. \tag{5.14}$$

Assemble these entries into upper triangular matrices

$$A := \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ 0 & a_{22} & \ldots & a_{2n} \\ \vdots & & & \vdots \\ 0 & 0 & \ldots & a_{nn} \end{bmatrix}, \quad B := \begin{bmatrix} b_{11} & b_{12} & \ldots & b_{1n} \\ 0 & b_{22} & \ldots & b_{2n} \\ \vdots & & & \vdots \\ 0 & 0 & \ldots & b_{nn} \end{bmatrix},$$

respectively. From (5.12) and (5.13), we now recast the QIEP as a system of inequalities.

THEOREM 5.1. *Solving for positive physical parameters c_i and k_i in terms of positive masses m_i, $i = 1, \ldots, n$, for the QIEP amounts to showing that the system of inequalities*

$$\begin{bmatrix} A \\ B \end{bmatrix} \mathbf{m} > 0 \tag{5.15}$$

is consistent *for some* $\mathbf{m} = [m_1, \ldots, m_n]^\top > 0$.

5.2. Real eigenpairs. Assume that the given pair of eigenvalues are real and distinct. We denote the prescribed eigenpairs via the matrices

$$\Lambda = \begin{bmatrix} \alpha & 0 \\ 0 & \gamma \end{bmatrix}, \qquad X = \begin{bmatrix} x_1 & x_2 & \ldots & x_n \\ y_1 & y_2 & \ldots & y_n \end{bmatrix}^\top, \tag{5.16}$$

where the two columns of X are the two eigenvectors associated with eigenvalues α and γ, respectively. The corresponding algebraic equation (1.3) can be formulated as the following n pairs of equations:

$$\begin{cases} \alpha^2 m_n x_n + \alpha X_n c_n + X_n k_n &= 0, \\ \gamma^2 m_n y_n + \gamma Y_n c_n + Y_n k_n &= 0 \end{cases} \tag{5.17}$$

and

$$\left\{ \begin{array}{rcl} \alpha^2 m_i x_i + \alpha X_i c_i - \alpha X_{i+1} c_{i+1} + X_i k_i - X_{i+1} k_{i+1} &=& 0, \\ \gamma^2 m_i y_i + \gamma Y_i c_i - \gamma Y_{i+1} c_{i+1} + Y_i k_i - Y_{i+1} k_{i+1} &=& 0 \end{array} \right. \tag{5.18}$$

for $i = 1, \ldots, n-1$. Defining the 2×2 matrices

$$\Pi_i := -\begin{bmatrix} \alpha X_i & X_i \\ \gamma Y_i & Y_i \end{bmatrix}, \qquad i = 1, \ldots, n, \tag{5.19}$$

we obtain equivalent equations in compact form:

$$\Pi_n \begin{bmatrix} c_n \\ k_n \end{bmatrix} = m_n \Lambda^2 \mathbf{z}_n, \tag{5.20}$$

$$\Pi_i \begin{bmatrix} c_i \\ k_i \end{bmatrix} = \Pi_{i+1} \begin{bmatrix} c_{i+1} \\ k_{i+1} \end{bmatrix} + m_i \Lambda^2 \mathbf{z}_i, \qquad i = n-1, \ldots, 1. \tag{5.21}$$

Assuming that all matrices Π_i are nonsingular, again we obtain a recursive relationship,

$$\begin{bmatrix} c_n \\ k_n \end{bmatrix} = m_n \Pi_n^{-1} \Lambda^2 \mathbf{z}_n, \tag{5.22}$$

$$\begin{bmatrix} c_i \\ k_i \end{bmatrix} = \Pi_i^{-1} \Lambda^2 \left[m_n \mathbf{z}_n + m_{n-1} \mathbf{z}_{n-1} + \cdots + m_i \mathbf{z}_i \right], \qquad i = n-1, \ldots, 1, \tag{5.23}$$

which is in the same format as that in (5.12) and (5.13). If we write

$$\Pi_i^{-1} \Lambda^2 \mathbf{z}_j = \begin{bmatrix} a_{ij} \\ b_{ij} \end{bmatrix}, \tag{5.24}$$

then determining positive c_i and k_i in terms of positive m_i would result in a system of inequalities similar to (5.15).

5.3. Numerical algorithm. We have shown above that solving a QIEP with damping from two given eigenpairs, both of which are either real-valued or complex-conjugate, is equivalent to solving a linear (strict) inequality system of the form (5.15). Without causing any ambiguity, we shall use the same notation A and B to represent the coefficient matrices involved in the inequality systems for both cases. Specifically, recall that both A and B are upper triangular matrices whose entries are defined by either (5.14) or (5.24). The question now is to determine the conditions on the prescribed eigenpairs under which the QIEP is solvable. In contrast to the QIEP with no damping in the preceding section, we are not aware of any analytic way to characterize the solvability of the QIEP with damping. To check the feasibility, we will have to resort to some numerical procedures.

Solving inequality systems of convex functions is a classical problem that has been discussed extensively in the literature, including the Farkas lemma [16], the von Neumann theorem [29, pp. 138–143], and the Ky Fan theorem [15, Theorem 1]. As an example, we state without proof the strong alternative theorem [4, pp. 260–262] as follows.

THEOREM 5.2. *The system of strictly linear inequalities* $\mathcal{A}\mathbf{x} < \mathbf{b}$ *is infeasible if and only if there exists a vector* $\boldsymbol{\omega} \geq 0$, $\boldsymbol{\omega} \neq \mathbf{0}$, *such that* $\mathcal{A}^\top \boldsymbol{\omega} = \mathbf{0}$ *and* $\mathbf{b}^\top \boldsymbol{\omega} \leq 0$.

For our application, we embed the need of $\mathbf{m} > 0$ in the inequalities and consider the augmented system $\mathcal{A}\mathbf{m} > 0$ with

$$\mathcal{A} = \begin{bmatrix} A \\ B \\ I \end{bmatrix}.$$

By Theorem 5.2, we see that $\mathcal{A}\mathbf{m} > 0$ is infeasible, and, hence, our (serially linked with damping) structured QIEP is not solvable if and only if there exist nonnegative vectors $\boldsymbol{\omega}_1, \boldsymbol{\omega}_2$, and $\boldsymbol{\omega}_3$, not all zero, such that $A^\top \boldsymbol{\omega}_1 + B^\top \boldsymbol{\omega}_2 + \boldsymbol{\omega}_3 = \mathbf{0}$. This is equivalent to the statement that $A^\top \boldsymbol{\omega}_1 + B^\top \boldsymbol{\omega}_2 \leq \mathbf{0}$ for some nonnegative, but not both zero, vectors $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$. Such a result of duality, though interesting in theory, might not be easy to realize in computation [2]. Instead, we propose a much simpler numerical approach.

Observe that the inequality system (5.15) is solvable for some $\mathbf{m} > 0$ if and only if it is solvable for $c\mathbf{m}$ with any $c > 0$. By scaling the variable $\mathbf{m}$ if necessary, we can recast our problem of solving an inequality system into the problem of finding a minimax solution of a multivariable function [19]. More specifically, we consider the problem

$$\max_{0 \leq \mathbf{m} \leq 1} \min \{A\mathbf{m}, B\mathbf{m}\}, \tag{5.25}$$

where for each given $\mathbf{m} \in \mathbb{R}^n$ the minimum is taken over the $2n$ entries of the vectors $A\mathbf{m}$ and $B\mathbf{m}$. It is not difficult to observe the following fact.

THEOREM 5.3. *If the inequality system* (5.15) *is consistent, then the maximal objective value in* (5.25) *is positive. Conversely, if* (5.15) *is inconsistent, then the optimal objective value to* (5.25) *is necessarily zero with* $\mathbf{m} = 0$.

Note that the function $\min\{A\mathbf{m}, B\mathbf{m}\}$ is concave in $\mathbf{m}$. Thus its maximal value over the convex domain $\{\mathbf{m} \in \mathbb{R}^n | 0 \leq m_i \leq 1,\ i = 1, \ldots, n\}$ is unique. We can easily convert (5.25) into a convex programming problem and there are readily available routines to solve (5.25). For example, the MATLAB routine fminimax implements a sequential quadratic programming method and seems capable of handling our problem reasonably well. With the equivalence of (5.15) and (5.25) in mind, we propose the following algorithm.

ALGORITHM 5.1. *Given two arbitrary complex-conjugate (or real-valued) eigenpairs, the following steps generally will either construct the masses* $m_1, \ldots, m_n$, *the damping constants* $c_1, \ldots, c_n$, *and the spring constants* $k_1, \ldots, k_n$, *all positive and* $m_i \leq 1$, *for the pencil* $\lambda^2 M + \lambda C + K$ *with the prescribed eigenpairs, where* M, C, *and* K *are of the structure specified in* (4.1) *and* (5.1), *or determine that such a system does not exist. In the rare case where the algorithm is terminated prematurely, the prescribed eigenpairs are not generic enough in the sense that one of the matrices* Θ_i *defined in* (5.8) *(or* Π_i *defined in* (5.19)*) is singular for some* $i = 1, \ldots, n$.

1. *Let* (Λ, X) *represent the given eigenpairs in the sense of* (5.2) *(or* (5.16)*).*
   (a) *For* $i = 1, \ldots, n$, *form vectors* $\mathbf{z}_i$ *according to* (5.9), *and compute the differences* X_i *and* Y_i *according to* (5.3) *and* (5.4), *respectively.*
   (b) *Check to see that* Θ_i *(or* Π_i*) is nonsingular for all* $i = 1, \ldots, n$. *Otherwise, report that the given spectral information is not generic and stop.*
2. *Form the upper triangular matrices* A *and* B *according to* (5.14) *(or* (5.24)*).*
3. *Solve the minimax problem* (5.25).
   (a) *If a solution* $\mathbf{m} > 0$ *exists, define* $\mathbf{c} = A\mathbf{m}$ *and* $\mathbf{k} = B\mathbf{m}$.
   (b) *If a solution* $\mathbf{m} > 0$ *does not exist, the given eigenpairs are not feasible.*

6. Numerical experiment. In this section we demonstrate the above ideas by some numerical examples.

Example 4. We first randomly generate the data

$$\mathbf{m} = [1.0660, 0.2593, 0.6809, 0.5453, 1.0000]^\top,$$

$$\mathbf{k} = [7.6210, 4.5647, 0.1850, 8.2141, 4.4470]^\top$$

and create an undamped pencil $\lambda^2 M + K$ in the form (4.1). The resulting system has natural frequencies

$$5.7620, 4.9207, 2.5971, 2.3485, 0.2768,$$

with columns in the matrix

$$\begin{bmatrix} -0.0080 & -0.5440 & -0.1126 & 0.7932 & 0.0153 \\ 0.0405 & 1.6238 & -0.1232 & 1.0958 & 0.0405 \\ -0.6480 & -0.0019 & 0.7796 & 0.0902 & 0.6579 \\ 1.1197 & -0.0347 & 0.3641 & 0.0263 & 0.6677 \\ -0.1732 & 0.0078 & -0.7046 & -0.1095 & 0.6794 \end{bmatrix},$$

as the corresponding natural modes. We point out earlier that the order of frequencies corresponds precisely to the number of sign changes in the eigenvector. Suppose now that we use the third eigenpair as the given eigenpair and apply Algorithm 4.1 to reconstruct the quadratic pencil. To illustrate the working of Algorithm 4.1, we allow the mass increasing factor s_p to vary from 1.001 to 10^5. The resulting allowable masses and the corresponding frequencies are plotted in the top and bottom graphs of Figure 6.1, respectively. Clearly, the solution to the QIEP is not unique. Note that in this particular example, only the second and the third masses are changed by our algorithm, whereas in all cases the third frequency stays invariant. The experiment also suggests an interesting observation that an increase on the second and the third masses, while maintaining the third eigenpair, has more effect on the first eigenvalue than on the fourth or the fifth eigenvalue.

Example 5. We randomly generate the data

$$\mathbf{m} = [3.4915, 4.4928, 5.2297, 4.1880, 1.0000]^\top,$$

$$\mathbf{c} = [0.0579, 0.3529, 0.8132, 0.0099, 0.1389]^\top,$$

$$\mathbf{k} = [4.0571, 9.3547, 9.1690, 4.1027, 8.9365]^\top$$

and create a pencil $\lambda^2 M + \lambda C + K$ in the form (5.1). The resulting system has 5 pairs of complex-conjugate eigenvalues,

$$-0.0863 \pm 3.3596i, -0.1792 \pm 2.6044i, -0.0792 \pm 1.7918i,$$

$$-0.0079 \pm 1.0365i, -0.0019 \pm 0.3736i,$$

not necessarily in any particular order, and the corresponding eigenvectors

$$\begin{bmatrix} 0.0005 \mp 0.0005i & 0.0696 \mp 0.2457i & -0.2044 \pm 0.3474i & -0.2495 \pm 0.3822i & -0.3451 \pm 0.1371i \\ -0.0012 \pm 0.0015i & -0.1084 \pm 0.2648i & -0.0325 \pm 0.0894i & -0.2572 \pm 0.3949i & -0.4763 \pm 0.1907i \\ 0.0019 \mp 0.0056i & 0.0566 \mp 0.1061i & 0.1583 \mp 0.3303i & -0.1457 \pm 0.1913i & -0.5767 \pm 0.2341i \\ -0.0176 \pm 0.0597i & -0.0018 \pm 0.0095i & -0.0335 \pm 0.0987i & 0.3470 \mp 0.5006i & -0.7014 \pm 0.2830i \\ 0.0676 \mp 0.2260i & -0.0181 \pm 0.0339i & -0.0575 \pm 0.1522i & 0.3943 \mp 0.5691i & -0.7125 \pm 0.2875i \end{bmatrix}.$$

Suppose we use the first complex-conjugate eigenpairs as the given eigenpairs and apply Algorithm 5.1 to reconstruct the quadratic pencil. We solve the minimax

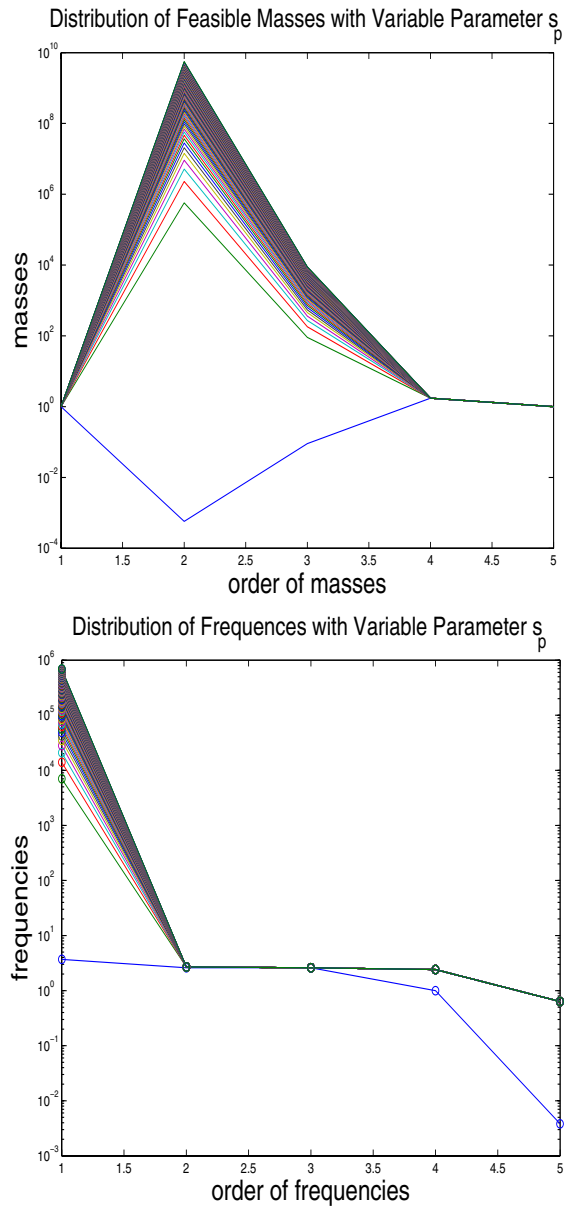


FIG. 6.1. *Allowable masses and frequency responses for the undamped system.*

problem (5.25) by employing the MATLAB code fminimax with tolerance TolFun = TolCon $= 10^{-8}$. We obtain a solution to the QIEP at

$$\mathbf{m} = [0.0512, 0.2413, 0.5096, 0.1653, 0.0326]^\top,$$

$$\mathbf{c} = [0.0045, 0.0045, 0.0588, 0.0045, 0.0045]^\top,$$

$$\mathbf{k} = [0.0045, 0.1520, 0.6021, 0.4316, 0.2912]^\top.$$

Note that $\mathbf{m}$ is limited to the range $[0, 1]$, whereas we could have normalized m_5 to unit mass without affecting the spectrum. An examination of the remaining eigenvalues does not seem to suggest that there is anything analogous to the Courant nodal line theorem for the damped case.

7. Conclusion. We have studied the QIEP arising from dynamical systems whose vibratory properties can be modeled after a serially linked mass-spring oscillator. The resulting QIEP must oblige a special matrix structure. In particular, the physical parameters must be positive. These constraints severely limit the solvability of the QIEP.

We show that for the undamped problem, the order of frequencies corresponds precisely to the number of sign changes in the eigenvectors. A make-or-break algorithm is proposed in this work. For the damped problem, we show that the solvability of the QIEP is equivalent to the consistency of a certain system of inequalities which can be solved via a minimax algorithm.

Thus far, the permissible eigenpair information is rather limited. For the undamped problem only one eigenpair is allowed, while for the damped problem only two eigenpairs are allowed. In either case, the masses $\mathbf{m}$ are used as the free parameters to express the other two physical parameters $\mathbf{c}$ and $\mathbf{k}$. In certain applications, it is desirable to express $\mathbf{m}$ and $\mathbf{c}$ in terms of $\mathbf{k}$. However, such a task is much harder to analyze, and we have not obtained significant results yet.

If the QIEP is solvable, usually there are multiple solutions. Although we are able to match the prescribed eigenpair information in our construction, our technique thus far cannot control the remaining eigenvalues or eigenvectors. It remains an open question whether additional eigenvalue information, not eigenvector information, can help to further restrict the number of solutions or control the remaining eigenstructure.

REFERENCES

[1] M. Baruch, *Optimization procedure to correct stiffness and flexibility matrices using vibration data*, AIAA J., 16 (1978), pp. 1208–1210.

[2] J. E. Beasley, *Advances in Linear and Integer Programming*, Oxford University Press, Oxford, UK, 1996.

[3] A. Berman and E. J. Nagy, *Improvement of a large analytical model using test data*, AIAA J., 21 (1983), pp. 1168–1173.

[4] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, Cambridge, UK, 2004.

[5] J. Carvalho, *State Estimation and Finite-Element Model Updating for Vibrating Systems*, Ph.D. dissertation, Northern Illinois University, Dekalb, IL, 2002.

[6] M. T. Chu, B. N. Datta, W.-W. Lin, and S.-F. Xu, *The spill-over phenomenon in quadratic model updating*, AIAA J., to appear.

[7] M. T. Chu and G. H. Golub, *Structured inverse eigenvalue problems*, Acta Numer., 11 (2002), pp. 1–71.

[8] M. T. Chu and G. H. Golub, *Inverse Eigenvalue Problems: Theory, Algorithms, and Applications*, Oxford University Press, Oxford, UK, 2005.

[9] M. T. Chu, Y.-C. Kuo, and W.-W. Lin, *On inverse quadratic eigenvalue problems with partially prescribed eigenstructure*, SIAM J. Matrix Anal. Appl., 25 (2004), pp. 995–1020.

[10] R. Courant and D. Hilbert, *Methods of Mathematical Physics*, *Vol.* 1, Interscience, New York, 1953.

[11] B. N. Datta, *Finite element model updating, eigenstructure assignment and eigenvalue embedding techniques for vibrating systems*, Mech. Sys. Signal Processing, Special Volume on "Vibration Control," 16 (2002), pp. 83–96.

[12] B. N. Datta, S. Elhay, Y. M. Ram, and D. R. Sarkissian, *Partial eigenstructure assignment for the quadratic pencil*, J. Sound Vibration, 230 (2000), pp. 101–110.

[13] B. N. Datta and D. R. Sarkissian, *Theory and computations of some inverse eigenvalue problems for the quadratic pencil*, in Structured Matrices in Mathematics, Computer Science, and Engineering, Contemp. Math. 280, AMS, Providence, RI, 2001, pp. 221–240.
[14] D. J. Ewins, *Adjustment or updating of models*, Sādhanā, 25 (2000), pp. 235–245.
[15] K. Fan, I. Glicksbert, and A. J. Hoffman, *Systems of inequalities involving convex functions*, Proc. Amer. Math. Soc., 8 (1957), pp. 617–622.
[16] J. Farkas, *Theorie der einfachen Ungleichungen*, J. Reine Angew. Math., 124 (1902), pp. 1–27.
[17] M. I. Friswell, D. J. Inman, and D. F. Pilkey, *The direct updating of damping and stiffness matrices*, AIAA J., 36 (1998), pp. 491–493.
[18] M. I. Friswell and J. E. Mottershed, *Finite Element Model Updating in Structural Dynamics*, Kluwer Academic Publishers, Boston, 1995.
[19] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, London, 1981.
[20] G. M. L. Gladwell, *Inverse Problems in Vibration*, 2nd ed., Martinus Nijhoff, Dordrecht, The Netherlands, 2004.
[21] Y.-C. Kuo, W.-W. Lin, and S.-F. Xu, *New methods for finite element model updating problems*, AIAA J., 44 (2006), pp. 1310–1316.
[22] Y.-C. Kuo, W.-W. Lin, and S.-F. Xu, *Solutions of the partially described inverse quadratic eigenvalue problem*, SIAM J. Matrix Anal. Appl., 29 (2006), pp. 33–53.
[23] P. Lancaster and U. Prells, *Inverse problems for damped vibration systems*, J. Sound Vibration, 283 (2005), pp. 891–914.
[24] N. K. Nichols and J. Kautsky, *Robust eigenstructure assignment in quadratic matrix polynomials: Nonsingular case*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 77–102.
[25] Y. M. Ram and S. Elhay, *An inverse eigenvalue problem for the symmetric tridiagonal quadratic pencil with application to damped oscillatory systems*, SIAM J. Appl. Math., 56 (1996), pp. 232–244.
[26] W. H. A. Schilders and H. A. van der Vorst, *Model Order Reduction, Coupled Problems and Optimzation*, http://www.lorentzcenter.nl/lc/web/2005/160/info.php3?wsid=160.
[27] L. Starek and D. J. Inman, *Symmetric inverse eigenvalue vibration problem and its applications*, Mech. Sys. Signal Process., 15 (2001), pp. 11–29.
[28] F. Tisseur and K. Meerbergen, *The quadratic eigenvalue problem*, SIAM Rev., 43 (2001), pp. 235–286.
[29] J. von Neumann and O. Morgenstern, *Theory of Games and Economic Behavior*, Princeton University Press, Princeton, NJ, 2004.
[30] F.-S. Wei, *Mass and stiffness interaction effects in analytical model modification*, AIAA J., 28 (1990), pp. 1686–1688.
[31] H.-M. Zhu, *Courant's Nodal Line Theorem and Its Discrete Counterparts*, Ph.D. dissertation, University of Waterloo, Waterloo, Ontario, Canada, 2000.

SIAM J. SCI. COMPUT.
Vol. 29, No. 6, pp. 2686–2704
© 2007 Society for Industrial and Applied Mathematics

PRESSURE SCHUR COMPLEMENT PRECONDITIONERS FOR THE DISCRETE OSEEN PROBLEM*

MAXIM A. OLSHANSKII† AND YURI V. VASSILEVSKI‡

Abstract. We consider several preconditioners for the pressure Schur complement of the discrete steady Oseen problem. Two of the preconditioners are well known from the literature and the other is new. Supplemented with an appropriate approximate solve for an auxiliary velocity subproblem, these approaches give rise to a family of the block preconditioners for the matrix of the discrete Oseen system. In the paper we critically review possible advantages and difficulties of using various Schur complement preconditioners. We recall existing eigenvalue bounds for the preconditioned Schur complement and prove such for the newly proposed preconditioner. These bounds hold both for LBB stable and stabilized finite elements. Results of numerical experiments for several model two-dimensional and three-dimensional problems are presented. In the experiments we use LBB stable finite element methods on uniform triangular and tetrahedral meshes. One particular conclusion is that in spite of essential improvement in comparison with "simple" scaled mass-matrix preconditioners in certain cases, none of the considered approaches provides satisfactory convergence rates in the case of small viscosity coefficients and a sufficiently complex (e.g., circulating) advection vector field.

Key words. Navier–Stokes equations, Oseen equations, finite element, iterative methods, preconditioning, pressure Schur complement

AMS subject classifications. 65F10, 65N22, 65F50

DOI. 10.1137/070679776

1. Introduction.

We consider the numerical solution of the steady Oseen problem: Given a divergence free advection velocity $\mathbf{w} : \Omega \to \mathbb{R}^d$, force field $\mathbf{f} : \Omega \to \mathbb{R}^d$, and boundary data $\mathbf{g} : \partial\Omega \to \mathbb{R}^d$, find a velocity $\mathbf{u} : \Omega \to \mathbb{R}^d$ and a pressure $p : \Omega \to \mathbb{R}$ such that

$$-\nu\Delta\mathbf{u} + (\mathbf{w}\cdot\nabla)\,\mathbf{u} + \nabla p = \mathbf{f} \quad \text{in } \Omega, \tag{1.1}$$

$$\operatorname{div}\mathbf{u} = 0 \quad \text{in } \Omega, \tag{1.2}$$

$$\mathbf{u} = \mathbf{g} \quad \text{on } \partial\Omega, \tag{1.3}$$

where $\Omega \subset \mathbb{R}^d$ $(d = 2, 3)$ is a bounded, connected domain with a piecewise smooth Lipschitz boundary $\partial\Omega$. The coefficient $\nu > 0$ is a given kinematic viscosity. For the sake of uniqueness of p one may impose some additional condition, such as $\int_\Omega p\,dx = 0$.

A necessity of solving Oseen equations numerically is commonly related to using Picard-type iterative methods to find a solution to steady Navier–Stokes problems. In this case $\mathbf{w}$ is an approximation of the velocity from previous iterative steps, so it is updated on every nonlinear iteration. Among another applications we mention Uzawa-type algorithms for the augmented variational inequality approach to the modeling of

*Received by the editors January 10, 2007; accepted for publication (in revised form) April 10, 2007; published electronically November 2, 2007. This research was supported by the Russian Foundation for Basic Research and the German Research Foundation through grants DFG-RFBR 06-01-04000, RFBR 05-01-00864.

http://www.siam.org/journals/sisc/29-6/67977.html

†Department of Mechanics and Mathematics, Moscow State M. V. Lomonosov University, Moscow 119899, Russia (Maxim.Olshanskii@mtu-net.ru).

‡Institute of Numerical Mathematics, Russian Academy of Sciences, Moscow 117333, Russia (vasilevs@dodo.inm.ras.ru).

Bingham fluids; see, e.g., [24]. Again one may need to solve discrete Oseen systems many times with different $\mathbf{w}$ and $\mathbf{f}$. Thus there is a demand for efficient iterative solvers for the discrete Oseen problem. We note that besides (1.3) other boundary conditions may be imposed in various models. Furthermore we remark that when using different boundary conditions special attention may be required.

In this paper we consider a finite element method to discretize (1.1)–(1.3). However, the linear algebraic solvers discussed here can be applied in a finite difference or finite volume context in the same manner. We assume that the finite element velocity (not necessarily conforming) and pressure spaces $\mathbb{V}_h$ and $\mathbb{Q}_h$ approximate $\mathbf{H}_0^1(\Omega)$ and $L_2^0(\Omega) := \{q \in L_2(\Omega) : (q,1) = 0\}$, respectively. Consider the following finite element problem: Find $\mathbf{u}_h \in \mathbb{V}_h$ and $p_h \in \mathbb{Q}_h$ satisfying

$$a_h(\mathbf{u}_h, \mathbf{v}_h) - (p_h, \operatorname{div} \mathbf{v}_h) + (q_h, \operatorname{div} \mathbf{u}_h) + c_h(p_h, q_h) = (\mathbf{f}_h, \mathbf{v}_h) + (g_h, q_h) \quad \forall\ \mathbf{v}_h \in \mathbb{V}_h,\ q_h \in \mathbb{Q}_h. \tag{1.4}$$

The bilinear form $a_h(\cdot,\cdot)$ may include some stabilizing terms for the advection dominated case. The nonnegative bilinear form $c_h(\cdot,\cdot)$ may be included in the finite element formulation if $\mathbb{V}_h$ and $\mathbb{Q}_h$ form an LBB unstable pair [21]; otherwise, $c_h(\cdot,\cdot) \equiv 0$. Denote by $(\cdot,\cdot)_{\mathbb{V}}$ the energy scalar product on $\mathbb{V}_h$ satisfying $(\psi,\phi)_{\mathbb{V}} = (\nabla\psi, \nabla\phi)$ for $\psi, \phi \in \mathbb{V}_h \cap \mathbf{H}_0^1$. For the bilinear forms a_h and c_h we assume ellipticity, continuity, and stability conditions

$$\alpha_1 \|\mathbf{v}_h\|_{\mathbb{V}}^2 \le a_h(\mathbf{v}_h, \mathbf{v}_h), \qquad a_h(\mathbf{v}_h, \mathbf{u}_h) \le \alpha_2 \|\mathbf{v}_h\|_{\mathbb{V}} \|\mathbf{u}_h\|_{\mathbb{V}} \quad \forall\ \mathbf{v}_h, \mathbf{u}_h \in \mathbb{V}_h, \tag{1.5}$$

$$\gamma_1^2 \|q_h\|^2 \le \sup_{\mathbf{v}_h \in \mathbb{V}_h} \frac{(q_h, \operatorname{div} \mathbf{v}_h)^2}{\|\mathbf{v}_h\|_{\mathbb{V}}^2} + c_h(q_h, q_h) \quad \forall\ q_h \in \mathbb{Q}_h, \tag{1.6}$$

$$c_h(q_h, p_h) \le \gamma_2 \|q_h\| \|p_h\|, \quad (q_h, \operatorname{div} \mathbf{v}_h) \le \gamma_3 \|q_h\| \|\mathbf{v}_h\|_{\mathbb{V}} \quad \forall\ q_h, p_h \in \mathbb{Q}_h,\ \mathbf{v}_h \in \mathbb{V}_h, \tag{1.7}$$

with positive mesh-independent constants α_1, α_2, γ_1, γ_2, and γ_3. We note that conditions (1.6) and (1.7) are common for the pressure stabilized finite element methods; see, e.g., the recent studies in [8]. For the LBB stable pairs (1.6) and (1.7) trivially hold.

Let $\{\phi_i\}_{1 \le i \le n}$ and $\{\psi_j\}_{1 \le j \le m}$ be bases of $\mathbb{V}_h$ and $\mathbb{Q}_h$, respectively. Define the following matrices:

$$A_{i,j} = a_h(\phi_j, \phi_i), \quad B_{i,j} = -(\operatorname{div} \phi_j, \psi_i), \quad C_{i,j} = c_h(\psi_j, \psi_i).$$

The linear algebraic system corresponding to (1.4) (*the discrete Oseen system*) takes the form

$$\begin{pmatrix} A & B^T \\ B & -C \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}. \tag{1.8}$$

We are interested in solving (1.8) by a preconditioned iterative method. Following the conventional approach [21, 29, 12] we consider the block triangular preconditioner for the system (1.8):

$$\mathcal{P} = \begin{pmatrix} \hat{A} & B^T \\ O & -\hat{S} \end{pmatrix}. \tag{1.9}$$

The matrix $\hat{A}$ is a preconditioner for the matrix A, such that $\hat{A}^{-1}$ may be considered as an inexact solver for linear systems involving A. The matrix $\hat{S}$ is a preconditioner for the pressure Schur complement of (1.8) $S = BA^{-1}B^T + C$. In an iterative algorithm one needs the actions of $\hat{A}^{-1}$ and $\hat{S}^{-1}$ on subvectors, rather than the matrices $\hat{A}$, $\hat{S}$ explicitly. Once good preconditioners for A and S are given, an appropriate Krylov subspace iterative method for (1.8) with the block preconditioner (1.9) is an efficient solver. In the literature one can find geometric or algebraic multigrid (see, e.g., [21] and references therein) or domain decomposition [23, 35] iterative algorithms which provide effective preconditioners $\hat{A}$ for a wide range of ν and various meshes. However, despite considerable recent effort and progress, building a preconditioner for S, which is robust for a wide range of parameters (especially viscosity), discretizations, and meshes, is still a challenge.

In this paper we recall two recent approaches to construct a preconditioner for S. One is due to Kay, Loghin, and Wathen [28], and another is from Elman and coauthors [16, 18]. Furthermore, in attempting to overcome some difficulties associated with these approaches we consider a new preconditioner for S. We give motivation for different choices of $\hat{S}^{-1}$, prove eigenvalue bounds, and present results of several numerical experiments.

As already mentioned, a good preconditioner $\hat{S}$ is necessary for building the block triangular preconditioner (1.9). Furthermore, there exist other numerical methods for the incompressible Navier–Stokes equations, where finding a proper approximation to S is vital. These methods are based on the Uzawa method and its variants [22, 40, 10], Arrow–Hurwicz [22, 1], SOR [14], and special factorizations [3, 5] for the linearized problems. Moreover, the performance of the widely used splitting algorithms like SIMPLE, projection, or pressure correction methods for the time integration of unsteady problem is also closely related with the issue of the Schur complement preconditioning [34, 38, 37].

There are also iterative methods for solving (1.8) which do not require consideration of the Schur complement S or its preconditioner, at least explicitly. Among these are coupled multigrid methods of Vanka type [27, 38], methods based on Hermitian splitting [4], augmented Lagrangian based preconditioning [6], and implicit-factorization preconditioning [15, 13]; see also the review article [5]. It is not the intention of this paper to discuss or compare these methods. We note only that implementing some of them in a purely algebraic manner may involve serious difficulties [41]. Thus the pressure Schur complement based block solvers remain attractive for treating "real-life" engineering problems and have potential for developing black-box algorithms.

The remainder of the paper is organized as follows. In section 2 we consider two well-known preconditioners for S and present a new approach. In section 3 we prove eigenvalue bounds. For the new preconditioner the h-independent bounds both for the LBB stable and the pressure stabilized discretizations are shown. In section 4 results of numerical experiments with different preconditioners are given for two-dimensional (2D) and three-dimensional (3D) problems discretized on simplicial meshes.

2. Schur complement preconditioners. In this paper all variants of preconditioner $\hat{S}$ are defined through their inverses. Before proceeding to the preconditioners we define the pressure mass, velocity mass, and Laplacian matrices:

$$(M_p)_{i,j} = (\psi_j, \psi_i), \quad (M_u)_{i,j} = (\phi_j, \phi_i), \quad L_{i,j} = (\phi_j, \phi_i)_{\mathbb{V}}.$$

2.1. Pressure convection-diffusion. We first consider the *pressure convection-diffusion* (PCD) preconditioner, proposed by Kay, Loghin, and Wathen [28] and studied by these and other authors (see [21]):

$$\hat{S}^{-1} := \hat{M}_p^{-1} A_p L_p^{-1}. \tag{2.1}$$

Here $\hat{M}_p^{-1}$ denotes an approximate solve with the pressure mass matrix. Matrices A_p and L_p are approximations to convection-diffusion and Laplacian operators in $\mathbb{Q}_h$, respectively. Note that both A_p and L_p need some boundary conditions to be prescribed.

In discretizations of (1.1)–(1.3) with continuous pressure approximations, one can use the conforming discretization of the pressure Poisson problem with Neumann boundary conditions. The corresponding finite element formulation for the case of the Neumann conditions is standard: Find $p_h \in \mathbb{Q}_h$ such that

$$(\nabla p_h, \nabla q_h) = (f, q_h) \qquad \forall\ q_h \in \mathbb{Q}_h \subset \mathrm{H}^1(\Omega).$$

Neumann boundary conditions are conventionally set for the convection-diffusion problem on $\mathbb{Q}_h$.

An alternative way to define L_p is to set $L_p = (B\hat{M}_u^{-1}B^T)$, where $\hat{M}_u$ is a diagonal approximation to the velocity mass matrix. Although a diagonal matrix might be a poor approximation to M_u in the case of anisotropic grids, this operator can be seen as a mixed discretization of the pressure Poisson problem with Neumann boundary conditions. Using $(B\hat{M}_u^{-1}B^T)$ is convenient in the case of discontinuous pressures. It is not straightforward to define A_p for the case of discontinuous pressures; see section 4 for a definition of A_p in the case of isoP_2-P_0 elements.

Analysis of the PCD preconditioner and numerical results found in the literature (see also section 4) recover several specific features of the method. In particular, the advantages of using this preconditioner are the following:

1. The PCD preconditioner provides mesh-independent convergence rates for moderate values of ν.
2. Dependence of the convergence rates on ν^{-1} is significantly improved in comparison to a scaled pressure mass matrix.
3. Numerical results [42, 30] suggest that the preconditioner is not very sensitive to grid anisotropy, at least for some discretizations. Thus in [42] the PCD preconditioning was used to solve the Oseen problem on every step of Picard iterations for the 3D driven cavity problem discretized with Q_2-Q_1 finite elements on a regular grid. The number of iterations was essentially independent of the aspect ratio of elements. In [30] the preconditioner was successfully used to compute a flow around an obstacle in a 3D cube, using isoP_2-P_1 elements on unstructured grid and allowing elements with high aspect ratios.
4. Some theoretical analysis of the preconditioner is available in [20], where the authors prove eigenvalue estimates for the preconditioned system. We recall these results in section 3.
5. The preconditioner can be used both for LBB stable and pressure stabilized discretizations.

There are also some open questions associated with using this method:

1. Degradation of the convergence rates as $\nu \to 0$ is easily seen even for the simplest constant flow: $\mathbf{w} = (1, 0)$ or $\mathbf{w} = (1, 0, 0)$.

2. The issue of proper boundary conditions in L_p and A_p is not very well understood. We address this question below in more detail.
3. The A_p matrix does not naturally arise in the original problem. Some effort may be needed to build it, especially for discontinuous pressure approximations.
4. The preconditioner is specifically oriented to the Oseen problem. Its extension to similar problems, like Newton linearization, quasi-Newtonian fluids, the Navier–Stokes system coupled with other equations, or general problems having the same 2×2 structure as (1.8), is not clear.

The choice of boundary conditions for the definition of A_p and L_p depends on boundary conditions in the Oseen problem. In [21] it is recommended that Neumann boundary conditions should be prescribed on those parts of $\partial\Omega$ where in the original formulation of the Oseen problem one has Dirichlet boundary conditions for $\mathbf{u}$, and Dirichlet boundary conditions in A_p and L_p should be used on those parts of $\partial\Omega$ where in the original formulation one has outflow boundary conditions for the stress tensor. A motivation for these recommendations comes from the symmetric case of the unsteady Stokes problem, where this choice proved to work well for the Cahouet–Chabard preconditioner [11, 32]. Our experiments and analysis suggest that for the case of dominating convection ($\nu \to 0$) this choice is not optimal. Below we give arguments that on the inflow boundary for small ν one may prefer using a Dirichlet homogeneous fictitious boundary for constructing A_p. Experimentally we observed that the change of boundary conditions on the outflow part of the boundary is not crucial. On the characteristic parts of $\partial\Omega$ and outflow parts with conditions on the stress tensor (the so-called do-nothing boundary conditions), Neumann boundary conditions in A_p and L_p are more appropriate.

We consider the continuous counterpart of the pressure Schur complement operator: $S := -\operatorname{div}(-\nu\Delta + (\mathbf{w}\cdot\nabla))_0^{-1}\nabla$, where $(-\nu\Delta + (\mathbf{w}\cdot\nabla))_0^{-1}$ is the solution operator for the vector convection-diffusion problem with Dirichlet boundary conditions. The operator S defines a mapping of L_0^2 onto L_0^2. For a given function $p \in L_0^2$ denote $\mathbf{v} = (-\nu\Delta + (\mathbf{w}\cdot\nabla))_0^{-1}\nabla p$, $\mathbf{v} \in \mathbf{H}_0^1(\Omega)$; then one can rewrite the relation $q = S\,p$ in the form

$$-\nu\Delta\mathbf{v} + (\mathbf{w}\cdot\nabla)\,\mathbf{v} - \nabla p = \mathbf{0}, \qquad -\operatorname{div}\mathbf{v} = q \quad \text{in } \Omega, \tag{2.2}$$

$$\mathbf{v}|_{\partial\Omega} = \mathbf{0}. \tag{2.3}$$

Let $\nu = 0$ and prescribe boundary condition (2.3) only on the inflow part Γ_{in} of $\partial\Omega$, i.e., $\Gamma_{in} = \{\mathbf{x} \in \partial\Omega \,:\, \mathbf{w}(\mathbf{x})\cdot\mathbf{n} < 0\}$. For the sake of simplicity we assume the following: the plain patch Γ_{in} is orthogonal to the x-axis, and $\mathbf{w}$ is sufficiently smooth, orthogonal to Γ_{in} at each point, and stays parallel to the x-axis in some neighborhood $\mathcal{O} \subset \Omega$ of Γ_{in}. For simplicity, we consider the 2D case (the 3D case is considered similarly). Integrating the first equation in (2.2) along characteristics in $\mathcal{O}$, using condition $\mathbf{v}|_{\Gamma_{in}} = 0$ and assumptions $\nu = 0$ and $\mathbf{w} = (1, 0)$ in $\mathcal{O}$ one gets the equality

$$\mathbf{v}(x, y) = \int_0^x \nabla p\,(s, y)\; d\,s \quad \text{in } \mathcal{O}. \tag{2.4}$$

Now we compute $\operatorname{div}\mathbf{v}$ in $\mathcal{O}$. If p is sufficiently smooth, the equality (2.4) yields

$$\operatorname{div}\mathbf{v} = \frac{\partial}{\partial x}\int_0^x \frac{\partial}{\partial s} p\,(s, y)\; d\,s + \frac{\partial}{\partial y}\int_0^x \frac{\partial}{\partial y} p\,(s, y)\; d\,s = \frac{\partial p}{\partial x} + \int_0^x \frac{\partial^2 p}{\partial y^2}\; d\,s \quad \text{in } \mathcal{O}.$$

The second equation in (2.2) implies $q = -\partial p/\partial x - \int_0^x \partial^2 p/\partial y^2 \, d\,s$ in $\mathcal{O}$. Therefore, we have in the sense of traces

$$q = -\frac{\partial p}{\partial x} \quad \text{on} \quad \Gamma_{in}. \tag{2.5}$$

On the other hand, the PCD approach suggests approximating $p = S^{-1}q$ by

$$p = -(\mathbf{w} \cdot \nabla)\Delta^{-1} q, \tag{2.6}$$

where Δ^{-1} is a solution operator to the Poisson problem with *some* boundary conditions. We want to define these conditions on Γ_{in} in a way consistent with (2.5). To this end we rewrite (2.6) in $\mathcal{O}$:

$$p = -\frac{\partial r}{\partial x}, \qquad \frac{\partial^2 r}{\partial x^2} + \frac{\partial^2 r}{\partial y^2} = q. \tag{2.7}$$

Relations (2.5) and (2.7) lead to $\partial^2 r/\partial y^2 = 0$ on Γ_{in}. The corresponding homogeneous boundary condition in the definition of Δ^{-1} is $r = 0$ on Γ_{in}. Therefore, for the practically important case when the flow $\mathbf{w}$ is orthogonal to Γ_{in} and ν is small, the reasonable boundary conditions for the PCD preconditioner on the *inflow* are *homogeneous Dirichlet boundary conditions*.

Although the above analysis studies the limit case of $h \to 0$ and $\nu \to 0$ (continuous inviscid problem), we believe it to be useful for understanding the issue of proper boundary conditions in the discrete preconditioner (2.1). In particular, it suggests that the choice of the optimal boundary conditions in $\hat{S}^{-1}$ depends not only on the type of conditions in (1.3), but also on ν and $\mathbf{w}$ (it may be different for inflow, outflow, and characteristic parts of $\partial\Omega$). From the practical standpoint there are still open questions. Thus in the continuous counterpart of the preconditioner ($\hat{S}^{-1} = (\nu\Delta - \mathbf{w}\cdot\nabla)\Delta^{-1}$) one has to prescribe boundary conditions *only* for the Poisson problem solution operator Δ^{-1}, while in the discrete case some boundary conditions are involved in the definition of *both* matrices A_p and L_p. The guess that conditions in A_p and L_p should be the same for optimal performance is not fully confirmed by our numerical experiments. In particular, the best convergence rates were observed with Neumann boundary conditions in L_p and different boundary conditions in A_p depending on ν and $\mathbf{w}$. We do not have a clear explanation of this phenomenon. Furthermore, from the implementation standpoint, Dirichlet boundary conditions for A_p may not be imposed on the nodes at Γ_{in} since these nodes have to contribute to the set of pressure degrees of freedom. For this reason one introduces outside Ω a fictitious one-cell layer attached to Γ_{in}. Dirichlet boundary conditions are assigned at layer nodes not belonging to Γ_{in}. Technical details can be found in section 4.

Results of numerical experiments suggest that the choice of boundary conditions on $\Gamma_{out} = \{\mathbf{x} \in \partial\Omega \,:\, \mathbf{w}(\mathbf{x})\cdot\mathbf{n} > 0\}$ in the preconditioner does not affect its performance in any substantial way. If in the Oseen problem (1.1)–(1.3) instead of Dirichlet conditions one sets the normal component of the stress tensor equal to zero on the outflow,

$$-\nu(\nabla\mathbf{u} + (\nabla\mathbf{u})^T)\cdot\mathbf{n} + 2p\,\mathbf{n} = 0 \quad \text{on} \quad \Gamma_{out}, \tag{2.8}$$

then for $\nu \to 0$ one gets $p = 0$ on Γ_{out}. Now relation (2.6) immediately gives $(\mathbf{w}\cdot\nabla)r = 0$ on Γ_{out}. For the case when $\mathbf{w}$ is orthogonal to Γ_{out} this results in the *homogeneous Neumann boundary conditions* for the matrices in the PCD preconditioner.

Using similar arguments one can show that Neumann boundary conditions "inside" $\hat{S}$ are appropriate for the characteristic part of $\partial\Omega$.

2.2. BFBt. Next we consider the BFBt preconditioner, proposed by Elman [16] and further developed by Elman and coauthors in [18]. The best available modification from [18] is

$$\hat{S}^{-1} := (B\hat{M}_u^{-1}B^T)^{-1}B\hat{M}_u^{-1}A\hat{M}_u^{-1}B^T(B\hat{M}_u^{-1}B^T)^{-1}, \tag{2.9}$$

where $\hat{M}_u$ is a diagonal approximation of M_u. In the case of continuous pressure elements the BFBt preconditioner may be reformulated as

$$\hat{S}^{-1} := L_p^{-1}B\hat{M}_u^{-1}A\hat{M}_u^{-1}B^TL_p^{-1}.$$

Below some observations on the BFBt preconditioner are listed. On the positive side one has the following:

1. In contrast to the PCD method, the preconditioner (2.9) is built from the matrices readily available. Indeed, matrices A and B are already in the system (1.8), and $\hat{M}_u$ is fairly easy to construct from M_u by the lumping procedure.
2. The preconditioner can be defined for general problems like (1.8), although in the general case one may need to find some other scaling matrices instead of the velocity mass matrix.
3. We found the BFBt preconditioner to be robust with respect to ν for the simplest parallel constant wind, $\mathbf{w} = (1, 0)$, and continuous pressure elements.
4. The issue of proper pressure boundary conditions does not arise explicitly.

On the other hand, one has the following:

1. The dependence on ν^{-1} is still observed for more complicated flows, like circulating flows.
2. h-independence of the convergence rate is observed for the special case of small ν, parallel constant wind with isoP_2-P_1 elements. Otherwise the BFBt method shows some h-dependence.
3. Two pressure Poisson problems should be solved instead of one, as for the PCD preconditioner.

We have no clear explanation of why, for certain discretizations, the mesh-dependent convergence rates occur for the BFBt preconditioner (see some arguments in section 2.3). In [16] and [17] Elman observed some h-dependence in the convergence rate of the GMRES method using the BFBt preconditioner for the finite difference (MAC) and finite element Q_2-Q_1 discretizations. Some h-dependence was also observed by Vainikko and Graham in [39] with Q_2-P_{-1} elements and by Hemmingsson and Wathen in [26] with a finite difference method. In all these papers the variant of the preconditioner with identity matrices instead of $\hat{M}_u$ in (2.9) was used. In the recent paper [18] it was noted that introducing $\hat{M}_u$ in (2.9) improves the situation significantly in the case of Q_2-Q_1 finite element discretizations, leading to virtually no h-dependence. The explanation of this phenomenon was partially heuristic. Furthermore, if one considers a uniform grid and applies finite differences or finite elements with piecewise linear velocity, then $\hat{M}_u$ is a scaled identity matrix (at least for the case of Dirichlet boundary conditions in (1.3)). Thus in this case the matrix $\hat{M}_u$ in (2.9) has no effect on the preconditioner. We note that a possible deterioration of convergence rates is consistent with available eigenvalue estimates for this preconditioner (see (3.2)) which also show the h-dependence.

An attempt to extend the preconditioner for the pressure-stabilized elements was recently made by Elman and coauthors [19]. It leads to a somewhat more complicated definition of $\hat{S}^{-1}$.

We have not found results in the literature about the performance of the BFBt preconditioner in the case of stretched grids.

2.3. More preconditioners. Our motivation is to build a preconditioner based on available matrices (blocks) in the spirit of BFBt with similar "robustness" properties with respect to ν, but without possible convergence failures for small h. To this end, let us consider the preconditioner (2.9) once again. If one ignores the velocity stabilization terms in A, the continuous counterpart of the BFBt preconditioner can be written as

$$\Delta_N^{-1}\mathrm{div}\,(-\nu\Delta+\mathbf{w}\cdot\nabla)\,\nabla\,\Delta_N^{-1}, \tag{2.10}$$

where Δ_N^{-1} is the solution operator for the Poisson problem with Neumann boundary conditions. Consider the operator (2.10) acting on some $p\in L_2(\Omega)$. Trouble might be caused by the lack of a proper tangential boundary condition for $\mathbf{v}=\nabla\,\Delta_N^{-1}p$. Indeed, for the normal component of $\mathbf{v}$ we have $\mathbf{v}\cdot\mathbf{n}=0$ on $\partial\Omega$, but the tangential component does not necessarily vanish on $\partial\Omega$. At the same time the computing of $(\nu\Delta+\mathbf{w}\cdot\nabla)\mathbf{v}$ requires $\mathbf{v}=0$ on $\partial\Omega$, since the discrete counterpart of $(\nu\Delta+\mathbf{w}\cdot\nabla)$—matrix A—was built assuming homogeneous Dirichlet boundary conditions for velocity. Similarly, for the vector function $\mathbf{u}=(\nu\Delta+\mathbf{w}\cdot\nabla)\mathbf{v}$ the normal condition $\mathbf{u}\cdot\mathbf{n}=0$ is not ensured, although this condition was used to define the matrix B—a discrete counterpart of div. Another relevant question is whether the operator in (2.10) is well defined in a bounded domain as an operator from L_0^2 to L_0^2. It can be shown that this requires the H^2-regularity assumption for the Poisson problem and proper homogeneous boundary conditions for the intermediate function $\mathbf{u}$ as discussed above.

It is necessary to point out that such regularity and boundary condition issues formally do not arise on the discrete level. However, the failure of the discrete operator to approximate a well-posed continuous counterpart as $h\to 0$ may be a reason for the h-dependence of the BFBt preconditioner for some discretizations.

The suggested remedy is to commute ∇ and div with Δ_N^{-1} in (2.10). We obtain

$$\mathrm{div}\,\Delta_0^{-1}(-\nu\Delta+\mathbf{w}\cdot\nabla)\,\Delta_0^{-1}\nabla, \tag{2.11}$$

where Δ_0^{-1} is a solution operator for the velocity vector Poisson problem with Dirichlet boundary conditions. Note that instead of the pressure Poisson problem like in the PCD and BFBt preconditioners the new approach involves the velocity Poisson problem. This allows one to avoid issues of pressure boundary conditions and regularity. The commutation property which we used holds only with special boundary conditions, e.g., periodic; this property is even more questionable for discrete operators. However, variants of such commutation arguments are often used in the literature to deduce PCD and BFBt preconditioners. The discrete operator corresponding to (2.11) is

$$\hat{S}^{-1}:=\hat{M}_p^{-1}BL^{-1}AL^{-1}B^T\hat{M}_p^{-1}. \tag{2.12}$$

Here L^{-1} is an approximate solve for the discrete velocity vector Poisson problem.

Our observations about this preconditioner are the following. On the positive side one has the following:

1. The preconditioner (2.12) is built from the matrices already available: matrix L is the diffusion part of matrix A.
2. The action of L^{-1} may be performed using the same technology as that of $\hat{A}^{-1}$ (MG, AMG, etc.).

3. Since the preconditioner does not use a discrete pressure Poisson solver, the issue of appropriate boundary conditions does not arise.
4. Preconditioner (2.12) can be easily extended for more general linearized Navier–Stokes-type problems.
5. In our experiments preconditioner (2.12) shows h-independent convergence results in a wider set of cases: both FE choices, various convection fields, up to $\nu = 10^{-3}$. This is supported by the h-independent eigenvalue estimates, which we prove in the next section.

On the other hand, one has the following:

1. Dependence on ν^{-1} is observed for more complicated flows, like circulating flows.
2. For the diffusion dominated case the condition number of the preconditioned matrix $\hat{S}^{-1}S$ is squared compared to the optimal mass matrix preconditioner. Indeed, if $\mathbf{w} = 0, \nu = 1$, and L^{-1} is the exact inverse, we have $A = L$ and
$$\hat{S}^{-1}S = \hat{M}_p^{-1}BL^{-1}AL^{-1}B^T\hat{M}_p^{-1}S = (\hat{M}_p^{-1}S)^2.$$
This results in nearly doubling the iteration numbers for the diffusion dominated case.
3. Compared to the PCD and BFBt preconditioners, the matrix L in (2.12) has a larger dimension than L_p or $(B\hat{M}_u^{-1}B^T)$.

Remark 2.1. Similar to BFBt, the new preconditioner (2.12) cannot be immediately used for the LBB unstable finite elements. However, for this case it admits a simple modification:
$$S^{-1} := \hat{M}_p^{-1}(BL^{-1}AL^{-1}B^T + C)\hat{M}_p^{-1}.$$
It is easy to show that for the symmetric case this modification ensures h-independent convergence. For the nonsymmetric case h-independent eigenvalue bounds will be proved in the next section.

Let us also mention a method from [9] for constructing a preconditioner $\hat{S}$. The approach is quite different from the techniques considered above. It is based on a hierarchical matrix technique for building approximate inverses for matrices. However, the numerical experiments in [9] show a large setup time in this approach, which makes it rather expensive in practice. Finally, we remark that effective pressure Schur complement preconditioners can be built for the linearized Navier–Stokes equations with nonlinear terms written in the rotation form [33, 31].

3. Eigenvalues estimates. It is well known that characterizing the rate of convergence of nonsymmetric preconditioned iterations can be a difficult task. In particular, eigenvalue information alone may not be sufficient to give meaningful estimates of the convergence rate of a method like preconditioned GMRES [25]. The situation is even more complicated for a method like BiCGStab, for which virtually no convergence theory exists. Nevertheless, experience shows that for many linear systems arising in practice, a well-clustered spectrum (away from zero) usually results in rapid convergence of the preconditioned iteration. Therefore, in this section we recall some known estimates for the eigenvalues of the preconditioned Schur complement with the PCD (2.1) and BFBt (2.9) preconditioners for LBB stable elements. Bounds for the PCD preconditioning will be extended to the pressure stabilized case. Also we prove analogous estimates for the new preconditioning (2.12).

Below we use the following notation: $\|\cdot\|$, $\langle\cdot,\cdot\rangle$ denotes the Euclidean norm and scalar product. We also define the norm $\|q\|_* := \langle M_p^{-1}q, q\rangle^{\frac{1}{2}}$. Note that for a matrix

$D \in \mathbb{R}^{m\times m}$ and corresponding matrix norms it holds that $\|D\|_* = \|M_p^{-\frac{1}{2}} D M_p^{\frac{1}{2}}\|$. Furthermore, the PCD preconditioner from (2.1) we denote by S_1, the BFBt preconditioner from (2.9) we denote by S_2, and S_3 will be the new preconditioner from (2.12).

Assume a quasi-uniform discretization (partition into triangles or quadrilaterals) of Ω. Let h denote a maximum diameter of elements. Assume that finite element spaces $\mathbb{V}_h$ and $\mathbb{Q}_h$ satisfy standard approximation properties and inverse inequalities. For the LBB stable case in [20] the following bounds for the eigenvalues of the preconditioned Schur complement were proved:

$$c_1 \le |\lambda(S{S_1}^{-1})| \le C_1, \tag{3.1}$$

$$c_2 \le |\lambda(S{S_2}^{-1})| \le C_2 h^{-2} \tag{3.2}$$

with positive constants c_1, c_2, C_1, C_2 independent of the meshsize h. For the pressure stabilized case ($C \neq 0$) we do not find in the literature any eigenvalue estimates with the PCD preconditioner S_1. Hence we give the proof of such bounds below in Theorem 3.2. As we already mentioned, using the BFBt preconditioner S_2 in the pressure stabilized case is not straightforward and requires some modifications [19]; this modified preconditioner is not considered in the paper. Also no eigenvalue estimates are known for the modified preconditioning.

Remark 3.1. The constants in (3.1), (3.2) may depend on other parameters; in particular, they depend on viscosity ν. It is hard to find this dependence in an optimal way. At the same time, one-dimensional analysis from [16] suggests that the upper bound in (3.2) may be tight with respect to h at least for some discretizations.

The following theorem extends results in (3.1) for the LBB unstable case and provides h-independent bounds for preconditioner (2.12). For the sake of brevity we prove the theorem assuming the exact inverses M_p^{-1} and L^{-1}. More practical choices involving the use of spectrally equivalent preconditioners do not change the result.

THEOREM 3.2. *For a quasi-uniform discretization of Ω assume that finite element spaces $\mathbb{V}_h$ and $\mathbb{Q}_h$, forming a* not *necessarily LBB stable pair, satisfy standard approximation and inverse inequalities. Assume conditions* (1.5)–(1.7) *for the bilinear forms of the discrete Oseen problem. The following estimates hold:*

$$c_1 \le |\lambda(S{S_1}^{-1})| \le C_1, \tag{3.3}$$

$$c_3 \le |\lambda(S{S_3}^{-1})| \le C_3 \tag{3.4}$$

with positive constants c_1, C_1, c_3, C_3 independent of the meshsize h.

To prove the theorem we will need several auxiliary estimates, which we put together in the following lemma.

LEMMA 3.3. *The ellipticity, continuity, and stability assumptions* (1.5)–(1.7)

yield the following estimates involving matrices A, L, B, C, and M_p:

$$\alpha_1\alpha_2^{-2}\|z\|^2 \le \langle A^{-1}L^{\frac12}z, L^{\frac12}z\rangle \qquad \forall\, z\in\mathbb{R}^n, \tag{3.5}$$

$$\|L^{\frac12}A^{-1}L^{\frac12}\| \le \alpha_1^{-1}, \tag{3.6}$$

$$\|L^{-\frac12}AL^{-\frac12}\| \le \alpha_2, \tag{3.7}$$

$$\|M_p^{-\frac12}BL^{-\frac12}\| = \|L^{-\frac12}B^TM_p^{-\frac12}\| \le \gamma_3, \tag{3.8}$$

$$\|M_p^{-\frac12}CM_p^{-\frac12}\| \le \gamma_2, \tag{3.9}$$

$$\langle (BL^{-1}B^T + C)q, q\rangle \ge \gamma_1^2\langle M_pq, q\rangle \qquad \forall\, q\in\mathbb{R}^m. \tag{3.10}$$

Proof. Note that the continuity and ellipticity estimates from (1.5) can be rewritten in the matrix-vector notation

$$\langle Au, v\rangle \le \alpha_2\|L^{\frac12}u\|\|L^{\frac12}v\| \qquad \forall\, u, v\in\mathbb{R}^n, \tag{3.11}$$

$$\alpha_1\|L^{\frac12}u\|^2 \le \langle Au, u\rangle \qquad \forall\, u\in\mathbb{R}^n. \tag{3.12}$$

For arbitrary $z\in\mathbb{R}^n$ consider $u = A^{-1}z$ and $v = L^{-1}z$. Due to (3.11) and (3.12) one gets

$$\begin{aligned}\langle L^{-1}z, z\rangle = \langle v, z\rangle = \langle v, Au\rangle &\le \alpha_2\|L^{\frac12}u\|\|L^{\frac12}v\| = \alpha_2\|L^{\frac12}u\|\langle L^{-1}z, z\rangle^{\frac12}\\ &\le \alpha_1^{-\frac12}\alpha_2\langle Au, u\rangle^{\frac12}\langle L^{-1}z, z\rangle^{\frac12} = \alpha_1^{-\frac12}\alpha_2\langle A^{-1}z, z\rangle^{\frac12}\langle L^{-1}z, z\rangle^{\frac12}.\end{aligned}$$

This yields $\langle L^{-1}z, z\rangle \le \alpha_1^{-1}\alpha_2^2\langle A^{-1}z, z\rangle$ for any $z\in\mathbb{R}^n$ which is equivalent to (3.5). Furthermore, consider the following relations:

$$\begin{aligned}\langle A^{-1}z, z\rangle = \langle u, z\rangle = \langle u, Lv\rangle &\le \|L^{\frac12}u\|\|L^{\frac12}v\| \le \alpha_1^{-\frac12}\langle Au, u\rangle^{\frac12}\|L^{\frac12}v\|\\ &= \alpha_1^{-\frac12}\langle A^{-1}z, z\rangle^{\frac12}\langle L^{-1}z, z\rangle^{\frac12}.\end{aligned}$$

Thus we obtain

$$\alpha_1\langle A^{-1}z, z\rangle \le \langle L^{-1}z, z\rangle \qquad \forall\, z\in\mathbb{R}^n.$$

We use this inequality and (3.12) to check

$$\langle A^{-1}z, y\rangle \le \alpha_1^{-1}\langle L^{-1}z, z\rangle^{\frac12}\langle L^{-1}y, y\rangle^{\frac12} \qquad \forall\, y, z\in\mathbb{R}^n. \tag{3.13}$$

Indeed, denoting $u = A^{-1}z$ and $v = L^{-1}y$, we have

$$\begin{aligned}\langle A^{-1}z, y\rangle = \langle u, y\rangle = \langle u, Lv\rangle &\le \|L^{\frac12}u\|\|L^{\frac12}v\| \le \alpha_1^{-\frac12}\langle Au, u\rangle^{\frac12}\langle L^{-1}y, y\rangle^{\frac12}\\ &= \alpha_1^{-\frac12}\langle A^{-1}z, z\rangle^{\frac12}\langle L^{-1}y, y\rangle^{\frac12} \le \alpha_1^{-1}\langle L^{-1}z, z\rangle^{\frac12}\langle L^{-1}y, y\rangle^{\frac12}.\end{aligned}$$

Now (3.6) follows from (3.13) through

$$\|L^{\frac12}A^{-1}L^{\frac12}\| = \sup_{x\ne0}\sup_{y\ne0}\frac{\langle L^{\frac12}A^{-1}L^{\frac12}x, y\rangle}{\|x\|\|y\|} = \sup_{x\ne0}\sup_{y\ne0}\frac{\langle A^{-1}x, y\rangle}{\|L^{-\frac12}x\|\|L^{-\frac12}y\|} \le \alpha_1^{-1}.$$

In the same way (3.7) follows from (3.11):

$$\|L^{-\frac{1}{2}}AL^{-\frac{1}{2}}\| = \sup_{u\neq 0}\sup_{v\neq 0}\frac{\langle L^{-\frac{1}{2}}AL^{-\frac{1}{2}}u,v\rangle}{\|u\|\|v\|} = \sup_{u\neq 0}\sup_{v\neq 0}\frac{\langle Au,v\rangle}{\|L^{\frac{1}{2}}u\|\|L^{\frac{1}{2}}v\|} \leq \alpha_2.$$

Finally, thanks to (1.7) we get

$$\|L^{-\frac{1}{2}}B^TM_p^{-\frac{1}{2}}\|$$

$$= \sup_{q\neq 0}\frac{\|L^{-\frac{1}{2}}B^Tq\|}{\|M_p^{\frac{1}{2}}q\|} = \sup_{q\neq 0}\sup_{v\neq 0}\frac{\langle L^{-\frac{1}{2}}B^Tq,v\rangle^{\frac{1}{2}}}{\|M_p^{\frac{1}{2}}q\|\|v\|} = \sup_{q\neq 0}\sup_{v\neq 0}\frac{\langle q,Bv\rangle^{\frac{1}{2}}}{\|M_p^{\frac{1}{2}}q\|\|L^{\frac{1}{2}}v\|} \leq \gamma_3$$

and $\|M_p^{-\frac{1}{2}}BL^{-\frac{1}{2}}\| = \|(L^{-\frac{1}{2}}B^TM_p^{-\frac{1}{2}})^T\| \leq \gamma_3$.

Inequalities (3.9) and (3.10) easily follow from the conditions (1.7) and (1.6), respectively. □

Now we are in position to prove Theorem 3.2.

Proof. The proof uses the technique of norm equivalence developed in [20]. In particular, we will show that

$$c_5\|M_pq\|_* \leq \|Sq\|_* \leq C_5\|M_pq\|_* \qquad \forall\ q\in\mathbb{Q}_h \tag{3.14}$$

and prove the estimates for S_k:

$$c_4\|M_pq\|_* \leq \|S_kq\|_* \leq C_4\|M_pq\|_*, \quad k=1,3, \qquad \forall\ q\in\mathbb{Q}_h. \tag{3.15}$$

From (3.15) and (3.14) one obtains the norm equivalence

$$c\|S_kq\|_* \leq \|Sq\|_* \leq C\|S_kq\|_*, \quad k=1,3, \qquad \forall\ q\in\mathbb{Q}_h \tag{3.16}$$

with mesh-independent positive constants c and C. To complete the proof one may consider the obvious inequalities

$$\|S_kS^{-1}\|_*^{-1} \leq |\lambda(SS_k^{-1})| \leq \|SS_k^{-1}\|_*. \tag{3.17}$$

As a consequence of (3.16) and (3.17) the estimate (3.4) follows from the eigenvalues of $S_k^{-1}S$ with some constants c_k, C_k independent of the meshsize h.

Therefore, we can focus on checking (3.14) and (3.15). First we prove (3.14). The upper bound in (3.14) follows from

$$\|SM_p^{-1}\|_* = \|M_p^{-\frac{1}{2}}(BA^{-1}B^T+C)M_p^{-\frac{1}{2}}\|$$

$$\leq \|M_p^{-\frac{1}{2}}BL^{-\frac{1}{2}}\|\|L^{\frac{1}{2}}A^{-1}L^{\frac{1}{2}}\|\|L^{-\frac{1}{2}}B^TM_p^{-\frac{1}{2}}\| + \|M_p^{-\frac{1}{2}}CM_p^{-\frac{1}{2}}\| \leq \gamma_3\alpha_1^{-1}+\gamma_2;$$

here we used estimates (3.6), (3.8), and (3.9). Next we find a lower bound for $\frac{\|SM_p^{-1}q\|_*}{\|q\|_*}$:

$$\inf_{q\neq 0}\frac{\|SM_p^{-1}q\|_*}{\|q\|_*} = \inf_{q\neq 0}\frac{\|M_p^{-\frac{1}{2}}SM_p^{-\frac{1}{2}}q\|}{\|q\|} \geq \inf_{q\neq 0}\frac{\langle M_p^{-\frac{1}{2}}SM_p^{-\frac{1}{2}}q,q\rangle}{\|q\|^2} \tag{3.18}$$

$$= \inf_{q\neq 0}\frac{\langle A^{-1}B^TM_p^{-\frac{1}{2}}q,B^TM_p^{-\frac{1}{2}}q\rangle + \langle CM_p^{-\frac{1}{2}}q,M_p^{-\frac{1}{2}}q\rangle}{\|q\|^2}$$

with use of the Cauchy–Schwarz inequality. Denoting $u = B^T M_p^{-\frac{1}{2}} q$ and applying (3.5) and (3.10), we continue with (3.18):

$$\inf_{q\neq 0} \frac{\|SM_p^{-1}q\|_*}{\|q\|_*} \geq \inf_{q\neq 0} \frac{\alpha_1\alpha_2^{-2}\langle L^{-1}B^T M_p^{-\frac{1}{2}} q, B^T M_p^{-\frac{1}{2}} q\rangle + \langle CM_p^{-\frac{1}{2}} q, M_p^{-\frac{1}{2}} q\rangle}{\|q\|^2}$$

$$= \inf_{q\neq 0} \frac{\langle (\alpha_1\alpha_2^{-2} BL^{-1}B^T + C)q, q\rangle}{\|M_p^{\frac{1}{2}} q\|^2} \geq \min\{\alpha_1\alpha_2^{-2}, 1\}\gamma_1^2.$$

Thus the lower bound in (3.14) is proved with the h-independent constant $c_5 = \min\{\alpha_1\alpha_2^{-2}, 1\}\gamma_1^2$.

For $k = 1$ inequalities (3.15) were proved in [20]. The proof in [20] does not use the LBB stability assumption. Thus we can use this result and conclude that eigenvalue bounds (3.3) hold with h-independent constants c_1, C_1.

To show (3.15) for $k = 3$ we first prove an upper bound on $\|M_p S_3^{-1}\|_*$. Thanks to (3.7), (3.8), and (3.9), one gets

$$\|M_p S_3^{-1}\|_* = \|M_p^{-\frac{1}{2}}(BL^{-1}AL^{-1}B^T + C)M_p^{-\frac{1}{2}}\|$$

$$\leq \|M_p^{-\frac{1}{2}} BL^{-\frac{1}{2}}\|\|L^{-\frac{1}{2}} AL^{-\frac{1}{2}}\|\|L^{-\frac{1}{2}} B^T M_p^{-\frac{1}{2}}\| + \|M_p^{-\frac{1}{2}} CM_p^{-\frac{1}{2}}\| \leq \gamma_3\alpha_2 + \gamma_2.$$

Hence the low bound in (3.15) holds with the h-independent constant $c_4 = \frac{1}{(\gamma_3\alpha_2+\gamma_2)}$.

Finally, we find an upper bound for $\|S_3 M_p^{-1}\|_*$:

$$\|S_3 M_p^{-1}\|_*^{-1} = \|M_p^{-\frac{1}{2}} S_3 M_p^{-\frac{1}{2}}\|^{-1} \tag{3.19}$$

$$= \inf_{q\neq 0}\sup_{p\neq 0} \frac{\langle M_p^{\frac{1}{2}} S_3^{-1} M_p^{\frac{1}{2}} q, p\rangle}{\|q\|\|p\|} \geq \inf_{q\neq 0} \frac{\langle M_p^{\frac{1}{2}} S_3^{-1} M_p^{\frac{1}{2}} q, q\rangle}{\|q\|^2}$$

$$= \inf_{q\neq 0} \frac{\langle AL^{-1}B^T M_p^{-\frac{1}{2}} q, L^{-1}B^T M_p^{-\frac{1}{2}} q\rangle + \langle CM_p^{-\frac{1}{2}} q, M_p^{-\frac{1}{2}} q\rangle}{\|q\|^2}.$$

Denote $u = L^{-\frac{1}{2}} B^T M_p^{-\frac{1}{2}} q$. The condition (3.12) yields $\langle AL^{-\frac{1}{2}} u, L^{-\frac{1}{2}} u\rangle \geq \alpha_1\|u\|^2$. Therefore, we continue with (3.19):

$$\|S_3 M_p^{-1}\|_*^{-1} \geq \inf_{q\neq 0} \frac{\alpha_1\langle L^{-\frac{1}{2}} B^T M_p^{-\frac{1}{2}} q, L^{-\frac{1}{2}} B^T M_p^{-\frac{1}{2}} q\rangle + \langle CM_p^{-\frac{1}{2}} q, M_p^{-\frac{1}{2}} q\rangle}{\|q\|^2}$$

$$= \inf_{q\neq 0} \frac{\langle (\alpha_1 BL^{-1}B^T + C)q, q\rangle}{\|M_p^{\frac{1}{2}} q\|^2} \geq \min\{\alpha_1, 1\}\gamma_1^2.$$

Thus the upper bound in (3.15) is proved with the h-independent constant $C_4 = (1 + \alpha_1^{-1})\gamma_1^{-2}$. ∎

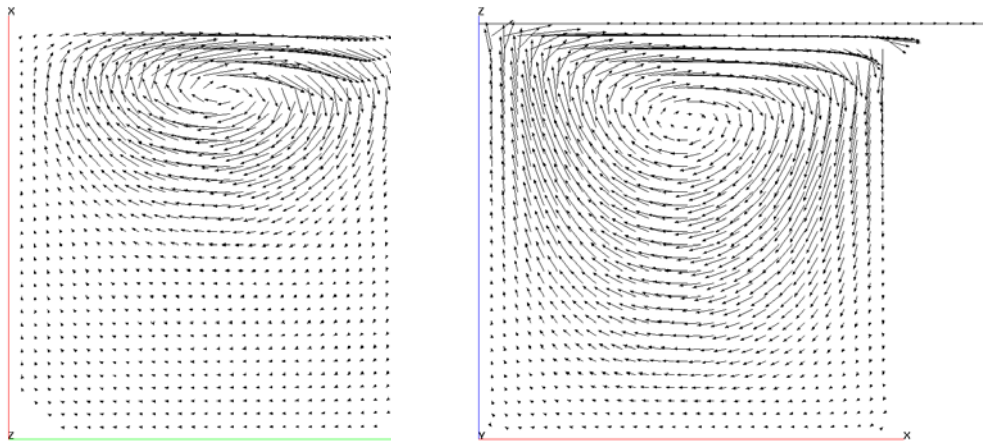


FIG. 4.1. *Left picture: The 2D velocity field* (4.2). *Right picture: The central cross-section by the OXZ-plane of the 3D velocity field.*

4. Numerical experiments. In this section we present numerical results for two model problems in $[0,1]^d$ for $d = 2, 3$. In the first problem the wind is parallel to the x-axis and constant:

$$\mathbf{w} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{or} \quad \mathbf{w} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}. \tag{4.1}$$

The second problem is the linearized "driven cavity" problem. In the 2D case the velocity function is suggested in [7]:

$$\mathbf{w} = \begin{pmatrix} \frac{r_2}{2\pi}\frac{e^{r_2 y}}{(e^{r_2}-1)}\sin\left(\frac{2\pi(e^{r_2 y}-1)}{e^{r_2}-1}\right)\left(1-\cos\left(\frac{2\pi(e^{r_1 x}-1)}{e^{r_1}-1}\right)\right) \\ -\frac{r_1}{2\pi}\frac{e^{r_1 x}}{(e^{r_1}-1)}\sin\left(\frac{2\pi(e^{r_1 x}-1)}{e^{r_1}-1}\right)\left(1-\cos\left(\frac{2\pi(e^{r_2 y}-1)}{e^{r_2}-1}\right)\right) \end{pmatrix}, \tag{4.2}$$

where $r_1 = 4$, $r_2 = 0.1$. This type of convection simulates a rotating vortex, whose center has coordinates (x_0, y_0), $x_0 \approxeq 0.831$, $y_0 \approxeq 0.512$, and $\max_\Omega |\mathbf{w}| \approxeq 1$ (Figure 4.1, left). In the 3D case the convection velocity field $\mathbf{w}$ is the solution of the "driven cavity" Stokes problem (Figure 4.1, right).

For the discretization method we use isoP_2-P_0 and isoP_2-P_1 finite elements defined on uniform triangulation (tetrahedrization) of a square (cubic) mesh in $[0,1]^d$. The velocity triangulation is built by connecting the midpoints on the edges of triangles or tetrahedra. In all the cases the convection term is stabilized by using the SUPG stabilization [36]. In the tables below h denotes the size for the pressure triangulations.

We use the block triangular matrix (1.9) as a right preconditioner in the Krylov subspace method for solving system (1.8). Some details of experiments may differ in the 2D and 3D cases (cf. below), since we used two different FE software packages to treat 2D and 3D problems, respectively. In the 2D experiments we used the BiCGStab and for 3D ones the full GMRES method was applied as an outer iterative solver. Note that the expense of one BiCGStab iteration approaches the cost of two GMRES iterations. The stopping criterion is the 10^{-6} decrease of the Euclidean norm of the residual. The approximate inverses involved in the application of the

TABLE 4.1
Number of the preconditioned iterations with the PCD preconditioner; isoP_2-P_1 FE; Neumann/Dirichlet boundary conditions in A_p.

Meshsize h	Viscosity ν			
	0.1	0.01	10^{-3}	10^{-4}
w is 2D cavity vortex, #BiCGStab				
1/32	12 / 12	24 / 24	57 / 67	248 / 769
1/64	12 / 12	22 / 22	64 / 81	619 / 1267
w is constant 2D wind, #BiCGStab				
1/32	12 / 13	27 / 21	124 / 28	1027 / 48
1/64	11 / 13	23 / 24	116 / 33	1425 / 51
w is constant 3D wind, #GMRES				
1/8	28 / 36	69 / 58	368 / 201	548 / 285
1/16	28 / 42	45 / 39	232 / 162	1108 / 430

preconditioner (1.9) were computed as follows. The application of $\hat{A}^{-1}$ to a vector is achieved via 3 multigrid W(1,1)-cycles with the alternating Gauss–Seidel method as a smoother. In the 2D case a W-cycle of geometric multigrid was used, whereas in the 3D case a V-cycle of algebraic multigrid was adopted. Both choices give a fairly good approximation to A^{-1} for all values of h and ν under consideration. Application of $(B\hat{M}_u^{-1}B^T)^{-1}$ and L_p^{-1} was evaluated using 10 V(4,4)-cycles in the 2D case and exact sparse factorization in the 3D case. We note that although the latter method is applicable to general meshes, it is asymptotically not optimal. If N is the number of the pressure degrees of freedom, then the method needs about $O(N^{2.2})$ flops for the computation of $O(N^{1.5})$ nonzero entries in triangular factors. This limits the size of L_p to several tens of thousands for the PC we used for running numerical tests. At the same time, in the case of isoP_2-P_1 elements, N is less than the size of matrix A by a factor of 24, and the practical limit for N is not very restrictive. L^{-1} was evaluated using interior iterations to provide a very good approximation of the inversion. Thus all the inverses involved in all Schur complement preconditioners were evaluated with pretty high accuracy.

In the first experiment we illustrate the effect of different boundary conditions in A_p on the performance of the PCD preconditioner. Recall that our analysis in section 2.1 suggests that for the problem with **w** from (4.1) one should implement Dirichlet conditions at the inflow boundaries, while for the problem with **w** from (4.2) one should use Neumann conditions on the entire boundary. In Table 4.1 we compare iteration counts for the PCD preconditioner with different boundary conditions in A_p in two and three dimensions and for two types of convection flow **w**. For the case of **w** from (4.1) we test two variants of boundary conditions in A_p: one consists in setting Neumann boundary conditions on the whole boundary; alternatively, we set Dirichlet boundary conditions on the inflow and Neumann conditions on the rest of the boundary. For the case of **w** from (4.2) the problem has only characteristic boundaries; thus we test setting either Dirichlet or Neumann conditions in A_p on the whole boundary. It was mentioned in section 2.1 that Dirichlet boundary conditions may not be imposed on the boundary nodes, since these nodes contribute to the set of pressure degrees of freedom. The Dirichlet condition is imposed on fictitious boundary nodes of an h-extension of the original mesh. Therefore, the actual boundary nodes are considered as interior in the extended mesh. This may be implemented in two ways. For a rectangular mesh we simply copy matrix entries for interior nodes to

TABLE 4.2
Number of preconditioned iterations for isoP_2-P_1 FE. PCD/BFBt/preconditioner (2.12).

Meshsize h	Viscosity ν			
	0.1	0.01	10^{-3}	10^{-4}
w is 2D constant wind, #BiCGStab				
1/32	13 / 8 / 22	21 / 6 / 35	28 / 8 / 39	48 / 10 / 46
1/64	13 / 12 / 27	24 / 8 / 34	33 / 7 / 34	51 / 12 / 40
1/128	13 / 17 / 23	24 / 16 / 30	41 / 6 / 37	62 / 9 / 37
w is 2D cavity vortex, #BiCGStab				
1/32	12 / 9 / 27	24 / 14 / 29	56 / 30 / 70	248 / 120 / 242
1/64	12 / 13 / 27	22 / 20 / 31	64 / 45 / 76	619 / 211 / 314
1/128	13 / 17 / 29	25 / 26 / 33	77 / 61 / 69	1053 / 349 / 446
w is 3D cavity vortex, #GMRES				
1/8	35 / 71 / 129	46 / 104 / 178	112 / 233 / 512	243 / 437 / 787
1/16	38 / 82 / 143	50 / 102 / 167	114 / 415 / 781	462 / 1402 / >2000

matrix entries for actual boundary nodes. However, for a general mesh one has to generate the fictitious mesh layer by reflecting the close-to-boundary layer of cells with respect to the actual boundary.

In L_p we use Neumann conditions only. We found that changing boundary conditions in L_p does not improve convergence rates. Results of the experiments in Table 4.1 are consistent with the analysis in section 2.1. In particular, better results with Neumann conditions for the cavity vortex test confirm that this is the right choice on characteristic boundaries, for the constant wind Dirichlet boundary conditions on the inflow show an advantage for small enough ν. This behavior is also expected, since the analysis in section 2.1, predicting this choice, was done for the limit case of $\nu = 0$. The same phenomenon was observed in numerical experiments for discretizations with isoP_2-P_0 finite elements. Thus, further in the experiments we will always define A_p with Neumann conditions for rotating **w** and Dirichlet conditions at the "inflow" for **w** from (4.1).

In Table 4.2 we compare convergence results for all three preconditioners tested for the isoP_2-P_1 discretization of the Oseen problem with **w** from (4.1) and (4.2), the latter case being examined in both two and three dimensions. All preconditioners demonstrate almost h-independent results except for the case of small ν. In general, rotating flow with small viscosity turns out to be a hard problem for all preconditioners.

The difficulty in treating the case of small ν may be related to the fact that Navier–Stokes flows become less stable in this case. For example, in [2] it was found that the first point of bifurcation for the 2D driven cavity problem occurs around $Re = 8.018$. Therefore, for the case of small viscosity solving unsteady, rather then steady, Navier–Stokes equations can be more appropriate sometimes. After linearization the unsteady problem would lead to a slightly modified Oseen problem (1.1)–(1.3) with an additional positive-definite reaction term in the momentum equations. This additional term, when also included in a Schur complement preconditioner, may improve the performance of the preconditioned iterations for small values of ν [21].

Next we proceed to approximations with the isoP_2-P_0 finite element method. For this discretization with discontinuous pressure approximation the BFBt preconditioner shows a strong h-dependence result even for the simplest constant parallel flow. This is illustrated in Table 4.3.

In Table 4.4 we examine PCD and (2.12) preconditioners which demonstrate h-

TABLE 4.3
Number of preconditioned BiCGStab iterations for the BFBt preconditioner with isoP$_2$-P$_0$ FE.

Meshsize h	Viscosity ν			
	0.1	0.01	10^{-3}	10^{-4}
w is 2D constant wind				
1/32	39	35	52	71
1/64	80	65	93	147
1/128	174	141	154	219

TABLE 4.4
Number of preconditioned BiCGStab iterations for PCD/(2.12) preconditioner with isoP$_2$-P$_0$ FE.

Meshsize h	Viscosity ν			
	0.1	0.01	10^{-3}	10^{-4}
w is 2D constant wind				
1/32	9 / 17	11 / 24	21 / 36	68 / 58
1/64	9 / 13	12 / 23	21 / 32	57 / 66
1/128	9 / 11	14 / 23	25 / 31	50 / 59
w is 2D cavity vortex				
1/32	10 / 18	18 / 22	66 / 60	>2000 / 191
1/64	9 / 21	19 / 23	74 / 63	>2000 / 306
1/128	9 / 21	19 / 23	78 / 58	>2000 / 453

independent convergence rates at least for the simplest case (4.1). We note that in the PCD preconditioner we now use mixed approximation for the pressure Poisson problem: $L_p = (B\hat{M}_u^{-1}B^T)$. To define A_p, we set $F_p := r_{u\to p}\, A_x r_{p\to u}$, where A_x is the x-subblock of A (maybe with different boundary conditions), $r_{u\to p}$ and $r_{p\to u}$ are suitable mappings from $\mathbb{Q}_h$ to $\mathbb{V}_h$, and vice versa. The type of boundary conditions in A_p was taken the same as for isoP_2-P_1 discretizations. We observe that up to $\nu = 10^{-3}$ both methods exhibit feasible convergence rates, and for $\nu = 10^{-4}$ the PCD method fails to converge.

Finally, we remark that although Theorem 3.2 guarantees that all eigenvalues of preconditioned matrices lie in the right half of the complex plane and are bounded independent of h (at least for the PCD and (2.12) preconditioners), iteration counts in Tables 4.2 and 4.4 have some increase with $h \to 0$ in the case of small ν. A most likely explanation of this phenomenon is that the implicit dependence of the constants from bounds (3.3) and (3.4) on ν makes them not useful when $\nu \to 0$.

5. Conclusions. This paper studies a preconditioning technique for finite element discretizations of the Oseen problem arising from Picard linearizations of the steady Navier–Stokes equations. The preconditioner is block triangular and requires an approximation to the inverse of the pressure Schur complement matrix. We focus on several approaches for building the pressure Schur complement preconditioner. Two of them are well known from the literature and one is new. The preconditioners differ in implementation and performance for various discretizations and flow patterns. The paper gives an account of their properties and available theoretical results. We prove missing eigenvalue estimates and discuss some open implementation problems, such as the choice of an appropriate pressure boundary condition in the method of Kay, Loghin, and Wathen. Numerical experiments show that all the methods work satisfactorily (with mild dependence on ν) in the range of small and modest Reynolds numbers; however, they may experience serious loss of efficiency when the Reynolds

number is larger.

REFERENCES

[1] G. P. Astrakhantsev, *Analysis of algorithms of the Arrow Hurwicz type*, Comput. Math. Math. Phys., 41 (2001), pp. 15–26.

[2] F. Auteri, N. Parolini, and L. Quartapelle, *Numerical investigation on the stability of singular driven cavity flow*, J. Comput. Phys., 183 (2002), pp. 1–25.

[3] R. E. Bank, B. D. Welfert, and H. Yserentant, *A class of iterative methods for solving saddle point problems*, Numer. Math., 56 (1990), pp. 645–666.

[4] M. Benzi and G. H. Golub, *A preconditioner for generalized saddle point problems*, SIAM J. Matrix Anal. Appl., 26 (2004), pp. 20–41.

[5] M. Benzi, G. H. Golub, and J. Liesen, *Numerical solution of saddle point problems*, Acta Numer., 14 (2005), pp. 1–137.

[6] M. Benzi and M. A. Olshanskii, *An augmented Lagrangian-based approach to the Oseen problem*, SIAM J. Sci. Comput., 28 (2006), pp. 2095–2113.

[7] S. Berrone, *Adaptive discretization of the Navier Stokes equations by stabilized finite element methods*, Comput. Methods Appl. Mech. Engrg., 190 (2001), pp. 4435–4455.

[8] P. B. Bochev, C. Dohrmann, and M. D. Gunzburger, *Stabilization of low-order mixed finite elements for the Stokes equations*, SIAM J. Numer. Anal., 44 (2006), pp. 82–101.

[9] S. Le Borne, *Hierarchical matrix preconditioners for the Oseen equations*, Comput. Vis. Sci., to appear.

[10] J. H. Bramble, J. E. Pasciak, and A. T. Vassilev, *Uzawa type algorithms for nonsymmetric saddle point problems*, Math. Comp., 69 (2000), pp. 667–689.

[11] J. Cahouet and J. P. Chabard, *Some fast 3D finite element solvers for the generalized Stokes problem*, Internat. J. Numer. Methods Fluids, 8 (1988), pp. 869–895.

[12] C. Calgaro, P. Deuring, and D. Jennequin, *A preconditioner for the generalized saddle point problems: Application to 3D stationary Navier-Stokes equations*, Numer. Methods Partial Differential Equations, 22 (2006), pp. 1289–1313.

[13] Z.-H. Cao, *A class of constraint preconditioners for nonsymmetric saddle point matrices*, Numer. Math., 103 (2006), pp. 47–61.

[14] X. Chen, *On preconditioned Uzawa methods and SOR methods for saddle-point problems*, J. Comput. Appl. Math., 100 (1998), pp. 207–224.

[15] H. S. Dollar, N. I. M. Gould, W. H. A. Schilders, and A. J. Wathen, *Implicit-factorization preconditioning and iterative solvers for regularized saddle-point systems*, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 170–189.

[16] H. C. Elman, *Preconditioning of the steady-state Navier–Stokes equations with low viscosity*, SIAM J. Sci. Comput., 20 (1999), pp. 1299–1316.

[17] H. C. Elman, *Preconditioning strategies for models of incompressible flow*, J. Sci. Comput., 25 (2005), pp. 347–366.

[18] H. C. Elman, V. E. Howle, J. Shadid, R. Shuttleworth, and R. Tuminaro, *Block preconditioners based on approximate commutators*, SIAM J. Sci. Comput., 27 (2006), pp. 1651–1668.

[19] H. C. Elman, V. E. Howle, J. Shadid, D. Silvester, and R. Tuminaro, *Least Squares Preconditioners for Stabilized Discretizations of the Navier-Stokes Equations*, Technical report, University of Maryland, CS Dept., CS-TR-4797, 2006.

[20] H. C. Elman, D. Loghin, and A. J. Wathen, *Preconditioning techniques for Newton's method for the incompressible Navier–Stokes equations*, BIT, 43 (2003), pp. 961–974.

[21] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics*, Numer. Math. Sci. Comput., Oxford University Press, New York, 2005.

[22] M. Fortin and R. Glowinski, *Augmented Lagrangian methods: Applications to the Numerical Solution of Boundary Value Problems*, Stud. Math. Appl. 15, North–Holland, Amsterdam, 1983.

[23] M. Garbey, Yu. A. Kuznetsov, and Yu. V. Vassilevski, *Parallel Schwarz method for a convection-diffusion problem*, SIAM J. Sci. Comput., 22 (2000), pp. 891–916.

[24] R. Glowinski and P. Le Tallec, *Augmented Lagrangian and Operator Splitting Methods in Nonlinear Mechanics*, SIAM Stud. Appl. Math. 9, SIAM, Philadelphia, 1989.

[25] A. Greenbaum, V. Pták, and Z. Strakoš, *Any nonincreasing convergence curve is possible for GMRES*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 465–469.

[26] L. Hemmingsson and A. J. Wathen, *A nearly optimal preconditioner for the Oseen equations*, Numer. Linear Algebra Appl., 8 (2001), pp. 229–243.

[27] V. John, *Higher order finite element methods and multigrid solvers in a benchmark problem for the* 3*D Navier Stokes equations*, Internat. J. Numer. Methods Fluids, 40 (2002), pp. 775–798.

[28] D. Kay, D. Loghin, and A. J. Wathen, *A preconditioner for the steady-state Navier–Stokes equations*, SIAM J. Sci. Comput., 24 (2002), pp. 237–256.

[29] A. Klawonn and G. Starke, *Block triangular preconditioners for nonsymmetric saddle point problem*, Numer. Math., 81 (1999), pp. 577–594.

[30] K. Lipnikov and Yu. V. Vassilevski, *Parallel adaptive solution of the Stokes and Oseen problems on unstructured* 3*D meshes*, in Parallel Computational Fluid Dynamics 2003, Elsevier, Amsterdam, 2004, pp. 153–162.

[31] M. A. Olshanskii, *Iterative solver for Oseen problem and numerical solution of incompressible Navier-Stokes equations*, Numer. Linear Algebra Appl., 6 (1999), pp. 353–378.

[32] M. A. Olshanskii, J. Peters, and A. Reusken, *Uniform preconditioners for a parameter dependent saddle point problem with application to generalized Stokes interface equations*, Numer. Math., 105 (2006), pp. 159–191.

[33] M. A. Olshanskii and A. Reusken, *Navier–Stokes equations in rotation form: A robust multigrid solver for the velocity problem*, SIAM J. Sci. Comput., 23 (2002), pp. 1683–1706.

[34] A. Prohl, *Projection and Quasi-Compressibility Methods for Solving the Incompressible Navier-Stokes Equations*, Teubner, Vienna, 1997.

[35] A. Quarteroni and A. Valli, *Domain Decomposition Methods for Partial Differential Equations*, Oxford University Press, Oxford, UK, 1999.

[36] H.-G. Roos, M. Stynes, and L. Tobiska, *Numerical Methods for Singularly Perturbed Differential Equations: Convection Diffusion and Flow Problems*, Springer Ser. Comput. Math. 24, Springer-Verlag, New York, 1996.

[37] F. Saleri and A. Veneziani, *Pressure correction algebraic splitting methods for the incompressible Navier–Stokes equations*, SIAM J. Numer. Anal., 43 (2005), pp. 174–194.

[38] S. Turek, *Efficient Solvers for Incompressible Flow Problems: An Algorithmic and Computational Approach*, Lect. Notes Comput. Sci. Engrg. 6, Springer-Verlag, Berlin, 1999.

[39] E. Vainikko and I. G. Graham, *A parallel solver for PDE systems and application to the incompressible Navier Stokes equations*, Appl. Numer. Math., 49 (2004), pp. 97–116.

[40] C. Vincent and R. Boyer, *A preconditioned conjugate gradient Uzawa-type method for the solution of the Stokes problem by mixed* $Q1$-$P0$ *stabilized finite elements*, Internat. J. Numer. Methods Fluids, 14 (1992), pp. 289–298.

[41] M. Wabro, *AMGe—coarsening strategies and application to the Oseen equations*, SIAM J. Sci. Comput., 27 (2006), pp. 2077–2097.

[42] A. J. Wathen, D. Loghin, D. Kay, H. C. Elman, and D. Silvester, *A Preconditioner for the* 3*D Oseen Equations*, Oxford University Computing Laboratory Report 02/04, Oxford, UK, 2002.